HAVA HARP OKULU
HAVACILIK VE UZAY TEKNOLOJİLERİ ENSTİTÜSÜ

ÇOKLU OTONOM İNSANSIZ HAVA ARAÇLARI İÇİN PARALEL PROGRAMLAMA TABANLI YOL PLANLAMASI

DOKTORA TEZİ

Ömer ÇETİN

Bilgisayar Mühendisliği Ana Bilim Dalı Başkanlığı

Yazılım Mühendisliği Programı

MAYIS 2015

HAVA HARP OKULU
HAVACILIK VE UZAY TEKNOLOJİLERİ
ENSTİTÜSÜ

ÇOKLU OTONOM İNSANSIZ HAVA ARAÇLARI İÇİN PARALEL PROGRAMLAMA TABANLI YOL PLANLAMASI

DOKTORA TEZİ

Ömer ÇETİN
(110202)

Bilgisayar Mühendisliği Ana Bilim Dalı Başkanlığı

Yazılım Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Hv. Müh. Alb. Güray YILMAZ

MAYIS 2015

Hava Harp Okulu Havacılık ve Uzay Teknolojileri Enstitüsünün 110202 numaralı Doktora Öğrencisi **Hv.Mu.Yzb. Ömer ÇETİN**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra Doç.Dr. Hv.Müh.Alb. Güray YILMAZ danışmanlığında hazırladığı **"ÇOKLU OTONOM İNSANSIZ HAVA ARAÇLARI İÇİN PARALEL PROGRAMLAMA TABANLI YOL PLANLAMA"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Doç. Dr. Hv. Müh. Alb. Güray YILMAZ**
Hava Harp Okulu

Jüri Üyeleri : **Prof. Dr. Can ÖZTURAN**
Boğaziçi Üniversitesi

Doç. Dr. Hv. Müh. Alb. Güray YILMAZ
Hava Harp Okulu

Doç. Dr. Hv. Müh. Yb. İlker BEKMEZCİ
Hava Harp Okulu

Yrd. Doç. Dr. Hüseyin ÜVET
Yıldız Teknik Üniversitesi

Yrd. Doç. Dr. Akhan AKBULUT
Kültür Üniversitesi

Teslim Tarihi : **MAYIS 2015**
Savunma Tarihi : **MAYIS 2015**

Bu tez çalışmasında belirtilen görüş ve yorumlar yazara aittir. Türk Silahlı Kuvvetleri'nin ya da diğer kamu kuruluşlarının görüşlerini yansıtmaz. Ayrıca bu tez çalışması bilimsel ahlak ve etik değerlere uygun olarak yazılmış olup, yararlanılan tüm eserler kaynaklarda gösterilmiştir.

MAYIS 2015 Ömer ÇETİN

Sevgili eşim ve bir tanem kızıma,

ÖNSÖZ

İnsansız Hava Aracı (İHA) sistemleri, sivil ve askeri birçok uygulama alanında son yıllarda artan bir ivme ile kullanılmaya başlanan otonom yapılar olmuşlardır. Yakın gelecekte İHA sistemlerinin otonom yeteneklerindeki gelişmeler ve risk faktörünü her zaman en aza indirgemeyi hedef olarak belirlemiş olan havacılık sektöründe insanlı (pilotlu) hava araçlarının yerlerini İHA sistemlerinin artan bir ivme ile alacağına şühe yoktur. Bu durum insanı sadece sistemi kontrol eden ve denetleyen bir operatör olmaya itecektir. İnsanlı sistemlerin son yıllarda yaygın olarak faydalandıkları formasyon uçuşu ve havada yakıt ikmali gibi görevler veya yüksek hızda çarpışmayı önleyici ve engellerden sakınarak seyrüsefer yapmayı destekleyen yol planlaması gibi yetenekler İHA sistemleri için olması gereken gerçek zamanlı özellikler ve otonom olarak icra edilmesi gereken görevler arasında yer almaktadır.

Çalışma kapsamında ortaya konulan yaklaşımın amacı, en az iki İHA platformunun dar bir alan içinde birbirleri ile çarpışmasını önleyici güvenlik tedbirlerini sağlayarak, kooperatif şekilde engellerden kaçınmasına imkan sağlayan nitelikte, formasyon uçuşu için otonom yol planlamasını gerçek zamanlı olarak sağlamaktır. Yüksek süratli ve agresif hareket edebilen otonom hava araçlarının gerçek zamanlı çözüm ihtiyacına istinaden, bu çerçeve içerisinde yer alan her bir alt problem çalışma kapsamında öncelikle tek tek ele alınmış, ardından topyekün bir sistem çözümü oluşturacak nitelikte uygulamalar ile desteklenerek ortaya konulmuştur.

Grafik işlemcilerin genel maksatlı paralel uygulamaların programlanmasında kullanımının artması ile karmaşık ve yüksek hesaplama maliyeti gerektiren problemlerin gerçek zamanlı olarak maliyet etkin paralel uygulamalar geliştirilerek ele alınmasına imkan sağlanmıştır. Ortaya konulan sistem çözümü, farklı programlama yapıları doğrultusunda gerçekleştirilmiş ve özellikle *NVIDIA firması tarafından çalışmanın desteklenmesi neticesinde elde edilen paralel programlama donanımı olarak kullanılabilen grafik işlemcilerden* faydalanılarak sınanmıştır.

Gerçekleştirilen çalışma kapsamında ortaya konulan sistem çözümünden elde edilen sonuçlar, *TÜBİTAK 1001 - Bilimsel ve Teknolojik Araştırma Projelerini Destekleme Programı kapsamında kabul edilen 112E281 no.lu proje desteği* ile sağlanan donanımlardan faydalanılarak simülasyon çalışmaları şeklinde ve uygulamalar ile sınanmış, yaklaşımın başarısı ortaya konulmuştur.

Çalışma süresince başta eşim ve kızım olmak üzere tüm aileme gösterdikleri sınırsız anlayış ve sabırdan dolayı teşekkürlerimi iletirim. Tez danışmanım Doç.Dr. Hv. Müh.Alb. Güray YILMAZ'a verdiği sonsuz destek, tez izleme komitesinde yer alan Prof.Dr. Can ÖZTURAN'a ve Doç.Dr. Hv.Müh.Yb. İlker BEKMEZCİ'ye çok değerli katkılarından ve yapıcı yönlendirmelerinden dolayı, ayrıca çalışmadaki katkıları nedeniyle Prof.Dr. Okyay KAYNAK'a teşekkürlerimi sunar, minnettarlığımı iletmek isterim.

MAYIS 2015 Ömer ÇETİN

İÇİNDEKİLER

Sayfa

KISALTMALAR

ABD : Amerika Birleşik Devletleri
ALU : Aritmetik Mantıksal Hesaplama Ünitesi (Aritmethic Logic Unit)
APM : Ardupilot Mega Kontrol Donanımı
ARIP : Havada Yakıt İkmali Başlangıç Noktası (Air Refueling Initiation Point)
ARCP : Havada Yakıt İkmali Kontrol Noktası (Air Refueling Control Point)
ARCT : Havada Yakıt İkmali Kontrol Zamanı (Air Refueling Control Time)
AST : Hava Üstünlüğü Hedef Programı (Air Superiority Target Program)
BLCB : Fırçasız Kontrol Kartı (Brushless Control Board)
CCL : Bağlı Bileşen Etiketleme (Connected Component Labeling)
CPU : Merkezi İşlem Birimi (Central Processing Unit)
CUDA : Compute-Unified Device Architecture
DARPA : The Defense Advanced Research Projects Agency
DDR3 : Double Data Rate Type Three Synchronous Dynamic Random Access Memory
DMA : Doğrudan Bellek Erişimi (Direct Memory Access)
DP : Komut Dağıtım Birimleri (Instruction Dispatch Units)
E A/R : Havada Yakıt İkmali Bitiş Noktası (End Air Refueling Point)
ECC : Hata Kotarma Kodu (Error Correction Code)
FP : Kayan Nokta (Floating Point)
GFOLP/s : Saniyede İcra Edilen Kayan Nokta Aritmetiği Sayısı
GNRON : Engellere Çok Yakında Bulunan Hedeflere Erişememe Problemi (Goals Non-Reachable with Obstacles Nearby)
GPGPU : Grafik İşlemci Tabanlı Çok Çekirdekli Paralel Programlama
GPU : Grafik İşlemci Ünitesi (Graphic Processing Unit)
GPS : Küresel Konumlandırma Sistemi (Global Positioning System)
HPC : Yüksek Performanslı Hesaplama (High Performance Computing)
HYİ : Havada Yakıt İkmali
ICAO : Uluslararası Sivil Havacılık Organizasyonu (International Civil Aviation Organization)
İHA : İnsansız Hava Aracı
IMU : Ataletsel Ölçme Birimi (Inertial Measurement Unit)
IR : Kızılötesi (Infrared)
LD/ST : Veri Erişim Ünitesi (Load/Store Unit)
LIDAR : Lazer ile Tespit ve Mesafe Ölçme Sistemi (Laser Detection & Ranging System)
LiPo : Lityum Polimer Pil (Lithium-Polymer Battery)
MALE : Orta İrtifa ve Uzun Menzil (Medium Altitude, Long Range)
NASA : National Aeronautics and Space Administration
OpenCL : Open Computing Language

P_CCL : Paralel Bağlı Bileşen Etiketleme (Parallel Connected Component Labeling)
PCI : Çevresel Bileşen Bağlantısı (Peripheral Component Interconnect)
PWM : Genlik Modülasyonu (Pulse With Modulation)
RADAR : Radyo Sinyali ile Tespit ve Mesafe Ölçme Sistemi (Radio Detection & Ranging System)
RTH : Eve Dönüş Noktası (Return To Home Point)
RDMA : Uzaktan Doğrudan Bellek Erişimi (Remote Direct Memory Access)
SIMD : Tek Komut Seti Çoklu Veri (Single instruction, multiple data)
SONAR : Ses Sinyali ile Tespit ve Mesafe Ölçme Sistemi (Sound Navigation & Ranging System)
SM : Fermi Mimarisi Akış Çoklu İşlemci (Stream Multi-Processor – Fermi)
SFU : Özel Fonksiyon Yürütme Ünitesi (Special Function Unit)
SMX : Kepler Mimarisi Akış İşlemci (Stream Multi Processor Unit - Kepler)
STL : Standart Şablon Kütüphaneleri (Standard Template Library)
TFR : Yüzey Takip Radarı (Terrain Following Radar)
TICAS : Trafik Çarpışma Önleme Sistemi (Traffic Collision Avoidance System)
UDP : Kullanıcı Veri Birimi Protokolü (User Datagram Protocol)
UVA : Birleştirilmiş Sanal Adresleme (Unified Virtual Addressing)
WSEP : Silah Sistemi Değerlendirme Programı (Weapon System Evaluation Program)
YPA : Yapay Potansiyel Alan
YKİ : Yer Kontrol İstasyonu

SEMBOL LİSTESİ

$\vec{F}_h$: Hedef Noktaya Ait Vektör Alan
F_h : Hedef Noktaya Ait Potansiyel Alan
$\vec{F}_e$: Engel Noktaya Ait Vektör Alan
F_e : Engel Noktaya Ait Potansiyel Alan
U : İki Boyutlu Yol
$\overrightarrow{F_t}$: Toplam Vektör Alan
$V(m, d)$: Vektör Temsili, *m* Vektörün Büyüklüğü (Şiddeti), *d* Vektörün Yönü
F_d : Düzenli Akış Alnının Potansiyel Alanı
$\vec{F}_d$: Düzenli Akışın Vektör Alanı
β : Alanın *x*-ekseni İle Yaptığı Açı
δ : Alan Kuvvet Çarpanı
∇ : Gradient İşlemi
F_p : Palenin Potansiyel Alanı
l : Panelin Uzunluğunu
$(L, -L)$: Panelin Başlangıç Ve Bitiş Noktalarını
xs, ys : Panelin Orjinden Kayış Miktarı
(x_h, y_h) : Hedef Noktanın Konumu
(x, y) : Mevcut İki Boyutlu Konum
(C_x, C_y) : Çeken Noktanın Konumu
(I_x, I_y) : İten Noktanın Konumu
(rp_x, rp_x) : Saat Yönünde Dönen Alan Oluşturan Hareketin Merkezi
(rn_x, rn_y) : Saat Yönünün Tersine Alan Oluşturan Hareketin Merkezi
γ : İki Boyutta Eliptiklik Çarpanı (Küçük Eksenin Büyük Eksene Oranı)
ε : Üç Boyutta Eliptiklik Çarpanı (Dikey Eksenin Yatay Eksene Oranı)
$F_{c_{rot}}$: Çeken Potansiyel Alan
$F_{I_{rot}}$: İten Potansiyel Alan
$F_{rp_{rot}}$: Saat Yönünde Dönen Potansiyel Alan
$F_{rn_{rot}}$: Saat Yönünün Tersine Dönen Potansiyel Alan
$F_{c_{3rot}}$: Üç Boyutlu Çeken Potansiyel Alan
$F_{I_{3rot}}$: Üç Boyutlu İten Potansiyel Alan
DX, DY : Vektör Alan Bileşenleri
DX_3, DY_3, DZ_3 : Üç Boyutlu Vektör Alan Bileşenleri
d : Öklid Uzaklığı
S_{a_b} : *a* Alanının İkili Sınır Fonksiyonu
$Slope$: Sigmoid Eğrinin Kırılım Noktası
ρ : Sigmoid Fonksiyonun Kaplama Alanı
S_{a_s} : *a* Alanının Sigmoid Sınır Fonksiyonu
S_{a3} : Üç Boyutlu a Alanının Sınır Fonksiyonu

SX, SY : Toplam Vektör Alan Bileşenleri
SX_s, SY_s : Sınırlandırılmış Toplam Alan Bileşenleri
R : Hava Aracı Dönüş Yarıçapı
φ : Hava Aracı Yatış Açısı
V : Hava Aracı Sürati
g : Yer Çekimi İvmesi
x_i : *i*'ninci Kartezyen Koordinat
$\emptyset$: Türevlenebilir Sayısal Fonksiyon
$Heading$: Uçuş Başı Tablosu
$Magnitude$: Uçuş Sürati Tablosu
r : Yarıçap
O : Orijin
$\propto$: Potansiyel Alanın Çözünürlük Değeri
Γ : Katman Sayısı
M : Platformun Menzili
A : Potansiyel Alan
Res : Potansiyel Alan Çözünütlüğü
$Size$: Boyutları Birbirine Eşit Alının Bir Eksen Boyutu
$q_u = x_u, y_u$: Yakıt İkmali Yapacak Platfom Konumu
$q_t = x_t, y_t$: Tanker Uçak Konumu
$q_c = x_{tc}, y_{tc}$: Tanker Uçağın Rota Merkezi
ax_{maj}, ax_{min} : Tanker Uçak Rotasının Eliptiklik Faktörleri
ω : Tanker Uçağın Eliptik Rotası Kısa Ekseni ile *y*-Ekseninin Yaptığı Açı
θ : Tanker Uçağın Bulunduğu Noktanın Uçuş Rotası Merkezi ile Yaptığı Açı
h_t : Tanker Uçağın Uçuş Başı
r_u : Tanker İle Alıcı Platform Arasındaki Mesafe
τ : Tankere Yaklaşma Açı Aralığı
τ_s ve τ_f : Tankere Yaklaşma Açı Aralığı Başlangıç Ve Bitiş Açıları
ϵ : Tanker Ve Alıcı Platform Arasındaki Açı
TBİ : Hesaplama İçin İhtiyaç Duyulan Toplam Bellek Miktarı
THS : Hesaplanan Alanda Yer Alan Toplam Hücre Sayısı
SFS : Single Float Veri Tipinin Bellekte Kapladığı Alan
PS : Alanda Yer Alan Platform Sayısı
TS : Bellekte Saklanacak Her Platform İçin Komut Tablo Sayısı
M : Platformun Kütlesi
a : Platformun ivmesi

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ÇOKLU OTONOM İNSANSIZ HAVA ARAÇLARI İÇİN PARALEL PROGRAMLAMA TABANLI YOL PLANLAMASI

ÖZET

İnsansız Hava Aracı (İHA) otonom platformları, insanlı benzer sistemlere nazaran daha etkin, ucuz ve emniyetli şekilde istenilen görev hedeflerini sağlayabilmeleri gibi nedenlerden dolayı, sivil ve askeri birçok alanda yaygın olarak kullanılan teknolojiler haline gelmişlerdir. Karmaşık olarak tanımlanan ve farklı tipte faydalı yük verileri (optik, IR vb.) ile icra edilebilecek görevlerin yerine getirilmesi, büyük ölçekli (taktik gibi) İHA sistemlerinin kullanılmasını gerektirmektedir. Ancak büyük ölçekli tek bir platform kullanılması yerine, birden fazla ve nispeten daha küçük ölçekli İHA sisteminin (mini gibi) bir arada iş birliği içinde formasyon uçuşu dahilinde kullanılması gerek maliyet gerekse de görevin başarı olasılığını arttırması gibi bir takım avantajlar sağlamaktadır.

Grup halinde bir formasyon dahilinde hareket edebilen otonom İHA sistemlerinin kullanımına olan ihtiyaç, görevin icrası sırasında coğrafi engellerden kooperatif olarak kaçınma ve dar alanlarda birbirleri ile çarpışmayı önleme gibi karmaşık problemleri, dolayısıyla kazanılması gereken ilave otonom yetenekleri ortaya çıkarmaktadır. Bu durum İHA sistemleri için dinamik yol planlaması kapsamında karmaşıklık düzeyi yüksek ve de hassas hareket planlaması ihtiyacını karşılayabilecek yol planlaması gereksinimi doğurur. Platformların birbirleri ile hassas bir konumlandırma dahilinde uyumlu hareket etmeleri ve koordinasyon ihtiyacı, gerçek zamanlı bir yol planlaması gereksinimini ortaya koyan bir diğer durumdur. Bu kapsamda gerçek zamanlı olarak çarpışmayı önleyecek ve dinamik engellerden kooperatif biçimde etkin olarak sakınmaya imkan sağlayacak, aynı zamanda belirlenen formasyon şemasını uçuş süresince mümkün olduğunca muhafaza etmeye çalışacak bir yaklaşım için, otonom yol planlaması ihtiyacı oluşmuştur.

Literatürde otonom hareket eden sistemlerin engellerden kaçınma ya da çarpışmayı önleme amaçlı yol planlaması problemlerinde yaygın olarak kullanılan çözüm yöntemlerden birisi yapay potansiyel alan (YPA) tabanlı yaklaşımlardır. Potansiyel alan yaklaşımı, nispeten küçük ölçekli olarak tanımlanmış iki boyutlu alanlar içerisinde ve yavaş hareket eden otonom araçlar için oldukça hızlı hesaplanabilir ve kolay uygulanabilir olması gibi nedenlerden dolayı otonom yol planlaması problemlerine yönelik bir çözüm yöntemi olarak literatürde yerini almıştır. YPA uygulamalarının karekteristik problemleri arasında yer alan yerel minimum problemi gibi sorunların çözümüne yönelik çalışmalar literatürde sıkça görülmektedir. Ender olarak ele alınan bir diğer YPA problemi ise; nispeten geniş alanlarda (özellikle genel yol planlaması ihtiyacı durumunda) ve çok hızlı hareket etme yeteneğine sahip üç boyutlu ortamlar içerisinde yer alan otonom sistemler için YPA'ların dinamik olarak gerçek zamanlı hesaplanmasına yönelik gereksinim duyulan hesaplama gücü ihtiyacının giderilmesi durumudur. YPA içerisindeki çözünürlük derecesi, hareketin hassasiyeti ve yol planlamasının etkinliği açısından önemli bir faktör olup,

çözünürlük arttıkça hareketin hassasiyeti artmakta ve dinamik olarak değişen üç boyutlu ortamlar için hesaplama hızına bağlı ihtiyaç duyulan hesaplama gücü ihtiyacı da fazlalaşmaktadır.

Ayrıca, takım halindeki kooperatif İHA sistemleri için hareket ortamı ve engel modellemesi üç boyutlu olduğundan, karmaşıklık düzeyi ve işlemci gücü ihtiyacı daha önemli bir problem haline gelmektedir. Bunun yanı sıra YPA kapsamında üç boyutlu olarak engellerin modellemesinde faydalanılacak olan sensor verisi hali hazırda İHA platformlarında kullanılan sensor mimarileri tarafından karşılanamamaktadır. Düşünülen yaklaşımın üç boyutlu alanlar içerisinde etkin olarak kullanılabilmesi amacıyla "Çok Katmanlı YPA" yaklaşımı çalışma kapsamında geliştirilen özgün bir çözüm yöntemi olup, İHA sistemlerinde engellerin tespit edilmesi amacıyla kullanılan sensor mimarilerinden üretilen veriler kullanılarak uygulanabilir bir yaklaşım elde edilmiştir.

Formasyon uçuşunun otonom olarak planlanması amacıyla farklı karakterlerdeki (iten, çeken, dönen vb.) temel potansiyel alan yapılarının bir arada kullanılmasına gerek duyulmuştur. Dinamik olarak değişen ve sürekli karakterini güncelleyen alan içerisindeki hareketin etkin olarak hesaplanabilmesi, gerçek zamanda uygulanması zor bir problemdir. Bunun ötesinde ortaya konulan çözümün uygulanabilir bir yaklaşım olması da etkin bir çözüm için gereklidir. Ortaya konulan otonom formasyon uçuşu amaçlı yol planlaması yaklaşımı, farklı karakterlerdeki İHA sistemleri tarafından uygulanabilir çözümler ortaya koymalıdır. Farklı özelliklerdeki temel potansiyel alanları bir araya getiren otonom yol planlaması yaklaşımında, alanlar arasındaki geçişler çalışma kapsamında öncelikle ikili değer üreten (binary) fonksiyonlar ile sağlanmış ancak bu durum alanlar arasında keskin dönüş karakteristiklerine ihtiyaç duyduğundan dolayı farklı tipte İHA sistemleri tarafından uygulanması güç paternler üretecek sonuçlar üretmiştir. Bu nedenle otonom yol planlaması modeli kapsamında ele alınan farklı potansiyel alanların sınırlarının belirlenmesi amacıyla ikili sınır fonksiyonları yerine sigmoid fonksiyonlardan faydalanılmış, böylece alan geçişleri esnasında platformun daha etkin ve uygulanabilir manevralar icra etmesini sağlayabilecek bir yaklaşım modellemesi gerçekleştirilmiştir. Her iki yöntemin başarısı simülasyon ortamında ölçülmüş ve etkinlik, performans açısından karşılaştırılarak sonuçlar üretilmiştir.

Üç boyutlu, geniş, yüksek çözünürlüklü ve hızlı olarak değişen dinamik potansiyel alanların hesaplanmasında; hızlı, geniş bellekli ve nispeten yüksek maliyetli merkezi işlemcilere dayalı yaklaşımlar problemin çözümü için yeterli olmamaktadır. Bunun yerine dağıtılmış yapıda çalışabilen esnek ve paralel hesaplama yapılarının kullanılmasının daha etkin sonuçlar üreteceği değerlendirilmiştir. Gerek maliyetleri, gerekse ortaya koydukları hesaplama kapasiteleri açısından son yıllarda yaygın olarak kullanılan paralel hesaplama amaçlı yapılardan birisi grafik işlemci tabanlı çok çekirdekli paralel programlama araçlarıdır. Son yıllarda grafik işlemcilerin paralel ve genel maksatlı olarak programlanabilmeleri sayesinde, öbek yapılı süper bilgisayar altyapılarının çok daha ucuz maliyetler ile gerçekleştirilmeleri sağlanmıştır. Karmaşık ve yüksek hesaplama gücü ihtiyacı duyan, yüksek çözünürlüklü dinamik olarak tanımlanmış YPA çözümleri gibi problemlerin, paralel olarak çözülmesi amacıyla grafik işlemcilerin genel maksatlı paralel programlanması (GPGPU) tabanlı yapılar gerek fiziksel özellikleri gerekse de etkin hesaplama yetenekleri nedeniyle İHA sistemlerinde kullanılabilir ideal platformlar olarak değerlendirilmektedir.

Çalışma ile otonom ve hızlı hareket kabiliyetine sahip, ancak kısıtlı manevra yeteneği olan, formasyon düzenindeki kooperatif İHA sistemleri için YPA yaklaşımının eksik kaldığı, dinamik olarak değişen ve önceden özellikleri bilinmeyen üç boyutlu alanlar içerisinde farklı otonom davranışların sergilenmesi esnasında vektörel alanların tekrar hesaplanmasının getirdiği yüksek hesaplama gücü ihtiyacını karşılayacak nitelikte GPGPU tabanlı paralel ve dağıtılmış bir çözüm yöntemi geliştirmek amaçlanmıştır. Çalışma kapsamında geliştirilen GPGPU tabanlı paralel çekirdek kodu sistem kaynaklarını minimum tüketirken, aynı zamanda sistemin maksimum hız ve verim ile çalışmasını sağlayacak gerekli optimizasyonları da içermektedir. Ayrıca, ortaya konulan yaklaşımın hali hazırda kullanılmakta olan İHA sistemleri üzerinde uygulanabilecek şekilde bir mobil donanım desteği vasıtasıyla uygulaması da sınanmıştır. Sonuç olarak, paralel programlama teknikleri uygulanarak problemin çözümünün mobil sistemler için ideal programlama ortamı sağlayan grafik işlemci (GPU) tabanlı yaklaşımlar ile hesaplanması amaçlanmaktadır. Bu konuda literatürde yeterli sayıda araştırmanın yer almadığı gözlemlenmiş ve literatüre katkı sağlanmıştır.

Ortaya konulan YPA tabanlı otonom yol planlaması çözüm yaklaşımının, gerçek zamanlı olarak sonuç üretebilmesine imkân sağlayacak nitelikte GPGPU tabanlı tek komut seti çolu veri (Single Instruction, Multiple Data - SIMD) tipinde paralel algoritmalar geliştirilerek uygulanması amaçlanmıştır. Otonom paralel formasyon düzenin sağlanması için geliştirilen yaklaşımın dinamik ve önceden konum ve davranışları bilinmeyen engellerin yer aldığı bir ortamda başarısının ölçülerek problemin karmaşıklık düzeyinin arttırılmasına yönelik çalışmalar gerçekleştirilmiştir. Önceden özellikleri bilinmeyen alan içerisinde yer alan engellerin algılanmasını modelleyen bir yaklaşım geliştirilmiştir.

Gerçekleştirilen yol planlama algoritmalarının faklı mobil ve sabit grafik işlemciler üzerinde başarısının ölçülerek algoritmaların donanımlara göre optimize edilmesi çalışmaları gerçekleştirilmiştir. Bu kapsamda mobil ve sabit grafik işlemciler üzerindeki yaklaşımın performansı, geleneksel işlemciler ile gerçekleştirilen seri hesaplama performansı ile karşılaştırmalı olarak ortaya konulmuştur. Paralel hesaplama performansı farklı türde grafik işlemciler üzerinde mobil (Tesla Quadro 1000M, NVIDIA GeForce GTX 670MX) ve iş istasyonu/masaüstü (Tesla K20c, GeForce GTX480) sınanarak performans değerleri üretilmiştir. Elde edilen sistemin hesaplama performansı, klasik olarak programlanmış sıralı hesaplama yaklaşımları ile karşılaştırılmış ve bir takım paralel hesaplamaya yönelik iyileştirmelerin uygulanması ile birlikte K20c grafik işlemcisi ile yaklaşık 17 kat seri hesaplama performansından daha iyi bir hesaplama performansı elde edilmiştir. Ayrıca birden fazla grafik işlemci bir arada kullanılarak, aynı ana makine üzerindeki grafik işlemcilerden faydalanılarak yaklaşık 41 kat performans elde edilmiştir. Ağ ortamı üzerinde yer alan çok sayıda grafik işlemciden bir arada YPA tabanlı hesaplama yapma olanağı sağlayacak yaklaşım ortaya konulmuş ve haberleşme maliyetleri göz ardı edildiğinde yaklaşık 48 kat seri hesaplamaya nazaran performans artışı elde edilmiştir.

İHA sistemleri üzerine geliştirilen akademik çalışmaların yoğunluğu, birçok otonom sistem için uygulama geliştirme amaçlı tekniğin, aracın ve yöntemin literatürde kolay ulaşılabilir hale gelmesini sağlamıştır. Yaklaşımın başarısı noktasal yüklü parçacığın hareket modeli tabanlı simülasyonlar ile sınanmıştır. Çok sayıda otonom döner kanatlı quad-rotor platformun otonom şekilde kontrolünün aynı anda sağlanmasının gerçekleştirilmesi ve YPA tabanlı otonom paralel formasyon yaklaşım teorisinin

uygulamalı olarak ortaya konulması kapsamında otonom uçuş kontrol sistemi yapısı simülasyon ve gerçek uygulama olarak ortaya konulmuştur. Çalışma kapsamında elde edilen sonuçlar, simülasyon ortamında sınandıktan sonra AR Drone quad-rotor platform sistemleri üzerinde uygulanarak gerçek sistemlerdeki başarımları ölçülmüş ve uygulamadaki problemleri ortaya konularak çözüm geliştirilmiştir.

Bu kapsamda tez çalışması ile geliştirilecek olan teorik çalışmaların başarısının uygulamalı olarak ölçülebilmesine imkân tanıyacak şekilde AR-Drone quad-rotor platformların otonom şekilde hareket etmelerine ve kontrol edilebilmelerine imkân tanıyacak şekilde gömülü yazılımlarında bir takım düzenlemelere gidilmiştir. Hareket kontrolünü düzenleyen uçuş bilgisayarı yapısı YPA ile desteklenmiş bir yol planlamasına destek verebilecek şekilde güncellenmiştir. Ayrıca platformun GPS desteği ile dış ortamlarda otonom uçuşunun sağlanabilmesi amacıyla Ardu Mega Pilot kontrol donanımı (APM) ile küresel konumlandırma sistemi (Global Positioning System - GPS) alıcı anteni ve XBee iletişim modülünün entegrasyonu sağlanmıştır. Böylece APM modülü üzerine bağlı olan GPS anteninin edindiği konum bilgilerinin Xbee kablosuz iletişim modülü üzerinden kontrol yazılımının koşturulduğu bilgisayar sistemine aktarımı sağlanmıştır. Farklı senaryolar dahilinde uçuş denemeleri gerçekleştirilerek yaklaşımın başarısı uygulamalı olarak ortaya konulmuştur.

Sonuç olarak, çalışma ile otonom uçuş gerçekleştirmeyi amaçlayan çok sayıdaki platformun bir arada formasyon teşkil edecek, birbirleri ve ortam dahilinde yer alan engeller ile çarpışmalarının önlendiği güvenli bir seyrüsefer gerçekleştirmelerine imkan sağlayacak yol planlaması yaklaşımı, grafik işlemciler üzerinde koşturulabilen paralel algoritmalar ortaya konularak gerçeğe en yakın zamanlı sonuçlar üretecek şekilde simülasyon ve uygulamalar ile sınanarak başarıyla özgün şekilde geliştirilmiştir.

PARALLEL PROGRAMMING BASED PATH PLANNING FOR MULTI AUTONOMOUS UNMMANED VEHICLES

SUMMARY

The autonomous Unmanned Aerial Vehicles (UAVs) are more efficient than similar manned platforms for the reasons such as to be able to provide cheap and safe solutions, so that technology has become widely used in many areas of civil and military. The missions which are defined as complex and requires different types of payload data together (like optical, IR, etc.), needs to usage of large scale of UAV platforms to fulfillment of the tasks. However using multiple and relatively small-scale UAV platforms (such as mini class) as a combination of cooperation in a formation flight instead of using a large scale single platform (such as tactical class) provides a number of advantages such as reducing operation costs and increasing probability of success of the mission.

The need for the use of autonomous UAV systems that can move within a formation in groups, brings out some complex problems like avoiding of geographical barriers in cooperative manner and prevention of collision with each other in narrow areas, furthermore needs additional autonomous capabilities. In this case, it also raises path planning requirements to meet the needs precise motion planning and a high level of complexity within the scope of dynamic path planning for UAV systems. Another case demonstrating the need for a real-time path planning is moving the platforms within a precise positioning and co-ordination with the others in the formation. In this context, autonomous path planning requirements has occurred because of the needs like to avoid collisions in real time and to avoid from the dynamic obstacles as in cooperative manner, but also to determine the formation scheme and to try to maintain it as much as possible during the flight. APF based path planning approaches are one of the widely used methods in the literature to avoid collision with other platforms and to generate obstacles free patterns. Potential field approach has taken its place in literature because of their easily applicable and computable features especially for autonomous platfroms in the small scale and two-dimensional space.

Efforts for generation of the solution for the APF based path planning approaches characteristic problems like local minimum problem are seen frequently in the literature. But another problem of APF based path planning (especially in the case of general path planning needs) is satisfaying the real-time computaiton needs especially for large size dynamic and three-dimensional fields. Resolution degree in the APF is and important factor by the means of movement precision and effectiveness in the path planning, it increases the sensivitiy of the movement but it affects the computation speed in negative way and it needs more power especially for three-dimensionan dynamic envoriments.

Furthermore the needs of computation power becomes important isuuse for the co-operated uav platoforms in a formation because they are in an envirement that

requires three dimensional modelling of obstacles and path planning. In addition, the sensors data which are used for modelling of the obstacles in three-dimensional have not been provided by the sensors on the UAV platforms yet. The sensors have not enough technical specs to produce three dimensional sensor data yet. To model the obstacles in three dimensional with current sensors that are used on UAV platforms, "multi layer APF" approach is designed and serve a unique solution in this work.

The basic strucutures of the potential field based path planning (like pushing, pulling, rotating, etc.) are used together in a combination to satisfy the needs of autonomous formation flight path planning. The computation of the movement is a difficult problem in real-time implementation especially for the dynamically changing enviroments that is updating its characteristics by the movements of the platforms in it. Moreover the real-time path planning solution must be feasible, effective and applicable by the platforms. The autonomous formation flight path planning approach must demonstrate practical solutions for the UAV systems in different characteristics. Firstly the basic structures of the potential fields are combined by using binary functions to achieve fast computation results, but these solution generates the paths which are hard to implement especially for different UAVs due to the requirements of the sharp turning manevuars, etc. Therefore, sigmoid functions are used instead of binary functions to limit the basic sub-potential fields while combining them to generated suitable path planning structures. With using sigmoid functions to limit the basic potential fields, more smooth patterns are generated that are suitable for most UAV flight charecteristics. The success and effectiveness of both of these methods have been measured in the simulation environment by comparing the results in terms of performance.

In the computation of the three-dimensional, large and rapidly dynamic potential field with high resolution value, high speed large memory based and reletively high cost central processors are not sufficient for solving the real time computation problem. Instead of central processors, the usage of flexible and distributed parallel computing structures are suggested to produce more effective computation performance results. GPU-based multi-core parallel programming tools has become one the widely used parallel computing structures in recent years due to both their low costs as well as demonstrating their parallel computing capacity. In recent years, thanks to parallel and general purpose programming ability on graphic processors, cluster based supercomputer infrastructures are provided with a much cheaper costs. Problems that needs high performance computing power like high resolution, dynamic and complex APF solutions, become able to solve by using parallel GPGPU based structures which are suitable computation frames for UAVs due to especilally their physical properties and efficent calculation capabilities.

The aim of this work meets the high speed re-computation requirements of the three-dimensional, large and rapidly dynamic potential field with high resolution value for the cooperative UAVs in a formation scheme with limited manevuar capabilities and high speed by benefiting from GPGPU based parallel and distributed computation approach. GPGPU based parallel kernel algorithm that is developed in this work, while consuming minimal system resources, but also includes optimizations necessary to make it work with maximum speed and efficiency for computation performance. In addition, the approach is tested on mobile hardwares which are accepted as suitable and applicable for the current UAV platforms today. As a result, solution of the problem is applied by using parallel programming techniques on the GPUs that are accepted as ideal mobile and parallel computing enviroments. It is

observed that there is not sufficient number of work in the literature about this subject and it is intend to provide support on literature with this work.

The proposed APF-based autonomous path planning approach are computed by developing and implementing SIMD type and GPGPU based parallel algorithms to provide real time solutions. The approach developed for establishing the autonomous formation, is tested under more complex conditions by applying in dynamic enviroments including unknown obstacles before flight. An approach is also developed to model for detection of the obstacles in unknown enviroment.

The performed path planning algorithms performances are measured on different mobile and on-board graphic processors by applying required optimizations that are specific for GPU processors hardware architectures. In this context, the performance of the approach on the on-board and mobile graphics processors, has tried to put forward in comparison with the serial computing performance achieved by conventional processors. Parallel computing performance values has been generated on mobile (Tesla Quadro 1000M NVIDIA GeForce GTX 670MX) and workstation / desktop (K20c Tesla, GeForce GTX480) graphics processors. The computing performance of the resulting system on K20c graphic processors achieved 17 times better computation performance results against sequentially programmed approach on classical CPUs computation performance. Moreover, by using a combination of multiple graphics processors on the same host machine approximately 41 times better computation performance has been achieved. Also, by using multiple graphic processors on a network with multiple hosts, approximately 48 times better computation performance has been achieved against sequentially programmed approach on classical CPUs computation performance by ignoring communication costs on network.

With the high intensity of the academic researches about autonomous systems like UAVs, the technique for application development for many autonomous systems, tool and methods become readily available in the literature. Firstly, the success of the approach is tested with the point mass particle motion model based simulations. Also, autonomous flight control system architecture has been putted forth with simulations and real time applications in the manner of controlling multiple quad-rotor platforms together. The obtained results is evaluated in the simulation studies and then the success of the approach has been evaluated in real time applications by using AR-Drone quadrotor platforms by this work.

Theoretical results obtained in this study has been evaluated with real-time applications on AR Drone platforms which become suitable by applying a number of regulations on embedded control software of the drones and developing control software that is allowing control of the multiple platforms together. Motion control of the governing structure of the flight computer has been updated so that it can support a path planning supported by the APF. Also to achieve the location of the platform on the outdoor environment a GPS antenna is integrated on the platform by using Ardu Pilot Mega (APM) flight computer and XBee communication equipment. Thus, the location information of the platform obtained by GPS antenna that is connected to the APM module is transferred via Xbee wireless communication module to the computer system that runs the control software. By implementing flight tests for different scenarios, the success of the approach has been putted forth experimentally.

As a result, autonomous path planning needs are satisfied for multiple platforms that aim autonomous flight on a defined formation scheme with safe navigation while prevention of collision with obstacles and other mobile platforms or each other. The path planning approach is successfully implemented as a real time original solution by using parallel suitable algorithms for graphic processors. Also the success of the genuine approach is tested with simulation and practical experiments.

1. GİRİŞ

Bu bölümde; çalışmaya duyulan ihtiyaç, ortaya konulan problem tanımı, çalışmanın amacı, uygulanan çözüm yaklaşımları ve araştırma kısıtları ortaya konulmuştur. Çalışma neticesinde elde edilen sonuçlar özetlenmiş, bu kapsamda çalışmanın uygulanabileceği alanlara değinilmiştir.

1.1 Motivasyon

Günümüzde İnsansız Hava Aracı (İHA) sistemleri, sivil ve askeri birçok uygulama alanında yaygın olarak kullanılmakta olan ve üzerinde çok sayıda bilimsel araştırmanın yürütüldüğü popüler araştırma konularından birisi haline gelmiştir. Özellikle otonom hareket eden mobil sistemler üzerine gerçekleştirilen araştırma çalışmalarının birçoğu uygulama alanı olarak İHA platformlarını seçmektedir [1][2].

Ticari veya askeri yeni nesil insanlı hava araçlarında birçok otonom yetenek kazandırılmış olup, bunların sayısı günden güne artmaktadır [3][4]. Uçuşun en riskli ve en karmaşık olarak tanımlanabilecek iniş, kalkış, seyrüsefer gibi bölümleri artık çoğunlukla insan kontrolü altında otonom olarak gerçekleştirilmektedir [5][6]. Otonom yeteneklerin geliştirilmesi ile özellikle yaşanan kaza/kırım olaylarındaki insan faktöründen kaynaklanan olayların azaltıldığı değerlendirilmektedir.

İHA platformları giderek insanlı hava aracı platformlarının yerini almaktadır [7]. Bu durum yakın gelecekte insanlı hava araçlarının halihazırda gerçekleştirebildiği formasyon veya yakın kol uçuşu, havada yakıt ikmali gibi en karmaşık görevlerin insan hayatını riske atmadan insansız platformlar ile gerçekleştirileceğinin en büyük göstergesidir.

Formasyon oluşturacak şekilde iş birliği halinde hareket edebilen otonom İHA sistemlerinin bir arada kullanımına ihtiyaç duyulmaktadır [8]. Tek bir platform için gerçekleştirmesi güç ve riskli olarak tanımlanan görevler formasyon uçuşu ile daha emniyetli ve kesin olarak gerçekleştirilebilir. Ayıca tek platform kullanıldığı takdirde nispeten daha büyük ölçekli hava araçlarına ihtiyaç duyan, farklı tipte faydalı yük

verilerine aynı anda gereksinim duyan görevlerin, maliyet etkin, hızlı ve görev başarısı açısından emniyetli şekilde yerine getirilmesi sağlanabilir. Bu tip görevlerin icrası, coğrafi engellerden kooperatif olarak kaçınma ve dar alanlarda birbirleri ile çarpışmayı önleme gibi ilave problemleri ortaya çıkarmaktadır. Bu problemlere çözüm oluşturabilecek gerçek zamanlı otonom yeteneklerin formasyon dahilinde giderilmesi ihtiyacı ortaya çıkmaktadır.

1.2 Problemin Tanımı

Bazı durumlarda otonom platform, çevresi hakkında önceden tam bir bilgi sahibidir ve buna dayalı olarak hareketini statik şekilde görev öncesinde planlar ve icra eder [9]. Ancak bazı durumlarda ise sadece hedef noktaya ait konum bilgisine sahiptir ve çevresi hakkında bilgi toplamak için üzerindeki algılayıcılardan faydalanarak yol planlamasını gerçek zamanlı olarak dinamik şekilde gerçekleştirir [10]. Yol planlaması yerel ve genel yol planlaması olarak iki farklı tanıma ayrılabilmekte ve her iki yaklaşımda da engellerden sakınarak varılmak istenen hedef noktaya giden en uygun yolu bulmak amaçlanmaktadır [11].

Vektör tabanlı yol planlaması yaklaşımları, sürekli bir hareket modellemesi izlemeleri ve matematik yoğunluğundan çok benzetim tabanlı modeller olmalarından dolayı, literatürde İHA sistemleri gibi hızlı, sınırlı ve agresif hareket edebilen yapılar için uygun yöntemler olarak değerlendirilmektedir. Vektör tabanlı yaklaşımlara, yaygın şekilde kullanılan Yapay Potansiyel Alanlar (YPA) yöntemi örnek olarak gösterilebilir ve sadece engellerden kaçınarak ulaşılmak istenen hedef noktaya yönelik yol planlaması amacıyla değil, aynı zamanda çarpışmayı önlemek maksadıyla veya istenen rotadan ayrılmadan hareket edilmesi amacıyla kullanılabilen yaklaşımlardır [12]. Bu nedenlerden dolayı YPA tabanlı yol planlaması yaklaşımı çoklu otonom hareket eden sistemlerin yol planlaması için uygun çözümlerden birisi olarak değerlendirilmektedir.

Ancak YPA tabanlı otonom yol planlamasının avantajlarının yanında yerel minimum, robot tuzağı, salınım gibi temel problemleri oluşmaktadır. Literatürde bu problemlerin önüne geçmek için genel yol planlaması yaklaşımı, farklı fonksiyon tipleri (harmonik fonksiyonlar gibi) ile modelleme gibi farklı çözüm önerileri mevcut olup, bu yaklaşımların her biri hesaplama maliyetine ilave süre eklemekte ve

dolayısıyla çözümün gerçek zamanlı olarak farklı mimarideki platformlar için uygulanabilir olmasından uzaklaşılmaktadır [1].

Özellikle üç boyutlu geniş ölçekli önceden davranışları kestirilemeyen dinamik alanlar içinde yer alan otonom sistemlerin ızgara tabanlı YPA yaklaşımı ile yol planlaması hesaplama süresi ihtiyacı, oldukça maliyetli bir süreçtir. Izgara çözünürlüğü performansa ve çözümün kalitesine doğrudan etki eden, optimize edilmesi gereken bir özelliktir. Dinamik olarak değişkenlik gösteren veya engel özellikleri bilinmeyen alanlar içerisinde gerçekleştirilen seyrüsefer esnasında gerçek zamanlı olarak tespit edilen engeller olması halinde, çözüm için alanın YPA tabanlı bir yaklaşım ile belirli bir sıklıkta defaten modellenmesi gerekmekte ve bu da hesaplama maliyetini oldukça arttırmaktadır. Yerel minimum noktalarının alan içerisinde kendiliğinden oluşması olasıdır. Modelleme neticesinde ortaya konulan alanın araç dinamikleri ile uyumlu olması ve buna bağlı olarak maliyetli bir takım işlemlerin gerçekleştirilmesi ihtiyacı vardır. Ayrıca araç salınım, çıkmaz yol ya da sonsuz yol problemi ile karşı karşıya kalabilir ve hedefe ulaşacağı garanti edilemeyebilir. YPA yaklaşımının getirdiği problemlere yönelik olarak literatürde farklı çözüm yöntemlerine rastlanmaktadır. Bu yöntemler tek başına problemlerin tamamını çözememektedir. Ancak literatür incelendiğinde Harmonik fonksiyonlar ile modellenen YPA yaklaşımının fark et-sakın vb. teknikler ile desteklendiğinde problemlerin birçoğuna çözüm geliştirilebileceği görülmektedir [13]. Geliştirilen çözüm yaklaşımları gerek hesaplama süre ihtiyacı gerekse de sistem bellek kullanımı açısından ilave maliyetler getirmekte ve problemin gerçek zamanlı olarak büyük ölçekli dinamik ortamlar için çözümünü zorlaştırmaktadır.

Bu tez çalışmasında ele alınmak istenen konuların başında YPA ile modellenmek istenen geniş ölçekli üç boyutlu alanların, dinamik çevre koşullarının ihtiyacına istinaden, sıralanan problemlerden etkilenmeyecek şekilde, gerçek zamanlı platform ihtiyaçlarına yanıt verebilecek şekilde hesaplanabilirliğini göstermektir. Yerel ya da genel yol planlaması çözümü için YPA tabanlı yol planlaması modellemesinde kullanılması değerlendirilen harmonik fonksiyonlar ile hücrenin potansiyel değeri, diğer hücrelerden ve önceki işlemlerden bağımsız olarak ve sadece pozisyon – seviye bilgisini gerektirerek hesaplanabilir olduğu şeklinde değerlendirilmektedir [14]. Ayrıca üç boyutlu YPA içerisinde yer alan her bir ızgara hücresinin potansiyel değerinin hesaplanmasında kullanılan aritmetik fonksiyon aynıdır. Farklılık gösteren,

değeri hesaplanan hücrenin pozisyonu, seviyesi ve dolayısıyla engellere ve hedefe olan uzaklığı gibi verileridir. Bu durumda SIMD tipindeki programlama sınıfına uygun olarak, sürecin birden fazla sayıda işlemci üzerine yayılarak paralel şekilde gerçekleştirilebilir olduğu değerlendirilmektedir. Çalışma kapsamında, SIMD yapısına uygun olduğu değerlendirilen YPA hesaplaması sürecinin paralel bir algoritma tasarımı geliştirilerek çeşitli sıralı ve paralel donanım mimarileri üzerindeki başarısı karşılaştırmalı olarak ölçülmesi amaçlanmıştır.

Ayrıca elde edilen sonuçların, karmaşıklık düzeyi diğer benzer uygulamalara göre nispeten yüksek olduğuna kanaat getirilen, İHA platformlarından oluşan bir formasyonun yol planlaması uygulaması üzerinde simülasyon ortamında ve uygulamalı olarak başarısının sınaması amaçlamaktadır. Formasyon, aynı zamanda kooperatif biçimde engellerden sakınan ve çarpışmayı engelleyici özellikleri bünyesinde barındıran özellikleri ihtiva edecektir. Platformların seyrüseferi süresince formasyon şemasının mümkün olduğunca koruması hedeflenmektedir.

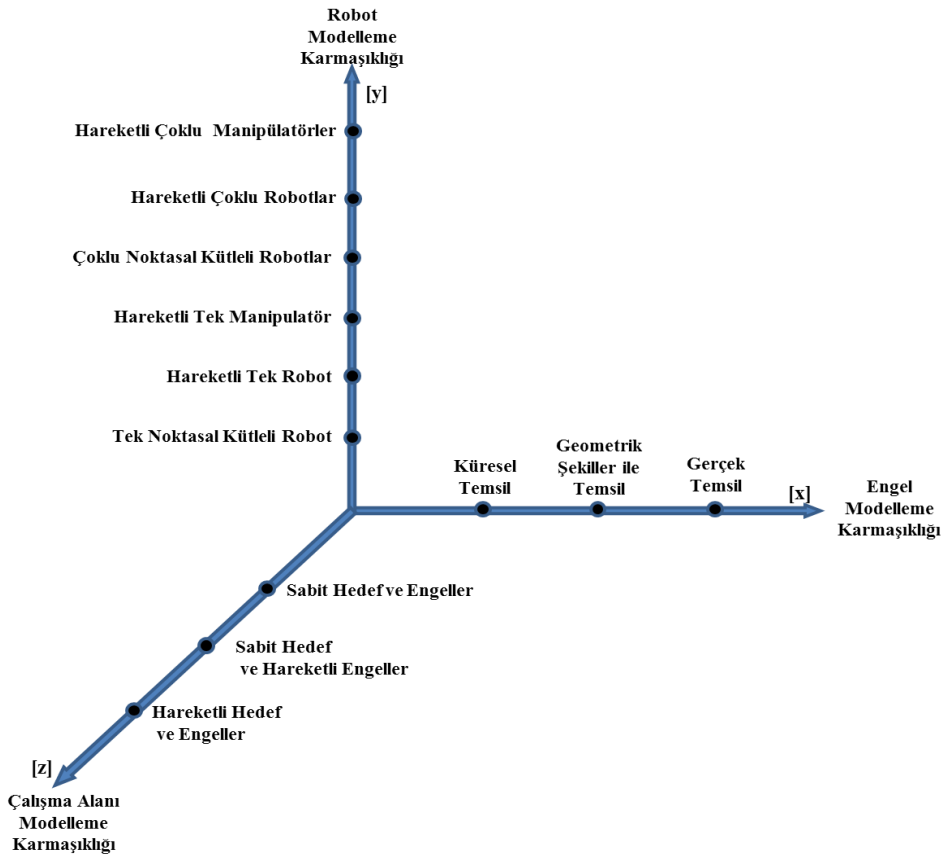
1.3 Çalışmanın Amacı

İHA platformlarının formasyon uçuşu amacıyla ortaya konulmak istenen otonom yol planlaması yaklaşımı; platformların birbirlerine göre konumlarını bir formasyon şeması dahilinde muhafaza etmesi esnasında önceden belirlenen seyrüsefer görevinin icra edilmesi, bu esnada karşılaşılacak önceden bilinmeyen engellerden sakınma ve platformların birbirleri ile dar alanlar içinde çarpışmasını engelleme özelliklerini sağlayabilmesi şeklinde özetlenebilir. Bu bölümde çalışmanın problem tanımı bölümünde ortaya konulan ve çalışma kapsamında ele alınması planlanan problemlerin çözümü olarak nelerin hedeflendiği tek tek başlıklar altında açıklanmıştır.

1.3.1 Formasyon amaçlı otonom yol planlaması

Formasyon uçuşu havada yakıt ikmali, yakın kol formasyonun muhafaza edilmesi gibi farklı insanlı/insansız uçuş görevlerinde kullanılan bir yaklaşımdır. Çok sayıda hızla değişen ve önceden tahmin edilemeyen girdi değerine bağlı dinamik bir modelleme yaklaşımdır. Özellikle formasyon amaçlı otonom yol planlaması yaklaşımının, gerçek zamanlı olarak çözüm sunabilecek yetenekte ve formasyon dahilinde yer alan farklı özelliklerdeki (heterojen) platformların farklı uçuş

yeteneklerini destekleyebilecek genel bir yaklaşım olması aranan özellikler olarak ön plana çıkmaktadır.



Şekil 1.1: Hareket planlama probleminde hiyerarşik zorluk grafiği [15].

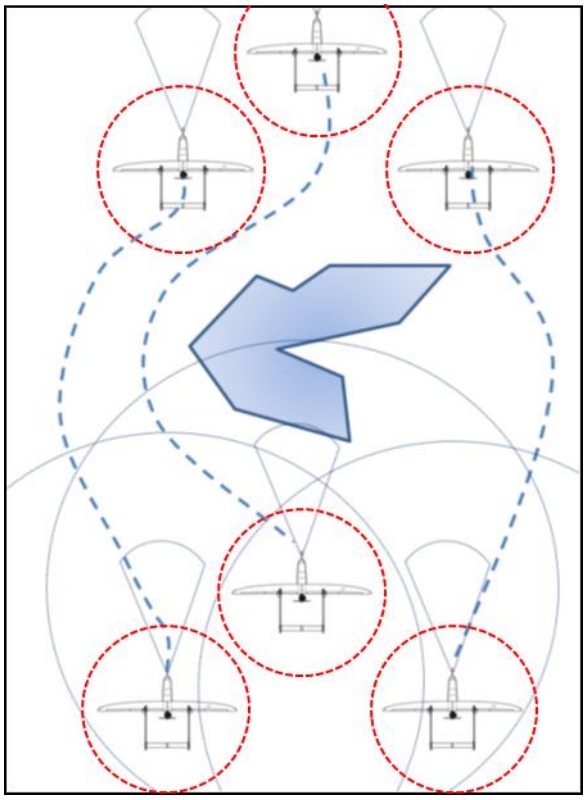
Şekil 1.1'de insansız araçlar için hareket planlama yönteminde kullanılan hiyerarşik zorluk grafiği gösterilmiştir. Bu grafik üzerinde *x* ekseni engel modelleme eksenidir. En basit durum, gerçek dünya engellerini yakalamaya çalışırken modelleme karmaşıklığının artmasıyla birlikte; tüm engellerin şekil içerisinde dairesel olarak modellendiği küresel temsil modelidir. Benzer olarak *y* ekseni robot modelleme eksenidir. Robot modelleme teknikleri robotun modellenmesine göre kabaca şu şekilde sınıflandırılabilir; noktasal yüklü tek robot, tekerlekli hareketli tek araç, tekerlekli hareketli tek manipülatör, noktasal yüklü çoklu robotlar, tekerlekli hareketli çoklu robotlar ve tekerlekli hareketli çoklu manipülatörler. *z* ekseni çalışma uzayı modelleme eksenidir. En basit çalışma uzayları sadece sabit engeller ve hedeflerin üstesinden gelirken, dinamik engeller ve dinamik hedeflerle oluşan çalışma uzayları hareket planlama için daha büyük zorluklara çözüm üretirler [15].

Bu tez çalışmasında, engel modelleme ekseninde geometrik şekiller ile temsil, robot modelleme ekseninde çoklu noktasal kütleli robotlar ve çalışma uzayı modelleme ekseninde ise hareketli hedef ve engeller üzerinde çalışılmıştır. Bu kapsamda Şekil 1.1'de yer alan hiyerarşik zorluk grafiğinde problemin yeri nispeten üst seviyelerde yer almaktadır.

Çalışma kapsamında otonom yol planlaması açısından amaçlanan; üç boyutlu ve önceden özellikleri (engel konumları) bilinmeyen bir alan içerisinde yer alan birden fazla yüksek hızlı ve kısıtlı manevra kabiliyetine sahip otonom platformun (İHA gibi) bir arada kooperatif şekilde hareket etmelerine olanak sağlayacak (formasyon uçuşu gibi) gerçek zamanlı yol planlaması yaklaşımının ortaya konulması ve uygulama ile başarısının sınanmasıdır.

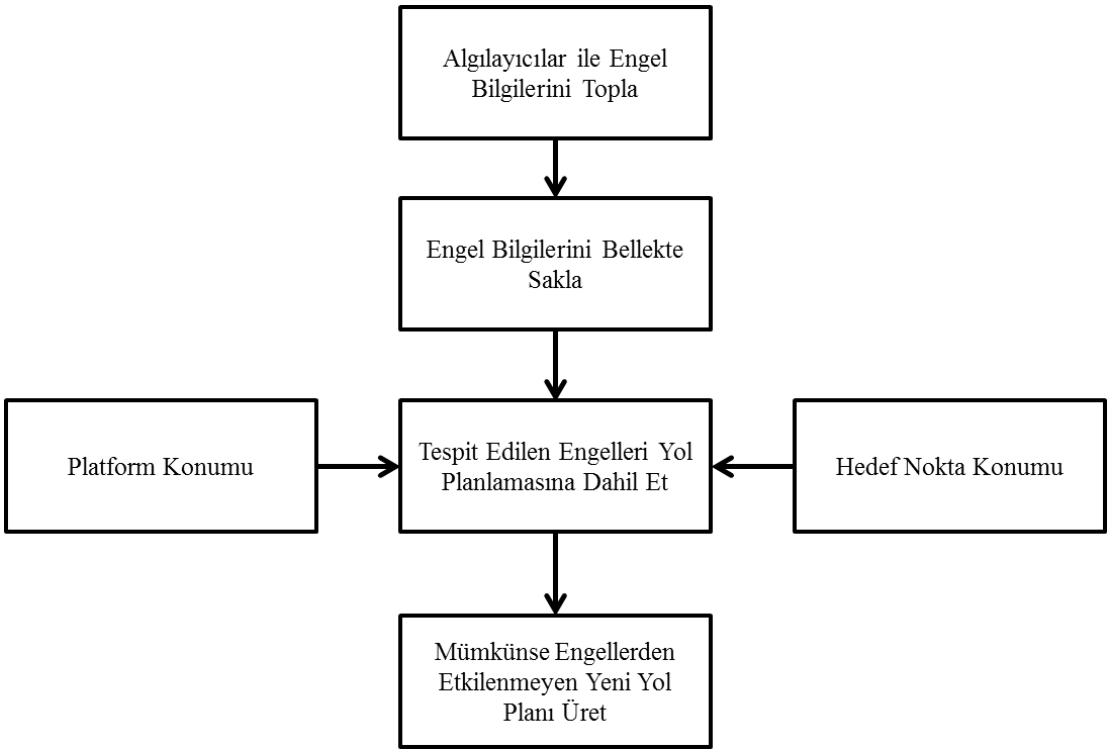
1.3.2 Engellerden sakınma

Çalışma kapsamında amaçlanan hedeflerden birisi otonom yol planlamasının durağan ve hareketli engellerden sakınma ihtiyacına cevap verebilecek nitelikte olmasıdır.



Şekil 1.2: Durağan engellerden formasyonu muhafaza ederek sakınma.

Durağan engeller, Şekil 1.2'de gösterilen biçimde, formasyon dahilindeki platformların icra ettikleri seyrüsefer esnasında gerek önceden bilinen gerekse de uçuş esnasında platformların üzerlerinde yer alan algılayıcılar ile tespit edilmiş olan doğal dağ, tepe gibi veya uçuş rotası üzerinde bulunan bina gibi engelleri temsil etmektedir. Hareketli engeller ise formasyon dahilinde bulunan platformlar haricindeki uçuş rotasında yer alan ve hareketli durumdaki diğer hava araçları gibi cisimleri temsil etmektedir.



Şekil 1.3: Engellerden sakınma sistemi operasyonel akış şeması [16].

Şekil 1.3 ile önceden coğrafi özellikleri bilinmeyen bir alan içerisinde otonom bir sistem tarafından icra edilen seyrüsefer esnasında platform üzerindeki algılayıcılar ile çevrenin tanınması ile gerçekleştirilen bir yol planlamasının ana unsurları görülmektedir [16]. Yaklaşım incelendiğinde görevine devam eden platformun üzerinde yer alan algılayıcılar ile (bunlar farklı teknolojide ve mimaride algılayıcılar olabileceği gibi süreç bundan bağımsızdır) çevresinde yer alan engelleri tespit ettiği görülmektedir. Bu bilgi görev süresince saklanmaktadır. Şayet tespit edilen engellerden bir ya da bir kaçı hedefe ulaşmak açısından bir engel teşkil ediyor ise yol planlamasının güncellenmektedir.

Bu tez çalışması kapsamında ortaya konulmak istenen otonom yol planlaması yaklaşımı önceki madde kapsamında açıklandığı şekilde platformların oluşturduğu

formasyonu muhafaza etmenin yanında seyrüsefer esnasında karşılaşılan ve önceden bilinen veya uçuş esnasında tespit edilen durağan ve hareketli engellerden sakınmayı amaçlamaktadır. Bu nedenle gerçekleştirilmesi planlanan yol planlaması bu desteği sağlar nitelikte olacaktır.

1.3.3 Çarpışmayı önleme

Tez ile ortaya konulmak istenen bir diğer hedef ise, formasyonu teşkil eden platformların bir engelden sakınmak için yaptıkları manevraya istinaden oluşabilecek çarpışma riskini ortadan kaldırmaktır. Ayrıca rüzgar gibi dış etkenlerden etkilenerek formasyon içindeki konumlarından sapmalarına neden olabilecek bir davranışta bulunmaları halinde çarpışmayı önleyici doğal tepkiler üretilebilmelidir. Hatta formasyondaki konumlarını alabilmek için birbirlerine yaklaştıklarında birbirlerine çarpmalarını engelleyecek gerekli tedbirleri alabilen bir otonom yol planlaması yaklaşımının ortaya konulması amaçlanmaktadır.

1.3.4 Gerçek zamanlı hesaplama

Çalışma kapsamında formasyonu teşkil ederek muhafaza eden yol planlamasının bu amacı mümkün olan en kısa sürede karşılayabilecek sonuçları üretmesi beklenmektedir. Ayrıca platformların engellerden sakınmalarını sağlayan ve birbirleri ile çarpışmalarını engelleyen, önceden tanımlanmış olan seyrüseferi otonom olarak icra etmelerine olanak sağlayan yol planlaması yaklaşımının, platformların hareket hızına bağlı olarak en hızlı (mümkün olduğunca gerçek zamanlı) sonuç üretebilecek nitelikte olması bu tez çalışması kapsamında amaçlanan hedeflerden bir tanesidir. Bu nedenle çalışma kapsamında ortaya konulan yaklaşımın hesaplama performansı seri ve paralel farklı donanım mimarilerinde hesaplama performansı olarak sınanmıştır. Bu donanım mimarileri üzerinde hesaplama yapılmasına imkan sağlayacak şekilde farklı programlama mimarilerinden faydalanılmıştır. Yaklaşımların başarısı simülasyon ve uygulamalı olarak ortaya konulmuştur.

1.4 Çözüm Yaklaşımı

Bir üst maddede açıklanan hedeflere bu tez çalışması kapsamında ulaşılabilmesi amacıyla literatürde Yapay Potansiyel Alanlar (YPA) olarak adlandırılan vektör

tabanlı yol planlaması yaklaşımından faydalanılmıştır. YPA tabanlı yol planlaması yaklaşımı vektör tabanlı bir hesaplama yöntemi olup, literatürde yer alan hesaplama veya geometri tabanlı yaklaşımlara nazaran daha esnek ve hızlı hesaplanabilir sonuçlar ürettiğinden seçilmiştir. Yol planlaması dahilinde yer alan çarpışmayı önleme, hedefe doğru seyrüsefer, engellerden kaçınma gibi birden fazla etkiyi tek bir matematiksel model ile ortaya koyabilen vektör tabanlı yaklaşımlar bu yönüyle diğer yaklaşımlara nazaran daha etkindirler. Ayrıca en optimum yolu bulma açısından hesaplama tabanlı yaklaşımlardan, en iyi temsil yeteneğiyle dolayısıyla uygulanabilirlik açısından ise geometri tabanlı yaklaşımlardan daha etkin sonuçlar üretmektedirler.

YPA tabanlı yol planlaması yaklaşımının literatürde yer alan tespit edilmiş olan problemlerinin (yerel minimum, salınım vb.) üstesinden gelebilecek şekilde (harmonik fonksiyonların kullanımı, yerel yerine genel yol planlaması vb.) ve gerçek zamanlı hesaplanmasına imkan sağlayacak SIMD tipinde paralel algoritmalar bu tez kapsamında geliştirilmiştir. Bu algoritmaların başarısı İHA sistemlerinde kullanılabileceği değerlendirilen GPU tabanlı paralel programlama donanımlarından faydalanılarak uygulamalı şekilde çözüm üretilmiştir. Ortaya konulan çözüm yaklaşımının başarısı simülasyon çalışmaları ile sınanmış ve AR Drone quadrotor platformlarından faydalanılarak uygulamalı şekilde ortaya konulmuştur.

1.5 Araştırma Kısıtları

Bu tez çalışması kapsamında gerçekleştirilecek olan yol planlaması, çarpışmayı önleme ve engellerden kaçınma desteği ile sunulan otonom formasyon düzenin sağlanmasına yönelik geliştirilecek olan yaklaşımlar ve uygulamalar kapsamında aşağıda maddeler halinde sıralanan hususlar çalışma kapsamı dışında bırakılmıştır. Aşağıda tanımlanan kısıtlar dahilinde çalışmalar gerçekleştirilmiştir:

a. *Platformların konum ve hareket bilgisi:* Formasyon dahilinde yer alan platformların üç boyutlu uzayda konumlarının tespiti için her bir platform üzerinde yer alan GPS sisteminden alınan konum bilgisinden (2-8 metre hassasiyetinde) ve platformlar üzerinde yer alan ve irtifanın ölçülmesi amacıyla faydalanılan ultrasonic ve barometrik algılayıcılardan faydalanılmıştır. Ataletsel ve moment ölçer algılayıcılar ile platformların davranışları ölçülebilmektedir. Ayrıca platformların üzerinde yer alan manyetik pusula ile uçuş başı bilgisi edinilmiştir. Tüm bu

platformların konum ve hareket bilgilerini üreten algılayıcıları üzerinden edinilen bilgilerin doğru olduğu kabul edilmiş ve herhangi bir düzeltme veya doğrulama işlemine tabi tutulmamıştır. Ayrıca bu bilgilerin simülasyon çalışmaları esnasında sürekli erişilebilir nitelikte oldukları kabul edilmiştir.

b. *Dış aerodinamik etkiler:* Rüzgar ve meteorolojik koşulların (yağmur, kar, fırtına) çalışma kapsamında kullanılacak olan platformlar üzerindeki aerodinamik etkileri göz ardı edilmiştir. Platformların uçuşu esnasında birbirleri üzerinde oluşturdukları aerodinamik etkiler görmezden gelinmiştir.

c. *Engel ve diğer platformların konumunu algılama:* Otonom platformların üzerinde etraflarında yer alan engelleri ve diğer hava araçlarını belirli bir mesafeye kadar algılayabildikleri algılayıcılar olduğu kabul edilmiştir. Bu algılayıcıların tarafından üretilen verilerin temsilen doğru olduğu kabul edilmiştir. Algılayıcıların yetenekleri iki boyutlu olarak çalışma kapsamında tanımlanan biçimde hâlihazırda kullanılan algılayıcıların yeteneklerine istinaden belirlenen mesafe ile sınırlandırılmıştır.

1.6 Sonuç Özeti

Çalışma kapsamında icra edilen araştırmalar neticesinde YPA tabanlı otonom yol planlaması tekniklerinden faydalanılarak dar bir alan içerisinde birbirleri ile çarpışmadan ve ortam dahilinde yer alan engellerden kooperatif şekilde sakınarak bir grup İHA platformunun güvenli bir formasyon teşkil edebildikleri gösterilmiştir.

Bu maksatla otonom yol planlaması gereksinimlerinin YPA tabanlı yaklaşımlar ile SIMD tipinde paralel algoritmalar geliştirilerek GPU tabanlı paralel donanımlar üzerinde etkinlikle hesaplanabildiği ortaya konulmuştur. Ortaya konulan yaklaşımın gerçek zamanlı sonuçlar üretebilmesi için GPU tabanlı paralel programlama mimarilerinden istifade edilmiş ve SIMD tipinde paralel programların koşturulması amacıyla maliyet etkin çözümler ortaya koyan ve aynı zamanda İHA sistemlerinde (platform üzerinde veya yer istasyonunda) kullanılabileceği değerlendirilen grafik işlemci donanımları üzerinde yaklaşımın uygulanabilir olduğu ortaya konulmuştur. Özgün yaklaşımın başarısı simülasyon ortamında ve gerçek uygulamalar ile doğrulanmıştır.

1.7 Uygulama Alanları

İHA sistemlerinde yaşanan gelişmeler ışığında gerçek zamanlı yol planlamasına olan ihtiyacın hali hazırda ortaya çıktığı açıktır. Halihazırda kullanılan yüksek süratli ve agresif hareket yeteneklerine sahip İHA platformları, geniş ölçekli alanlar içerisinde yüksek hassasiyette hareket planlamasına ihtiyaç duymaktadır. Bu ihtiyaç modelleme açısından karmaşık uygulamalara ve de yüksek hesaplama gücü ile bunlara uygun sonuç üretebilecek gerçek zamanlı hesaplama algoritmalarına ihtiyaç duymaktadır.

Ortaya konulan otonom yol planlaması yaklaşımı ile İHA platformlarının bir arada bir formasyon teşkil edebilecek şekilde otonom yol planlaması ihtiyacının karşılanabileceği gösterilmiştir. Bunun yanı sıra güvenli bir formasyon teşkil edilebileceği örnekler ile simülasyon ortamında ve uygulamalı olarak açıklanmıştır. Bu bilgiler ışığında ortaya konulan yaklaşım sivil veya askeri birçok uygulamada özelleştirilerek uygulanılabilir. Bunlara örnek verilmesi gerekirse; İHA platformlarının havada yakıt ikmali, formasyon halinde bir hedefin görüntülerinin alınması, formasyon dahilinde yer alan İHA platformlarının bir arada kullanılarak haberleşme menzilinin arttırılması veya bir link oluşturulması gibi örnekler verilebilir.

1.8 Tez Organizasyonu

Tez çalışmasının *ikinci* bölümünde; çalışma kapsamında ele alınan hususlara istinaden literatür taramasına yer verilmiş, önceden yapılan akademik çalışmaların sonuçları irdelenerek ortaya konulmuştur. Ayrıca aynı bölüm dahilinde, çalışma kapsamında hedeflenen hususların hangi yöntemlerden istifade edilerek çözüm oluşturulacağı nedenleri ile ortaya konulmuştur.

Çalışmanın *üçüncü* bölümünde YPA tabanlı yol planlaması yaklaşımı dahilinde potansiyel ve vektör alanların nasıl modellenerek hesaplandığı, birden fazla potansiyel alanın ne şekilde sınırlandırılarak bir arada kullanıldığı gibi temel modelleme ve hesaplama hususlarına yer verilmiştir. Literatürde yer alan YPA tabanlı yol planlaması yaklaşımlarının karşılaştıkları sorunlara ve bunların çözümü için önerilerine değinilmiştir.

Çalışmanın *dördüncü* bölümünde ise, YPA tabanlı yol planlaması ile üç boyutlu ortam içerisinde yer alan mobil araçların yol planlamasının nasıl

gerçekleştirileceğine, üç boyutlu yol planlamasının hesaplama performansına etkilerine değinilmiştir. Ayrıca çalışma kapsamında özgün olarak ele alınan çok katmanlı YPA tabanlı yol planlaması yaklaşımı açıklanarak tanımlanmıştır.

Beşinci bölümde ise, YPA tabanlı yol planlaması ile dinamik modellemelerin nasıl gerçekleştirileceğine değinilmiş, yaklaşım matematiksel modeller üretilerek ortaya konulmuştur. Özellikle çalışma kapsamında örnek yol planlaması ihtiyacı duyan senaryolar olarak belirlenen İHA sistemleri için HYİ ve formasyon uçuşu modellemelerine değinilmiştir. Bu senaryolar dahilinde ihtiyaç duyulan yol planlaması gereksinimine YPA tabanlı yapılar ile nasıl cevap verilebileceği irdelenmiştir. YPA tabanlı yol planlaması yaklaşımının ortaya konulabilmesi için gerekli seri hesaplama fonksiyonlarının algoritmaları tasarlanmıştır.

Çalışmanın *altıncı* bölümünde YPA tabanlı yol planlamasının gerçekleştirilebilmesi amacıyla GPGPU tabanlı paralel algoritmaların nasıl geliştirildiğine değinilmiştir. Bölüm dahilinde vektör alanların hesaplanabilmesi için farklı hesaplama stratejisine sahip özgün paralel algoritmalar üretilmiş ve sonuçları karşılaştırmalı olarak irdelenmiştir.

Yedinci bölümde ise, formasyon uçuşunun gerçekleştirileceği alan dahilinde yer alan platformlar haricindeki engellerin nasıl tespit edileceğine, tespit edilen engelleri algılayan algılayıcıların çalışma kapsamında nasıl modellendiğine değinilmiştir. Formasyon yaklaşımının yanı sıra önceden özellikleri bilinmeyen dinamik alan içinde yer alan formasyonun engellerden sakınarak hareketini güncellemesine imkan sağlayan paralel algoritma tasarımları ortaya konulmuştur. Platformların birbirleri ile çarpışmalarını engelleyecek nitelikte önlemler almak amacıyla ilave fonksiyonlar tanımlanmıştır. Tespit edilen engellerin gruplanmasına, geometrik formlar ile temsil edilmesine ve olası çözüm uzayları ile engel modellemesi arasındaki optimizasyon gereksinimine değinilmiştir.

Çalışmanın *sekizinci* bölümünde ise formasyon uçuşu amacıyla ihtiyaç duyulan otonom yol planlaması yaklaşımının gerçek zamanlı olarak hesaplanmasına imkan sağlayacak nitelikte ortaya konulan algoritmaların GPGPU programlama yapıları ile GPU donanımları üzerinde koşturulmasına olanak sağlayacak program geliştirmelerine değinilmiştir. Ortaya konulan sistemin başarısı simülasyon çalışmaları ile sınanmış, hesaplama performansı farklı GPU donanımları üzerinde

seri hesaplama ile karşılaştırmalı olarak irdelenmiştir. Ayrıca birden fazla GPU donanımı bir arada kullanılarak hesaplama performansına etkileri incelenmiştir.

Çalışmanın *dokuzuncu* bölümünde ortaya konulan sistem çözümünün simülasyon çalışmaları ile doğrulanması yapılarak AR Drone quadrotor platformlarından faydalanılarak uygulamalı şekilde başarısı sınanmıştır. Simülasyon ve uygulama sonuçları bu bölüm dahilinde tartışılmıştır.

Son bölümde ise çalışma kapsamında ortaya konulan sistem çözümünün hedeflere ne derece yanıt verdiği, tez çalışmasının amaçlarına ne derece cevap üretilebildiği irdelenmiş, elde edilen sonuçlar tartışmalı olarak ortaya konulmuştur. Çalışmanın devamı niteliğinde neler yapılabileceğine değinilmiştir.

2. TANIMLAR VE LİTERATÜR ÖZETİ

Bu bölüm kapsamında, çalışma dahilinde ele alınan terim ile yaklaşımların literatürde yer alan tanımlarına ve bu kapsamda önceden gerçekleştirilen çalışmalara yer verilmiştir. Ayrıca tez çalışması kapsamında gerçekleştirilen çalışmaların ve uygulamaların hangi yöntemlerden faydalanılarak hangi sistem donanımları üzerinde gerçekleştirildiği detaylı biçimde tanımlanmıştır.

2.1 İnsansız Hava Aracı Sistemleri

İnsansız Hava Aracı platformları, yaygın olarak bilinen diğer adıyla "*Drone*", Uluslararası Sivil Havacılık Organizasyonu (International Civil Aviation Organization - ICAO) tanımlarına göre "*Pilotsuz Hava Aracı"* (*Unpiloted Aerial Vehicle, UAV*) veya "*Uzaktan Pilotlu Hava Aracı"* (*Remotely Piloted Aircraft, RPA*) olarak farklı şekillerde adlandırılmış olsa dahi özetle; üzerinde fiziki olarak pilot barındırmayan ve uçabilen hava aracı olarak tanımlanmıştır. ICAO tarafından insansız hava araçları iki temel kategoriye ayrılmıştır [17]:

a. *Otonom Hava Aracı*; halihazırda hukuki dayanak ve sorumluluk konularındaki eksiklere istinaden mevcut mevzuat kapsamında sivil amaçlı uçuşu uygun olmayan sistemlerdir. Herhangi bir insan kontrolünde olmayan, üzerinde yer alan algılayıcıların desteği ile karar vererek uçuşunu sürdüren, önceden planlanmış veya uçuş esnasında yapay zeka tarafından karar verilen uçuş görevini icra eden robot platformlar şeklinde tanımlanabilir.

b. *Uzaktan Pilotlu Hava Aracı*, ICAO ve ulusal havacılık otoritelerinin özel düzenlemelerine tabi olarak geliştirilen legal sistemlerdir. Özetle her ne kadar fiziksel olarak üzerinde bir pilot (insan) barındırmasa dahi uzak bir mesafeden bir insan (pilot) tarafından uçurulan hava araçlarıdır. Platform ve yer istasyonu arasında yer alan özel bir haberleşme sistemi ile kontrol edilen, platform üzerinde yer alan algılayıcı ve telsiz/uydu haberleşme sistemlerinin kontrol istasyonunda yer alan pilota aktarılması ile görevini icra eden platformlar şeklinde tanımlanabilirler.

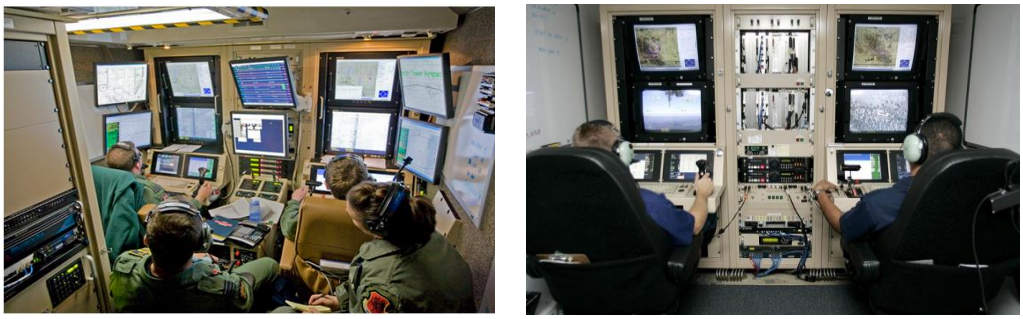
ICAO tarafından yapılan sınıflandırmaya istinaden otonom hava araçlarının uluslararası havacılık hukukunda tanımı yoktur. Oysa uluslararası havacılık hukukunda tanımı olmayan hava araçlarının, bir grup şeklinde ayrıca kategorize edilebilecek bir sayıya ulaşmış olması, yaygın olarak kullanıldığının ve hukuki tanımlama ihtiyacının ortaya çıktığının en büyük göstergesidir. Otonom hava araçları hakkında çok yakın bir gelecekte sivil havacılık mevzuatında gerekli değişiklikler ve eklentiler yapılarak hukuki zemin oluşturulacağı kaçınılmazdır. Bu hukuki dayanakların oluşturulması hiç şüphesiz otonom sistemlerin akredite olabilmeleri için bir takım standartlara (özellikle güvenlik konularında) sahip olmalarını gerektirecektir. Sivil havacılık uygulamalarında günümüz İHA sistemlerine benzeyen geleneksel insanlı sistemlerin akredite olabilmeleri için gereksinim duyulan konular incelendiğinde; yer alan birçok başlık arasında Trafik Çarpışma Önleme Sistemi (Traffic Collision Avoidance System - TICAS) ve durumsal farkındalığa bağlı farklı uçuş davranışlarını (görerek veya alet) destekleyebilme gibi kıstasların yer aldığı görülmektedir. Bu nedenle akredite edilmek istenen otonom hava aracı kesinlikle bu iki kısıtla birlikte çok sayıda ilave yeteneği bünyesinde belirli bir standart dizisinin tanımladığı biçimde bulundurmak zorunda kalacaktır.

Şekil 2.1: ABD Silahlı Kuvvetleri İHA sistemlerinin bazıları gösterimi [18].

Şekil 2.1'de hâlihazırda Amerika Birleşik Devletleri (ABD) Silahlı Kuvvetleri tarafından kullanılmakta olan farklı tipte İHA platformları görülmektedir. İHA platformları tek başına bir anlam ifade etmemektedir. Esasında İHA platformu bir sistemin sadece bir bileşenini oluşturmaktadır.

Şekil 2.2: Askeri amaçlı taktik sınıfı İHA sisteminin temel bileşenleri [19].

Askeri amaçlı kullanılan taktik sınıfı İHA sisteminin bileşenleri Şekil 2.2 ile gösterilmektedir. Şekilden de görüleceği üzere İHA platformu bu bileşenlerden sadece birisini oluşturmaktadır. Platform ister otonom hareket etsin, isterse de uzaktan bir pilot tarafından kontrol edilsin bu bileşenlerin büyük kısmına ihtiyaç duymaktadır.



(a) MQ-1 Predator YKİ. (b) RQ-2B Pioneer YKİ.

Şekil 2.3: Farklı İHA sistemleri için yer kontrol istasyonu iç gösterimi [18].

İHA sistemlerinin başlıca bileşenlerinden birisi de Yer Kontrol İstasyonu (YKİ) olarak adlandırılır. Şekil 2.3'de ise İHA sisteminde yer alan YKİ'nun içinden farklı İHA sistemleri için görünümlere yer verilmiştir. Görüldüğü gibi platformun uzaktan yönetilmesi ve uçurulması için çok sayıda insan desteğine ihtiyaç duyulmaktadır. Ancak bu personelin hiç birisinin platform üzerinde yer almasına gereksinim duyulmaktadır. İşte bu husus İHA sistemlerinin kullanımının artmasına neden olan en büyük etkendir.

İHA sistemleri kullanım amaçlarına ve farklı ülkelerin gerçekleştirdikleri araştırma çalışmalarında kullandıkları terminolojiye bağlı olarak çok farklı isimler ile anılmaktadırlar; örneğin "*Pilotsuz Hava Aracı Sistemi"* (*Unpiloted Air System, UAS*), *"Uzaktan Pilot Kontrollü Hava Aracı Sistemi"* (*Remote Piloted Aircraft System, RPAS*), *"İnsansız Savaş Hava Araçları"* (*Unmanned Combat Air Systems, UCAS*), gibi. İster platform üzerindeki uçuş bilgisayarı tarafından isterse de yer kontrol istasyonunda yer alan pilot tarafından uçurulursa uçurulsun ve ayrıca adının ne olduğundan bağımsız olarak, drone sistemlerinin ortak noktası platform üzerinde fiziksel olarak platformu uçuran pilot (insan) bulunmayışıdır.

Her ne kadar İHA sistemleri silahlı kuvvetler tarafından yaygın olarak kullanılsalar dahi, insan hayatının riske atılması riskli bulunan veya uzun görev saatlerinden dolayı insan sınırlarından etkilenmeden görev icra edilebilmesi amacıyla güvenlik, yangın izleme, radyasyon ölçümü, ilaçlama gibi sivil görevlerde de kullanımları giderek artmaktadır [20].

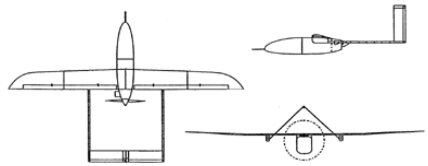
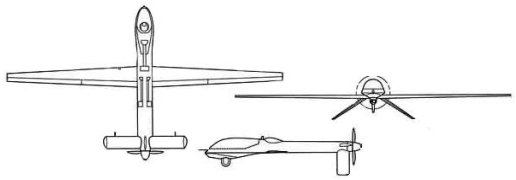
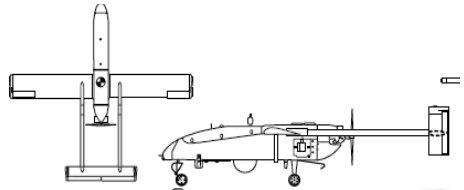
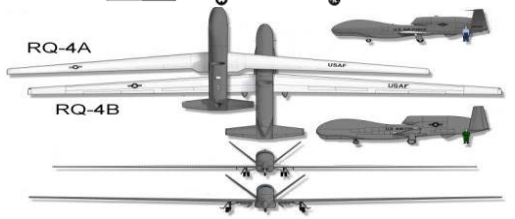
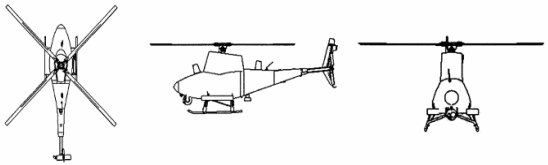
İnsansız hava araçlarına olan talepteki artışın belki de en önemli nedeni bu araçların insanların emniyetle gidip dönemeyecekleri her göreve gönderilebiliyor olmasından kaynaklanmaktadır. Bir diğer önemli neden ise insanlı sistemler ile icra edilen görevlere istinaden maliyet-etkinlik faktörüdür. Örneğin havadan keşif amaçlı farklı platformların kullanım maliyeti açısından Çizelge 2.1'de bir karşılaştırma yapılmıştır. Görüldüğü üzere en maliyet etkin yaklaşım İHA sistemlerinden faydalanmaktır.

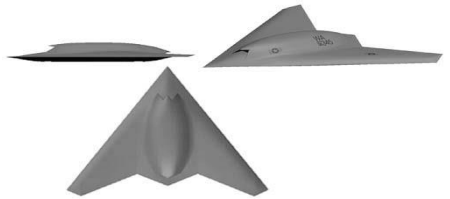
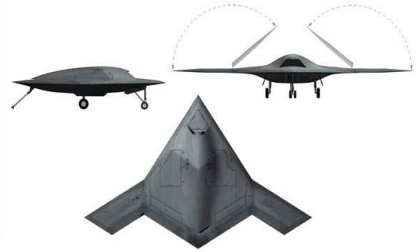
Çizelge 2.1: Literatürdeki İHA ve diğer hava araçlarının uçuş maliyetleri [14].

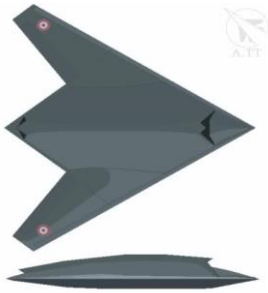
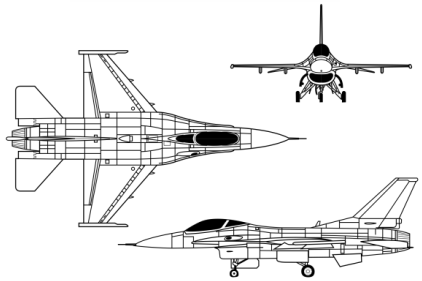
Hava Aracı	1 Saatlik Uçuş Maliyeti
GNAT-750 İHA Sistemi	400 Dolar
UH-1H Helikopteri	1000 Dolar
UH-60 Skorsky Helikopteri	3075 Dolar
RF-4E Phantom Keşif Uçağı	12.500 Dolar

İHA platformlarının birbirlerinden ayrıldıkları bir diğer nokta ise kalkış biçimleridir. Hali hazırda kullanılan İHA platformları incelendiğinde koşturarak pisten kalkan, dikine kalkış yapabilen ve destek bir sistem tarafından fırlatılarak kalkış yapan olmak üzere farklı kalkış sistematikleri ile donatılmışlardır. Tüm bu ayrı kalkış sistemleri için otonom çözümler geliştirilmiş ve üzerinde çalışılmaya devam edilmektedir.

Çizelge 2.2: Son nesil İHA platformları ve özellikleri.

Adı/Modeli	Maksimum Sürat (kt – m/sn)	Maksimum İrtifa (ft – m)	Görev Yarıçapı (nm – km)	Algılayıcıları	Temsil
Aerosonde Mark 4.7	80 – 40	15000 – 4500	100 – 150	EO/IR	
MQ-1 Predator	118 – 60	25000 – 7600	500 – 926	EO – IR – SAR	
RQ-2B Pioneer	110 – 57	15000 – 4500	100 – 185	EO – IR	
RQ-4B Global Hawk	350 – 180	60000 – 18000	5400 – 10000	EO – IR – SAR – MTI	
RQ-8A/B Fire Scout	125 – 64	20000 – 6000	150 – 278	EO – IR – LDRF	

Adı/Modeli	Maksimum Sürat (kt – m/sn)	Maksimum İrtifa (ft – m)	Görev Yarıçapı (nm – km)	Algılaıcıları	Temsil
Boeing X-45C	460 – 237	40000 – 12192	1200 – 2200	ESM – SAR – MTI	
MQ-9 Predator B	225 – 116	50000 – 15000	2000 – 3700	SAR – MTI Weapons (4x500lb)	
Northrop Grumman X-47B	460 – 237	40000 – 12192	1600 – 3000	ESM – SAR – GMTI – EO – IR	
BAE - Taranis	660 – 340	45000 – 13700	2000 – 3700	Weapons (2x500 lbs)	
TAİ – ANKA (TİHA)	117 – 60	30000 – 9150	108 – 200	EO-IR-LRF-LD-SAR/MTI	

Adı/Modeli	Maksimum Sürat (kt – m/sn)	Maksimum İrtifa (ft – m)	Görev Yarıçapı (nm – km)	Algılaıcıları	Temsil
Dassault nEUROn	530 – 237	45900 – 14000	2000 – 3700	Weapons (2x500 lbs)	
Boeing Unmanned QF-16 Viper	1041 – 536	40000 – 12192	1700 – 3150	LANTIRN – LITENING – AN/APG-68/83 radar	
BAE Mantis	300 – 155	55000 – 16764	2000 – 3700	EO – IR – SAR	

Otonom bir İHA platformu için yol planlaması (path planning), seyrüsefer (navigation) ortamında yer alan engellerden kaçınarak (obstacle avoidance) varılmak istenen üç boyutlu evrendeki bir noktaya ulaşmak olarak tanımlanabilir [21]. Bazı durumlarda platform, çevresi hakkında tam bir bilgi sahibidir ve buna dayalı olarak hareketini planlar. Ancak, bazı durumlarda sadece hedef noktaya ait konumsal bilgiye sahiptir ve çevresi hakkında bilgi toplamak için üzerindeki algılayıcılardan faydalanarak yol planlamasını dinamik olarak gerçekleştirir [21].

İHA platformları kullanım amaçlarına uygun olarak farklı sürat, tavan irtifası, görev yarıçapı gibi teknik özelliklerde üretilirler. Bu değerleri belirleyen temel neden İHA sistemi ile gerçekleştirilecek görevlerin ihtiyacıdır. Aynı zamanda görev ihtiyaçlarına uygun olarak farklı tipte faydalı yükler ile donatılırlar. Son yıllarda geliştirilmiş ve sınıflarının öncüsü olarak tanımlanan İHA platformları ile bu platformlara ait teknik veriler, üzerlerinde taşıyabildikleri faydalı yük algılayıcı sistemleri Çizelge 2.2 ile gösterilmiştir

2.2 İHA Sistemlerinin Geleceği

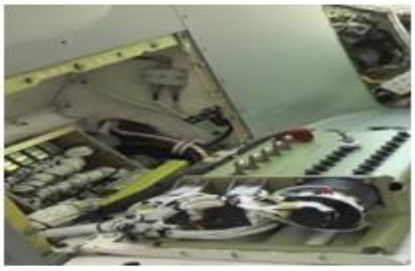
İHA sistemleri üzerine gerçekleştirilen son çalışmalar ışığında İHA platformları kıtalar arası görev icra edebilen, ses hızına yakın ve üzerinde süratlere çıkabilen yeteneklere sahip olmaktadır. Bu yeteneklerdeki gelişim, otonom sistemler ile desteklenmelidir. Ancak bu şekilde İHA sistemleri farklı görevlere hizmet etme imkanına ulaşabilir.

Halihazırda kullanılmakta olan İHA platformlarının sürat, irtifa ve manevra kabiliyetleri insanlı görevleri icra etmek açısından yeterlidir. Ancak İHA sistemleri tarafından karmaşık insanlı görevlerin icra edilmesi için platformun fiziki yetenekleri tek başına yeterli değildir. Tüm bu fiziki yeteneklerin aynı zamanda otonom yetenekleri planlayacak, yönetecek ve icra etme amacıyla kullanılacak sistemler ile desteklenmesi gerekmektedir. Çizelge 2.2'de görüleceği üzere İHA platformlarının fiziki ve aerodinamik yetenekleri insanlı hava platformlarının yeteneklerine ulaşmıştır. Hatta bazı insanlı hava platformlarının insansız otonom sistemlere veya uzak kontrollü sistemlere dönüştürülmesi ile daha üstün yetenekler elde edilmiştir. Örneğin Boeing firması tarafından geliştirilen Unmanned QF-16 Viper olarak adlandırılan platform insanlı bir sistemden modifikasyonlar ile geliştirilmektedir ve insanlı sisteme nazaran daha hızlı, daha dar dönüş yarıçapına sahip, daha yüksek G

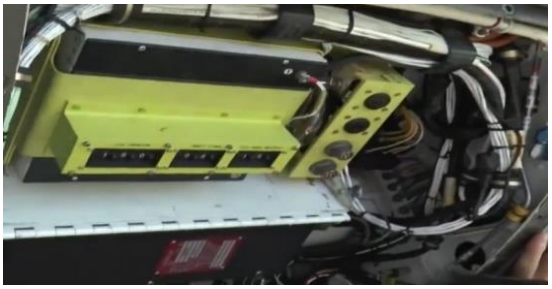
yüklerine çıkabilen, dolayısıyla daha agresif manevraları icra etme yeteneğine sahip bir platform haline gelmiştir [23].

(a) İnsansız platformun otonom kalkışı.

(b) İnsansız platform ile insanlı F-16 uçuş kolu.



(c) Platform otonom uçuş sistemi ile aviyonik bağlantıları.



(ç) Otonom uçuş bilgisayarı.

(d) İnsansız QF-16 alçak irtifa uçuşu.

(e) Uçuş anında kokpit görüntüsü.

(f) Türk Hava Kuvvetlerine ait insanlı F-16 uçakları kol uçuşu [24].

Şekil 2.4: Boeing Unmanned QF-16 Viper platformundan görünümler [23].

Boeing tarafından önce insanlı olarak geliştirilen F-16 muharip uçağı Şekil 2.4 ile gösterildiği üzere ABD hava kuvvetleri envanterinde yer alan blok 15, 25 ve 30

tipindeki uçakların Hava Üstünlüğü Hedef Geliştirme (Air Superiority Target - AST) programı kapsamında drona dönüştürülmesi ile insansız hale getirilmiştir [23].

Bu AST projesi kapsamında geliştirilen QF-16 dronlarının Silah Sistemi Değerlendirme Programı (Weapon System Evaluation Program - WSEP) kapsamında havadan havaya geliştirilen füzelerin testleri amacıyla kullanılması planlanmaktadır. Aynı zamanda havadan havaya füze atışlarında pilotların eğitimi için kullanılacağı açıklanmıştır. Bu kapsamda modernize edilerek insansız hale getirilen QF-16 insansız hava aracı platformları tanımlanan bu görevleri yerine getirebilmek için yüksek süratte (Mach 1.47/1,800 km/s) ve yüksek irtifalarda (yaklaşık 40,000 ft/12,2 km) agresif manevraları yapabilmesi gerekmektedir. Proje kapsamında toplamda 126 adet QF-16 insansız platformunun geliştirilmesi planlanmaktadır [23].

Şekil 2.4 (a)'da insansız platformun otonom kalkışı görülmektedir. Şekil 2.4 (b) ile insansız QF-16 platfromu ile insanlı F-16 uçağının kol uçuşu görülmektedir. Şekil 2.4 (c) ve (ç) ile platform otonom uçuş sistemi ile aviyonik bağlantıları ile otonom uçuş bilgisayarı gösterilmektedir. Şekil 2.4 (d) ile insansız QF-16 platformunun deniz üzerinde alçak irtifa uçuşu, Şekil 2.4 (e) ile bu uçuş esnasındaki kokpit görüntüsü gösterilmektedir. Gelişmeler göstermektedir ki Şekil 2.4 (f) ile de gösterildiği üzere Türk Hava Kuvvetlerine ait insanlı F-16 uçakları tarafından icra edilen kol uçuşu görevi çok yakın bir gelecekte insansız platformlar tarafından gerçekleştirilecektir.

(a) İnsansız platformların kol uçuşu.

(b) İnsansız platformlar ile insanlı platformun oluşturduğu uçuş kolu.

Şekil 2.5: Stealth filminde yer alan İHA platformlarından görünümler [25].

Stealth filmi, 2005 yılı Amerikan yapımı olup Rob Cohen tarafından yönetilmiş ve Columbia Pictures firması tarafından dağıtımı yapılmış, yaklaşık 135 milyon ABD doları bütçeli bir bilim kurgu filmidir [25]. Film senaryosu kapsamında Şekil 2.5 (a) ile gösterildiği şekilde İHA platformları tarafından icra edilen yakın kol uçuşu görevi

ve Şekil 2.5 (b) ile gösterilen biçimde insanlı hava aracının kolunda yer alan ve yönetiminde uçan otonom İHA platformları görülmektedir.

Gerek QF-16 platformunun geliştirilmesi neticesinde oluşan durum gerekse de Stealth bilim kurgu filmi ile ortaya konulan tanımlama göstermektedir ki, özellikle askeri alanda kullanılan uçak sistemlerinin yerini çok yakın bir gelecekte insansız platformlar alacaktır. Bu durum Lockheed Martin firması tarafından geliştirilmesine devam edilen F-35 Lightning muharip uçağı hakkında ABD Savunma bakanlığı tarafından yapılan üretilen son insanlı muharip hava aracı olacağı yönündeki açıklama ile de bir kez daha doğrulanmıştır. Bu gelişmeler gerçekleştirilen bu tez çalışmasının önemini bir kez daha ön plana çıkarmaktadır.

2.3 İHA Platformları İçin Formasyon Uçuşu

İHA platformları için formasyon uçuşu yeteneğine niçin ihtiyaç duyulduğuna değinmeden önce formasyon kavramının tanımını yapmakta fayda olacaktır. Bu anlamda formasyon, birden fazla aracın bir arada belirli bir düzeni muhafaza ederek bir bütün şeklinde davranış sergilemesi olarak özetle tanımlanabilir. Formasyonun kara, deniz, hava araçları hatta uydu ve denizaltı sistemleri gibi farklı tipte araçlar için değişik uygulamalarına kullanımda rastlamak mümkündür.

(a) Kuş sürüsünün oluşturduğu formasyon.

(b) Karınca sürüsünün oluşturduğu formasyon.

(c) Balık sürüsünün oluşturduğu formasyon.

Şekil 2.6: Doğada yer alan farklı formasyon teşkillerinden örnekler.

Formasyon en az iki araç ile oluşturulabilen ve görevin gereksinimlerine cevap verecek formda icra edilen grup hareketidir. Grup hareketinden kasıt, belirli bir düzen içinde hareket etmektir. Düzen dahilinde yer alan her grup üyesi aracın belirgin bir konumu görev süresince muhafaza etmesine ihtiyaç duyar. Sürüden ayrıldığı noktada budur. Sürü dahilinde yer alan araçlar birbirlerinin yerini rastgele

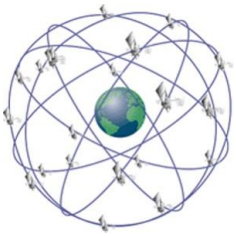
alabilirler. Ancak formasyon dahilindeki araçlar ise kendilerine tanımlı olan konumda yer almak zorundadır. Konum değişimleri mümkün olmak ile birlikte bu durum sadece formasyon yönetimi tarafından istendiğinde mümkün olmaktadır. Formasyon dahilinde özel olarak adlandırılan pozisyonlar da mevcuttur; örneğin lider gibi. Hatta formasyon dahilinde sanal üyelerde yer alabilirler.

Temel grup hareketleri doğada sıkça görülmektedir. Oluşan doğal canlı grupları farklı ihtiyaçlara cevap vermek üzere teşkil edilirler. Örneğin Şekil 2.6 (a) da görüldüğü biçimde bir arada bir formasyonu koruyarak göç eden kuş sürüsü, daha az efor sarf ederek daha uzun menzilli uçuşlar gerçekleştirebilir. Veya Şekil 2.6 (b) görüldüğü biçimde bir formasyonu muhafaza ederek hareket eden karınca sürüsü kaybolmadan seyrüsefer gerçekleştirebilir. Şekil 2.6 (c) de görülen formasyon halinde hareket eden balık sürüsü kendilerine yönelen tehlikelerden daha az etkilenirler.

(a) Hava aracı formasyonu.

(b) Deniz aracı formasyonu.

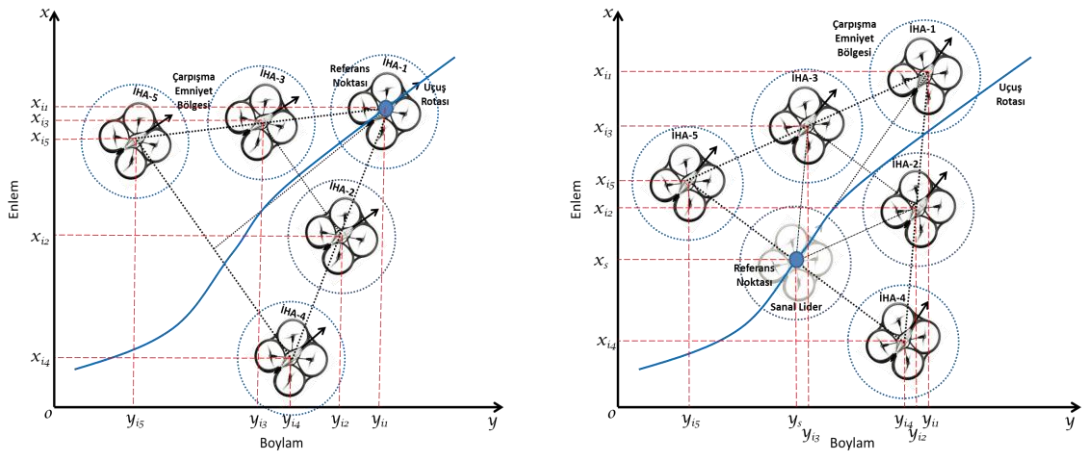
(c) Kara aracı formasyonu.



(ç) Takım uydu formasyonu.

Şekil 2.7: Farklı formasyon teşkillerinden örnekler.

Şekil 2.7'de farklı platformların farklı amaçlar doğrultusunda oluşturdukları formasyonlardan örnekler verilmiştir. Örneğin Şekil 2.7 (a)'da üç hava aracının havada yakıt ikmali için oluşturdukları formasyon, Şekil 2.7 (b)'de çok sayıda deniz taşıtının Şekil 2.7 (c)'de ise kara taşıtlarının birlikte seyrüsefer icra ederken oluşturdukları formasyonlar görülmektedir. Şekil 2.7 (d)'de ise farklı yörüngelerde yer alan ancak belirli bir uyum içinde hareket eden çok sayıda uydudan teşkil edilmiş olan bir formasyon görülmektedir. Şekil 2.7 (a)'da formasyon dahilindeki pozisyonlara net örnekler verilebilir. Bu durumda formasyonun ilk pozisyonunda yakıt aktarımı yapan tanker hava aracı, diğer pozisyonlarda ise yakıt almayı bekleyen diğer platformlar yer almaktadır. Yakıt alma görevini icra eden araç arkada ve soldaki konumda yer almalıdır.



(a) Sanal lider yaklaşımı olmadan formasyon düzeni.

(b) Sanal lider yaklaşımı ile formasyon düzeni.

Şekil 2.8: İHA sistemleri için V tipi formasyon düzeni.

Gerek doğada yer alan canlılar tarafından oluşturulmuş olsun gerekse de insanlı veya insansız araçların oluşturduğu formasyonlar olsun hepsinde bir lider yer almaktadır. Lider grup içinde yer alan üyelerden birisi veya sanal bir lider olabilir. Hatta liderlik görevini farklı grup üyeleri ya da tamamı sırayla icra edebilirler. Formasyonun sağlanması üzeri ne literatürdeki farklı yaklaşımlar incelendiğinde temelde iki tip seyrüseferi destekleyen modele rastlanılmaktadır. Bunlardan ilki Şekil 2.8 (a)'da gösterildiği biçimde grup içerisindeki bir aracın lider olarak seçilmesi ve de diğer araçların bu liderin davranışlarına göre göreceli olarak davranmalarının sağlanmasıdır. Bu yaklaşımın dezavantajları da liderin grup içinden seçilmesi, liderin her hangi bir nedenler uçuşa devam edememesi durumunda yeni liderin belirlenmesi, liderin davranışlarının diğer platformlara göre koordine edilmesi olarak sıralanabilir. Bir diğer yaklaşım ise Şekil 2.8 (b)'de gösterildiği biçimde seyrüsefer yapan ancak gerçekte var olmayan sanal bir lidere göre formasyonun sağlanması yaklaşımıdır. Bu yaklaşımın en önemli avantajı olarak, platformların birbirlerine göre konumu yerine referans teşkil edebilecek nitelikte bir nirengiyi baz almaları söylenebilir. Ancak bunun yanında sanal liderin davranış modellemesinin yapılması ayrıca bir sisteme yük getirmektedir.

Bu tez kapsamında her iki yaklaşımın problemlerinin çözümünden çok yaklaşımın yukarıda sıralanan özelliklere sahip şekilde bir kontrol mekanizmasının geliştirilmesi amaçlanmıştır. Bu nedenle çalışma kapsamında Şekil 2.8 (a) ile gösterilen ve sanal lider olmayan yaklaşım baz alınmıştır.

Formasyonlar daha alt gruplara ayrılarak nispeten küçük ölçekli yeni formasyonlar ya da küçük ölçekli formasyonlar bir araya gelerek nispeten daha büyük ölçekli yeni formasyonlar oluşturabilirler. Ancak tüm bu koşullar altında formasyon tanımı ve mantalitesi değişiklik göstermez.

Formasyon yaklaşımı, insanlı askeri ve sivil uçaklarda kullanılan bir uçuş tekniğidir. İnsanlı hava aracı platformlarının formasyon uçuşu gerçekleştirmelerinin farklı amaçları vardır. Bunları örneklemek gerekir ise;

a. Formasyon dahilinde yer alan platformlar, sürükleme (drag) kuvveti etkisi gibi yakıt tüketimini arttıran etkilerden daha az etkilenirler ve daha uzun uçuş menziline sahip olabilirler;

b. Havada yakıt ikmali gibi değişik görevlerin icra edilebilmesi için platformlar arasında önceden tanımlı bir formasyonun sağlanması görevin emniyeti açısından kaçınılmazdır;

c. Özellikle askeri amaçlı hava platformları bir formasyon teşkil ederek radar kesitinde ve gözle görülme esnasında tespit edilmek açısından avantaj sağlarlar;

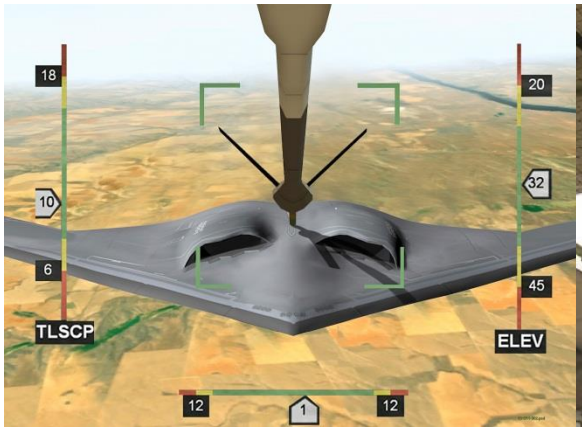
ç. Savaş uçakları etkin bir savunma ve de ateş gücünden faydalanmak amacıyla formasyon uçuşundan faydalanır,

d. Formasyon dahilinde hareket eden hava araçları farklı faydalı yükler taşıyarak aynı hedefin farklı algılayıcı verilerine sahip olabilirler ya da aynı algılayıcıları ile hedefin farklı açılardan edinilmiş bilgilerine ulaşabilirler.

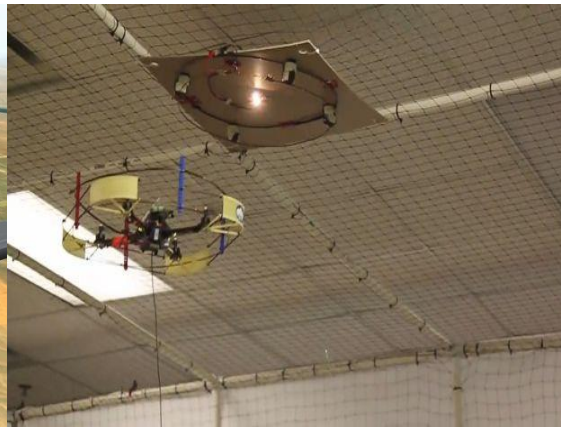
Bu örnekleri arttırmak mümkündür. İHA platformları da diğer insanlı hava aracı sistemleri gibi belirli bir formasyonu muhafaza edecek şekilde uçmayı görevleri gereği amaçlarlar [20]. Özellikle son yıllarda artan bir ivme ile İHA sistemlerinin formasyon şeklinde uçmalarına yönelik çalışmaların sayısında önemli bir artış olmuştur. İHA platformlarının belirli bir formasyonu muhafaza edecek şekilde uçmalarını amaçlayan temel motivasyon sivil ya da askeri uygulamalarda maliyeti azaltmak ve var olan teknolojik imkanlara alternatifler oluşturmak olarak sıralanabilir.

(a) DARPA tarafından İHA sistemlerinin havada yakıt ikmali projesi [26].

(b) NASA Global Hawk RQ-4 İHA platformları arasında havada yakıt ikmali [27].



(c) ABD Hava Kuvvetleri Araştırma Laboratuvarı tarafından yürütülen İHA sistemleri için havada yakıt ikmali çalışması [27].



(ç) NIMBUS laboratuvarı (Nebraska-Lincoln Üniversitesi) tarafından yürütülen uzaktan quadrotor platformu elektrik şarj istasyonu projesi [28].

Şekil 2.9: İHA sistemleri için havada yakıt ikmali formasyon düzenleri.

İHA platformları tarafından formasyon uçuşunun önemini arttıran bir diğer başlık insanlı sistemleri icra ettiğine benzer Şekil 2.9 (a), (b) ve (c) ile gösterildiği biçimde veya kendilerine has Şekil 2.9 (ç) ile gösterilene benzer bir sistem yaklaşımı ile havada yakıt ikmali görevini icra etme ihtiyaçlarıdır. Hiç şüphe yoktur ki İHA platformlarının havada yakıt ikmali görevini gerçekleştirebilmeleri için öncelikle formasyon uçuşu kabiliyeti kazanmaları gerekmektedir. Çünkü yakıt ikmali görevinde yakıt ikmali yapan platform ile yakıt ikmali desteği sunan platform arasında yakın kol uçuşuna benzer bir formasyon teşkil edilmek ve de yakıt ikmali süresince muhafaza edilmek zorundadır.

Özellikle dağıtılmış algılayıcı sistemlerin bir arada uyum içerisinde kullanımına imkan sağlaması arama, hasar tespiti, keşif, gözetleme, kimyasal ve biyolojik hasar tespiti, hasat kontrolü, topografya izlenmesi gibi farklı uygulama alanlarında daha etkin ve ucuz çözümler üretilmesi açısından formasyon uçuşu yeteneği önem arz

etmektedir. Birden fazla algılayıcı sistemi tek bir İHA platformu üzerinde taşımak doğal olarak platform boyutlarının ve kabiliyetlerinin arttırılması ihtiyacını beraberinde getirecek ve bununla birlikte önemli ölçüde maliyeti arttırıcı bir durum ortaya çıkacaktır. Oysa aynı sayıda farklı algılayıcı sistemi nispeten daha küçük ve çok sayıda ancak belirli bir formasyonu muhafaza ederek hareket edebilme kabiliyetine sahip platform üzerinde taşımak daha esnek ve etkin çözümler üretilmesine imkan sağlayacaktır. Tek bir platform kullanımı görevin başarısı açısından yüksek riskler teşkil ederken, aynı görev için formasyon dahilinde birden fazla platform kullanımı görevin başarısına mutlaka katkıda bulunacaktır.

İnsansız Hava Aracı (İHA) platformları için güvenli formasyon kontrolü son yıllarda üzerinde çalışılan bir konu haline gelmiştir [8][29]. Yakıt tasarrufu ve de uçuş menzilinin arttırılması dışında formasyon uçuşunun İHA platformları için avantajlarından birisi de birden fazla platformun daha fazla faydalı yükü birlikte taşıyabilmesidir. İş birliği içinde bir arada davranılması görevin başarı şansını arttıracak ve de daha kısa sürede tamamlanmasına imkân sağlayacaktır. Daha yüksek faydalı yük kapasitesine sahip üst sınıf bir İHA kullanımı yerine, göreceli olarak daha küçük birden fazla platformun bir arada aynı görev için kullanılması daha maliyet etkin bir çözüm olacaktır. Özel bir görev için gerekli olan sensor ve teçhizatı birden fazla platform üzerine dağıtarak daha düşük faydalı yük kapasitesine sahip platformlar ile görevin icrasına imkân sağlayacak ve otonomi ile performansın arttırılmasına olanak verecektir.

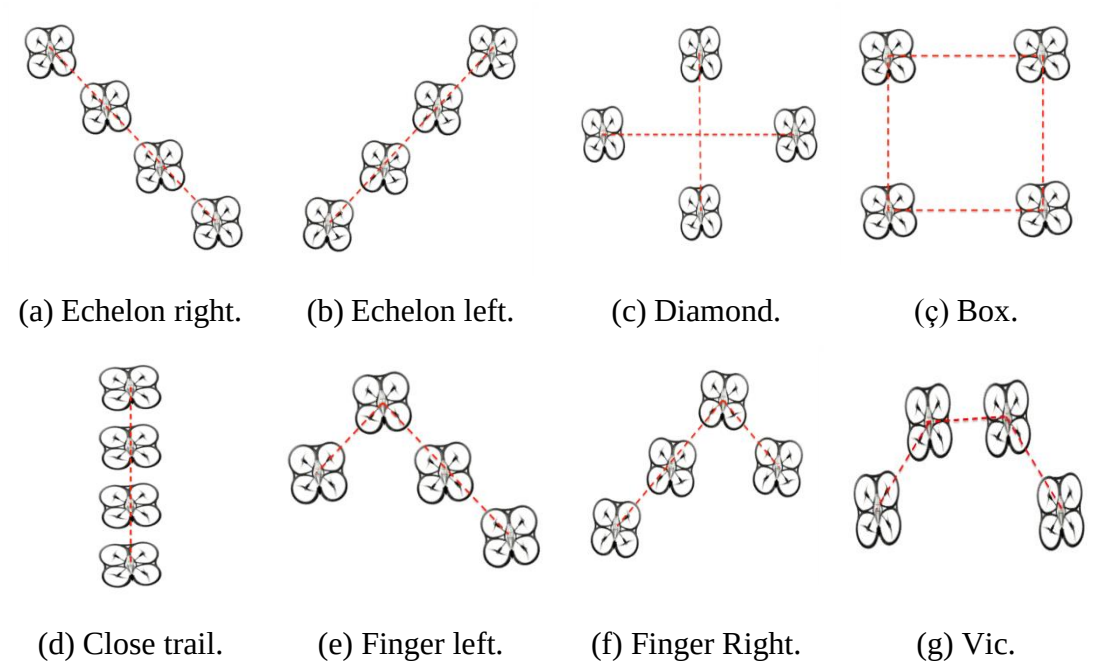
Çoklu insansız araç sistemlerinin karmaşık yapısına rağmen büyük ilgi çekmelerinin birçok nedeni vardır [30]. Özetle sıralamak gerekirse:

a. Görev tek İHA platformunun başarabilmesi için çok karmaşık olabilir,

b. Tek bir platform kullanmak yerine kooperatif şekilde görev icra edebilen çoğul İHA platformları kullanılarak görevin tamamlanma süresi açısından önemli bir performans avantajı sağlanabilir,

c. Görev ihtiyaçlarını karşılayabilmek için nispeten büyük ölçekli ve yüksek faydalı yük kapasitesine sahip İHA platformu görevlendirmek yerine; çeşitli sayıda basit, küçük insansız araçlarını bir arada kullanmak daha maliyet etkin, daha esnek ve hata toleransı daha düşük bir çözüm olabilir. Ayrıca nispeten küçük ölçekli İHA platformları kullanılarak farklı görev ihtiyaçlarına daha esnek çözümler üretilebilir,

ç. İHA platformlarından teşkil edilmiş formasyon tek bir İHA kullanımına nazaran daha güvenilir ve sağlam olabilir. Örneğin bir araç ile icra edilen görevlerde platformdan kaynaklanan bir sorun oluşursa görev kesinlikle başarısızlığa uğrarken, formasyon halinde hareket eden araçlardan birisinde oluşabilecek bir sorun diğer araçlardan birisinin onun görevini devralması sağlanarak görevin devam edilmesine olanak sağlayabilir. Görev başarısızlığa uğramadan ya da kısmi bir başarı ile devam ettirilebilir.



(a) Echelon right. (b) Echelon left. (c) Diamond. (ç) Box.

(d) Close trail. (e) Finger left. (f) Finger Right. (g) Vic.

Şekil 2.10: Farklı formasyon teşkillerinden örnekler.

Formasyon uçuşu; özetle birden fazla uçan platformun gerçekleştirdikleri seyrüsefer süresince lider olarak tanımlanan platforma göre ya da birbirlerine göre uçuş süresince belirli bir konumu muhafaza ederek hareket etmelerine denir. Şekil 2.10'da gösterildiği şekilde değişik uçuş formasyonları mevcuttur ve bunların sayısını arttırmak mümkündür.

Farklı formasyon uçuş düzenleri kullanılmasında sivil ve askeri nedenler vardır. Bu tez çalışması kapsamında farklı uçuş formasyon düzenlerinden faydalanılmasındaki amaç, uçuş süresince engellerden kaçınmak, değişik algılayıcılara sahip olduğu düşünülen İHA platformlarının bir düzene istinaden görev icra etmelerine olanak sağlamak ve de belirli bir mesafede kalarak birbirleri ile sürekli ilteşimi muhafaza edecek bir uçuşa imkan sağlamak olarak özetlenebilir.

Bu tez çalışmasının amacı platformları birbirlerine göre doğru konumda tutmayı amaçlayan otonom formasyon kontrolü yaklaşımını gerçek zamanlı olarak ortaya koymaktır. Özetle İHA sistemleri için formasyon kontrolü, iki veya daha fazla sayıda platformun lider platforma ve de diğer platformlara göre doğru konumu alarak, ortak bir görev doğrultusunda bir arada amaçlanan seyrüseferi güvenle icra etmesi olarak tanımlanabilir. Yakıt tasarrufu, dar bir alandan geçiş, aynı hedefe ait farklı sensorlar ile bilgi almak gibi değişik nedenlere istinaden uygulanan Şekil 2.10'da örnekleri gösterilen farklı formasyon yapıları mevcuttur. Bu tezde, Şekil 2.10 (a) ile gösterilen kademe sağ formasyonu yakıt tasarrufu için, Şekil 2.10 (d) ile gösterilen yakın iz oluşumu dar aralıktan geçişler için ve Şekil 2.10 (ç) ile gösterilen kutu oluşumu farklı sensorlar ile aynı hedefin görüntüleri almak için kullanılmaktadır. Ayrıca genel olarak insansız hava araçlarının formasyon uçuşunu sürdürebilmeleri için birbirleriyle iletişim halinde kalmalarına imkân sağlayacak şekilde platformları dar bir mesafede tutmak amaçlanmıştır. Her platform Şekil 2.10'da görüldüğü gibi uçuş sırasında doğru konumunu korumak için çalışır ve lider görev sırasında seyrüsefere devam etmektedir. Ayrıca, her bir platform uçuş sırasında çarpışmayı önlemek için gerekli manevraları gerçekleştirmelidir. Bu tez çalışması kapsamında geliştirilmek istenen formasyon kontrol mekanizması, her bir platformun doğru pozisyonunu korumak ve çarpışmayı önlemek için ihtiyaç duyduğu gerçek zamanlı komutların üretilmesini amaçlamaktadır. Formasyonda her platform, diğerleri için engel teşkil eder ve Şekil 2.10'de tarif edildiği gibi formasyon içindeki olunması gereken doğru pozisyonunun potansiyel çeken bir kaynak olduğu kabul edilir.

Çoklu otonom İHA sistemlerinin kullanımına olan ihtiyaç, karmaşık problemleri ve kazandırılması gereken ilave otonom yetenekleri de ortaya çıkarmaktadır. Bu yeteneklerden öne çıkan bazıları; görevin icrası sırasında coğrafi engellerden kooperatif olarak kaçınma ve dar alanlarda birbirleri ile çarpışmayı önleyici (collision avoidance) sistemler ile desteklenmiş otonom yol planlaması yetenekleridir. Bu araçların kontrolünde hiçbir insan etkileşiminin gerçek zamanlı olarak olmadığı düşünüldüğünde, yapay zeka bir çarpışmanın meydana geleceği bir durumun oluşmamasını sağlamaktan sorumlu olur. Otonom İHA sistemleri için çarpışma önleme planı;

a. Bir hava sahasında yer alan birden fazla araç arasında (insanlı ya da insansız) oluşabilecek çarpışma riski ve bunu önlemek için uyulması gereken minimum ayırma mesafesinin oluşturulması ve uyulması ihtiyacı,

b. İHA için yol planlaması esnasında karşılaşabileceği fiziki engellerden (coğrafi ya da suni engeller) sakınarak seyrüsefer icra etmesi ihtiyacı ve

c. Aynı görevin icrası esnasından faydalanılan birden fazla insanlı ya da insansız platformun bir uyum içerisinde birlikte işbirliği çerçevesinde hareket etmesi esnasında karşılaşılabilecek çarpışma riskini çözümleyebilecek nitelikte çözüm sunacak bir yöntem ihtiyacını karşılayabilir nitelikte bir yaklaşım olmalıdır.

Formasyonu uçuşu karmaşık bir kontrol problemi olarak karşımıza çıkmakta ve çok sayıda girdiye bağlı olarak platformların göreceli konumlarının muhafaza edilmesi temeline dayanmaktadır. Özellikle gerçek zamanlı uygulamalarda yüksek işlemci gücüne ihtiyaç duymakta ve platform kabiliyetleri ile doğrudan ilişkili sonuçlar üretmektedir. Bu noktadan hareket ile aşağıda sıralanan dar boğazların çözüme kavuşturulması ortaya konulacak olan yaklaşımın etkinliğini belirleyecektir. Bu kapsamda ortaya konulmak istenen formasyon amaçlı otonom yol planlaması yaklaşımı;

a. Değişik sayıda platformun (homojen veya heterojen özelliklerde sistemlerin) bir arada ve formasyonu muhafaza edebilecek nitelikte hareketine imkan sağlayacak özellikte olmalı,

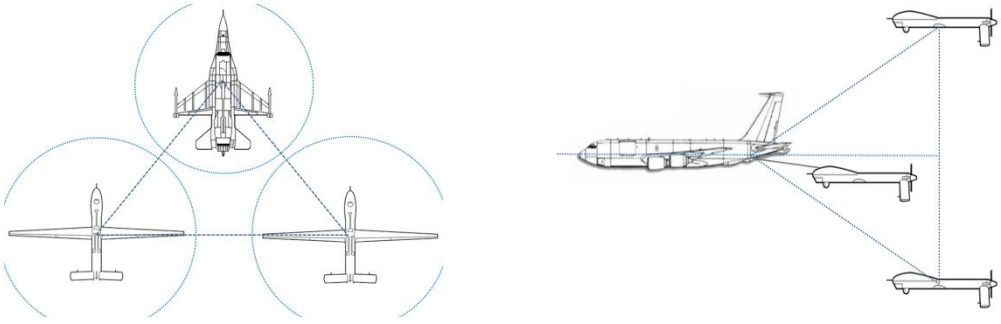
b. Belirli kurallar ile tanımlanmış bir seyrüsefer yaklaşımını destekleyebilmeli,

c. Değişik formasyon tiplerini (yakın formasyon, esnek formasyon, sıralı ve V formasyon vb.) destekleyebilicek ve bu tipler arasında geçişe izin verebilecek nitelikte olmalı,

ç. Formasyon uçuşu özellikle platformların dar bir alanda bir arada seyir halinde olmalarına ihtiyaç duyduğundan, çarpışmayı önleme gibi güvenlik özelliklerini bünyesinde doğal olarak taşımalı,

d. Seyrüsefer esnasında karşılaşılabilinecek doğal ve yapay engellerden etkilenmeyecek ve bu duruma reaksiyon gösterebilecek nitelikte esnek olmalı,

e. Platformların her birisine gerçek zamanlı kontrol imkanı tanıyabilecek uygulanabilir çözümler üretmeli ve platform kabiliyetlerine bağlı olarak esnek ve uyumlu sonuçlara imkan sağlamalıdır.



(a) İnsanlı – insansız sistemlerin bir arada oluşturdukları formasyon düzeni.

(b) İHA formasyonunun havada yakıt ikmali görevini icra etmesi.

Şekil 2.11: İnsanlı-insansız hava aracı entegrasyonu.

Havada yakıt ikmali veya insanlı-insansız hava aracı entegrasyonu gibi farklı ihtiyaçlara istinaden değişik formasyonlar ortaya konulabilir [1]. Formasyon yaklaşımı ayrıca değişik sistemlerin (insanlı – insansız) bir arada kullanılmasına da imkan tanıyabilirler. Örneğin Şekil 2.11 (a)'da görüldüğü üzere hareketleri önceden planlanmamış insanlı bir sistemin kolunda uçuşa devam edecek nitelikte bir formasyon yaklaşımına ihtiyaç duyulabileceği gibi, Şekil 2.11 (b)'de temsil edilen hareketleri önceden kestirilebilen HYİ görevindeki tanker platformun etrafında bir formasyonu teşkil edebilecek şekilde bir formasyon ihtiyacı da tanımlanabilir. İnsanlı ve insansız sistemler bir formasyon dahilinde bir arada kullanılabilir, buna örnek olarak havada yakıt ikmali görevi gösterilebilir. Geliştirilecek yaklaşımın esnekliği bu tip farklı kombinasyonlardan ve amaçlara istinaden teşkil edilmiş bir düzeni de sağlayabilecek özellikte olması yaklaşımının başarısını arttıracaktır. Tüm bunların ötesinde farklı ölçeklerde değişik sınıflara ait insanlı ve insansız sistemler bir arada da ya da çok sayıda küçük ölçekli sistem tek bir büyük ölçekli sisteme nazaran aynı görev dahilinde kullanılabilir.

2.4 Otonom Yol Planlaması

Otonom tabiri her ne kadar literatürde çok farklı şekillerde tanımlanmış olsa dahi özetle; otonom bir sistem "belirli bir görevi insan katkısı olmadan karar vererek yerine getiren yapı" olarak tanımlanabilir. Otonomi, güzergah tespiti ile hedef seçimi ve hedef noktalara intikali, üzerinde bulunan algılayıcılar, kontrol ve navigasyon

yazılımları ile hiçbir dış müdahaleye ihtiyaç duymadan kendi başına yapabilme ve karar verebilme yeteneğidir [13].

Kapalı ve açık olmak üzere planlama algoritmaları iki temel sınıfa ayrılır. Açık hareket planlama yaklaşımı literatürde en yaygın olan yaklaşımdır ve Şekil 2.12 ile gösterildiği biçimde üç adıma gerçekleştirilir; yol planlama, yörünge planlama ve robot kontrolü.

Şekil 2.12: Açık hareket planlaması bileşenlerinin gösterimi [15].

Açık hareket planlaması yaklaşımının ilk adımını oluşturan yol planaması, robotun başlangıç konumuna ve içinde bulunduğu ortam bilgisine bağlı olarak mevuct konumdan hedefe doğru bir eğri oluşturur. Oluşturulan bu eğri robotun kinematiklerinden ve dinamiklerinden bağımsız olarak sadece ortam bilgisi doğrultusunda hesaplanır. İkinci adımda ise oluşturulan yol planlamasına bağlı olarak robotun uygulayabileceği yörünge planlaması gerçekleştirilir. Son adım olarak da ortaya konulan yörünge robot kontrolü tarafından uygulanır.

Kapalı hareket planlaması yaklaşımında ise bu işlem basamakları bir arada ele alınır. Yörünge ve yol planlaması, robotun kinematikleri ve dinamikleri göz önünde bulundurularak bir arada gerçekleştirilir. Oluşturulan hareket planı çerçevesinde robotun kontrolü gerçekleştirilir. Kapalı hareket planlama yöntemleri bu basamakların eş zamanlı olarak gerçekleşmesine imkan tanıyacak nitelikte farklı teknolojilere ihtiyaç duymaktadır. Bu nedenle uygulaması her robot sistemi için mümkün olmamaktadır [31].

Bu tanım doğrultusunda mobil bir sistem için otonom yol planlaması, görev kapsamında hedefe ulaşmak için izlenmesi gereken güzergahı tanımlayan alt sistem olarak özetlenebilir. Otonom sistemin hedefe ulaşması için gerekli yolu planlarken görev süresince hedefe ulaşmayı engelleyecek durumların üstesinden gelmeyi de amaçlar. Hedef noktaya ulaşmak isteyen mobil bir araç için izlenilen seyrüsefer süresince kaşılaşılacak doğal veya yapay engeller olabilir. Bu engeller mobil robotun üstesinden gelebileceği özellikte ise, amaçlanan bu engellerin otonom olarak bertaraf

edilerek navigasyona devam edilmesi ve en etkin şekilde hedefe ulaşılması amacıyla izlenecek alternatif yolun planlanmasıdır.

İnsansız hava araçlarında otonom uçuş, kalkışından inişine kadar tanımlanmış tüm görev profilini, paterni belirlemenin ötesinde hiçbir insan etkisi olmaksızın kendiliğinden tamamlaması olarak tanımlanabilir [13].

İlk insansız araçlarda otonomi kullanılmamıştır. Bu İHA sistemleri daha çok uzaktan insan kontrolünde görev icra eden platformlar olmuşlardır. Teknolojinin hızla gelişmesiyle otonomi bir çok alanda olduğu gibi insansız araçlar üzerinde de kullanılmaya başlamıştır. Birçok gelişmiş ülke tehlikeli görevlerde insan kaybı olmadan görevlerin icrası için otonomi üzerine büyük çalışmalara başlamışlardır. Tam otonom bir sistemi elde edebilmek için öncelikle aşağıdaki ihtiyaçları karşılamak gerekmektedir [32];

a. Alıcı bilgileri; platformun etrafındaki çevre bilgilerinin ve platformun davranış bilgilerinin elde edilmesini sağlar,

b. Haberleşme altyapısı, insansız platformlar ile diğer sistem bileşenşeri arasında kesintisiz iletişimin olmasını sağlar,

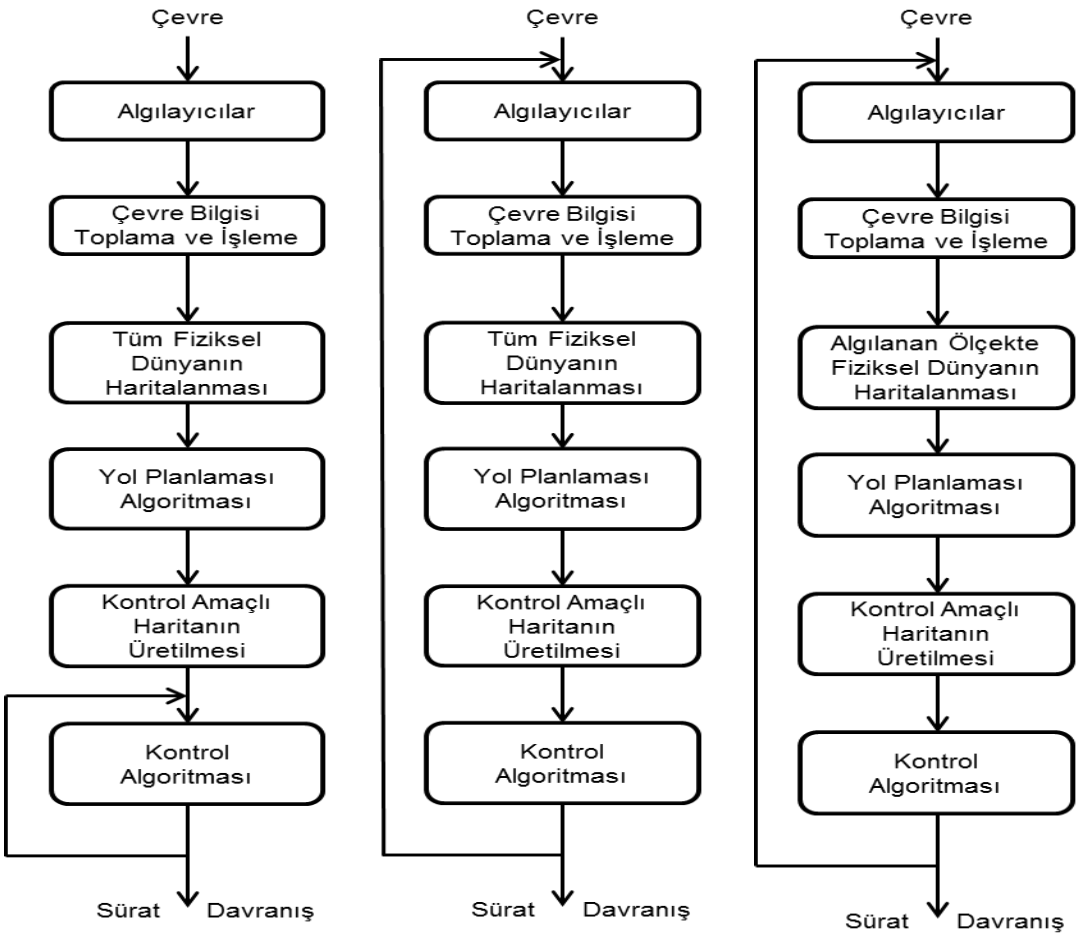
c. Yol planlama, engellerden sakınarak ve çarpışmadan kaçınarak görevin icrasi için en uygun yolun tanımlanması ve hedefe ulaşılmasını sağlar,

ç. Yol üretme, otonom aracın güvenli bir şekilde hedefe gittikten sonra geri gelebilmesini sağlar,

d. Yol düzenleme, hareket kısıtlılığının olduğu durumlarda en optimum kararları vererek görevin devamlılığını sağlar,

e. Görev tahsisi ve planlama, zaman ve ekipman açısından görevlerin planlanmasını ve görev planlama dahilinde yer alan her bir araç için ayrı ayrı gerçekleştirilmesini sağlar,

f. Çoklu görev planlaması, görev planlamalarında bazen birden fazla otonom araç görevlendirilebilir ve bu durumda çoklu görev planlaması takımdaki bütün araçların hem kendilerine tahsis edilen hem de grubun ortak görevlerini yapmak üzerine planlar, iletişimin kesilmesi gibi olağan dışı durumlarda görevlerin aksaksız yürütmesi için grup içerisinde görev değişimine ve yeniden planlamaya imkan sağlarlar.



(a) Durağan genel yol planlaması. (b) Dinamik genel yol planlaması. (c) Durağan veya dinamik yerel yol planlaması.

Şekil 2.13: Otonom sistemler için yol planlaması.

Yukarıda yer alan listede de görüldüğü üzere otonom bir sistemin temel bileşenlerinden birisi de yol planlamasıdır. Tanım olarak yol planlaması, yerel (local) ve genel (global) yol planlaması olarak iki farklı başlığa ayrılmaktadır [33]. Genel yol planlaması bulunulan noktadan varılmak istenen hedef noktaya ulaşmak için izlenmesi gereken yol planlamasını bir bütün olarak yaparken, yerel yol planlaması daha dar alanda bu yol planlamasının izlenmesi gereken alt evrelerinin aşama aşama planlanmasını ele almaktadır. Her iki yaklaşımda da engellerden sakınarak varılmak istenen hedef noktaya giden en uygun yolu bulmak amaçlanır. Ancak aralarındaki fark, yerel yol planlamalarının evrensel yol planlamalarına göre daha dar tanımlanmış alanlar için dinamik ve hızlı güncellenen yapılar olmasından ve gerçek zamanlı olarak hesaplanması gereksiniminden kaynaklanmaktadır.

İHA sistemlerinin diğer otonom sistemlere nazaran yol planlamaları, üç boyutlu bir ortam içinde sürekli hareketli ve yüksek süratli yapılar (sabit kanatlı platformlar için)

olmaları ve limitli manevra yetenekleri (minimum dönüş kütürü vb.) olması gibi nedenlerden dolayı nispeten daha zor kabul edilebilir. Bu nedenle çok daha fazla sayıda yol planlamasına etki eden faktör göz önünde bulundurulmalıdır.

Yol planlaması probleminin çözümü için değinilmesi gereken bir diğer nokta ise seyrüseferin icra edileceği ortam bilgisidir. Bu noktada farklı tanımlamalara gitmek mümkündür:

a. Özellikleri önceden bilinen ve bilinmeyen ortamlarda seyrüsefer ve

b. Durağan veya dinamik olarak değişen bir ortam içerisinde seyrüsefer, şeklinde bir gruplama yapılabilir.

Sınırları belli bir alan içerisinde otonom olarak hareket edecek olan araç için izlemesi gereken temel kontrol yapısı Şekil 2.13'de gösterilen şemalar şeklinde modellenebilir. Kontrol yapısının giriş bilgisi aracın anlık komunu ve algılayıcıları tarafından algılanabilen çevre koşulları olacaktır. Çıktı ise aracın hareketlerini belirleyecek olan sürat ve davranış bilgisi olacaktır. Şekil 2.13 (a) durağan bir alan içerisinde, Şekil 2.13 (b) dinamik bir alan içerisinde; genel yol planlamasını temel alarak hareket eden bir sistemi modellerken, Şekil 2.13 (c) durağan veya dinamik yerel yol planlaması doğrultusunda hareket eden bir sistemi temsil etmektedir.

Ayrıca ulaşılmak istenen hedef ile ilgili şu ayrımı yapmak, izlenecek olan yerel ve genel yol planlamasının yapılmasında kullanılacak yöntemin geliştirilmesi açısından önem arz etmektedir;

a. Hareketli ve hareketsiz hedeflere doğru seyrüsefer ve

b. Hareketleri önceden bilinen veya anlık olarak değişebilen bir hedefe doğru seyrüsefer.

Bu ayrımlar geliştirilecek olan gerçek zamanlı yol planlaması yaklaşımının başarısını doğrudan etkilemektedir. Örneğin;

a. Üç boyutlu sayısal haritalar ile önceden özellikleri bilinen, sınırlı bir coğrafi bölge üzerinde ve sadece durağan dağ ve/veya tepe gibi engellerden kaçınarak, statik olarak tanımlanmış hedef koordinata ulaşmak için gerçekleştirilen seyrüseferi amaçlayan yol planlaması,

b. Önceden engel tanımlamasının yapılmadığı ve uçuş esnasında algılayıcılar ile gerçek zamanlı olarak tespit edilen engellerin yer aldığı sınırlı bir arazide, önceden

hareketleri kestirilemeyen dinamik bir hedefin takibini amaçlayan bir seyrüseferi amaçlayan yol planlaması arasında ciddi tasarımsal farklılıklar olacaktır.

Şekil 2.13'de görüldüğü üzere, yol planlaması amacıyla en hızlı yanıt verebilecek olan sistem yaklaşımı; genel yol planlaması ile çözüme kavuşturulacak, çevre koşullarının başlangıç konumunda tümüyle algılanabildiği ve durağan olarak kabul edildiği, dolayısıyla koşulların hareket boyunca değişmediği "Durağan Genel Yol Planlaması" yaklaşımıdır. Yanıt süresi olarak en yavaş sonuç üretecek olan yaklaşım ise; çevre koşullarının sürekli olarak değişebildiği ve hareket alanının tamamında bu değişimin algılanabildiği "Dinamik Genel Yol Planlaması" yaklaşımıdır. Ancak dinamik çevre koşulları altında en uygun (optimum) ve sorunsuz çözümü de yine "Dinamik Genel Yol Planlaması" yaklaşımı üretecektir.

Bu tez çalışması kapsamında geliştirilmek istenen yol planlaması yaklaşımı aşağıdaki problemlere yanıt verebilecek bir çözüm sunmayı amaçlamaktadır:

a. Önceden engel tanımlamasının yapılmadığı ve uçuş esnasında algılayıcılar ile gerçek zamanlı olarak tespit edilebilen engellerin yer aldığı (arazide yer alan dağ veya tepeler gibi) sınırlı üç boyutlu bir ortamda, birden fazla otonom İHA sisteminin ve/veya insanlı sistemin bir formasyon oluşturacak şekilde, önceden hareketleri bilinen veya kestirilemeyen hareketli bir hedefi takip etmesine imkan tanıyacak otonom dinamik yerel yol planlaması,

b. Önceden durağan engel tanımlamalarının yapıldığı (fiziki engeller gibi) ve ilaveten uçuş esnasında hareketli diğer engellerin davranış bilgilerinin de gerçek zamanlı alınabildiği (diğer hava araçlarının konum bilgisi gibi) sınırlı üç boyutlu bir ortamda, birden fazla otonom İHA sisteminin ve/veya insanlı sistemin bir formasyon oluşturacak şekilde, önceden hareketleri bilinen veya kestirilemeyen hareketli bir hedefi takip etmesine olanak tanıyacak otonom dinamik genel yol planlaması.

Otonom yol planlama problemlerinin büyük bir kısmı, zaman ve yönlendirme kısıtlarından dolayı, NP-hard problemlerdir [34]. Bu yaklaşımlar literatürde her ne kadar sezgisel ve klasik yaklaşımlar olarak sınıflandırılsalar dahi bu sınıflandırmayı genişletmek mümkündür [34]. Klasik algoritmalar eğer varsa, en optimal çözümü hesaplamak veya hiçbir uygulanabilir yol olmadığını kanıtlamayı hedeflemektedirler. Ancak, klasik yaklaşımların çoğu özgür yapılandırma alanı (free configuration space - CSpace) yani engellerden sakınarak hedefe ulaşma, kavramında zorluklar ile

karşılaşmaktadır. Özellikle dinamik ortamların ihtiyaç duyduğu "uyarlanabilirlik" (adaptivity) ve "sağlamlık" (robustness) özelliklerinin eksikliği, yeni bir çözümü sıfırdan elde etme ihtiyacını doğurabilir ve bu durum dinamik ortamlar için uygun değildir [35]. Öte yandan, sezgisel algoritmalar kısa zamanda kaliteli bir çözüm arayışı bulmak girişimini güderler. Ancak sezgisel algoritmaların da güçlüğü dinamik probleme bağlı olarak "kilitlenme" (deadlock) ve "salınım" (oscillation) durumlarının kolaylıkla gerçekleşmesi olduğu için, iyi bir çözüm bulmakta başarısız olabilirler [35]. Bu genel sınıflandırmanın dışında yol planlaması amaçlı çözüm yaklaşımları genel olarak üç temel grupta toplanmaktadırlar [36]:

a. Hesaplama tabanlı yaklaşımlar (Calculus-based Approaches)
b. Geometri tabanlı yaklaşımlar (Geometry-based Approaches)
c. Vektör tabanlı yaklaşımlar (Vector-based Approaches)

Bundan sonraki bölümlerde sırasıyla bu yaklaşımlar incelenmiş ve birbirlerine göre otonom yol planlaması probleminin çözümü açısından avantaj ve dezavantajları oratay konmuştur.

2.4.1 Hesaplama tabanlı yaklaşımlar

Hesaplama tabanlı yaklaşımlar en optimum yol planlamasını ortaya koymayı amaçlayan yaklaşımlardır. Ancak hesaplama tabanlı yaklaşımlar ihtiyaç duydukları yoğun işlemci gücü ve hızı nedeniyle gerçek zamanlı çalışma ihtiyacı duyan yol planlaması çözümlerinde uygulanmaları oldukça zor yaklaşımlardır.

Evrimsel algoritmalara dayanan çözümler, hesaplama tabanlı yaklaşımlara bir örnektir. Rathbum ve arkadaşları tarafından ortaya konulan [37] evrimsel algoritma yaklaşımı sınırlı bir alan içerisinde çarpışmayı önler nitelikte bir yerel yol planlaması çözümünü ortaya koymaktadır. Yaklaşım 20 kadar olası yol planını çözüm popülasyonu olarak ele almakta ve mutasyon ile yeni alternatif yollar üretilmesini sağlamaktadır. Platformun hareket yeteneklerine, çevre koşullarına, rota üzerindeki kısıtlara vb. bağlı bir maliyet fonksiyonuna istinaden üretilen alternatif yollar değerlendirilmektedir. En yüksek skor değerine sahip belirli sayıdaki yol bir sonraki nesli oluşturacak şekilde seçilmektedir. Bu süreç yaklaşık olarak yeni 50 nesil üretecek şekilde iteratif olarak gerçekleştirilmekte ve en iyi yol belirlenmeye çalışılmaktadır. Her ne kadar rastlantısal değerlerin yer aldığı bir yaklaşım olsa dahi

bu yöntem oldukça etkin sonuçlar üretebilmektedir. Ancak gerek rastsal değerlerin üretilmesinin maliyeti, gerekse de iteratif bir yaklaşım olması ve ayrıca çok sayıda alternatifin bir arada değerlendirilmesi, gerçek zamanlı bir uygulamada oldukça yüksek bir hesaplama gücü ve hızı ihtiyacına gereksinim duymaktadır ki bu durum uygulanması güç bir sonuç yaratmaktadır.

2.4.2 Geometri tabanlı yaklaşımlar

Geometri tabanlı yaklaşıma örnek olarak Alejo ve arkadaşları [38] veya Meng ve arkadaşları [39] veya Xia ve arkadaşları [40] tarafından ortaya konulan ızgara-tabanlı (grid-based) yaklaşımlar örnek olarak verilebilir. Bu yaklaşımlar uçuş uzayını sabit büyüklükteki ızgara şeklinde alt bileşenlere böler ve A* ve/veya Dijkstra [41] gibi graph tabanlı algoritmalar ile en uygun yolu bulmak için işler [42]. Yapılan ilk işlem ızgara şeklinde uçuş alanını bölümlere ayırmaktır ve daha sonra bu alt bölümler alan içerisinde yer alan engeller ve diğer araçların konumları girilerek işaretlenir. Bunlar dışında kalan alanlar ise gezilebilir alanlar olarak belirlenir. Hedef konum bulunduktan sonra mevcut konumdan hedefe ulaşan en uygun yol graph gezilerek bulunmaya çalışılır [42]. Başlangıç noktasından itibaren uygun komşu noktalara geçişler yapılarak hedefe ulaşan en uygun yol tespit edilir. Bu yaklaşımlar hesaplama tabanlı yaklaşımlara istinaden daha hızlı ve neredeyse aynı etkinlikte sonuçlar üretebilir. Ancak performans ile ters orantılı fakat en uygun yolun bulunması ile doğru olan bir nokta, belirlenecek olan sabit büyüklükteki ızgaranın ne hassasiyet ile seçileceği ve dinamik ortamlarda nasıl güncelleneceğidir. Çünkü temsil edilecek olan graph çözünürlüğü çözümün kalitesi ve performansı açısından oldukça belirleyicidir. Bir diğer nokta ise çözüm komşuluk ilişkisindeki düğümleri gezen bir yaklaşım olarak ele alındığından platformun icra edemeyeceği çözümleri içermesi mümkündür. Örneğin ortaya konulan sonuç çözüm ızgaranın bir karesinde 90^{o} derecelik bir baş değişikliğini içeren bir manevraya ihtiyaç duyabilir ancak platform bu manevrayı istenilen alan dahilinde gerçekleştiremeyebilir. Ayrıca bu durumun önüne geçmek için ızgara bileşen alanlarını büyütmek olası çözümlerin birçoğundan vazgeçmek anlamına gelebilir. Bu çözüm kapsamında oluşturulacak olan ızgaranın çözünürlüğü ciddi bir optimizasyon maliyeti getirmektedir.

Tüm bu problemlerin ötesinde ortam dahilinde oluşabilecek olan her türlü değişiklik (hedef noktanın dinamik olması, mobil engellerin var oluşu gibi) tüm değerlerin

yeniden yerleştirilmesine ve sonucun tekrar hesaplanmasına ihtiyaç duyacaktır. Bu nedenle yaklaşımın hesaplama maliyeti oldukça yüksek olacaktır.

2.4.3 Vektör tabanlı yaklaşımlar

Literatürde yaygın olarak yol planlaması amacıyla faydalanılan yaklaşımlardan bir diğeridir. Vektör tabanlı yaklaşımlara örnek olarak en yaygın şekilde kullanılan YPA yöntemi verilebilir. Potansiyel alanlar engellerden kaçınırken hedefe giden yolları planlamada yaygın olan tekniklerinden birisidir. Bu fikir ilk olarak, çekici potansiyel olarak bir hedef ve itici potansiyel olarak da durağan engelleri kullanarak, mobil robot kontrolünü tasarlayan Khatib tarafından öne sürülmüştür [12]. YPA teknikleri geometri tabanlı yaklaşımlardakine benzeyen ızgara (grid) tabanlı bir yaklaşımdır. Ancak ayrıldıkları nokta, oluşan hücrelerin her birinin geometri tabanlı yaklaşımlara örnek olan A* veya Dijkstra benzeri yaklaşımlardaki gibi bağımsız olarak değerlendirilmeyişidir. Bir başka tanımla, ızgara içindeki her bir düğüm tek tek ele alınmak yerine, genel bir fonksiyon ile topyekün hesaplanabilir. Hareketi modellenen alanın genelinde bütün bir akış olarak planlamayı hedefleyen YPA yaklaşımı manyetik alan modellemesine ya da gaz yayılması modellemesine benzerlikler göstermektedir [43]. Basit anlamda potansiyel alan metodu, küçük ölçekler için kolay uygulanabilir ve çok değişikliğe ihtiyaç duymadan kabul edilebilir sonuçlar sağlayabilir. YPA metodu ile yol planlamasının en büyük avantajlarından birisi yol planlaması ve kontrol yaklaşımını, dolayısıyla bu iki süreci birleştirerek top yekûn bir çözümü tek seferde üretebilmesidir. Benzetme tabanlı bir yaklaşım olması hesaplama maliyeti açısından önemli avantajlar katmaktadır.

2.4.4 Yol planlaması yaklaşımlarının karşılaştırılması

Hesaplama tabanlı yaklaşımlara How ve arkadaşları [44][45] tarafından ortaya konulan Mixed Integer Lineer Programming tabanlı yaklaşım ya da geometri tabanlı yaklaşımlara Han tarafından ortaya konulan algoritmalar [46] örnek gösterilebilir ve bu sayı daha da arttırılabilir. Ancak neredeyse tüm bu yaklaşımlarda yerel yol planlaması için yukarıda sayılan gerekçelere benzer nedenlerle problemler üretilebilmekte ve gerçek zamanlı uygulamalarda yüksek hesaplama gücü ihtiyaçları nedeniyle özellikle yüksek süratli ve agresif hareket yeteneğine sahip araçlar için etkin olarak kullanılabilir çözümler sunmaktan uzaklaşmalarına neden olmaktadır. Geometri tabanlı yaklaşımların ürettikleri seri yön değişimi gibi sonuçların, limitli ve

sürekli manevra yeteneklerine sahip İHA sistemlerinde uygulanması güç olarak değerlendirilmektedir. Ancak vektör tabanlı yaklaşımlar, sürekli bir hareket modellemesi izlemeleri ve matematik yoğunluğundan çok benzetim tabanlı modeller olduklarından İHA sistemleri için daha uygun yöntemler olarak değerlendirilmektedir [47][48]. Özellikle gerçek zamanlı veri üretilmesi ihtiyacı duyan ve özellikleri önceden bilinmeyen alanlar için kullanılacak olan yerel yol planlaması çözümlerinde vektör tabanlı yaklaşımlar sıkça kullanılan tekniklerdir.

Bu noktadan hareket ile aşağıda sıralanan ve tez önerisi kapsamında problem olarak tanımlanan dar boğazların çözüme kavuşturulması, ortaya konulacak olan otonom formasyon kontrolü yaklaşımının etkinliğini belirleyecektir;

a. Otonom formasyon kontrolü yaklaşımı değişik sayıda platformun (homojen veya heterojen özelliklerde sistemlerin) bir arada ve istenen formasyonu muhafaza edebilecek nitelikte hareketine imkân sağlayacak özellikte olmalıdır (tek model ile heterojen platform desteği özelliği).

b. Otonom formasyon kontrolü yaklaşımı belirli kurallar ile tanımlanmış bir seyrüsefer yaklaşımını doğal olarak destekleyebilmelidir (Seyrüsefer desteğini kendiliğinden verebilme özelliği).

c. Formasyon uçuşunu sağlayan kontrol yaklaşımı değişik formasyon tiplerini (yakın formasyon, esnek formasyon, sıralı ve V formasyon vb.) destekleyebilmeli ve bu tipler arasında istendiğinde geçişe izin verebilecek nitelikte olmalıdır (Esnek olma özelliği).

ç. Formasyon uçuşu özellikle platformların dar bir alanda ve bir arada seyir halinde olmalarından dolayı otonom formasyon kontrolü yaklaşımı çarpışmayı önleme gibi güvenlik özelliklerini bünyesinde doğal olarak taşımalıdır (Dinamik modelleme özelliği).

d. Otonom formasyon kontrolü yaklaşımı seyrüsefer esnasında karşılaşılabilecek doğal ve yapay dinamik engellerden etkilenmeyecek ve bu duruma gerçek zamanlı reaksiyon gösterebilecek nitelikte esnek olmalıdır (Gerçek zamanlı dinamik modelleme özelliği).

e. Otonom formasyon kontrolü yaklaşımı platformların her birisine gerçek zamanlı kontrol imkânı tanıyabilecek ve uygulanabilir çözümler üretmelidir.

Platform kabiliyetlerine bağlı olarak esnek ve uyumlu sonuçlara imkân sağlamalıdır. (Gerçek zamanlı esnek hesaplanabilme özelliği)

Tanımlanan bu ihtiyaçları yukarıda belirtilen yöntemlerden hangisinin ne derece karşıladığı problemin çözümü için belirlenecek yaklaşımın seçiminde belirleyici olmuştur. Bu nedenle probleme yönelik aranan niteliklerin hangi yaklaşım ile ne derece karşılandığı ortaya konulmaktadır.

Çizelge 2.3: Literatürdeki çözüm yaklaşımlarının karşılaştırması.

Karşılanmak İstenen Özellik	**Yol Planlaması Yöntemi**		
	Hesaplama Tabanlı Yaklaşımlar	**Geometri Tabanlı Yaklaşımlar**	**Vektör Tabanlı Yaklaşımlar**
Heterojen Platform Desteği	Hayır	Evet	Evet
Doğal Seyrüsefer Desteği	Evet	Hayır	Evet
Esnek Olma Özelliği	Hayır	Hayır	Evet
Dinamik Modelleme Özelliği	Hayır	Evet	Evet
Gerçek Zamanlı Dinamik Modelleme Özelliği	Hayır	Evet	Evet
Gerçek Zamanlı Esnek Hesaplanabilme Özelliği	Hayır	Hayır	Evet

Bu kapsamda yukarıda tanımlanan yaklaşımların belirlenen problem özelliklerine ne derece cevap verebildikleri literatürdeki örnekleri incelenerek Çizelge 2.3 ile gösterilmiştir. Problem olarak tanımlanan ve çözüm için aranan özelliklere yaklaşımların ne derece yanıt verebildikleri karşılaştırmalı olarak ortaya konulmuştur. Bu sonuçların da ortaya koyduğu gibi gerçek zamanlı dinamik bir kontrol probleminin etkin olarak çözüme kavuşturulması için faydalanılması planlanan yaklaşım vektör tabanlı bir yaklaşımdır.

Vektör tabanlı yaklaşımlara literatürde verilebilecek bir örnek de YPA tabanlı yol planlaması ve otonom kontrol yapılarıdır. Bu tez çalışması kapsamında YPA tabanlı bir çözüm yöntemi seçilmesinin temel nedeni budur.

Ancak problemin boyutları büyüdükçe klasik YPA tabanlı yaklaşımların hesaplama performansları gerçek zamanlı çözümlerin üretilmesi açısından yetersiz olabilirler.

Ayrıca dinamik ortamlar hesaplanan yol planlanmasının tekrarlanmasına dolayısıyla işlem yükünün artmasına neden olabilirler. Bu nedenle tez kapsamında hesaplama performansını geliştirecek nitelikte paralel algoritmaların geliştirilmesine ve de bu paralel algoritmaların GPGPU yaklaşımından faydalanılarak uygulanmasına değinilmiştir [49].

2.5 Hesaplama Mimarileri

Yol planlaması amacıyla kullanılan olan yöntem ne olursa olsun bir matematiksel model temelinde hesaplanması gerekecektir. Bu bağlamda ortaya konulan yöntemin hesaplanması amacıyla farklı mimarilerde hesaplama donanımlarına ihtiyaç duyulacaktır. Hesaplama donanımları olarak işlemciler, bellek donanımları ve veri yolları sayılabilir. Bu donanımların hesaplama amacıyla kullanımı temelde iki alt kategoriye ayrılmıştır. Bunlar seri ve paralel hesaplama mimarileri olarak adlandırılmaktadır. Seri mimariler hesaplamayı belli bir sıra dahilinde her bir süreci ve süreç dahilinde ortaya konulan işlemleri tek tek ele alarak işlerken, paralel mimariler birden fazla süreci veya süreç dahilindeki bağımsız işlemi aynı anda işleyebilirler.

Çalışma kapsamında ortaya konulan yaklaşımların, uygulama alanı olarak İHA platformları gibi yüksek süratte agresif hareket edebilen otonom sistemler seçilmiş olması nedeniyle mümkün olduğunca gerçek zamanlı çözümler üretmesi beklenmektedir. Bu nedenle geliştirilen yaklaşımların hesaplama performansları seri ve paralel hesaplama mimarileri üzerinde ölçülecektir. Geliştirilen yaklaşım kapsamında ortaya konulan algoritmalar seri ve paralel hesaplama mimarileri üzerinde koşturulacak şekilde geliştirilecektir. Bu nedenle öncelikle seri ve paralel hesaplama mimarileri tanımlanacaktır.

2.5.1 Seri hesaplama mimarileri

Çalışma kapsamında ortaya konulan yöntemler, seri hesaplama mimarisine uygun olarak geliştirilen algoritmaların uygulanmasıyla sınanmış ve hesaplama performansları değerlendirilmiştir. Geliştirilen algoritmalar Çizelge 2.4 ile gösterilen seri hesaplama sistemi üzerinde hesaplama performansı açısından sınanmıştır.

Çizelge 2.4: Seri hesaplama mimarisi donanım özellikleri.

Donanım	Özellik
İşlemci	Intel® Core™ i7-3630QM CPU @2.40GHz (6M Cache, up to 3.40 GHz)
Bellek	32 GB DDR3 800MHz
Veri Yolu	Intel Ivy Brigde
İşletim Sistemi	MS Windows 7 ® Ultimate 64 bit
MATLAB Versiyonu	MATLAB R2013b
C++ Versiyonu	Microsoft Visual C++ 2010 (v.10.0)

Çizelge 2.4'de de gösterildiği şekilde seri hesaplama donanımı olarak Intel tabanlı işlemciler ve DDR3 mimarisinde standart bellek mimarileri seçilmiştir. Seçilen seri hesaplama donanımı ticari olarak kolay temin edilebilen, halihazırda İHA sistemlerinde kullanılan veya kullanılabilinecek donanımlar arasından belirlenmiştir.

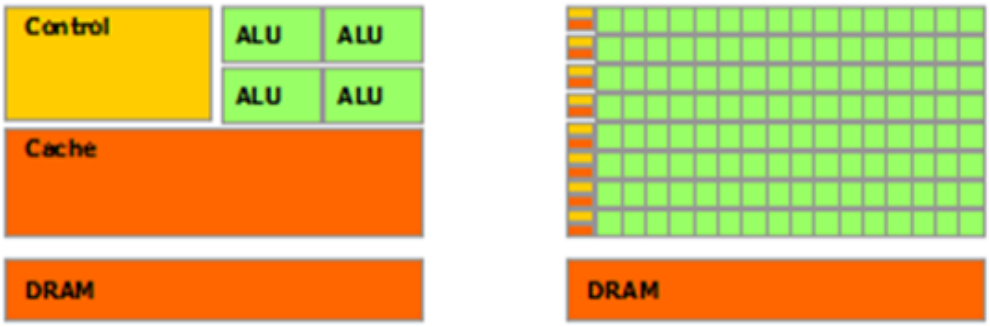
2.5.2 Paralel hesaplama mimarileri

Seri hesaplama mimarisine uygun olarak geliştirilen algoritmalar, paralel hesaplama mimarilerine uygun olarak tekrar güncellenmiş ve de farklı paralel hesaplama donanımları üzerinde hesaplama performansı açısından değerlendirilmek ve yaklaşımın başarısı ölçülmek üzere sınanmıştır. Çalışmanın ilerleyen bölümlerinde açıklandığı şekilde geliştirilen algoritmalar yaklaşımın SIMD tipinde paralel programlama mimarisine uygun olduğu değerlendirildiğinden ve son yıllarda popüler paralel programlama mimarileri arasında yer alan grafik işlemciler üzerinde sınanmıştır. Grafik işlemcilerin paralel programlama mimarisi olarak seçilmesinin temel iki nedeni vardır; bunlardan ilki, grafik işlemcilerin seri hesaplama amacıyla faydalanılan klasik işlemcilerden çok sayıda işlemcinin bir arada kullanılması yöntemine nazaran grafik işlemcilerin daha maliyet etkin çözümler üretmesi ve İHA sistemlerinde güç ihtiyacı, ağırlık ve ölçekleri açısından daha kullanılabilir olmasıdır.

2.5.3 Grafik işlemcilerin paralel programlama amacıyla kullanımı

Gerçek zamanlı grafik uygulama programlama ara yüzleri (Graphic APIs), bir nesnenin yüzeylerini temsil etmek için temel olarak noktaları, çizgileri ve üçgenleri kullanırlar. Grafik işleme hattı (Graphic Processing Pipeline) bu temsili, monitör üzerinde gösterebilmek için piksel tabanlı dizilere dönüştürür. Bu dönüşüm shader fonksiyonları adı verilen bir dizi kısa komuttan oluşan fonksiyonlar tarafından grafik işlemci üzerinde paralel olarak yürütülür. Paralel olmaları sayesinde dizi üzerinde yer

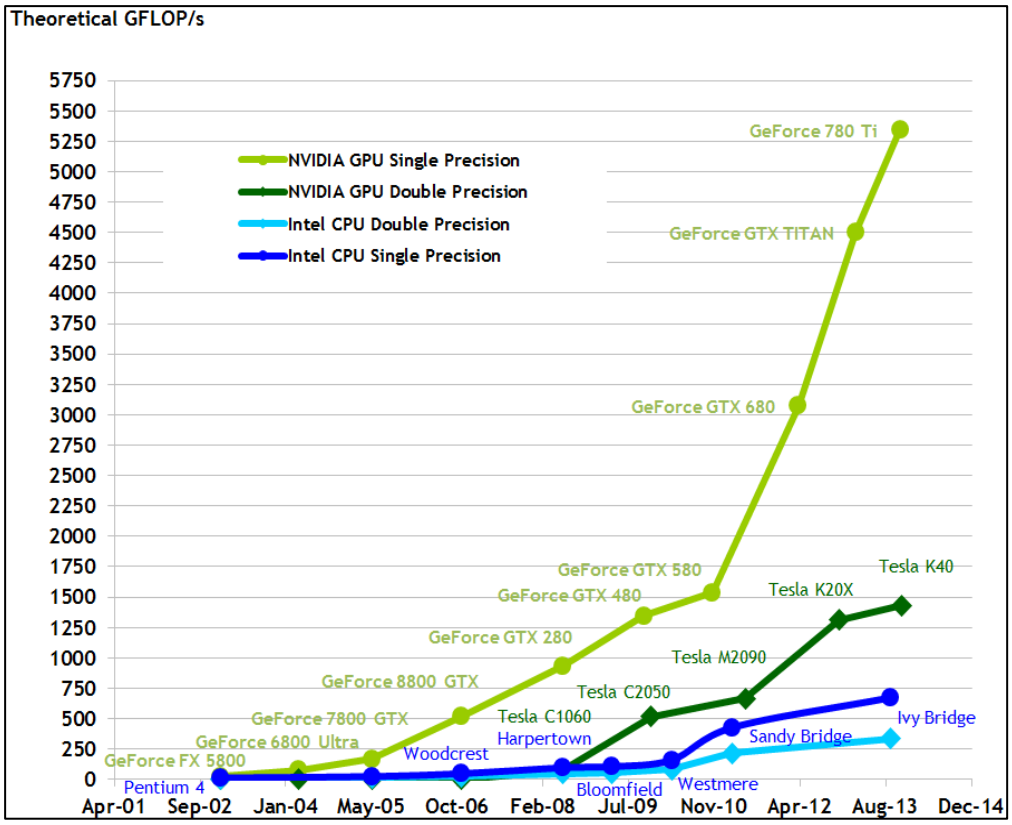
alan birden fazla piksel değerini aynı anda hesaplama imkânına sahiptirler. Bu paralel işlem bir nevi SIMD tipinde paralel bir uygulamadır.



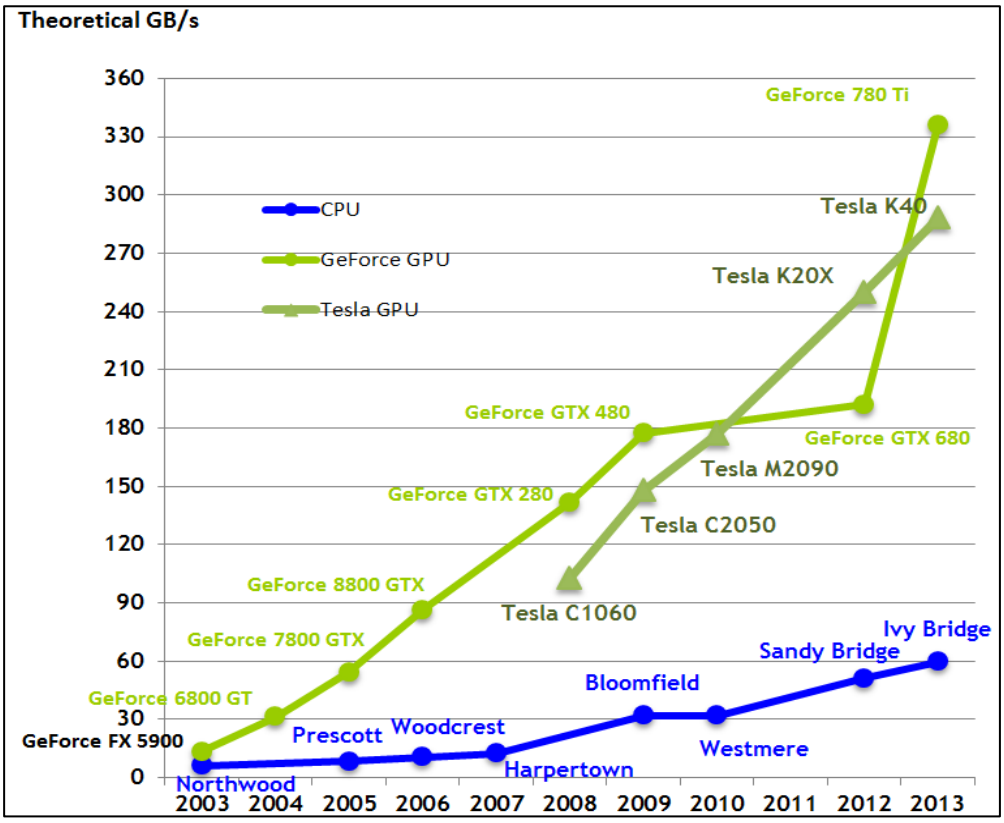
(a) CPU mimarisi. (b) GPU mimarisi.

Şekil 2.14: CPU ve GPU mimarilerinin karşılaştırılması [50].

Şekil 2.14 ile gösterildiği biçimde bir grafik işlemcisinde bu tip paralel işleme imkân tanıyacak klasik merkezi işlemci birimi (Central Processing Unit – CPU) mimarilerine nazaran çok daha fazla sayıda Aritmetik Mantıksal Hesaplama Ünitesi (Aritmethic Logic Unit – ALU) içeren çekirdek (core) yapıları yer almaktadır. Her ne kadar GPU bünyesinde yer alan ALU birimlerinin hesaplama hızı CPU dahilinde yer alan ALU birimlerine göre nispeten düşük olsa da SIMD tipinde bir paralel uygulamada performans sonuçlarını etkileyecek olan bir diğer etken de kullanılabilir ALU sayısıdır. SIMD yapısına uygun şekilde geliştirilmiş geleneksel algoritmaların (sıralama, arama vb.) paralel şekilde GPU donanımları üzerinde koşturulması sonucu elde edilen GFOLP/s sayısı Şekil 2.15 (a) ile gösterildiği biçimde CPU performansına nazaran yaklaşık beş kat daha iyi olabilir [51].



(a) Hesaplama hızı.



(b) Bellek erişim hızı.

Şekil 2.15: CPU ve GPU karşılaştırılması [51].

Bir tek resim karesi oldukça fazla sayıda piksel verisini içermektedir ve dolayısıyla GPU tüm bu veriyi paralel şekilde işlemek için tasarlanmıştır. Bu nedenle veri erişiminin bir dar boğaz oluşturmasını engellemek maksadıyla çok iplikli (multi-threading) yapıya müsaade edecek şekilde yüksek bant genişliğine sahip hızlı bellek mimarileri ile donatılmışlardır. Bellek erişim hızı GPU üzerinde koşturulan klasik algoritmalar (sıralama, arama vb.) için CPU üzerinde koşturulan uygulamalara nazaran Şekil 2.15 (b) ile gösterildiği biçimde yaklaşık dört kat daha iyi sonuçlar üretebilir [51].

Şekil 2.14 ile gösterildiği gibi, donanım mimarisi olarak CPU ve GPU karşılaştırıldığında, GPU'ların merkezi işlemcilere kıyasla [51];

a. Veri işleme amaçlı çok daha fazla sayıda transistöre sahip oldukları,

b. Geliştirilen kodun bellek paylaşımlı haberleşebilen yüzlerce çekirdek üzerine binlerce iplik oluşturularak yayılabileceği,

c. GPU üzerinde gerçekleştirilen işlemlerin CPU'dan tümüyle bağımsız olarak koşturulabileceği görülmektedir.

Ancak tüm bunların yanında CPU'ların da grafik işlemcilere nazaran [51];

a. Daha büyük ve hızlı ön bellek mimarilerine sahip oldukları,

b. Program içerisinde dallanma yeteneklerinin daha uygulanabilir olduğu,

c. Çekirdek hızlarına bakıldığında daha performanslı işlemler gerçekleştirebildikleri söylenebilir.

GPU'nun bu eksiklikleri sahip oldukları çok sayıda ALU, daha hızlı dahili (on-board) bellek üniteleri ve paralel mimarileri ile giderilmeye çalışılmıştır. Sonuç olarak bu özelliklere istinaden şöyle bir kanıya varmak doğru olacaktır; CPU mimarileri görev paralellik (task parallelism) için ideal platformlar ve GPU mimarileri veri paralelliği (data parallelism) için ideal platformlar olarak kabul edilebilirler.

Veri paralel algoritmalar, GPU mimarilerinin aşağıdaki özniteliklerinden faydalanılarak performanslarını maksimize edebilirler;

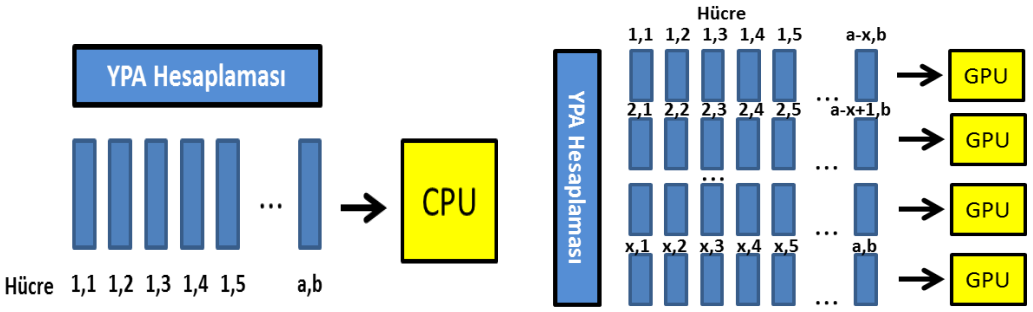
a. Büyük veri dizileri, yüksek veri akışı hacmi,

b. İyi ayrılmış veri tanımlaması ve hızlı veri erişimi ile SIMD paralellik,

c. Düşük gecikme süresi ile Kayan Nokta (Floating Point- FP) işlem hesaplama yeteneği.

Çizelge 2.5: Seri ve paralel hesaplama karşılaştırılması .

Seri Hesaplama (CPU üzerinde)	**Paralel Hesaplama (GPU üzerinde)**
1. Tek bir merkezi işlemci üzerinde tüm hesaplama algoritması koşturulur. 2. Komutlar (instruction) aşağıdaki şekilde gösterildiği gibi ardışıl (sıralı) olarak koşturulur. 3. Her bir saat vuruşunda (Clock Cycle) tek bir komut koşturulur.	1. Çok çekirdekli bir grafik işlemci üzerinde hesaplama algoritması koşturulur. 2. Komutlar aşağıdaki şekilde gösterildiği gibi grafik işlemci üzerindeki çekirdek sayısı kadar aynı anda paralel olarak koşturulur. 3. Her bir saat vuruşunda çok sayıda komut koşturulur.



Bu durumda uygulama geliştirme esnasında GPU ve CPU donanımlarının avantajları Çizelge 2.5 ile gösterildiği biçimde gerçekleşecektir.

Grafik işlemciler düşük güç tüketiminin yanı sıra fiziksel olarak küçük ebatlarda olan ve mobil olarak kullanılabilen paralel programlama donanımı özellikleri sağlayan maliyet etkin yapılar haline gelmişlerdir. Tüm bu performans özelliklerinin yanında grafik donanımları üzerinde genel maksatlı paralel uygulamaların geliştirilmesi bir takım sıkıntıları da içermektedir [51]. Örneğin grafik işlemcileri üzerinde paralel bir programlama geliştirmek isteyen programcı kesinlikle tüm özel kütüphanelere hakim olmalı ve donanım mimarisini çok iyi bilmelidir. Çünkü her bir grafik işlemcisinin kendisine has ve standart olmayan özellikleri vardır. Geliştirilen paralel uygulamanın performansı doğrudan üzerinde koşturulacağı donanımın yeteneklerine bağlıdır. Bu yetenekler paralel iplik (thread) sayısı, komut seti sayısı, depolanabilen ve işlenebilen veri miktarı, bellek mimarileri ve aralarındaki transfer performansı gibi

sıralanabilir ve bu sayı daha da arttırılabilir. Bu durum doğrudan paralel uygulamanın performansına etki edecektir. Bir başka değiş ile geliştirilen paralel algoritmalar doğrudan donanım özellikleri temel alınarak donanıma uygun geliştirilmeli ve dolayısıyla donanım değişikliği durumunda donanım özelliklerine bağlı olarak tekrar optimize edilmelidir. Sonuç olarak GPU üzerinde geliştirilen paralel uygulamalar için platforma bağımlılık donanımsal olarak en üst seviyededir. Ayrıca GPU üzerinde yer alan bellek mimarileri rastgele erişimli bellek okuma ve yazma işlemlerini desteklemediklerinden dolayı program modelinde oldukça yüksek seviyede veri erişim sınırlamalarını beraberlerinde getirmektedir. Bu nedenle geliştirilen paralel uygulama dahilinde detaylı bir bellek erişim yöntemi tanımlanarak izlenmelidir.

Önde gelen grafik işlemci üreticileri olan NVIDIA, ATI ve INTEL markaları son yıllarda görüntü işlemenin ötesinde grafik işlemcilerin kayan nokta hesaplaması yapılmasına imkan tanıyarak programlanabildikleri ve dolayısıyla paralel algoritmaların koşturulabilmesine imkan sağlayacak GPGPU teknolojileri geliştirmektedirler. Bu yaklaşımların Open Computing Language (OpenCL) [52], ATI STREAM veya Yüksek Performanslı Hesaplama (High Performance Computing – HPC) [53] ve NVIDIA Compute-Unified Device Architecture (CUDA) [54] olarak sayılabilir.

2006 yılının Kasım ayında, NVIDIA, CUDA adı verilen genel amaçlı paralel programlama işlem platformu tanıttı ve GPU üzerinde daha verimli bir şekilde birçok karmaşık bilgi işlem problemini paralel olarak çözmek için NVIDIA GPU'ların paralel hesaplama motorunu güçlendirdi [18]. CUDA, grafik işlem biriminin gücünden yararlanarak, hesaplama performansında önemli artışlara olanak veren NVIDIA tarafından geliştirilmekte olan paralel hesaplama mimarisidir. CUDA, genel maksatlı hesaplamaların grafik işlemcisi üzerinde gerçekleştirilmesi amacıyla CPU'dan zaman bağımsız olarak ve yüzlerce çekirdek üzerinde binlerce iplik kullanarak paralel hesaplamaya olanak veren son nesil NVIDIA ekran kartlarında doğrudan desteklenen bir paralel yazılım geliştirme desteğidir.

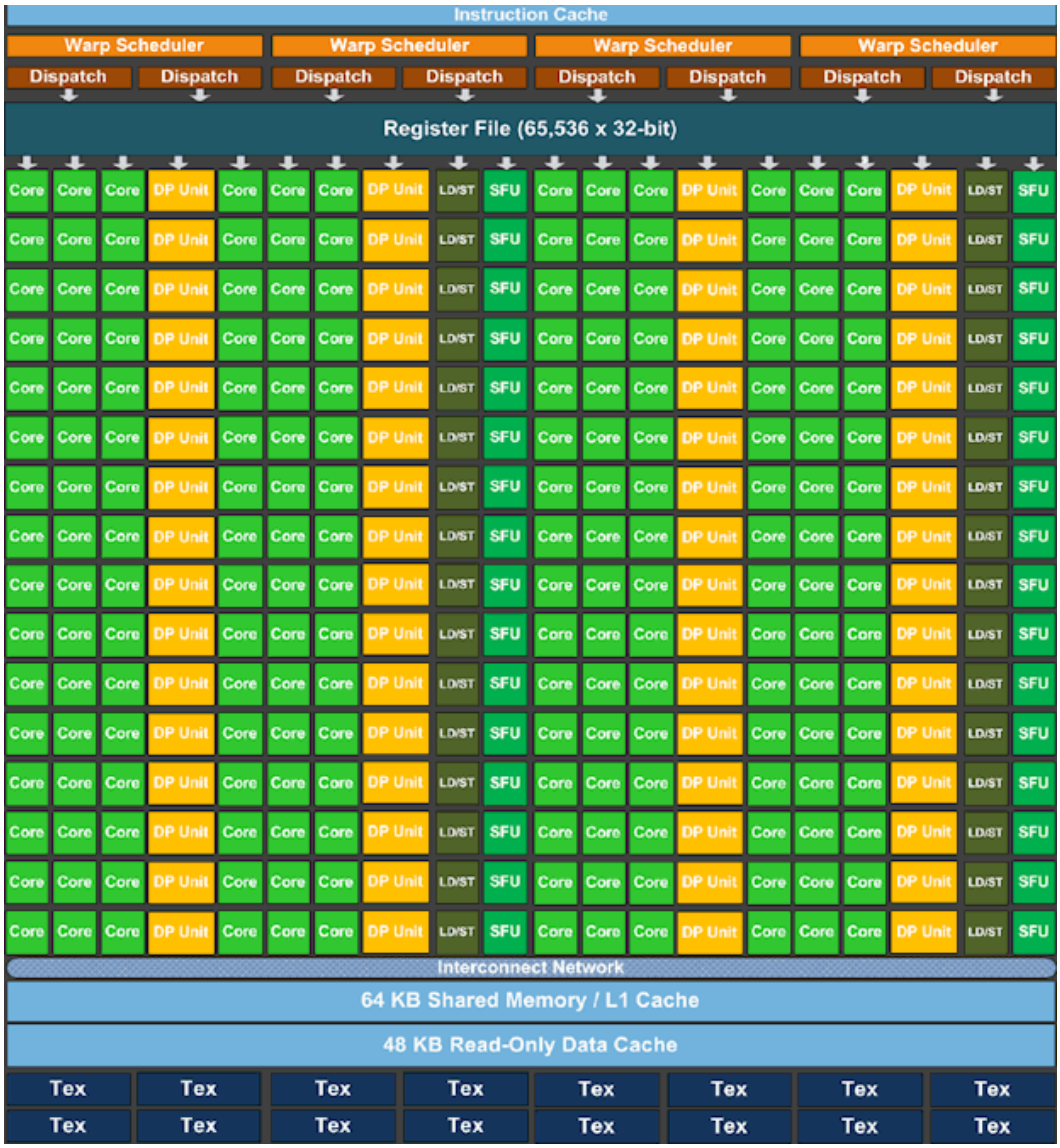
CUDA ile paralel yazılım geliştirme sürecinde donanım özelliklerine bağlı olarak üzerinde durulması gereken iki ana unsur yer almaktadır;

a. Kaç adet paralel koşturan eş zamanlı iplik oluşturulacağı ve bunların organizasyonu (iplik sayısı, blok sayısı ve grid sayısı olarak),

b. İplik haberleşmesinde hangi tip bellek mimarisinden faydalanılacağı ve verinin işlenmesi için ipliklerin ve grupların nasıl organize edileceği.

GPGPU programlama performansının doğrudan donanım mimarisine bağlı olduğu açıktır. Bu nedenle aynı kod yapısı farklı donanım mimarileri üzerinde değişik performans sonuçları üretecektir. CUDA programlama dili C-tabanlı bir programlama dili türü olmasına rağmen, görsel çıktıları üretmek oldukça zordur. Görsel grafikler üreterek matematik fonksiyonlarını analiz etmek CUDA ortamında hazır kütüphaneler ile desteklenmediğinden kolay değildir.

CUDA, C programlama dili ile aşina geliştiricilerin kolayca GPU cihazı tarafından yürütülebilecek programlar yazmaları için basit bir yol sağlayan, yüksek seviyeli bir programlama dili olarak C tabanlı bir yazılım ortamı sağlamaktadır. CUDA ile geliştirilen C programı sadece bir kez çağrıldığında düzenli olarak bağımsız C fonksiyonları gibi, farklı *n* adet CUDA çekirdeği tarafından paralel şekilde *n* kez çalıştırılır, yani çekirdek adı verilen komut setleri bir grup olarak C fonksiyonlarını yürütür. CUDA komutları ile yürütme sırasında birden fazla bellek alanına aynı anda erişebilir. Her iplik için özel yerel hafıza alanı vardır. Her bir iplik bloğu içerisinde yer alan ipliklerin birbirleri arasında iletişimine imkân sağlayan ve de blok ömrü boyunca aktif olan ortak bir bellek alanını da sahiptir. Ayrıca faklı bloklarda yer alan tüm ipliklerin erişebildiği küresel bir bellek alanı da mevcuttur: Constant ve Texture bellek alanları. SIMD paralel hesaplama mimarisinde açıklandığı üzere multiprocessor işlemci iplikleri Warp ardı verilen otuz ikili gruplara ayırarak yaratır, yönetir, planlar ve koşturur [51]. Adreslenebilir bellek (yani küresel, yerel, paylaşılan, constant veya texture bellek alanları) alanlarına erişen bir komut çözgü içindeki iplikler arasında bellek adreslerinin dağılımına bağlı olarak birden fazla kez yeniden planlanabilir.



Şekil 2.16: NVIDIA Kepler GK110 GPU Mimarisi (SMX) [55].

Şekil 2.16'da ve Şekil 2.17'de görüldüğü üzere mimariye göre değişiklik göstermek ile birlikte çok sayıda Fermi Mimarisi Çoklu Akış İşlemci (Fermi Stream Multi-Processor – SM) oluşmaktadır [55][56]. Her bir SM çok sayıda komut yürütme ünitesi Komut Dağıtım Birimleri (Instruction Dispatch Units – DP), Özel Fonksiyon Yürütme Ünitesi (Special Function Unit – SFU), Veri Erişim Ünitesi (Load/Store Unit - LD/ST) ve işlemci çekirdeği (core) ile donatılmıştır. Koşturma esnasında her bir komut yürütme elemanı SIMD tipinde çok sayıda komutu SM bünyesinde yer alan çekirdekler üzerinde paralel olarak koşturabilir. Şekil **2.16**'da NVIDIA tarafından Kepler olarak adlandırılan grafik işlemci donanım mimarisi görülmektedir [55]. Şekil 2.17'de ise NVIDIA tarafından Fermi olarak adlandırılan bir önceki nesil grafik işlemci mimarisi görülmektedir [56].

Şekil 2.17: NVIDIA Fermi GF100 GPU Mimarisi (SM) [56].

Şekil 2.16'da gösterildiği üzere SM'ler Kepler mimarisi ile birlikte Kepler Mimarisi Çoklu Akış İşlemci (Kepler Stream Multi Processor Unit – SMX) olarak adlandırılmıştır. Kepler GK110 mimarisinde Tesla K20c de uygulandığı gibi her bir SMX dahilinde 6 adet 64-Bit bellek denetleyicisi bulunur. Her bir SMX 192 adet single-precision CUDA çekirdeğine, 64 adet double-prescision ünitesine ve 32 adet SFU ile 32 adet LD/ST ünitesine sahiptir. Her bir SMX'in erişiminde konfigüre edilebilir nitelikte 64 KB paylaşılan bellek mimarisi tanımlıdır ve iplikler arasındaki haberleşme ve senkronizasyon bu bellek ünitesi üzerinden sağlanmaktadır. NVIDIA CUDA farklı mimariler için farklı seviyelerde hesaplama yazılım desteği sunmaktadır. Kepler mimarisi için K20c için desteklenen sürüm 3.5 ve GT650M için sürüm 3.0'dır. NVIDIA GeForce GTX480 grafik işlemcisi Fermi GF100 mimarisinde bir donanım olup, toplam 15 adet SM ve her bir SM üzerinde 32 adet

single-precision CUDA çekirdeği ile 4 adet SFU ve 16 adet LD/ST ünitesi ile donatılmıştır. Fermi mimarisi versiyon 2.0 hesaplama desteği sunmaktadır.

NVIDIA firması tarafından farklı mimarilerde değişik genel maksatlı programlanabilen GPU mimarileri ticari olarak geliştirilmiştir. Bu tez çalışması kapsamında aşağıdaki Çizelge 2.6 ile gösterilen ve özellikleri paylaşılan mimarilerden faydalanılacaktır. Bu mimarilerin performansa etkileri tez kapsamında ortaya konulmak istenen noktalardan birisidir.

Çizelge 2.6: Çalışma kapsamında kullanılan NVIDIA GPU mimarileri.

	Tesla K20c	**GeForce GTX480**	**GeForce GT650M**
Mimari	Kepler GK110	Fermi GF100	Kepler GK107
SM/SMX Sayısı	13 SMX	15 SM	2 SMX
SM / Toplam Çekirdek Sayısı	13 x 192 = 2496	15x 32 = 480	2 x 192 = 384
SFU Sayısı	32 (Her bir SMX)	4 (Her bir SM)	32 (Her bir SMX)
LD/ST Sayısı	32 (Her bir SMX)	16 (Her bir SM)	32 (Her bir SMX)
Bellek Yapısı	GDDR5 320 208 bit GB/s max.	GDDR5 384 Bit 177.4 GB/s max.	DDR3 128 Bit 28.8 GB/s max.
Paylaşılan Bellek Konfigürasyonu (smem/cache – L1)	16/32/48 Toplam 64 KB	16/48 Toplam 64 KB	16/32/48 Toplam 64 KB
Hesaplama Mimarisi (Compute Capability)	3.5	2.0	3.0

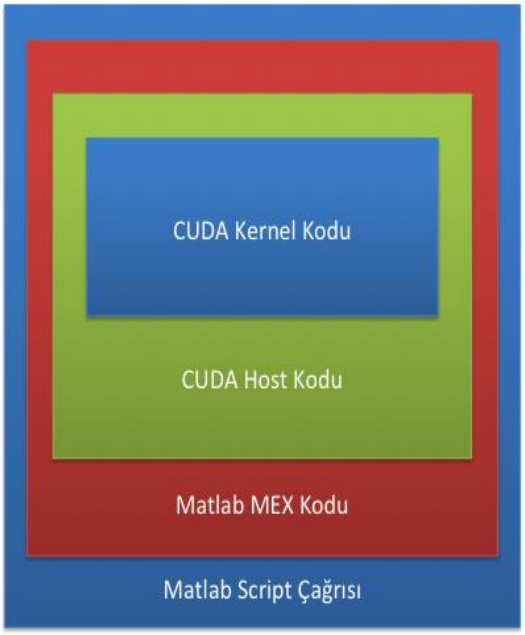
CUDA yazılımcılara yüksek seviye bir dil olan C desteği sunacak bir yazılım mimarisi desteği ile gelmektedir. C diline ve kütüphanelerine bir ilave olarak karşımıza çıkar. Böylelikle C dili notasyonunda ve özel olarak tanımlanmış fonksiyonları kullanarak çekirdek kodu çağrısı yapılmasını ve bu çekirdek kodun *n* adet CUDA ipliği olarak *n* farklı CUDA çekirdeği üzerinde paralel şekilde koşturulmasına imkân tanır. Koşturma esnasında her bir CUDA ipliği farklı mimarilerde birden fazla bellek alanına erişebilme yeteneğine sahiptir. Ayrıca her bir iplik için özel olarak tanımlanmış bellek alanı da bulunur. Her bir iplik bloğu (thread block) yaşam süresi boyunca içinde bulundurduğu ipliklerin haberleşmesi ve senkronizasyonu için erişebilecekleri özel bellek alanlarına sahiptirler. Ayrıca tüm iplikler genel bellek (global memory) adı verilen bellek mimarisine doğrudan erişim yetkisine sahiptirler.

CUDA ile gerçekleştirilecek olan paralel yazılımların geliştirilmesi esnasında donanım yeteneklerine bağlı olarak üzerinde durulması gereken başlıca iki ana unsur vardır;

a. Aynı anda paralel olarak kullanılacak iplik sayısı ve

b. Bu iplikler ile oluşturdukları grupların haberleşme, senkronizasyon ve veri işleme amacıyla erişebildikleri bellek seviyeleri ve miktarları.

Bu noktadan hareket ile bu ihtiyaçlara istinaden donanım mimarilerine olan bağımlılık öne çıkmaktadır.



Şekil 2.18: MATLAB destekli CUDA yazılım geliştirme ortamı.

Her ne kadar CUDA yazılım geliştirme ortamı C tabanlı programlama dilini doğrudan desteklese dahi, C++ programlama dili ile görsel çıktılar üretmek, IDE benzeri yapıları kullanmak, grafikler ile desteklenmiş matematik fonksiyonlarını analiz etmek güç olacaktır. Bu nedenlerden ötürü çalışma kapsamında MATLAB programlama dilinin özelliklerinden de faydalanılabilmesi için farklı bir yazılım geliştirme ortamından istifade edilmiştir.

Şekil 2.18 ile gösterildiği üzere tez çalışması kapsamında geliştirilen CUDA çekirdek kodları, C++ programlama dilinde geliştirilmiş, CUDA Host Kodu adı verilen ve CUDA mimarisi tarafından doğal olarak desteklenen bir üst seviye kod ile sarmalanmıştır. C++ programlama dilinde hazırlanan programa MATLAB ile erişmek için MEX adı verilen ara yüz yazılımından faydalanılmış ve dolayısıyla bu

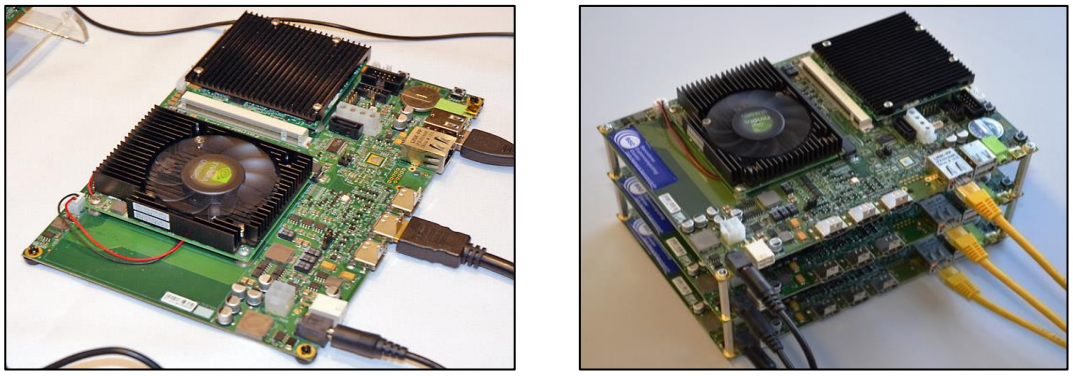
ara yüzü oluşturacak MEX kodları ve fonksiyonları kullanılmıştır. MEX fonksiyonlarına doğrudan MATLAB ortamında geliştirilen scriptler ile erişim sağlanmıştır. Bu yapı çalışmanın ilerleyen bölümlerinde ayrıntılı olarak açıklanmıştır.

2.5.4 Farklı NVIDIA grafik işlemci mimarileri

Önceki bölümler kapsamında açıklandığı şekilde grafik işlemciler üzerinde gerçekleştirilen paralel hesaplama performansı donanım özelliklerine oldukça bağlıdır. Grafik işlemcilerin donanım özellikleri kullanım yerlerine göre (mobil, masaüstü, iş istasyonu vb.) değişiklik göstermektedir. Bu kapsamda tez çalışması dahilinde geliştirilen yaklaşımlara ait algoritmaların hesaplama performansı aşağıda başlıklar halinde açıklanmıştır.

2.5.4.1 NVIDIA mobil-GPU hesaplama mimarileri

Bu tez kapsamında gerçekleştirilen tüm yaklaşımlar İHA sistemleri üzerinde kullanılabilir şekilde tasarlanmaktadır. Bu kapsamda ortaya konulan teorik yaklaşımların uygulanabileceği platformlar arasında düşük güç ihtiyacı duyan ve fiziksel olarak hafif ve küçük ölçekli grafik donanımlar ve bu donanımları destekleyen mimariler üzerinde sınanması bu bölüm kapsamında ele alınmıştır.



(a) Tek Kart ile geliştirme ortamı. (b) Çok sayıda kartın bir arada kullanımı.

Şekil 2.19: NVIDIA CARMA geliştirme kartı.

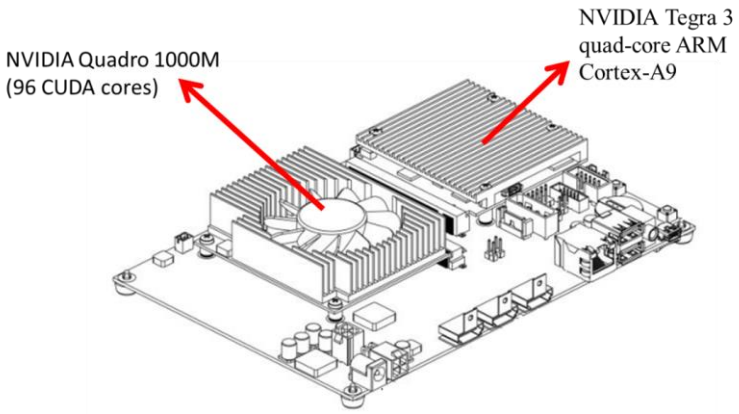
Tez çalışmasında kullanılması için NVIDIA firması tarafından gönderilen NVIDIA CARMA geliştirme kartı Şekil 2.19 (a) ile gösterilmektedir. Bu kart Orta İrtifa ve Uzun Menzil (Medium Altitude, Long Range – MALE) sınıfı İHA konseptine uygun olarak geliştirilen Taktik İnsansız Hava Aracı Sistemlerinde rahatlıkla kullanılabilecek özelliktedir. Fiziksel özellikleri ile hafif ve küçük ölçekte olması

nedeniyle ve ayrıca düşük güç tüketimine ihtiyaç duyması nedeniyle İHA sistemlerinde etkin kullanılabileceği değerlendirilmektedir.

Çizelge 2.7: NVIDIA CARMA geliştirme kartı temel özellikleri [57].

Özellik	**Değer**
CPU	NVIDIA Tegra 3 quad-core ARM Cortex-A9
GPU	NVIDIA Quadro 1000M (96 CUDA cores)
Bellek	2GB (CPU) + 2GB (GPU)
GPU Performans	270 GFLOP/s (single precision)
Ölçüler	200 x 140 mm
Ağırlık	~350gr
Güç İhtiyacı	19 V DC 45 W

Çizelge 2.7 ile CARMA geliştirme kartının teknik özellikleri gösterilmiş olup, Şekil 2.19 (b) ile gösterilen biçimde çok sayıda kartı bir ağ ortamı üzerinden birleştirerek daha geniş ölçekli problemlerin çözümü için kullanmak mümkündür.



Şekil 2.20: CARMA geliştirme kartı üzerinde CPU ve GPU gösterimi [57].

Şekil 2.20 ile gösterildiği biçimde CARMA geliştirme kartı üzerinde 96 adet CUDA çekirdeğine sahip genel maksatlı programlama yapabilen Şekil 2.21 ile gösterilen NVIDIA Quadro 1000M grafik donanımı bulunmaktadır [58]. Ayrıca geliştirme kartı üzerinde Ubuntu 10.04 işletim sistemini ve C++ dilinde programlama imkânı sağlayan, ağ donanımının ve diğer işlemlerin yürütülmesine olanak sağlayan CUDA sürücüleri ile desteklenmiş NVIDIA Tegra 3 Cortex A9 işlemci yer almaktadır.

Şekil 2.21: NVIDIA Quadro 1000M grafik donanımı [58].

NVIDIA Quadro 1000M grafik donanımı temel mimari özellikleri Çizelge 2.8 ile gösterilmiştir. Fermi mimarisindeki donanım, 2.1 versiyon hesaplama mimarisinde ve CUDA versiyon 5.0 destekleyen bir grafik işlemcidir [58].

Çizelge 2.8: NVIDIA Quadro 1000M GPU temel özellikleri [58].

Özellik	**Değer**
Compute Capability	2.1
Driver Ver.	5.0
Max. ThreadsPerBlock	1024
Max.ShmemPerBlock	49152
Max.ThreadBlockSize	[1024 1024 64]
MaxGridSize	[65535 65535 65535]
Total Mem.	2048 Mbytes
Multiprocessor Count	2
ClockRateMHz	700
Power	45 W
Memory	128 Bit
CUDA Cores	96 Cores

Çalışma kapsamında gerçekleştirilen paralel algoritmaların sınandığı bir diğer mobil grafik donanımı ise Şekil 2.22 ile gösterilen NVIDIA GeForce GTX 670MX'dir. CARMA geliştirme ortamından farklı olarak bu grafik işlemcisi, paralel programlama amacıyla kullanılabilmesi için standart PC konfigürasyonuna ve ilave merkezi işlemci birimi ile belleğine ihtiyaç duymaktadır [59].

Şekil 2.22: NVIDIA GeForce GTX 670MX grafik donanımı [59].

NVIDIA GeForce GTX 670MX grafik işlemcisinin temel paralel programlamaya yönelik özellikleri Çizelge 2.9 ile gösterilmiştir. Kepler mimarisindeki paralel işlemci hesaplama versiyon 3.0 özelliklerini ve CUDA versiyon 6.5 yazılım ortamını desteklemektedir [59].

Çizelge 2.9: NVIDIA GeForce GTX 670MX GPU temel özellikleri.

Özellik	**Değer**
Compute Capability	3.0 (Kepler)
Driver Ver.	6.5
Max. ThreadsPerBlock	1024
Max.ShmemPerBlock	49152
Max.ThreadBlockSize	[1024 1024 64]
MaxGridSize	[65535 65535 65535]
Total Mem.	2048
Multiprocessor Count	7
ClockRateMHz	700
Power	75 W
Memory	67.2 GB/s 192 Bit
CUDA Cores	960

2.5.4.2 NVIDIA iş istasyonu/masaüstü-GPU hesaplama mimarileri

Çalışma kapsamında performansları değerlendirilen mobil grafik işlemciler dışındaki bir diğer grafik işlemci sınıfı ise masaüstü/iş istasyonu mimarilerinde kullanılan mobil işlemciler gibi doğrudan platformun üzerinde yer almasından çok İHA yer kontrol merkezinde kullanılabilecek özelliklerde donanımlardır. Mobil işlemcilere nazaran daha ağır ve büyük ölçekte olmalarının yanı sıra, daha yüksek güç ve daha donanımlı çevre ünitelerine ihtiyaç duymaktadırlar. Bu nedenlerden dolayı platformun uzaktan yönlendirilmesinde etkin olarak kullanılabilirler.

Şekil 2.23: NVIDIA Tesla K20c paralel hesaplama donanımı [60].

Şekil 2.23 ile gösterilen NVIDIA Tesla K20c grafik işlemci donanımı daha çok iş istasyonlarında kullanılmak üzere üretilmiş ve doğrudan grafik işlemciler ile genel maksatlı paralel programlama amacına yönelik olarak geliştirilmiş bir donanımdır [60].

Çizelge 2.10: NVIDIA Tesla K20c GPU temel özellikleri [60].

Özellik	**Değer**
Compute Capability	3.5 (Kepler)
Driver Ver.	6.5
Max. ThreadsPerBlock	1024
Max.ShmemPerBlock	49152
Max.ThreadBlockSize	[1024 1024 64]
MaxGridSize	[65535 65535 65535]
Total Mem.	5120 Mbytes
Multiprocessor Count	13
ClockRateMHz	1500
Power	225 W
Memory	208 GB/s 320 bit
CUDA Cores	448

Çizelge 2.10 ile Kepler mimarisinde çalışan hesaplama versiyon 3.5 destekleyen ve CUDA 6.5 sürümünü ile geliştirilmiş paralel kodları koşturabilen NVIDIA Tesla K20c grafik işlemcisinin temel programlama ve fiziksel özellikleri gösterilmiştir. Yüksek grafik bellek miktarı, hızlı bellek erişimi olanağı ve yüksek güç tüketimi donanımı diğer örneklerinden ayıran başlıca özellikleridir [60].

Bu K20c gibi Kepler GK110 mimaride her SMX altı bellek denetleyicisi (64 Bit) içerir. Çizelge 2.6 ile gösterildiği gibi her SMX biriminde 192 tek kesinlikli CUDA çekirdeği, 64 çift hassasiyetli birimleri ve 32 yükle/sakla birimleri vardır [60].

Şekil 2.24: NVIDIA GeForce GTX 480 grafik işlemci donanımı.

Çalışma kapsamında paralel programlama amacıyla faydalanılan bir diğer donanım ise Şekil 2.24 ile gösterilen NVIDIA GeForce GTX 480 grafik işlemcisidir [61]. Bu donanım daha çok standart masaüstü bilgisayarlarında kullanılan grafik işlemcisidir.

Çizelge 2.11: NVIDIA GeForce GTX 480 GPU temel özellikleri [61].

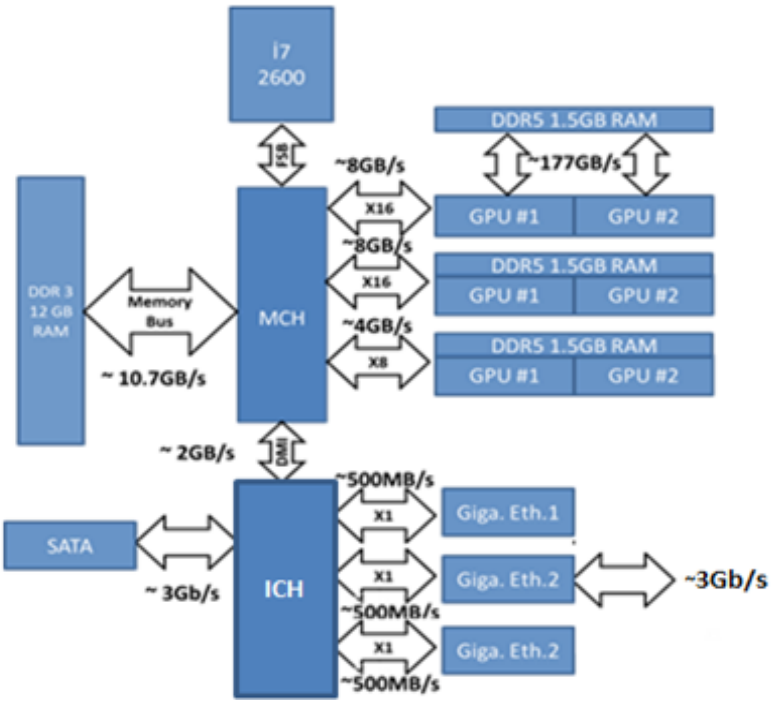
Özellik	**Değer**
Compute Capability	2.0 (Fermi)
Driver Ver.	6.5
Max. ThreadsPerBlock	1024
Max.ShmemPerBlock	49152
Max.ThreadBlockSize	[1024 1024 64]
MaxGridSize	[65535 65535 65535]
Total Mem.	1536
Multiprocessor Count	15
ClockRateMHz	1400
Power	225 W
Memory	177 GB/s 384 Bit
CUDA Cores	480 Cores

Çizelge 2.11 ile GTX 480 grafik işlemcisinin temel paralel programlama ve fiziksel özellikleri gösterilmiştir. Hesaplama versiyon 2.0 desteği sunan donanım Fermi mimarisinde geliştirilmiş bir grafik işlemcisidir. Yüksek sayıda akış işlemcisi ve CUDA hesaplama çekirdeği sunması ile diğer grafik işlemcilerden ayrılmaktadır [61].

2.5.4.3 Multi-GPU ile hesaplama mimarileri

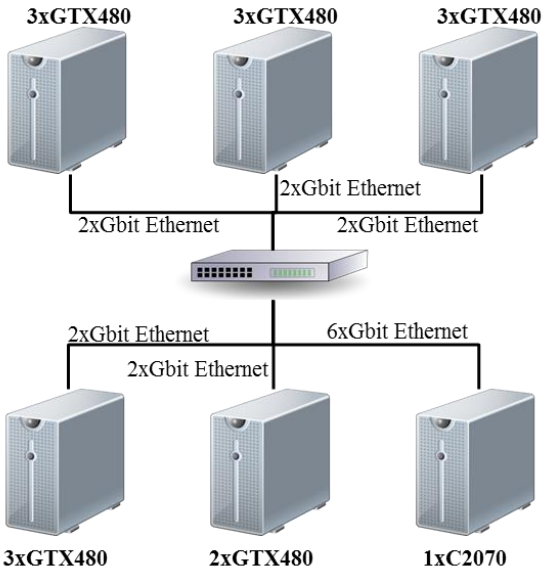
GPU üzerinde paralel kod geliştirmenin daha da hızlandırılması amacıyla faydalanılabilecek yaklaşımlardan bir diğeri de çok sayıda GPU işlemciyi aynı

problemin çözümüne yönelik bir arada kullanmaktır. Dolayısıyla bu durum probleme bir dağıtılmış çözüm yaklaşımını da sunacaktır.



Şekil 2.25: Grafik işlemcilerinin ana makine ile fiziksel bağlantısının gösterimi.

Şekil 2.25 ile gösterilen biçimde sistem üzerinde çok sayıda grafik işlemcinin fiziksel bağlantısı mümkündür. Grafik işlemcinin ana makine üzerindeki bağlantı şekli çok sayıda GPU'nun hesaplama amacıyla bir arada kullanılması esnasında hesaplama performansına doğrudan etki edecektir.



Şekil 2.26: Grafik işlemciler yer alan sunucuların ağ bağlantısının gösterimi.

Aynı ana makine üzerinde yer alan grafik işlemcilerin yanısıra Şekil 2.26 ile gösterildiği biçimde çok sayıda üzerinde grafik işlemci içeren sunucuyu bir ağ ortamı üzerinden birbirleri ile iletişim haline getirerek aynı uygulama doğrultusunda bir arada kullanmak mümkündür. Bu durum da karşımıza üç farklı çok sayıda GPU'nun bir arada kullanımı durumu çıkacaktır. İlerleyen alt başlıklar altında her bir durum ayrı ayrı incelenmiştir.

2.5.4.3.1 Ortak PCI veri yoluna bağlı GPU işlemciler

Şekil 2.25 ile gösterilen biçimde ana makine üzerindeki her Çevresel Bileşen Bağlantısı (Peripheral Component Interconnect – PCI) veri yoluna sadece bir NVIDIA grafik kartı takılabilir. Ancak bazı NVIDIA grafik kartları üzerinde birden fazla GPU yer almaktadır. Bu iki GPU ortak genel bellek üzerinden yaklaşık 177 GB/s hızında haberleşebilirler. Bu iki GPU'yu bir arada kullanma imkanı vardır.

2.5.4.3.2 Aynı fiziksel yonga seti üzerinden haberleşen ve farklı PCI veri yollarına bağlı GPU işlemciler

Bir diğer birden fazla GPU'yu bir arada aynı paralel uygulama dahilinde kullanma yöntemi ise ana sunucu üzerinde yer alan farklı PCI veri yollarına takılmış NVIDIA kartlarında yer alan GPU'ları kullanmaktır. Bu durum yine

Şekil 2.25 ile gösterilmektedir. Dikkat edilmesi gereken husus sunucu ana kartının kuzey köprüsü üzerinde yer alan PCI veri yollarının hepsi ya da bir kısmı kullanıldığında ortaya çıkan veri aktarım hızlarıdır. Çünkü CUDA tabanlı bir paralel uygulamada başlangıç verisi ile işlemler gerçekleştirildikten sonra elde edilen sonuç verileri grafik genel belleğinden sunucu belleğine PCI veri yolu üzerinden aktarılacaktır. Ayrıca kullanılan farklı PCI veri yollarına bağlı GPU'lar arasındaki haberleşme amaçlı veri aktarımı da yine bu PCI veri yolları üzerinden yapılacaktır. Çalışma kapsamında kullanılan ana kart özellikleri incelendiğinde

Şekil 2.25 ile gösterildiği şekilde kuzey köprüsü üzerine 3 adet PCI veri yolu ile bağlı grafik donanımı takıldığında bunların ilk ikisi 16x (yaklaşık 8GB/s) hızında, birisi ise 8X (yaklaşık 4 GB/s) hızında veri aktarımı imkanı sağlayacaktır. Bu veri aktarım hızları teorik olup, ölçülen veri aktarım hızları Ek A'da yer alan tablolarda verildiği şekilde gerçekleşmektedir.

2.5.4.3.3 Farklı fiziksel yonga seti üzerinde yer alan PCI veri yollarına bağlı ve bir ağ üzerinden haberleşen GPU işlemciler

Son birden fazla GPU kullanarak paralel hesaplama gerçekleştirme imkanı ise birden fazla sunucu üzerinde yer alan çok sayıdaki grafik işlemciden faydalanmaktır. Bu durumda farklı sunucular üzerinde yer alan grafik işlemcileri birbirleri ile ortak ağ bağlantısı üzerinden Şekil 2.26 ile gösterildiği biçimde haberleşeceklerdir. Her sunucu üzerinde yer alan üç gigabit ethernet bağlantısı tek bir veri yolu oluşturacak şekilde konfigüre edilerek ağ üzerinden yaklaşık 3 Gbit/s hızında haberleşme imkanı sağlanmıştır. Elde edilen bu veri aktarım hızı daha önceden sıralanan çok sayıda GPU kullanım yöntemlerine kıyasla en yavaş haberleşme kanalını ortaya koymaktadır. Ayrıca grafik işlemciler tarafından birbirlerine aktarılmasına ihtiyaç duyulan verinin öncelikle sunucu belleklerine aktarılmasına ardından da ağ ortamı üzerinden diğer sunuculara aktarılmasına ihtiyaç duyulmaktadır. Yüksek miktardaki veri aktarımlarında bu durum bir dar boğaz teşkil edebilmektedir.

3. YAPAY POTANSIYEL ALAN TABANLI YOL PLANLAMASI

Bu bölüm dahilinde otonom yol planlaması amacıyla faydalanılan YPA tabanlı yapıların temel özelliklerine, nasıl modellendiklerine, hareketi nasıl belirlediklerine, yol planlaması amacıyla faydalanılan temel formlarına ve bu formların birleştirilerek görevin planlanması ve platformların yönlendirilmesi için nasıl sınırlandırılarak bir arada kullanıldığına değinilmiştir. Ayrıca YPA tabanlı yol planlaması esnasında karşılaşılan literatürdeki problemler ile çözüm yöntemleri belirtilmiş, çalışma kapsamında nasıl uygulandığı açıklanmıştır.

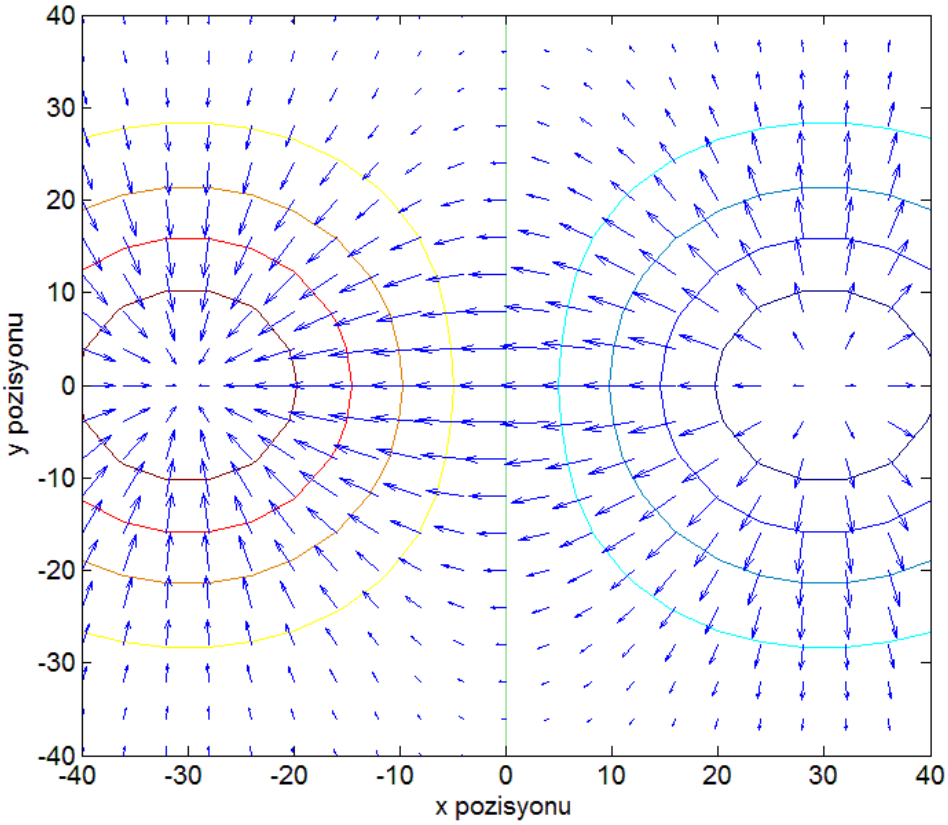
3.1 Yapay Potansiyel Alanlar

Yol planlaması amacıyla faydalanılan yaklaşımlardan birisi vektör tabanlı yaklaşımlardır. Vektör tabanlı yaklaşımlara örnek olarak kullanılan en yaygın yol planlaması yöntemi Yapay Potansiyel Alanlar (YPA) yöntemidir. Potansiyel alanlar engellerden kaçınırken hedefe giden yolları planlamada yaygın olan faydalanılan benzetim tabanlı tekniklerden birisidir.

Bu fikir ilk olarak, çekici potansiyel olarak bir hedef ve itici potansiyel olarak durağan engelleri kullanarak, mobil robot kontrolü tasarlayan Khatib tarafından öne sürülmüştür [12]. YPA teknikleri Geometri tabanlı yaklaşımlardakine benzeyen ızgara (grid) tabanlı bir yaklaşımdır. Ancak ayrıldıkları nokta oluşan hücrelerin her birinin geometri tabanlı yaklaşımlara örnek olan A* veya Dijkstra benzeri yaklaşımlardaki gibi bağımsız olarak değerlendirilmeyişidir [41]. Hareketi modellenen alanın genelinde bütün bir akış olarak planlamayı hedefleyen YPA yaklaşımı manyetik alan modellemesine ya da gaz yayılması modellemesine benzerlikler göstermektedir [43]. Basit anlamda potansiyel alan metodu, küçük ölçekler için kolay uygulanabilir ve çok değişikliğe ihtiyaç duymadan kabul edilebilir sonuçlar sağlayabilir.

Temel olarak YPA yaklaşımını tarif etmek için bir hareket alanı tanımlamasını şu şekilde yapabiliriz: “Sınırları belirlenmiş bir hareket alanı içerisinde, hareket edebilen bir platform olduğu ve yer çekimi kuvvetinin etkisine bağlı olarak yüksek

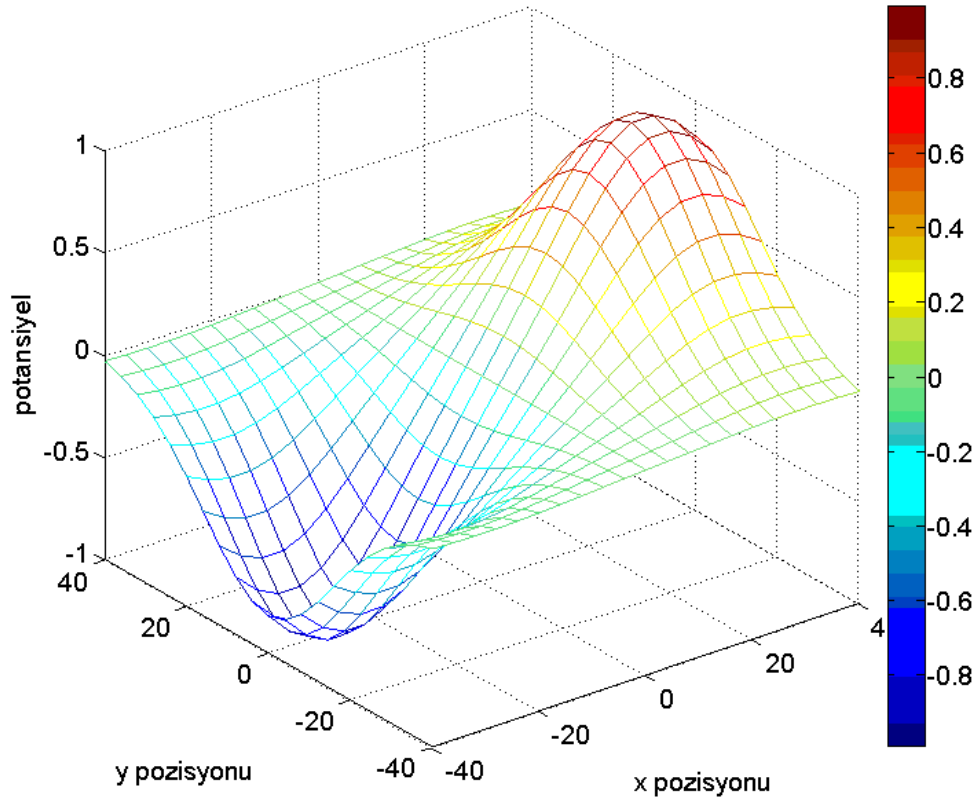
bir konumdan alçak bir konuma doğru hareket edebildiği düşünüldüğünde, mobil platform hareketini tepelerin arasından geçerek en düşük noktaya varacak şekilde sürdürecek ve bu hareket esnasında daima çevresindeki yükseltiler ile temsil edilen engellerden kaçınarak alçak noktaları seçerek hareketini sürdürecek ve alan içerisindeki en alçak noktaya ulaştığında ise hareketini sonlandıracaktır". Bu tanım çerçevesinde YPA tabanlı yol planlaması yaklaşımı, mobil platformun seyahati süresince hareketini şekillendiren tüm sürat ve davranış komutları verilerini üretecektir.



Şekil 3.1: Toplam potansiyel alandaki bileşke kuvvet vektörlerinin gösterimi.

YPA tabanlı yol planlaması yaklaşımında, çevre koşullarında yer alan iki grup temel kavram tanımlaması yapılmaktadır; hedef noktalar ve engeller. Hedef noktalar en düşük potansiyel değerli konumları, engeller ise daha yüksek potansiyelleri temsil etmektedir. Hedef noktalar yol planlamasının sonunda robotun ulaşmak istediği konumları gösterir ve YPA içerisinde belirli bir çözünürlük değerine bağlı olarak tanımlanmış vektör alanında çeken yönlü vektörlerin işaret ettiği konumdur. Dolayısıyla bu noktaya Şekil 3.1'de (*-30,0*) konumu ile gösterildiği biçimde çeken kuvvetlerin merkezi de denilebilir ve $\vec{F}_h$ ile temsil edilir. YPA ile modellenmiş vektör alan içerisinde yer alan bir diğer tanımlama da engellerin oluşturduğu ve

çeken kuvvetlere ters yönlü olarak tanımlanmış iten kuvvet vektörleridir. İten kuvvet vektörlerinin merkez noktasını da Şekil 3.1'de (*30,0*) konumu ile gösterildiği şekilde engeller oluşturur ve $\vec{F}_e$ ile temsil edilir.



Şekil 3.2: Toplam potansiyel alandaki bileşke potansiyel değerlerin gösterimi.

Khatib'in yaklaşımında; potansiyel alan, hedef noktasında bir minimuma sahip olan konfigürasyon uzayı içinde tanımlanır. Şekil 3.2 ile gösterildiği biçimde hedef minimum potansiyel noktasındayken (F_h), tüm engeller yüksek potansiyel tepesi (F_{e_i}) olarak farz edilir. Öyle ki potansiyel alanda, robot çalışma alanı içerisinde aynı anda engeller tarafından itilirken hedef tarafından çekilecektir. Robotun yer çekimi etkisinde hareket eden elemanın hareketinde olduğu gibi bir davranış izlemesi beklenmektedir. Tüm kuvvetlerin toplamı mobil robotun hareket yönünü ve hızını belirleyecektir.

Şekil 3.1'de görüldüğü gibi çeken nokta etrafında oluşturulan potansiyel hızlandırıcılar Şekil 3.2'de görüldüğü gibi yapay vadiler ile gösterilebilen güç vektörlerine dönüştürülür. Şekil 3.1'de görüldüğü gibi engelleri temsil eden itici güç engelleri temsil eden noktaların etrafında Şekil 3.2'de gösterildiği şekilde yapay tepeler oluşturarak, ölçülebilir büyüklüklerden oluşan alanlar ortaya konulur.

Potansiyel alanlardan vektör alanlara dönüşüm temel gradient işlemi vasıtasıyla gerçekleştirilir. Bu durum iki boyutlu alanlarda yer alan hedef ve engel potansiyelleri için sırasıyla Denklem (3.1) ve (3.2) ile gösterildiği şekilde hesaplanabilir.

$$\vec{F}_h(x,y) = \nabla F_h(x,y) = \frac{\partial F_h}{\partial x}\hat{\imath} + \frac{\partial F_h}{\partial y}\hat{\jmath} \quad \textbf{(3.1)}$$

$$\vec{F}_e(x,y) = \nabla F_e(x,y) = \frac{\partial F_e}{\partial x}\hat{\imath} + \frac{\partial F_e}{\partial y}\hat{\jmath} \quad \textbf{(3.2)}$$

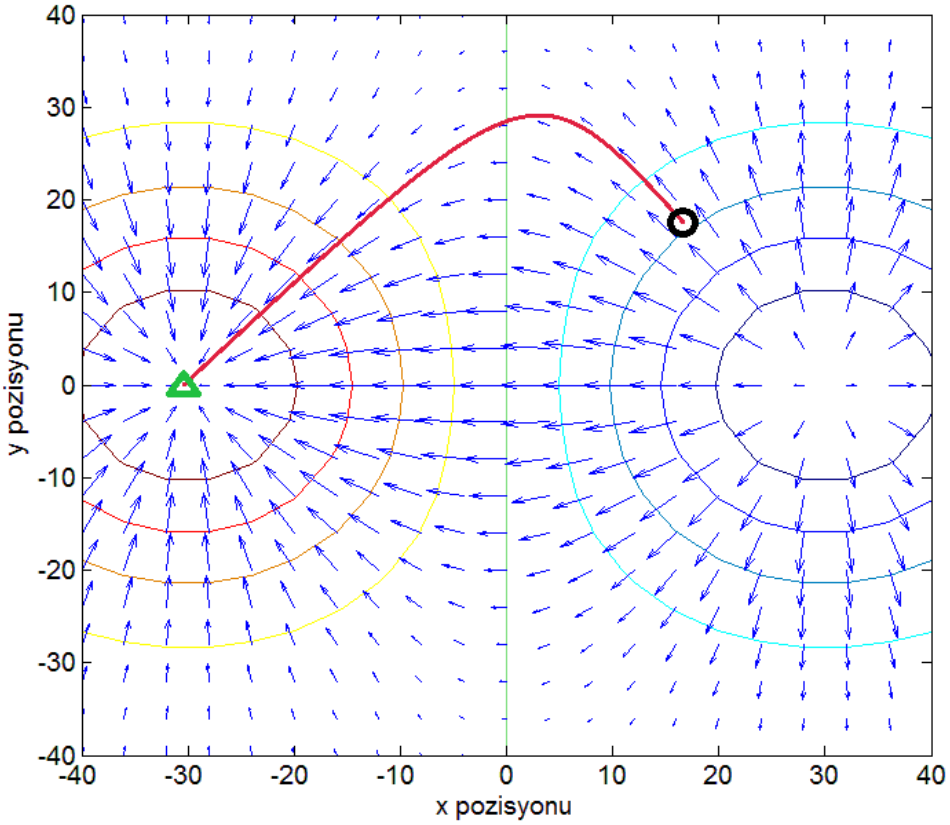
Vektör alan içerisinde yer alan çeken yönlü hedef nokta ($\vec{F}_h$) ve iten ters yönlü engellere ($\vec{F}_e$) ait vektör kuvvetlerin oluşturduğu toplam bileşke vektörler en genel anlamı ile robotun alan içerisinde herhangi bir noktadan hedefe ulaşması için izlemesi gereken doğrultuyu ve robota etki eden kuvvetin şiddetini temsil etmektedir. Bu durumda *n* adet engel ve bir hedef noktanın yer aldığı alanın içerisinde, her bir (x, y) noktasında yer alan robota etki eden toplam kuvvet ($\vec{F}_t$), Denklem (3.3)'deki şekilde gösterilebilir.

$$\vec{F}_t(x,y) = \vec{F}_h(x,y) + \sum_{i=0}^{n} \vec{F}_{e_i}(x,y) \quad \textbf{(3.3)}$$

Ölçülebilir alanların birleştirilmesi ile engellerden sakınarak hedefe ulaşmayı öngören bir yapı elde edilir. Bu yöntem Denklem (3.4)'de ifade edildiği şekilde ve Şekil 3.3 ile gösterildiği biçimde, (x_0, y_0) başlangıç konumundan başlayan ve engellerden kaçınırken hedefe giderken izlenen her bir (x, y) noktasının oluşturduğu yolu planlamada ölçülebilir alanların bileşenlerinin toplanması sonucu, engellerden sakınarak hedefe ulaşmayı öngören ve otonom sistemler için yaygın olarak kullanılan yol planlaması (U) tekniklerinden birisidir:

$$U(x,y) = \int_{(x_0,y_0)}^{(x,y)} \overrightarrow{F_t}.dl \quad \textbf{(3.4)}$$

Denklem (3.4) ile Şekil 3.3'de daire sembolü ile gösterilen başlangıç noktasından üçgen sembolü ile temsil edilen hedef noktaya ulaşmak için izlenmesi gereken yol planlaması hesaplaması gösterilmektedir.



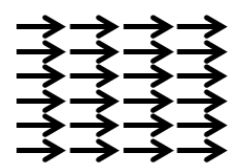
Şekil 3.3: YPA tabanlı yol planlamasının gösterimi.

(*x*,*y*)'nin tüm kombinasyonları için oluşturulacak olan tasvir, *U* olarak adlandırılan iki boyutlu diziyi, dolayısıyla izlenmesi gereken yolu hücre konumları ile ortaya koyacaktır. Potansiyel alan kapsamında kullanılan grid çözünürlüğünün her bir hücresi bir vektörel büyüklüğü temsil etmektedir ve bu vektörel büyüklük *V(m,d)* şeklinde ifade edildiği taktirde *m* parametresi vektörün büyüklüğünü dolayısıyla gridin o noktasında robota etkin eden kuvvetin şiddetini, *d* parametresi ise vektörün yönünü dolayısıyla kuvvetin etki ettiği yönü dolayısıyla robotun hareket etmesi istenen istikameti temsil etmektedir. Bu yöntemde güç kontrolü bu iki potansiyelin bileşkesi olan gradientidir [12]. Bu tanımdan da anlaşılacağı gibi YPA yaklaşımı yol planlaması çözümünün ötesinde aynı zamanda robotun hareketini kontrol eden bir kontrol yaklaşımı olarak kabul edilebilir [62]. Elde edilen toplam potansiyel alanın bir noktadaki gradienti o nokta için robota etki eden potansiyel kuvvet vektörünün değerini ve yönünü ifade edecektir.

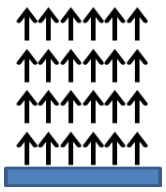
3.2 Yapay Potansiyel Alan Tabanlı Temel Hareket Modelleri

YPA tabanlı yol planlaması yaklaşımının temelini engel ve hedef modellemelerinin yapılması oluşturmaktadır. Ancak engel tanımlamalarının kusursuz yapılabilmesi

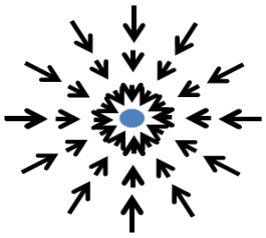
birçok şarta bağlıdır. Bu şartlardan birisi geometrik olarak engellerin ve hedeflerin modellenmesi durumudur. Dikkat edilmesi gereken bir diğer husus, modelleme hassasiyeti ne kadar artarsa hesaplama maliyetininde o derece artacağıdır. Yukarıda yapılan açıklamalara istinaden YPA yaklaşımında geometrik olarak engellerin ve hedeflerin farklı şekillerde modellenmesine imkan tanıyan yaklaşımlar literatürde mevcuttur ve genel olarak bu yaklaşım örnekleri Şekil 3.4 ile gösterilmektedir.



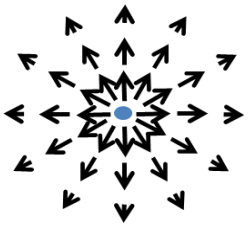
(a) Düzenli (Uniform).



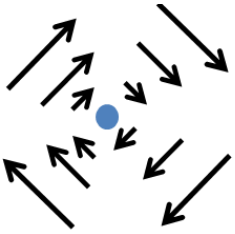
(b) Engele Dik (Perpendicular).



(c) Noktasal/Dairesel Çeken Kuvvet (Point/Circular Attraction).



(ç) Noktasal/Dairesel İten Kuvvet (Point/Circiular Repulsion).



(d) Noktasal/Dairesel Teğet Kuvvet(Point/Circular Tangential).

Şekil 3.4: YPA içerisinde hedef/engel modellemesinde kullanılan temsiller.

YPA ile modellenen alanlar her zaman engellerden kaçınarak ulaşılmak istenen hedef noktaya yönelik yol planlaması amacıyla faydalanılan yaklaşımlar değildir. Örneğin aynı alan içerisinde yer alan birden fazla hava aracı için her araç kendisi dışındaki diğer araçları, çarpışmayı önlemek maksadıyla, dinamik birer engel olarak görebilir [63]. Bu durumda yol planlaması gerçekleştirilen platformun kendisi dışındaki diğer araçları Şekil 3.4 (ç)'de gösterildiği biçimde noktasal/dairesel iten kuvvet olarak YPA içerisinde modellemek mümkündür. Ya da aracın belirli bir alan içerisinde istenen doğrusal rotadan ayrılmadan hareket etmesi beklenebilir. Bu durumda izlenmek istenen doğrusal rota boyunca iki taraflı bir kanal benzere engele dik yapıda Şekil 3.4 (b)'de gösterildiği şekilde vektörel alanlar ile modellenmiş sanal

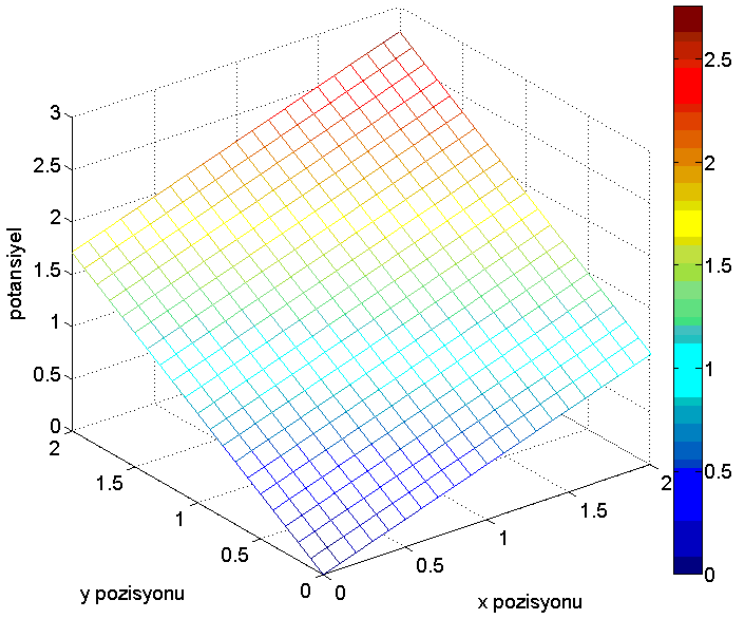
engeller oluşturulabilir ve aracın sürekli bu rotada kalması sağlanabilir. Bir diğer örnek ise aracın bir nokta etrafında örneğin ulaşılan hedef üzerinde dairesel bir bekleme rotası izlemesi olarak verilebilir. Bu durumda hedef nokta Şekil 3.4 **(**d)'deki gibi noktasal/dairesel teğet kuvvet olarak modellenir ve aracın bu nokta etrafında belirli bir mesafede yörünge hareketini yapması sağlanır. Bundan sonraki bölümlerde bu temel alanların modellenmesine maddeler halinde değinilmiştir.

3.2.1 Düzgün akış potansiyel alanı

Akış yönü boyunca potansiyelin düzgün olarak değiştiği potansiyel fonksiyon, "düzgün akış" olarak adlandırılır. Düzgün akıştan çalışma uzayının toplam potansiyelini modellemek için faydalanılır. Akış iki boyutta x-ekseni ile β açısı yaptığında, düzgün akış (F_d) için potansiyel fonksiyon denklemi Denklem (3.5) ile ifade edilebilir;

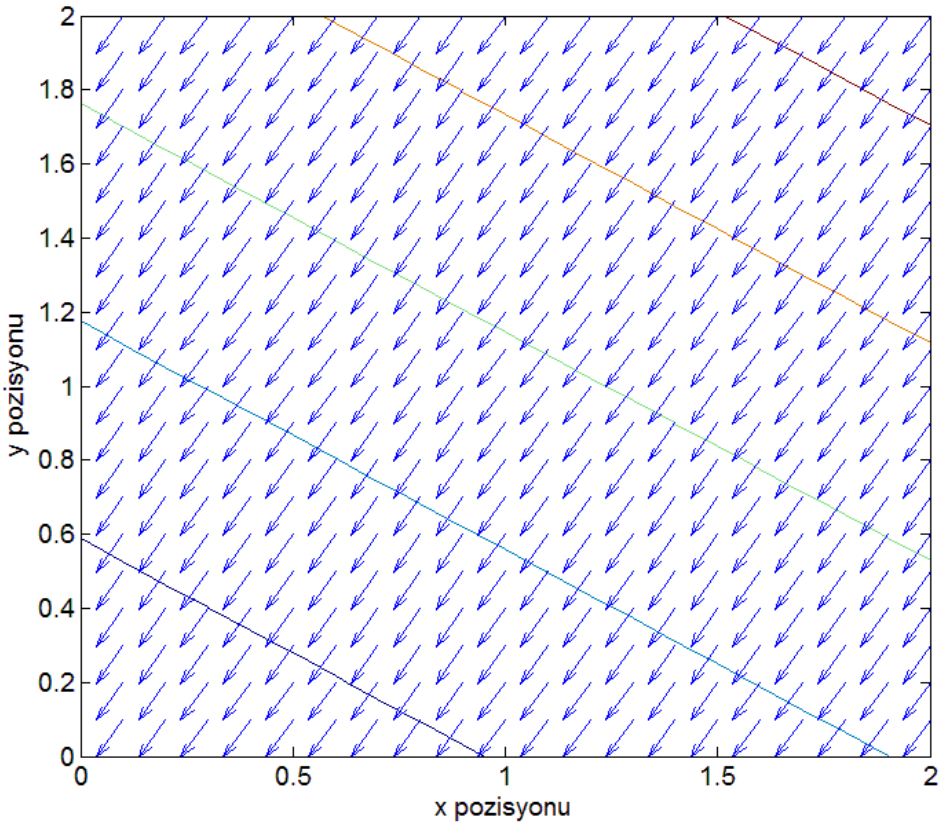
$$F_d = -\delta(x cos\beta + y sin\beta) \qquad \textbf{(3.5)}$$

Burada δ parametresi düzgün akışın şiddetini belirtir. Düzgün akış, nötr durumdaki sınırsız bir çevre için, başlangıç konumundan hedef konumuna kadar etkin potansiyel akış türetmek için kullanılır. x-ekseni ile 45 derecelik açı ($\beta = \pi/4$) yapan ve kuvvet çarpan değeri $\delta = -1$ olarak belirlenen alan için Denklem (3.5) eşitliğini kullanarak Şekil 3.5 ile gösterilen potansiyel düzgün akış elde edilir.



Şekil 3.5: Düzgün potansiyel akış gösterimi.

Denklem (3.5) ile gösterilen düzgün akış potansiyelinin gradienti (∇) alındığında Şekil 3.6 ile gösterilen düzgün akışa ait vektör alan ($\vec{F}_d$) elde edilir.



Şekil 3.6: Düzgün akış vektör alan gösterimi.

3.2.2 Engele dik oluşan yapay potansiyel alan

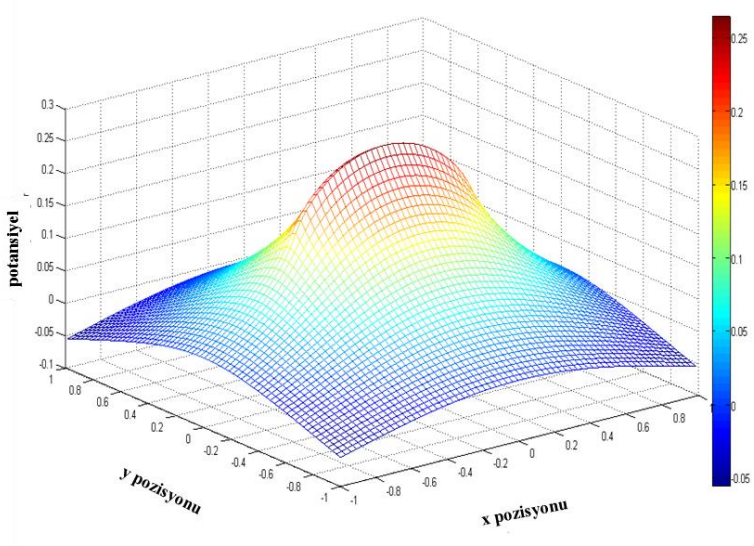
Bu bölüm kapsamında iki boyutlu uzayda yer alan bir doğrusal engelin oluşturduğu itici potansiyelin nasıl hesaplandığı tanımlanacaktır. Doğrusal parçanın potansiyeli akışkanlar mekaniğinde panel olarak adlandırılır. Şayet bir panel düzenli bir potansiyel alan kaynağı olarak kabul edilir ise, panelin oluşturduğu potansiyel kuvvet büyüklüğü F_p olarak tanımlanabilir. Herhangi bir (x, y) noktasındaki potansiyel değer kaynak panele ait etkilerin o noktadaki (dl) toplamıdır [63][64] ve Denklem (3.6) ile hesaplanabilir;

$$\nabla F_p = \frac{\delta\, dl}{2\pi} ln\, r = \frac{-\delta}{2\pi} ln \sqrt{x^2 + (y-l)^2}\, dl \qquad \textbf{(3.6)}$$

Bu durumda bütün panel için vektör alan ($\overrightarrow{\mathrm{F_p}}$) Denklem (3.7) yardımı ile hesaplanabilir;

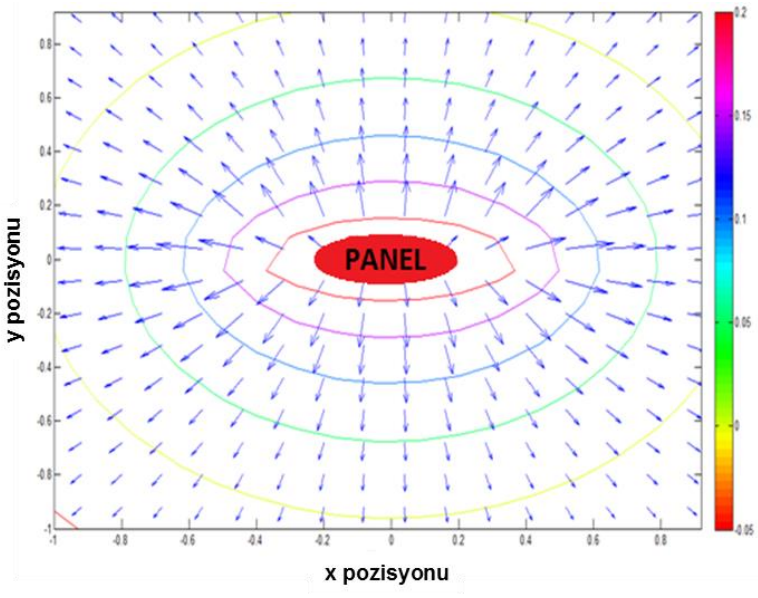
$$\overrightarrow{F_p}(x,y) = \frac{\delta}{4\pi} \int_{-L}^{L} ln\, x^2 + (y-l)^2\, dl \qquad \textbf{(3.7)}$$

$(L, -L)$ panelin başlangıç ve bitiş noktalarını, l panelin uzunluğunu temsil eder. Şekil 3.7'de Denklem (3.7) eşitliğini kullanarak, birim uzunlukta panelin potansiyel değeri hesaplanarak gösterilmiştir. Panel için kuvvet bileşeni değeri $\delta = -1$ olarak alınmıştır.



Şekil 3.7: Panel potansiyelinin gösterimi.

Şekil 3.8 ile Şekil 3.7'de yer alan panel potansiyelinin, dolayısıyla Denklem (3.7)'de yer alan eşitliğin gradienti (∇) alınarak üretilmiş olan panele ait vektör alan gösterilmektedir.



Şekil 3.8: Panelin oluşturduğu vektör alanın gösterimi.

Potansiyel fonksiyonun kısmi difaransiyelini alarak panel tarafında oluşan hız alanları hesaplanabilmektedir. *x*-eksenine ve *y*-eksenine bağlı olarak difaransiyel hız bileşkeleri yani $\overrightarrow{F_{p1}}$ ve $\overrightarrow{F_{p2}}$'yi kartezyen koordinatlar verecektir.

$$\overrightarrow{F_{p1}}(x,y) = -\frac{\partial F_p}{\partial x} = \frac{-\delta}{2\pi}\int_{-L}^{L}\frac{x}{x^2+(y-l)^2}\,dl \qquad \textbf{(3.8)}$$

$$\overrightarrow{F_{p2}}(x,y) = -\frac{\partial F_p}{\partial y} = \frac{-\delta}{2\pi}\int_{-L}^{L}\frac{y-l}{x^2+(y-l)^2}\,dl \qquad \textbf{(3.9)}$$

Buradan sonuçla $\overrightarrow{F_{p1}}(x,y)$ ve $\overrightarrow{F_{p2}}(x,y)$ değerleri Denklem (3.10) ve (3.11) hesaplanarak elde edilebilir;

$$\overrightarrow{F_{p1}}(x,y) = \frac{-\delta}{2\pi}\left(arctan\,\frac{y+l}{x} - arctan\frac{y-l}{x}\right) \qquad \textbf{(3.10)}$$

$$\overrightarrow{F_{p2}}(x,y) = \frac{-\delta}{4\pi}\ln\left(\frac{x^2+(y+l)^2}{x^2+(y-l)^2}\right) \qquad \textbf{(3.11)}$$

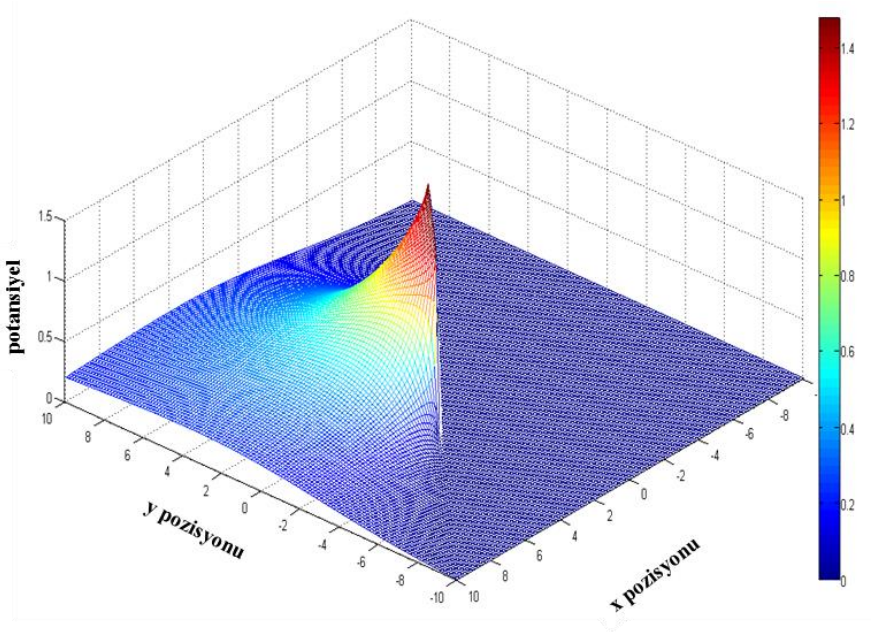
Denklem (3.10)'deki $\overrightarrow{F_{p1}}$ değeri limitlenirse $\overrightarrow{F_{p1}}(0^-, y) = -\delta/2$ ile panelin sol tarafındaki değerler elde edilir. Panelin sağ tarafındaki değerler ise $\overrightarrow{F_{p1}}(0^+, y) = -\delta/2$ şeklinde olacaktır. $\overrightarrow{F_{p1}}$ fonksiyonunun sadece sol tarafını alıp, sağ tarafını sıfıra eşitleyebiliriz. Böylece panelin tek tarafında oluşan potansiyel değerleri elde edebiliriz [14]. Bir diğer husus ise panalin rotasyonu yani panele açı verme durumudur ki bunun için Denklem (3.12) ve (3.13)'den faydalanılabilinir;

$$x(d,e) = (xx(d) - xs).\sin(\beta) - (yy(e) - ys).\cos(\beta) \qquad \textbf{(3.12)}$$

$$y(d,e) = (yy(e) - ys).sin(\beta) - (xx(d) - xs).cos(\beta) \qquad \textbf{(3.13)}$$

Denklem (3.12) ve (3.13) eşitliklerinde panelin bulunduğu *x* ve *y* koordinatlarını değiştirmek için uygulanacak dönüşüm mevcuttur. *xs* ve *ys* parametreleri paneli orijin (*0,0*) noktasından ne kadar kayacağını belirlerken β açısı ise hangi yöne rotasyon yapılacağını ifade eder.

Şekil 3.9'da açılı olarak panel yerleşimi gösterilmiş olup β (rotasyon) açısı 135 derece olarak alınmıştır. Ayrıca panel gücü değeri $\delta = 100$ ve panel boyu (genişliği) değeri $l = 4$ olarak belirlenmiştir.



Şekil 3.9: Panelin oluşturduğu açılı potansiyel alanın gösterimi.

3.2.3 Noktadan çeken ve iten yapay potansiyel alan

Denklem (3.14)' deki gibi verilen iki bilinmeyenli normal iki değişkenli fonksiyon oval/eliptik şekilli bir form üretir [13]. Bu fonksiyon çeken yönlü kuvvetlerin oluşturulması için kullanılabilir.

$$F_c(x,y) = -\log\left(\delta((x-x_h)^2 + \gamma(y-y_h)^2)\right) \quad \textbf{(3.14)}$$

Robotun pozisyonunun x, y noktası olduğu kabul edildiğinde Denklem (3.14)'deki eliptik çeken alan fonksiyonun merkez noktası (x_h, y_h) ile temsil edilmiştir. Kontrol değişkeni γ, kümenin dış merkezini etkileyen, küçük eksenin (y-ekseni yönünde) büyük eksene olan oranını gösterir. γ değişkeninin aldığı değere göre bu eşitlik, dairesel veya eliptik bir form meydana getirmektedir. Eğer $\gamma = 1$ olursa fonksiyon dairesel şekilde, aksi taktirde elips şeklinde bir form oluşturacaktır. δ katsayısı potansiyel alanın kuvvet çarpanıdır. Bu katsayı çeken alan için, her zaman toplam potansiyel alanda hedefe gidilmesi amacıyla, iten alanların katsayılarına nispeten, çok büyük bir değer seçilir.

Denklem (3.3) ile ifade edildiği üzere, her bir (x, y) noktasındaki toplam potansiyel değer, parametrik olarak yönlendirilebilir ve şiddeti bir kat sayıya istinaden ayarlanabilirdir. Bu durum yol planlaması açısından ele alındığında potansiyel alan

içerisinde yer alan engellere ve hedeflere farklı vektörel tanımlamaların yapılabilmesine olanak sağlar.

İten ve çeken potansiyellerin her hangi bir (x, y) noktasındaki toplamının gradienti alınarak her bir noktadaki toplam potansiyel kuvvet bileşenleri hesaplanabilir. Bu durumda iki boyutlu olarak çeken potansiyel alanları oluştururken kullandığımız Denklem (3.14) için her bir (x, y) noktasının gradienti (∇) Denklem (3.15)'de gösterildiği şekilde hesaplanabilmektedir:

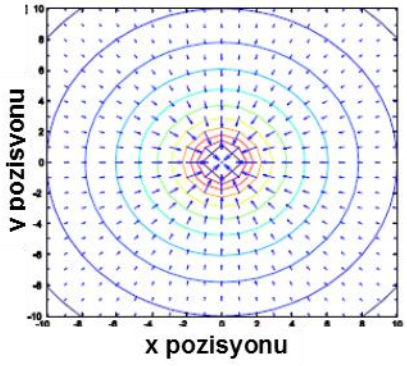
$$d_x = -2(x - x_h)/((x - x_h)^2 + \gamma(y - y_h)^2)$$
$$d_y = -2\gamma(y - y_h)/((x - x_h)^2 + \gamma(y - y_h)^2) \quad \textbf{(3.15)}$$

İki boyutlu yapay potansiyel alan içerisinde tanımlanacak olan eliptik biçimli çeken ya da iten nitelikteki bir engel yada hedef nokta bir açısal eksende döndürülebilmelidir. Bu döndürme hareketi elde edilen potansiyel alan vektörlerinin şiddetlerini ve yönlerini değiştirmek için kullanılabilir.

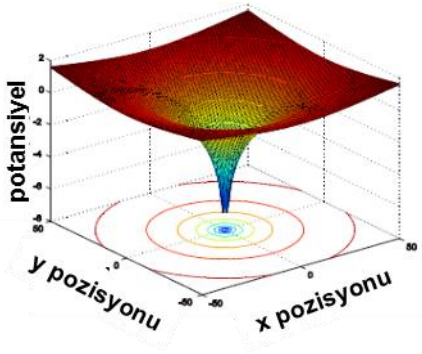
$$x_{rot} = \cos(\beta)\,(x - x_c) - \sin(\beta)\,(y - y_c)$$
$$y_{rot} = \sin(\beta)\,(x - x_c) + \cos(\beta)\,(y - y_c) \quad \textbf{(3.16)}$$

Dairesel bir alan için ($\gamma = 1$ olduğu durumda) döndürmeye gerek olmasa da eliptik biçimde tanımlanmış bir alan için ($\gamma \neq 1$ olduğu durumda) bu parametre bilgisi Denklem (3.16)'dan faydalanılarak üretilebilir [13]. β açısı iki boyutlu alan içerisinde tanımlanan eliptik potansiyel alanın y ekseni ile yaptığı açıyı göstermektedir.

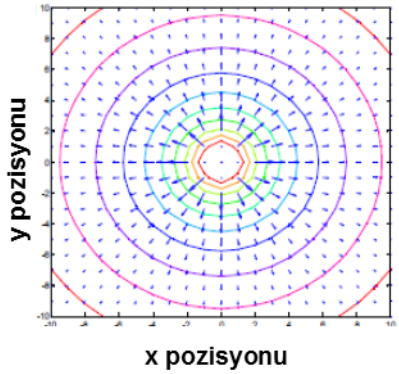
Engelleri modellemek için aynı eşitlik (-) değer ile çarpılarak kullanılmıştır. Bu bölümde tanımlanan iten ve çeken potansiyel alanlar ile bu potansiyel alanların gradienti alınarak hesaplanan vektör alanlar Şekil 3.10'da gösterilmiştir. Burada engelin etrafında kuvvet alan vektörlerinin daha şiddetli, engelden uzaklaştıkça şiddetinin azaldığı görülmektedir. Bu durum potansiyel alan içerisinde engelleri tanımlarken istenilen bir durumdur. Böylelikle aracın engele yaklaşması başka bir alan etkisinde kalındığında dahi mümkün olmayacaktır.



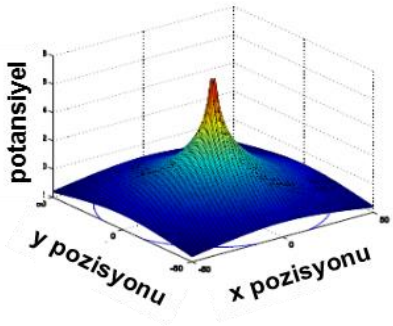
(a) Çeken kuvvet vektörlerinin gösterimi.



(b) Çeken kuvvet alanının potansiyeli.



(c) İten kuvvet vektörlerinin gösterimi.



(ç) İten kuvvet alanının potansiyeli.

Şekil 3.10: YPA yaklaşımının gösterilmesi.

Bu tez çalışması kapsamında noktadan çeken alanların potansiyel değerlerinin ($F_{c_{rot}}$) hesaplanması için Denklem (3.17)'den, noktadan iten alanların potansiyel değerlerinin ($F_{I_{rot}}$) hesaplanması için ise Denklem (3.18)'den faydalanılmıştır.

$$F_{c_{rot}}(x,y) = -log\left(\delta_c.\begin{pmatrix}\left(sin(\beta_c).(y-C_y)-cos(\beta_c).(x-C_x)\right)^2 - \\ \gamma_c.\left(sin(\beta_c).(x-C_x)+cos(\beta_c).(y-C_y)\right)^2\end{pmatrix}\right) \quad \textbf{(3.17)}$$

$$F_{I_{rot}}(x,y) = log\left(\delta_I.\begin{pmatrix}\left(cos(\beta_I).(y-I_y)-sin(\beta_i).(x-I_x)\right)^2 - \\ \gamma_I.\left(sin(\beta_I).(x-I_x)+cos(\beta_I).(y-I_y)\right)^2\end{pmatrix}\right) \quad \textbf{(3.18)}$$

Sırasıyla çeken ve iten potansiyel alanların δ_c ve δ_I parametreleri güç çarpan değerlerini, γ_c ve γ_I parametre değerleri alanların eliptik formlarını (eksen oranlarını), β_c ve β_I parametre değerleri ise alanların y-ekseni ile yaptıkları dönme açısını temsil etmektedir.

Noktadan çeken alanların potansiyel fonksiyonlarından ($F_{c_{rot}}$) iki boyutlu çeken vektör alanlar ($\overrightarrow{F_{c_{rot}}}$) elde edilmek istenir ise Denklem (3.19)'dan, noktadan iten alanların potansiyel fonksiyonlarından ($F_{I_{rot}}$) iki boyutlu iten vektör alanlar ($\overrightarrow{F_{I_{rot}}}$) elde edilmek istenir ise Denklem (3.20)'den faydalanılır.

$$\nabla F_{c_{rot}}(x,y) = \frac{\partial F_{c_{rot}}}{\partial x}\hat{\imath} + \frac{\partial F_{c_{rot}}}{\partial y}\hat{\jmath} \tag{3.19}$$

$$\nabla F_{I_{rot}}(x,y) = \frac{\partial F_{I_{rot}}}{\partial x}\hat{\imath} + \frac{\partial F_{I_{rot}}}{\partial y}\hat{\jmath} \tag{3.20}$$

Denklem (3.19) ve (3.20) sonucunda *x*-ekseninde oluşan değişim sırasıyla çeken ($DX_{c(k)}$) ve iten ($DX_{I(k)}$) alanlar için her bir *(x,y)* noktası için Denklem (3.21) ve (3.22) hesaplanarak elde edilebilir.

$$DX_{c(k)}(x,y) = \frac{\partial F_{c_{rot}}(x,y)}{\partial x}$$

$$= \frac{\left(2\gamma_c.\left(cos(\beta_c) + sin(\beta_c)\right).\left(cos(\beta_c).(C_y - x) + sin(\beta_c).(C_x - x)\right)\right)}{\left(\left(cos(\beta_c).(C_x - y) - sin(\beta_c).(C_y - y)\right)^2 + \gamma_c.\left(cos(\beta_c).(C_y - x) + sin(\beta_c).(C_x - x)\right)^2\right)} \tag{3.21}$$

$$DX_{I(k)}(x,y) = \frac{\partial F_{I_{rot}}(x,y)}{\partial x}$$

$$= -\frac{\left(2\gamma_I.\left(cos(\beta_I) + sin(\beta_I)\right).\left(cos(\beta_I) * (I_y - x) + sin(\beta_I).(I_x - x)\right)\right)}{\left(\left(cos(\beta_I).(I_x - y) - sin(\beta_I).(I_y - y)\right)^2 + \gamma_I.\left(cos(\beta_I).(I_y - x) + sin(\beta_I).(I_x - x)\right)^2\right)} \tag{3.22}$$

Aynı şekilde Denklem (3.19) ve (3.20) sonucunda *y*-ekseninde oluşan değişim sırasıyla çeken ($DY_{c(k)}$) ve iten ($DY_{I(k)}$) alanlar için her bir *(x,y)* noktası için Denklem (3.23) ve (3.24) hesaplanarak elde edilebilir.

$$DY_{c(k)}(x,y) = \frac{\partial F_{C_{rot}}(x,y)}{\partial y}$$

$$= \frac{2.\left(cos(\beta_c) - sin(\beta_c)\right).\left(cos(\beta_c).(C_x - y) - sin(\beta_c).(C_y - y)\right)}{\left(\left(cos(\beta_c).(C_x - y) - sin(\beta_c).(C_y - y)\right)^2 + \gamma_c.\left(cos(\beta_c).(C_y - x) + sin(\beta_c).(C_x - x)\right)^2\right)} \tag{3.23}$$

$$DY_{I(k)}(x,y) = \frac{\partial F_{I_{rot}}(x,y)}{\partial y}$$

$$= -\frac{\left(2.\left(cos(\beta_I) - sin(\beta_I)\right).\left(cos(\beta_I) * (I_x - y) - sin(\beta_I).(I_y - y)\right)\right)}{\left(\left(cos(\beta_I).(I_x - y) - sin(\beta_I).(I_y - y)\right)^2 + \gamma_I.\left(cos(\beta_I).(I_y - x) + sin(\beta_I).(I_x - x)\right)^2\right)} \tag{3.24}$$

İten veya çeken nitelikteki birden fazla alanı bir arada kullanabilmek için *k* parametresi tanımlanmıştır.

3.2.4 Saat yönünde ve tersine dönen yapay potansiyel alan

Çalışma kapsamında faydalanılan bir sonraki temel yapay potansiyel alan saat yönünde veya tersine dönen bir vektör alan oluşturmak için faydalanılan potansiyel alanlardır. Noktadan iten veya çeken potansiyel alanların tanımına benzer bir yaklaşım ile davranıldığında saat yönünde ($F_{rp_{rot}}$) ve tersine ($F_{rn_{rot}}$) döndürme hareketini iki boyutlu ortamda sağlayan potansiyel alan denklemleri sırasıyla Denklem (3.25) ve (3.26) ile gösterilmiştir.

$$F_{rp_{rot}}(x,y) = 1 - 2.\left(log\left(-\delta_{rp}.\left(\left(x - rp_x\right)^2 + \left(y - rp_y\right)^2\right)\right)\right) \quad \textbf{(3.25)}$$

$$F_{rn_{rot}}(x,y) = -\left(1 - 2.\left(log\left(-\delta_{rn}.\left((x - rn_x)^2 + \left(y - rn_y\right)^2\right)\right)\right)\right) \quad \textbf{(3.26)}$$

Denklem (3.25) ve (3.26)'dan görüldüğü üzere saat yönünde dönen alan oluşturan hareketin merkezi (rp_x, rp_x) parametreleri, saat yönünün tersine alan oluşturan hareketin merkezi ise (rn_x, rn_y) parametreleri ile gösterilmektedir. Aynı şekilde sırasıyla saat yönünde ve saat yönünde dönen potansiyel alanların güç çarpanı δ_{rp} ve δ_{rn} parametreleri ile temsil edilmektedir.

$$\nabla F_{rp_{rot}}(x,y) = \frac{\partial F_{rp_{rot}}}{\partial x}\hat{\imath} - \frac{\partial F_{rp_{rot}}}{\partial y}\hat{\jmath} \quad \textbf{(3.27)}$$

$$\nabla F_{rn_{rot}}(x,y) = \frac{\partial F_{rn_{rot}}}{\partial x}\hat{\imath} - \frac{\partial F_{rn_{rot}}}{\partial y}\hat{\jmath} \quad \textbf{(3.28)}$$

Denklem (3.27) ve (3.28) sonucunda *x*-ekseninde oluşan değişim sırasıyla çeken ($DX_{Rp(k)}$) ve iten ($DX_{Rn(k)}$) alanlar için her bir *(x,y)* noktası için Denklem (3.29) ve (3.30) hesaplanarak elde edilebilir.

$$DX_{Rp(k)}(x,y) = \frac{\partial R_p(x,y)}{\partial x} = \frac{\left(4.\left(rp_x - x\right)\right)}{\left(\left(rp_x - x\right)^2 + \left(rp_y - y\right)^2\right)} \quad \textbf{(3.29)}$$

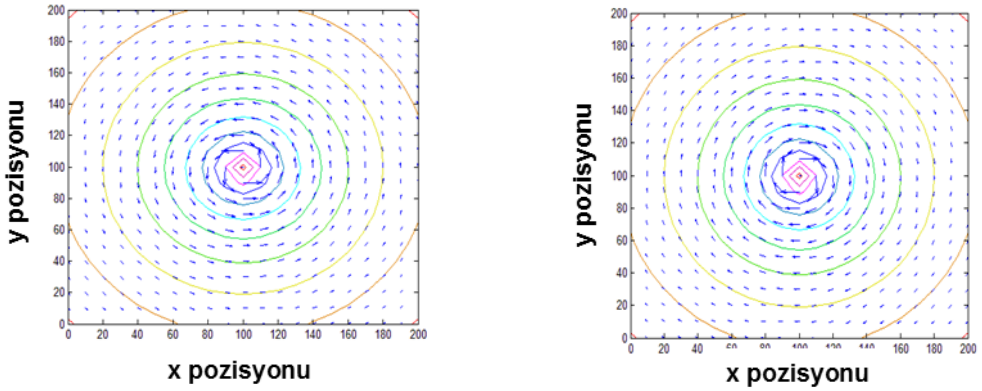
$$DX_{Rn(k)}(x,y) = \frac{\partial R_n(x,y)}{\partial x} = -\frac{\left(4.\left(rn_x - x\right)\right)}{\left((rn_x - x)^2 + \left(rn_y - y\right)^2\right)} \quad \textbf{(3.30)}$$

Aynı şekilde Denklem (3.27) ve (3.28) sonucunda *y*-ekseninde oluşan değişim sırasıyla çeken ($DY_{Rp(k)}$) ve iten ($DY_{Rn(k)}$) alanlar için her bir *(x,y)* noktası için Denklem (3.31) ve (3.32) hesaplanarak elde edilebilir.

$$DY_{Rp(k)}(x,y) = \frac{\partial R_p(x,y)}{\partial y} = \frac{\left(4.\left(rp_y - y\right)\right)}{\left(\left(rp_x - x\right)^2 + \left(rp_y - y\right)^2\right)} \tag{3.31}$$

$$DY_{Rn(k)}(x,y) = \frac{\partial R_n(x,y)}{\partial y} = -\frac{\left(4.\left(rn_y - y\right)\right)}{\left(\left(rn_x - x\right)^2 + \left(rn_y - y\right)^2\right)} \tag{3.32}$$

Sonuç olarak Denklem (3.25) ve (3.26) ile tanımlanan saat yönünde ve tersine dönme hareketini oluşturmak amacıyla faydalanılan potansiyel alan denklemlerinden Şekil 3.11 ile gösterilen vektör alanlar elde edilmektedir.



(a) Saat yönünün tersine dönen vektör alan gösterimi.

(b) Saat yönünde dönen vektör alan gösterimi.

Şekil 3.11: Dönen vektör alanların gösterilmesi.

3.3 Birleştirilmiş Yapay Potansiyel Alanlar

Çalışmanın bu kısmına kadar yol planlaması amacıyla davranışı yönlendiren temel potansiyel alanlar ile bu alanlardan üretilmiş vektör alanaların nasıl elde edildiğine değinilmiştir. Çalışmanın bundan sonraki bölümünde ise bu temel alanların nasıl bir araya getirilerek bir arada kullanıldığına değiniliecektir.

3.3.1 Temel yapay potansiyel alan sınır fonksiyon tanımlamaları

Birden fazla temel potansiyel alanı bir arada kullanmak için önce temel potansiyel alan etkilerinin nasıl sınırlandırıldığı incelenmesi gereken bir noktadır [65]. Bu

amaçla temel potansiyel alan etkilerini hareket sahası içinde nasıl sınırlandırılacağını belirleyen genel fonksiyon tiplerine değinilmiştir.

(a) Keskin sınırlandırma fonksiyonu. (b) Doğrusal sınırlandırma fonksiyonu. (c) Üssel sınırlandırma fonksiyonu.

Şekil 3.12: Potansiyel alan içerisindeki kuvvetlerin sınırlandırılması.

Literatürde sıklıkla kullanılan ve potansiyel alan etkilerinin sınırlandırılması amacıyla faydalanılan örnek sınır fonksiyon temsilleri Şekil 3.12'de gösterilmektedir. Örneğin bir engele ait vektör alanı ele alındığında, kuvvet çarpanı engele olan uzaklığa bağlı olarak bir mesafeden önce veya sonra Şekil 3.12 (a)'da gösterildiği şekilde sıfırlanabileceği gibi mesafeye bağlı olarak değişkenlik gösterebilen Şekil 3.12 (b) veya Şekil 3.12 (c)'dekine benzer bir formda azalıp, artabilir. Bu değişim belirlenen mesafeye bağlı sınır fonksiyonuna bağlı olarak lineer ya da üssel bir karaktere sahip olabilir. Sınır fonksiyonlarına örnek fonksiyonları arttırmak mümkündür. Ancak özetle sınır fonksiyonları, alan üzerinde yer alan noktalarda oluşan potansiyel alan etkisini potansiyel alan kaynağına olan mesafeye bağlı olarak bir fonksiyon karakterinde değiştiren yapılar olarak tanımlanabilir.

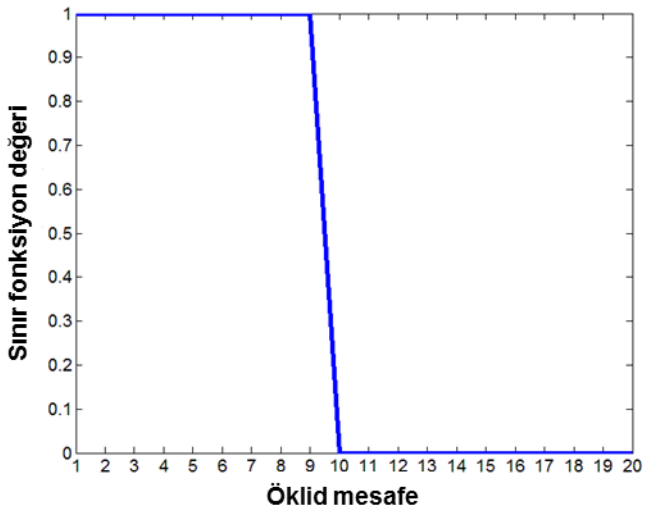
Bu tez kapsamında iki farklı sınır fonksiyon tipinden faydalanılmıştır. Bunlar hızlı hesaplanabilir olması ve keskin sonuçlar üretebilmesi açısından faydalanılan ikili (binary) sınır fonksiyonları ile mobil platformların hareketlerine daha uygun yumuşak geçişlere sahip sigmoid fonksiyonlardır.

İkili fonksiyonlar mantık operatörleri vasıtasıyla oldukça kolay üretilebilir. Örneğin çeken potansiyel alan kaynağı (C_x, C_y) ile halihazırda içinde bulunulan hücre koordinatları (x, y) arasındaki öklid uzaklığı d parametresi ile temsil edilsin. Örnek olarak belirlenen çeken potansiyel kaynağın etkisi d parametresi ile belirlenmiş bir değerden uzak olan noktalar için sıfır kabul edilsin. Bu durumda her bir (x,y) noktası için sınır fonksiyon değerleri $(S_C\ (x, y))$ Denklem (3.33) ile gösterildiği şekilde hesaplanabilir.

$$S_{C_b}(x,y) = \begin{cases} 0 \; for \left(\sqrt{(x-C_x)^2 + \left(y-C_y\right)^2} \right) > d \\ 1 \; for \left(\sqrt{(x-C_x)^2 + \left(y-C_y\right)^2} \right) \leq d \end{cases} \quad \textbf{(3.33)}$$

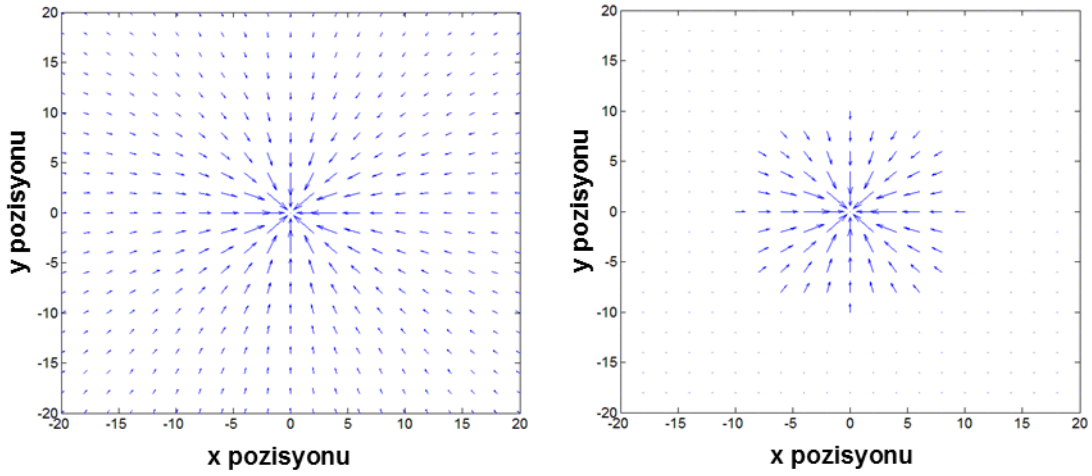
Şayet elde edilen sınır fonksiyon değerleri potansiyel alandan üretilmiş gradient işlemi sonucundaki vektör değerlerine Denklem (3.34) ile gösterildiği şekilde etki ettirilir ise elde edilen çeken vektör alan değerleri Denklem (3.33) ile gösterilen *d* parametresine bağlı ikili fonksiyon ile sınırlandırılmış olur.

$$\begin{aligned} DX_{c(k)}(x,y) &= \frac{\partial F_{c_{rot}}(x,y)}{\partial x} . S_{C_b}(x,y) \\ DY_{c(k)}(x,y) &= \frac{\partial F_{c_{rot}}(x,y)}{\partial y} . S_{C_b}(x,y) \end{aligned} \quad \textbf{(3.34)}$$



Şekil 3.13: İkili sınır fonksiyon değer değişimi.

Bu durumda, *d* parametre değeri 10 olarak belirlenir ise mesafeye bağlı olarak tek boyutta sınır fonksiyon değişimi Şekil 3.13 ile gösterildiği biçimde gerçekleşir.

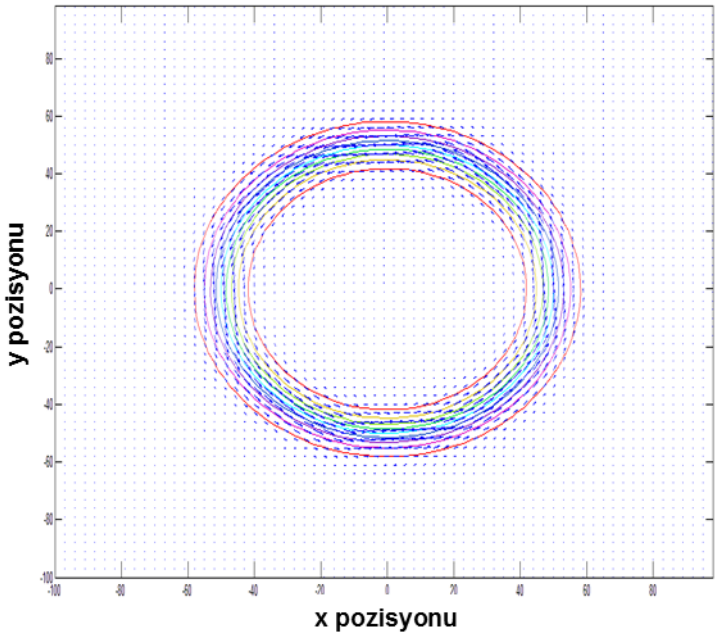


(a) Sınırlandırılmamış çeken alan gösterimi.

(b) Sınırlandırılmış çeken alan gösterimi.

Şekil 3.14: Vektör alan sınırlandırılması.

Şekil 3.14 (a) ile çeken potansiyel alandan elde edilmiş ve sınırlandırılmamış vektör alan gösterilmektedir. Aynı çeken potansiyel alan ikili sınırlandırma fonksiyonu ile ve *d* parametre değeri 10 olarak belirlenerek sınırlandırıldığında Şekil 3.14 (b) de gösterilen vektör alan elde edilebilir.

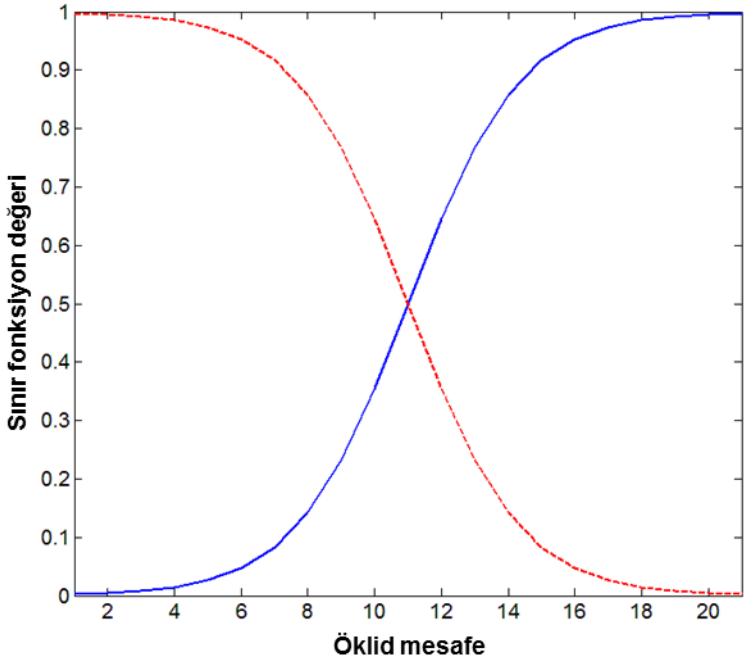


Şekil 3.15: Sınırlandırılmış saat yönü tesine dönen alan gösterimi.

Çeken vektör alan değerleri Denklem (3.33) ile gösterilen *d* parametresine bağlı ikili fonksiyon ile sınırlandırılmıştır. Ancak sınırlandırma her zaman tek taraflı olmak zorunda değildir. İki boyutlu ortamda sınırlandırma her iki taraftan, içten ya da dıştan gerçekleştirilebilir. Şekil 3.15 ile saat yönü tersine dönen vektör alanın hem içten

hem de dıştan sınırlandırıldığı görülmektedir. Yaklaşım Denklem (3.33) ile gösterilen ile aynı olup sınırlandırma değerleri bu kez d_1 *ve* d_2 olarak belirlenen iki farklı parametreye göre gerçekleştirilmiştir.

İkili fonksiyonlar kullanarak potansiyel alanı sınırlandırmak her ne kadar uygulaması basit ve de hesaplaması kolay olsa dahi özellikle İHA gibi otonom sistemler tarafından uygulaması güç sonuçlar oluşturabilmektedir. Çünkü alan değerlerinde oluşan keskin değişimler yol planlamasında da hızlı yön ve sürat değişimlerine sebebiyet verecektir ki bu durum otonom sistemler tarafından arzu edilen bir sonuç değildir. Ancak ikili fonksiyonlar hızlı etki üretmesi istenen durumlarda da faydalanılacak yapılardır. Bu nedenle ikili sınır fonksiyonlar ile birlikte potansiyel alan sınırlamalarını daha yumuşak ortaya koyan sigmoid fonksiyonların kullanımına ihtiyaç oluşmuştur. Tez kapsamında ikili değer üreten mantıksal denklemlerden oluşturulmuş sınır fonksiyonları ile birlikte sigmoid fonksiyonlardan faydalanılmıştır. Böylece özellikle alan geçişleri esnasında daha yumuşak manevra komutları üreten yapıda ve bölge sınırlarına yakın bölgelerde geçişlere göre değişiklik gösterebilen sonuçlar üretilmiştir.



Şekil 3.16: Temel azalan ve artan sigmoid sınır fonksiyon gösterimi.

Sigmoid sınır fonksiyonları aynı zamanda bulanık mantık tabanlı kontrol sistemlerinde üyelik fonksiyonlarının tanımlanmasında da kullanılan bir yaklaşımdır [66]. Temel azalan (kesikli çizim) ve artan (düz çizim) sigmoid sınır fonksiyonları

Şekil 3.16'da gösterilmiştir. Artan sigmoid sınır fonksiyonu Denklem (3.35) ile gösterilen fonksiyon yardımıyla, azalan sigmoid sınır fonksiyonu ise Denklem (3.36) ile gösterilen fonksiyon yardımıyla elde edilebilir. *Slope* değeri eğrinin kırılım noktasını, ρ değeri ise fonksiyonun kaplama alanını temsil etmektedir.

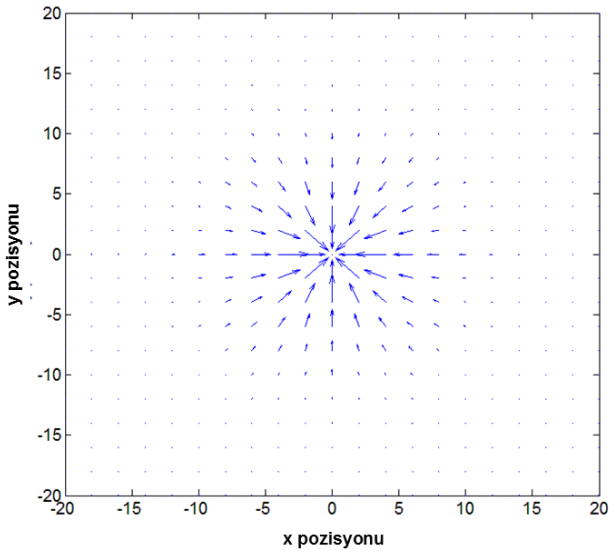
$$S^{+}(d) = \frac{1}{1 + e^{-(Slope*d)*\rho}} \tag{3.35}$$

$$S^{-}(d) = 1 - \left(\frac{1}{1 + e^{-(Slope*d)*\rho}}\right) \tag{3.36}$$

Aşağıdaki Denklem (3.37)'den faydalanılarak her bir (x, y) noktasındaki sigmoid sınır fonksiyon değeri hesaplanabilir. Burada ayırt edilmesi gereken nokta alanın sönümlenmek ya da oluşturulmak istenmesidir.

$$S_{C_S}(x,y) = \begin{cases} S^{+}\left(\sqrt{(x - C_x)^2 + \left(y - C_y\right)^2}\right) & güçlenen\ alan\ için \\ S^{-}\left(\sqrt{(x - C_x)^2 + \left(y - C_y\right)^2}\right) & sönümlenen\ alan\ için \end{cases} \tag{3.37}$$

Bu durumda sınırlandırılmamış formu Şekil 3.14 (a) ile gösterilmiş çeken potansiyel alanın ρ değeri 10 olarak belirlenmiş sigmoid fonksiyon ile sınırlandırılması sonucunda elde edilen vektör alan Şekil 3.17 ile gösterilmiştir.



Şekil 3.17: Sigmoid fonkisyon ile sınırlandırılmış çeken vektör alan gösterimi.

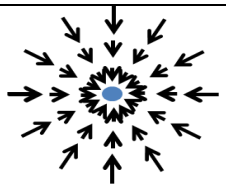
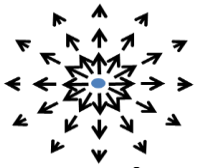
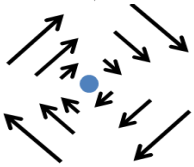
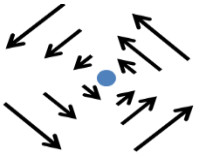
Şekil 3.17 ile gösterilen sigmoid fonksiyon ile sınırlandırılmış çeken alan ile Şekil 3.14 (b) ile gösterilen ikili fonksiyon ile sınırlandırılmış çeken alan

karşılaştırıldığında alanın başlangıç noktasındaki keskin durum ile yumuşak geçiş çok net gözlemlenmektedir. Bu durum özellikle birden fazla temel alanın bir arada kullanıldığı ve de artan ile azalan sigmoid fonksiyonların karşılıklı olarak kullanıldığı durumlarda çalışmanın ilerleyen kısımlarında örneklendiği şekilde otonom araçlar tarafından büyük uygulanabilirlik avantajı sağlamaktadır.

3.3.2 Birleştirilmiş toplam vektör alanın hesaplanması

Çalışmanın bu bölümünde YPA yaklaşımı ile vektör alanın hesaplanmasına yönelik olarak Çizelge 3.1 ile gösterilen temel fonksiyonlardan birden fazlasının bir arada nasıl kullanılacağına değinilmiştir. Bu kapsamda kullanılacak olan temel potansiyel modellerinden aynı modelden birden fazlası veya farklı temel modellerden birkaçı aynı alanın modellenmesinde bir arada kullanılabilir. Çizelge 3.1 ile çalışma kapsamında kullanılan potansiyel alanlar ile bunların oluşturulmasında kullanılacak olan potansiyel ve vektör modelleri özetlenmiştir.

Çizelge 3.1: Temel potansiyel alan modellemeleri.

No	Adı	Potansiyel Model	Vektör Model	Grafik Temsili
1	Noktasal Kaynağa Çeken Potansiyel Alan Modeli	Denklem (3.17)	Denklem (3.21) ve (3.23)	
2	Noktasal Kaynaktan İten Potansiyel Alan Modeli	Denklem (3.18)	Denklem (3.22) ve (3.24)	
3	Noktasal Kaynağa Teğet Alan Modeli (Saat Yönünde)	Denklem (3.25)	Denklem (3.29) ve (3.31)	
4	Noktasal Kaynağa Teğet Alan Modeli (Saat Yönü Tersine)	Denklem (3.26)	Denklem (3.30) ve (3.32)	

Bir alan içerisinde yer alan çok sayıdaki temel potansiyel alan değerlerinin *(x, y)* noktasına etki eden değerlerin toplamı o noktadaki toplam potansiyel değeri üretir. Bu noktadan hareket ile *m* adet noktasal çeken, *n* adet noktasal iten, *p* adet saat yönünde dönen, *r* adet saat yönü tersine dönen alan için *(x, y)* noktasındaki toplam

potansiyel değerlerin x bileşeni ($SX(x,y)$) ve y bileşeni ($SY(x,y)$) değerleri Denklem (3.38)'de gösterilen şekilde elde edilir.

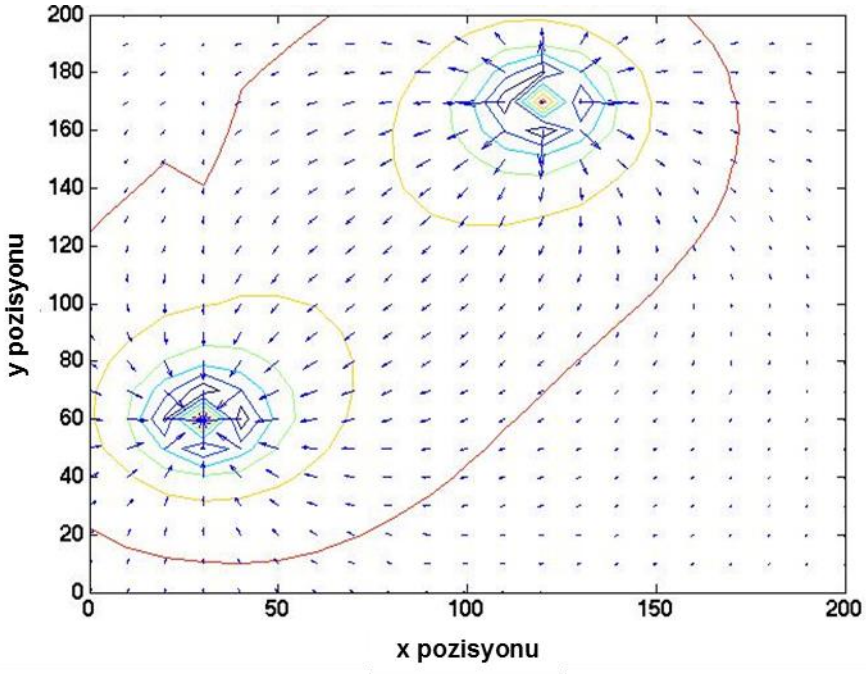
$$SX(x,y) = \sum_{k=0}^{m} DX_{c(k)}(x,y) + \sum_{k=0}^{n} DX_{I(k)}(x,y) + \sum_{k=0}^{p} DX_{Rp(k)}(x,y) + \sum_{k=0}^{r} DX_{Rn(k)}(x,y)$$
$$SY(x,y) = \sum_{k=0}^{m} DY_{c(k)}(x,y) + \sum_{k=0}^{n} DY_{I(k)}(x,y) + \sum_{k=0}^{p} DY_{Rp(k)}(x,y) + \sum_{k=0}^{r} DY_{Rn(k)}(x,y) \quad \textbf{(3.38)}$$

Örnek olarak içerisinde bir adet çeken ve bir adet de iten noktasal potansiyel kaynağın yer aldığı alan modellenmiştir. Aynı alan içerisinde bir adet iten ve bir adet çeken noktasal potansiyel kuvvet olduğu varsayıldığında ve alanın hesaplanmasına yönelik olarak Çizelge 3.1 verilen ilgili denklemlerde Çizelge 3.2 ile verilen parametre değerleri kullanarak yapılan hesaplama sonucunda Şekil 3.18 ile gösterilen vektör alan elde edilir.

Çizelge 3.2: Birleştirilmiş potansiyel alanda yer alan temel alan parametreleri.

Parametre	**Değer**
$size\ (Alan\ Boyut)$	200x200
$Res\ (Alan\ \text{Çöznürlük})$	10
δ_c	1.0
γ_c	1.0
C_x, C_y	30 60
β_c	π
δ_I	1.0
γ_I	1.0
I_x, I_y	120,170
β_I	π

Denklem (3.38) incelendiğinde ve bu denklem ile hesaplanan Şekil 3.18 ile gösterilen toplam vektör alan göz önüne bulundurulduğunda toplam alan içerisinde yer alan temel alanların her hangi bir sınır fonksiyonu ile sınırlanmadan toplandığı görülmektedir.



Şekil 3.18: Bir iten ve bir çeken temel alan ile oluşturulan toplam alan gösterimi.

Oysa daha önceden de açıklandığı üzere birden fazla temel alanın bir arada kullanıldığı toplam alanlarda temel potansiyel alan değerlerinin sınırlandırılması ihtiyacı tanımlanmıştır. Bu nedenle Denklem (3.38) ile gösterilen eşitlik Denklem (3.39)'da görüldüğü üzere sınır fonksiyon parametreleri eklenerek güncellenmiştir. Bu örnekte sigmoid sınır fonksiyonları seçilmiş olup, ihtiyaca göre ikili sınır fonksiyonları da aynı şekilde kullanılabilmektedir.

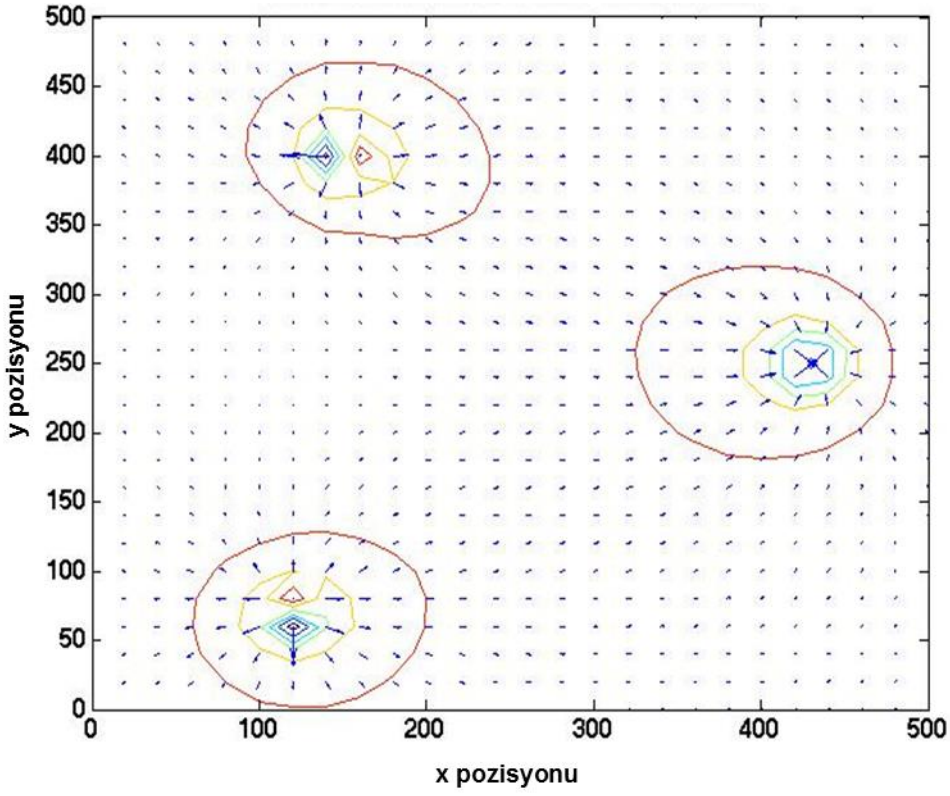
$$SX_s(x,y) = \sum_{k=0}^{m} S_{C(k)_s}(x,y).DX_{c(k)}(x,y) + \sum_{k=0}^{n} S_{I(k)_s}(x,y).DX_{I(k)}(x,y) + \sum_{k=0}^{p} S_{Rp(k)_s}(x,y).DX_{Rp(k)}(x,y) + \sum_{k=0}^{r} S_{Rn(k)_s}(x,y).DX_{Rn(k)}(x,y)$$

$$SY_s(x,y) = \sum_{k=0}^{m} S_{C(k)_s}(x,y).DY_{c(k)}(x,y) + \sum_{k=0}^{n} S_{I(k)_s}(x,y).DY_{I(k)}(x,y) + \sum_{k=0}^{p} S_{Rp(k)_s}(x,y).DY_{Rp(k)}(x,y) + \sum_{k=0}^{r} S_{Rn(k)_s}(x,y).DY_{Rn(k)}(x,y) \quad \textbf{(3.39)}$$

Aynı alan içerisinde iki adet sigmoid fonksiyon ile sınırlandırılmış iten ve bir adet çeken noktasal potansiyel kuvvet olduğu varsayıldığında ve alanın hesaplanmasına yönelik olarak Çizelge 3.3 ile verilen parametre değerleri kullanılarak Denklem (3.39)'da belirtildiği şekilde hesaplandığında Şekil 3.19 ile gösterilen vektör alan elde edilir.

Çizelge 3.3: Birleştirilmiş potansiyel alanda yer alan sınırlandırılmış temel alan parametreleri.

Parametre	**Değer**
$size\ (Alan\ Boyut)$	500x500
$Res\ (Alan\ \text{Çöznürlük})$	20
δ_c	1.0
γ_c	1.0
C_x, C_y	30, 250
β_c	Π
$\delta_{I(1)}$	1.0
$\gamma_{I(1)}$	1.0
$I_{x(1)}, I_{y(1)}$	150, 400
$\beta_{I(1)}$	Π
$\delta_{I(2)}$	1.0
$\gamma_{I(2)}$	1.0
$I_{x(2)}, I_{y(2)}$	120, 0
$\beta_{I(2)}$	Π
$Slope$	0.5
ρ	10



Şekil 3.19: Sınırlandırılmış iki iten ve bir çeken temel alan ile oluşturulan toplam alan gösterimi.

3.4 YPA Tabanlı Yol Planlaması Problemleri Ve Çözüm Yöntemleri

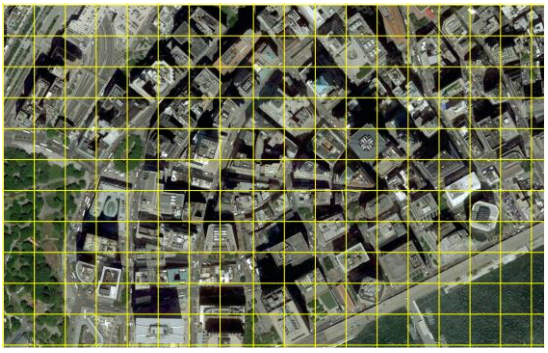
YPA yaklaşımının diğer yaklaşımlardan farklı olarak sürekli bir akışı modellemesi, alan içerisinde yer alan tüm araçlar ve engeller için bir arada modelleme imkanı sunması ve dinamik tanımlamalar için hızlı ve kolay olması gibi nedenlerden dolayı robotların yol yapılandırması için kullanışlı bir teknik haline getirmiştir [65]. Ancak tüm bu avantajlarının yanında YPA yaklaşımının temel formunun beraberinde getirdiği bir takım problemlerde mevcuttur. Bu problemler ilerleyen bölümde başlıklar halinde ele alınmış, literatürden faydalanarak ve çalışma kapsamında ortaya konulan yaklaşımlar ile çözüm üretilmeye çalışılmıştır.

3.4.1 Boyut problemi

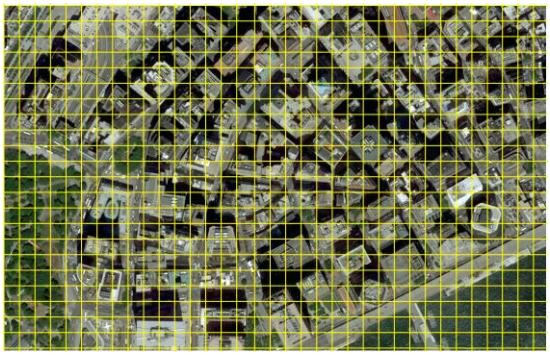
Tanımlanan YPA yaklaşımları literatür incelendiğinde daha çok iki boyutlu ve küçük ölçekli bir alan içerisinde etkin olarak kullanılan fonksiyonlar olarak görülmekte ve bu durumda karmaşıklığı 2'nci dereceden olan potansiyel fonksiyonlardan faydalanılmaktadır, oysa İHA gibi otonom sistemler üç boyutlu ve nispeten çok daha büyük ölçekli bir evrende yol planlaması ihtiyacı duymaktadırlar [67]. Üç boyutlu ve geniş ölçekli alanlar içinde doğal olarak çok daha fazla sayıda ızgara hücresi oluşacaktır. Bu durum hesaplama performansını olumsuz etkileyecektir. Bu nedenle bu ihtiyacı karşılamak ve de yüksek süratli İHA platformlarının gerçek zamana en yakın yol planlaması çözümünü üretmek amacıyla ilerleyen bölümlerde açıklandığı şekilde özgün olarak bu tez çalışması kapsamında ortaya konulan çok katmanlı modelleme ve paralel hesaplama tekniklerinden faydalanılmıştır. Bu yaklaşımların boyut problemine nasıl çözüm oluşturduğu simülasyon ortamında ve uygulamalı olarak ortaya konulmuştur.

3.4.2 Izgara çözünürlüğü

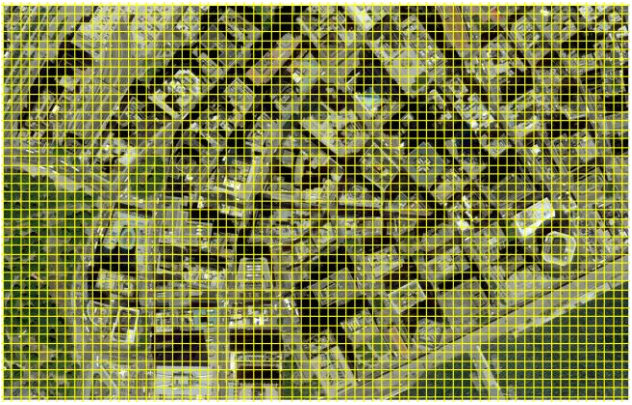
YPA tabanlı yol planlaması kapsamında karşılaşılan bir diğer problem ise ızgara çözünürlüğünün ne seçileceğidir. Aslında tüm ızgara tabanlı yaklaşımlarda optimize edilmesi gereken hususların başında yer alan problemlerden birisi ızgaranın çözünürlüğü dolayısıyla hücrelerin temsil ettiği büyüklüktür [11].



(a) Düşük grid çözünürlüğü.



(b) Uygun grid çözünürlüğü.



(c) Yüksek grid çözünürlüğü.

Şekil 3.20: Grid çözünürlüğü etkisinin gösterimi.

Şekil 3.20'de görüldüğü üzere hücre büyüklüğü çözümün sonucuna doğrudan etki etmektedir. Örneğin Şekil 3.20 (a)'daki gibi nispeten büyük olarak seçilmiş hücre ölçütlerinde, yani çözünürlüğün düşük olması durumunda olası birçok alternatif çözümden başlangıç durumunda alanın özelliklerine bağlı olarak vazgeçilmiş olunur. Hatta Şekil 3.20 (a)'daki örnekte görüldüğü üzere binaların bir engel olduğu kabul edildiğinde bir çözüm olmasına kaşın olmadığı sonucuna dahi varılabilir. Ancak bu durumun Şekil 3.20 (c)'deki gibi tam tersi ele alındığında da ciddi hesaplama performansı problemleri ile karşılaşılacaktır. Nitekim Şekil 3.20 (b)'deki gibi en iyi çözünürlük değeri tespit edilerek doğru çözüme ulaşmak ile en hızlı çözüme ulaşmak arasında önemli bir optimizasyon parametresi grid çözünürlüğüdür.

Bu tez kapsamında ele alınan esas problem YPA tabanlı yol planlamasının hesaplama performansıdır. Bu nedenle çalışma kapsamında ele alınan grid çözünürlükleri farklı değerlerde seçilerek deneysel biçimde en iyi sonuçları ve performans kriterlerini karşılayacak sonuca ulaşılmaya çalışılmıştır.

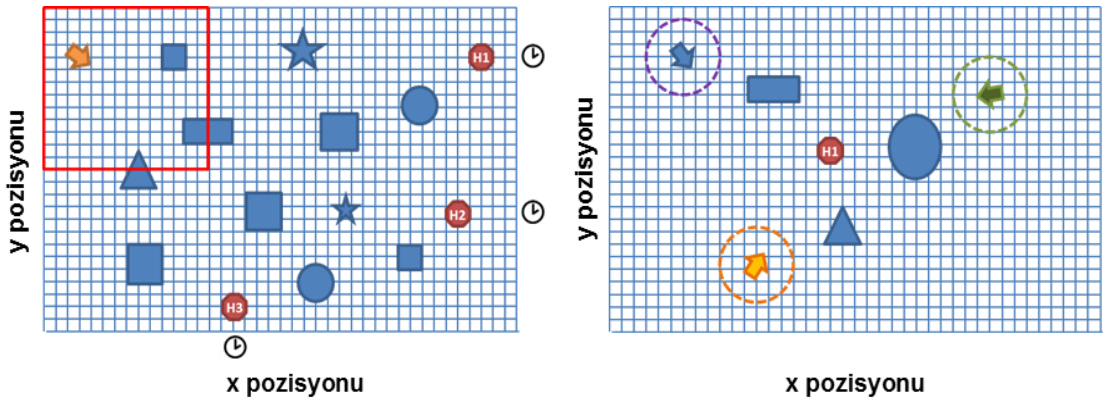
3.4.3 Yerel minimum problemi

Yerel minimum hedef dışında vektör kuvvetlerinin toplamının sıfır değerini aldığı ve kısmen hedef benzeri davranışlar sergileyen istenmeyen noktalar olarak tanımlanabilir. YPA yaklaşımının temel formu hedef nokta dışında düşük potansiyellerin oluşmasına, diğer bir değişle yerel minimum (local minima) durumu ile karşılaşılmasına neden olabilir ve istenmeyen hedef dışındaki hedef ile benzer özellikler gösteren takılma noktaları oluşması muhtemeldir. Robotun hedefe ulaşacağı bu durumda garanti edilemez [68]. Yol yapılandırması için potansiyel alanların genel formu hedef dışındaki lokal minimumların oluşumunu engellemez. Robot bu minimumlara maruz kalabilir ve hedef dışındaki sabit bir konuma ulaşabilir. Kodistcek geometrik argümanları kullanarak, en azından belirli etki alanları için, her zaman insansız araca neredeyse başlangıç noktasından gidilecek hedef noktaya kadar rehberlik edebilecek potansiyel fonksiyonlar olduğunu göstermiştir [65]. Yerel minimum durumundan robotun etkilenmesini engellemek için gürültü durumu oluşturulması (rastgele kuvvetler içeren geçici bir alan oluşturma durumu), Fark et-Sakın (avoid-past) yaklaşımı (robot her adımdan önce bulunduğu konum ve hareket ettiği hücre bilgisini tutar), teğet engel modellemesi (tangential obstacle) ile kaçınma yöntemi gibi farklı yaklaşımlar geliştirilmiştir [68]. Ancak bu yaklaşımların tamamı yerel minimum oluşmasını engellemek yerine robotun bu duruma maruz kaldığında yapması gerekenler üzerine geliştirilmiş veya yerel minimum durumundan kurtulmak üzerine geliştirilmiş yaklaşımlardır. Ancak literatürde yerel minimum olmasını engelleyecek şekilde modelleme yaklaşımlarına da rastlanmaktadır. Bunlardan birisi de harmonik fonksiyonlar olarak tabir edilen fonksiyonların potansiyel alanların modellemesinde kullanılması yaklaşımıdır [31][63]. İlerleyen bölümlerde harmonik fonksiyonların tanımı yapılmış ve YPA tabanlı yol planlamasında potansiyel alanların modellenmesinde nasıl kullanıldıklarına değinilmiştir.

3.4.4 Dinamik modelleme problemi

Khatib tarafından ortaya konulan YPA tabanlı yol planlaması yaklaşımının temel formu durağan şekilde tanımlanmış ve sadece bir hedef ile yine durağan olarak tanımlanmış engelleri modelleyebilir. Oysa bu tez kapsamında ortaya konulan yol

planlaması ihtiyacı özellikle dinamik özelliklere sahip hareketli engellerin ve hedef nokataların modellenmesine ihtiyaç duymaktadır.



(a) Yerel yol planlaması. (b) Genel yol planlaması

Şekil 3.21: Dinamik modelleme problemi gösterimi.

Şekil 3.21 (a) ile iki problem bir şekil üzerinde gösterilmiştir. Bunlardan ilki yerel yol planlaması durumunda ortaya çıkmaktadır. Mobil robotun özelliklerini bir başka değiş ile alan içerisinde yer alan engelleri önceden bilmediği ve kırmızı çerçeve ile gösterilen alan içerisindeki engelleri üzerindeki algıyacılar ile görevin icrası esnasında tespit edebildiği kabul edildiğinde icra edebileceği yol planlaması çerçeve ile tanımlanan alan içindeki yerel yol planlaması olacaktır. Robot alan içinde *H1* ile gösterilen hedefe doğru ilerledikçe yeni engelleri tespit edecektir. Bu durumda her ilerleme adımında yerel yol planlamasını tekrarlamasının yanı sıra her yeni engel noktası tespitinde bu yeniden hesaplamayı gerçekleştirmesi gerekecektir. Bu durum hesaplama maliyetleri açısından oldukça yük getirecek bir durumdur. Bir diğer problem ise yine Şekil 3.21 (a) ile gösterildiği biçimde alan içerisinde zamana bağlı olarak uğranması gereken birden fazla hedef nokta olması durumudur. Bir başka değiş ile hedef konumun dinamik ya da hareketli olması halidir. Bu durumda robot daha önceden geçerek tespit ettiği engeller olmasına rağmen yerel yol planlamasından faydalandığı için her defasında tekrar yerel yol planlamasını tekrar etmek durumunda kalacaktır.

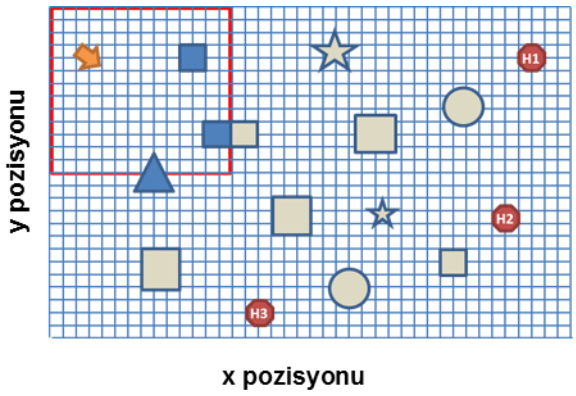
Bir diğer problem ise Şekil 3.21 (b) ile gösterildiği biçimde alan içerisinde birbirlerine engel teşkil eden, bir başka ifade ile çarpışmamaları gereken birden fazla araç olduğu durumdur. Bu durumda genel yol planlaması yapılması ve araçlar dışındaki alanın durağan olması ve hatta tüm alanın özelliklerinin görev öncesinde biliniyor olması yeniden hesaplama ihtiyacını ortadan kaldırmayacaktır. Çünkü alan

içerisinde yer alan araçlardan her hangi birisinin konum değiştirmesi alan dahilinde yer alan tüm araçlar için ayrı ayı potansiyel alanın tekrar hesaplanmasına sebebiyet verecektir. Bu durum yine hesaplama performansı açısından bir maliyet oluşturmaktadır.

Özellikle Şekil 3.21 (b) ile tanımlanan problemin çözümüne yönelik geçici çözümler üretilebilir. Örneğin araçların birbirleri ile belirli bir mesafeye gelene kadar çarpışma olasılığının önemsenmemesi gibi. Ancak bu durum sürekli geçerli olmayacaktır. Çarpışma olasılığının oluştuğu değerlendirildiği andan itibaren hesaplama performansı araçların hareket hız ve yeteneklerine de bağlı olarak yetersiz kalabilecektir. Bu tez kapsamında bu nedenle hesaplama performasının arttırılmasına yönelik özellikle paralel algoritmalar geliştirilmesi şeklinde çözüm sunulması amaçlanmıştır.

3.4.5 Önceden bilinmeyen ortamlarda yol planlaması problemi

YPA tabanlı yol planlamasının Khatib tarafından ortaya konulan temel yaklaşımı dahilinde, bir yol planlaması yapılmak istendiğinde, tüm uygulama alanı özellikleri (engellerin ve hedef noktaların kesin konumu) planlama öncesinde bilinmelidir. Ayrıca hedef noktasının yerinin ve engellerin yerlerinin sabit olduğu kabul edilir.



Şekil 3.22: Bilinmeyen ortamda yerel yol planlaması gösterimi.

Yerel yol planlaması dahilinde bu durum üstteki problem kapsamında ele alınmıştır. Alan özellikleri önceden bilinse dahi Şekil 3.22 ile gösterildiği biçimde yerel yol planlaması durumunda alanın tekrar hesaplanması ihtiyacı aracın her hareketinde ortaya çıkacaktır.

Bu tanım maalesef tez kapsamında ortaya konulan problemin çözüm ihtiyaçlarına uymamaktadır. Çarpışmayıcı önleyen bir yaklaşım dahilinde ve formasyon

durumunda çok sayıda araç bir arada yer almaktadır ve hepsi hareketlidir. Hareketlerinin ne olacağıda önceden bilinmediği kabul edilmiştir. Ayrıca özellikleri önceden bilinmeyen bir alan içerisinde hareket eden otonom platformun çevresini tanımlayabildiği algılayıcılarının mesafesi sınırlıdır. Bu nedenle dinamik bir modelleme ihtiyacı ortaya çıkar ve bu da alanın yeniden hesaplanmasına, yol planlamasının sürekli olarak güncellenmesine gereksinim duyar.

3.4.6 Araç dinamiği problemi

Yol planlamasının başarılı olabilmesi için potansiyel alan içerisinde hareket eden aracın vektörel kuvvetlerin tamamına koşulsuz riayet edeceği garanti edilmelidir, aksi taktirde araç manevra kabiliyeti göz ardı edilir ve bu durum yeniden hesaplanma ihtiyacını aracın her hareketinde tekrar ortaya çıkaracaktır. Ancak modellemeye bir de araç dinamiklerinin eklenmesi hesaplama performansını olumsuz yönde etkileyecektir. Bu dinamikler İHA platformları için ele alındığında, üç boyutlu bir ortamda yer alan mobil platformun hareket kabiliyetleri dönüş yarıçapı ve tırmanış açısı ile sınırlanmaktadır.

$$R = V^2/g.tan(\varphi) \tag{3.40}$$

Örneğin sabit kanatlı bir İHA platformu için dönüş performansı (R dönüş yarıçapı) Denklem (3.40) ile ifade edildiği şekilde platformun o anki yatış açısına (φ), süratine (V) ve üzerine binen yer çekimi ivmesine (g) bağlı olarak değişiklik gösterecektir. Oluşturulacak olan potansiyel alan modellemesi neticesinde elde edilecek olan her vektör alan değerleri platformun hızına ve yatış açısına bağlı olarak düzenlenmesi gerekecektir. Bu durumda olası çözüm uzayı platformun yeteneğine bağlı olarak güncellenebilir. Ancak bunun yerine platfor dinamikleri dahil edilmeden hesaplanan bir vektör alan neticesinde platformun mümkün olduğunca verilen komuta uygun davranması ve potansiyel alanın buna bağlı olarak tekrar hesaplanması ortaya konulacak olan paralel hesaplama yaklaşımı ile daha maliyet etkin olarak değerlendirilmektedir. Çünkü İHA benzeri hava platformlarında hareketi etkileyen platformun yetenekleri dışında rüzgar etkisi gibi çok daha fazla etken vardır ve bunların tamamını hesaplama modeline etkilemek oldukça zordur.

3.4.7 Modelleme problemi

Klasik YPA tabanlı yol planlaması uygulamalarında engeller ve hedef bir ızgara hücresinde yer alan nokta formunda modellenir, genellikle engellerin ve hedefin merkez noktaları alınarak dairesel profiller ile hassasiyeti çok düşük modellemeler kullanılır ki bu durum olası en uygun çözüme ulaşılmasını bazı durumlarda engeller. Ancak alan içerisinde yer alan tüm engelleri paneller kullanarak gerçeğe en yakın şekilde modelleme çabası hesaplama performasını oldukça olumsuz yönde etkileyecektir. Tez kapsamında gerçekleştirilen engel modellemelerinde bu nedenle eliptik formlar kullanılmıştır. Ancak eliptik formların ne derece engelleri kapsayıp kapsamadığı göz önünde bulundurularak birden fazla eliptik form ile tek engelin modellenmesine çalışılmıştır.

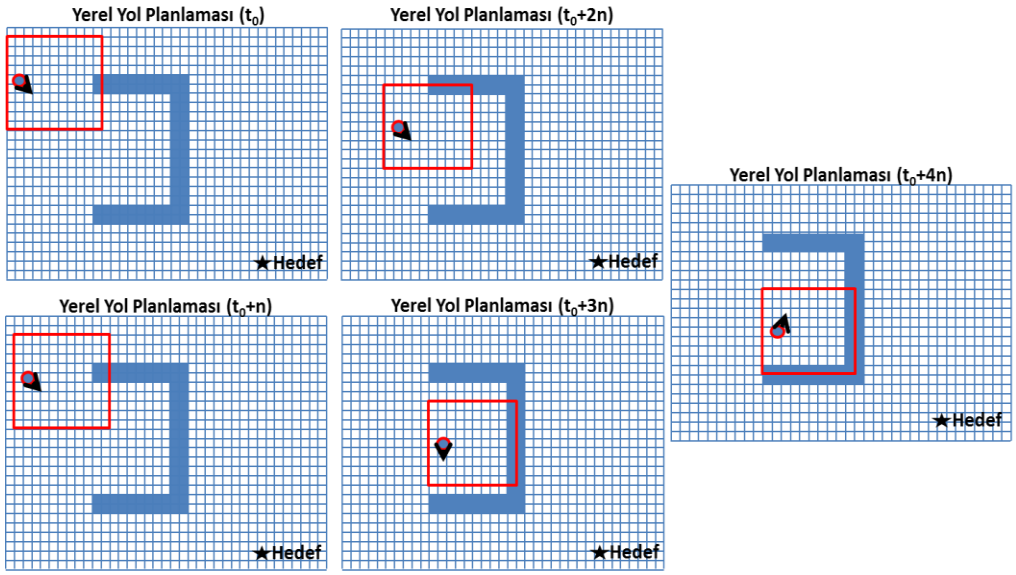
Bir diğer modelleme açısından karşılaşılan problem ise İHA sistemleri gibi üç boyutlu ortamda hareket eden otonom yapıların karşılaştıkları engellerinde doğal olarak üç boyutlu olmasıdır. Üç boyutlu engel modellemesi ve hareketin üç boyutlu olarak planlanması İHA sistemlerinde halihazırda kullanılan algıyacı teknolojileri göz önüne alındığında ve YPA tabanlı yol planlaması hesaplama maliyeti değerlendirildiğinde problem oluşturmaktadır. Bu nedenle tez kapsamında sınırlı İHA algılayıcıları ile etkin olarak yol planlaması amacıyla faydalanılabileceği değerlendirilen çok katmanlı yol planlaması modelinden faydalanılmıştır.

3.4.8 Salınım problemi

Engeller ve hedef nokta öyle konumlarda yer alabilirler ki mobil robot bir noktada sürekli benzer hareketi tekrar etmeye başlar ve bu durum salınım (oscillation) durumu olarak adlandırılırken, robotun istenen hedef noktaya ulaşmasını engeller [68]. Bu problem daha çok statik olarak tanımlanmış alanlar için geçerlidir. Çünkü engellerin ve hedefin hareketli olması bu durumun oluşmasını engelleyen başlıca koşullardan birisidir. Çünkü alan dahilinde hareket eden ve YPA tabanlı yol planlaması kapsamında modellenen her engelin hareketi şayet genel yol planlaması yapılması durumunda alan içinde yer alan her noktanın potansiyel değerini güncelleyecektir.

3.4.9 Çıkmaz yol problemi

Engeller öyle bir konumda yer alabilirler ki potansiyel akış robotu çıkmaz bir yola doğru sürükleyebilir [68]. Bu durum robot tuzağı veya çıkmaz yol problemi olarak adlandırılır ve daha çok yerel yol planlamasına göre hareketi planlanan sistemlerde görülmektedir. Şekil 3.23 ile gösterilen biçimde önceden bilinmeyen alanlar içirisinde yerel yol planlaması icra ederek hareket eden mobil platformun şayet tespit ettiği engelleri hafızasında tutma gibi bir imkanı yok ise rastlanması mümkün bir durumdur.

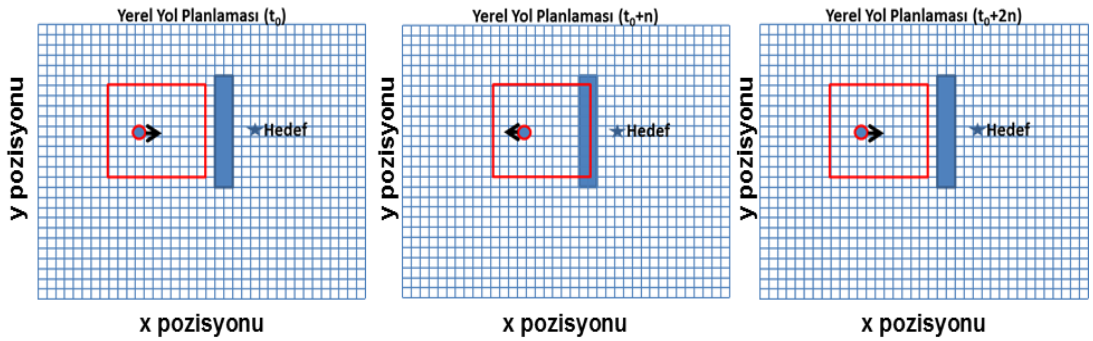


Şekil 3.23: Yerel yol planlamasında çıkmaz yol problemi gösterimi.

Bu durumun önüne geçmek için öncelikli olarak genel yol planlaması ile hareketi planlamak ve tespit edilen engel konumlarını hafızada tutmak yeterli olacaktır. Doğal olarak bu çözüm hesaplama maliyetini olumsuz yönde etkileyecektir.

3.4.10 Sonsuz yol problemi

Engeller ve hedef öyle bir konumda yer alabilirler ki robot bir noktada takılı kalmasa dahi hiç bir zaman istenen noktaya ulaşamayabilir. Bu durum literatürde farklı isimler ile anılmaktadır. En yaygın olarak kullanılan tanımı Engellere Çok Yakında Bulunan Hedeflere Erişememe Problemi (Goals Non-Reachable with Obstacles Nearby – GNRON) olarak adlandırılan tanımıdır [69].



Şekil 3.24: Yerel yol planlamasında sonsuz yol problemi gösterimi.

Sonsuz yol problemi en iyi Şekil 3.24 ile açıklanmaktadır. Özellikle yerel yol planlaması ile yönlendirilen ve önceden özellikleri bilinmeyen alanlar içerisinde karşılaşılabilinecek bir durumdur. Şekil 3.24 (a) ile gösterildiği biçimde mobil robot karşısındaki engelden habersiz biçimde yerel yol planlaması doğrultusunda hedef noktaya doğru ilerler. Şekil 3.24 (b) ile gösterildiği biçimde engeli farkettiği bir konuma geldiğinde yerel yol planlaması robota engelden uzaklaşacak yeni bir komut üretir. Ancak robot tekrar engel yerel yol planlaması çerçevesinin dışına çıktığında Şekil 3.24 (c) ile gösterildiği biçimde sadece hedef noktanın etkisi altında kalır ve ilk pozisyona geri döner. Bu çevrim sürekli tekrarlandığında robot hedefe varamadığı gibi yolculuğu da hiç sonlanmaz.

Bu durumun önüne geçmenin çözümü çıkmaz yol probleminde de olduğu gibi öncelikli olarak genel yol planlaması ile hareketi planlamak ve tespit edilen engel konumlarını hafızada tutmak olacaktır. Doğal olarak bu çözüm hesaplama maliyetini olumsuz yönde etkileyecektir.

3.4.11 YPA tabanlı yol planlaması problemlerine çözüm yaklaşımları

Yukarıda sıralanan problemler incelendiğinde, bu problemler ile karşılaşmadan çözüm üretebilecek olan yol planlaması yaklaşımı aşağıda sıralanan nitelikte olmalıdır:

a. Üç boyutlu ortamda seyrüsefer icra eden platform için mümkün olduğunca iki boyutlu modellemelerden faydalanılarak hesaplama performansı arttırılmalı,

b. Alan çözünürlüğü tüm olası çözüm uzayını kapsayabilmek açısından mümkün olduğunca yüksek belirlenmeli,

c. Harmonik fonksiyonlar veya benzeri fonksiyonlardan faydalanılarak yerel minimum oluşmasına sebebiyet vermeyecek bir potansiyel alan modellemesi gerçekleştirilmeli,

ç. Bilinmeyen ortamlar dahilinde hareket edilirken tespit edilen engel bilgisi sürekli saklanmalı,

d. Araç dinamiğinin hesaplama modeline dahil edilmesi yerine aracın komutlara verdiği tepkilere bağlı olarak yol planlaması yaklaşımı sürekli yenilenmeli,

e. Yerel yol planlaması yerine genel yol planlaması icra edilmeli, salınım, çıkmaz ve sonsuz yol durumlarından etkilenmemek için genel yol planlaması icra edilmelidir.

Tipik olarak YPA yaklaşımında, çekici ve itici potansiyel alan kuvvetleri ayrı ayrı formüle edilmektedir ve alanın toplam potansiyeli, bu iki alanın doğrusal olarak süper pozisyonundan elde edilmektedir. Bu temel yaklaşımla potansiyel kuvvetlerin oluşturulmasında faydalanılan fonksiyon örnekleri; Krogh'un genelleştirilmiş potansiyel alan (GPF) fonksiyonu [71], Khatib'in FIRAS fonksiyonu [12], süperquadrik potansiyel fonksiyon [70], Ge ve Cui'nin yeni potansiyel fonksiyonu [62][69], harmonik potansiyel fonksiyon [63][64] ve Beard'ın çekici ve itici potansiyelleri [72] olarak sıralanabilir. Bu fonksiyonların formüle edilebilmesi için, sadece yerel gradyant bilgisi yeterli olduğundan, yerel veya genel potansiyel yaklaşımlar için kullanılabilir. Hareket için önceki işleme gerek olmadığından, bir başka değiş ile hücrelerin hesaplanmasında veri bağımsızlığı özelliğinden dolayı, bu yaklaşımlar hesaplama açısından çok caziptir ve ayrıca engellerden kaçınma eğilimindeki dinamik bir davranışı belirlemek için kullanışlıdır. Potansiyel alanların modellenmesinde yaygın olarak faydalanılan potansiyel fonksiyon tipleri ve yol planlaması açısından ele alınan genel özellikleri Çizelge 3.4 kapsamında gösterilmektedir.

Çizelge 3.4: Potansiyel fonksiyon tipleri ve özelliklerinin gösterimi [15].

	FIRAS Fonksiyonu	GPF Fonksiyonu	Süperquadrik Potansiyel	Harmonik Fonksiyon	Seyrüsefer Fonksiyonu	G&C Fonksiyon
Yerel Minimum Yok – Tek Engel İçeren Alanda	Hayır	Hayır	Evet	Evet	Evet	Hayır
Yerel Minimum Yok – Birden Fazla Engel İçeren Alanda	Hayır	Hayır	Hayır	Evet	Evet	Hayır
GNRON Problemini Çözer	Hayır	Evet	Evet	Evet	Evet	Evet
Sadece Pozisyon ve Seviye Bilgisine İhtiyaç Duyar	Evet	Hayır	Evet	Evet	Evet	Evet
Kaynaklar	[12]	[71]	[70][71]	[63][64]	[65][73]	[62][69]

Harmonik potansiyel alan, harmonik fonksiyon tabanlı potansiyel alan yaklaşımının yerel minimum bağımsız çözüm sağlayan önemli bir sınıfıdır. Harmonik fonksiyonları güncelleme yeteneği statik ortamların yanı sıra dinamik ortamlarda da onların kullanılmasını sağlar. Ancak, kullanımlarını kısıtlayan bazı durumlar ve dezavantajları da bulunmaktadır. Örneğin, harmonik fonksiyonlar boyunca elde edilen bir akım hattını takip ederken hareketteki titreşim, bir akım hattından diğerine sapmaya neden olabilir [63]. Ayrıca, en kısa (optimum) çözümü bulmak genel arama (global search) tekniği gerektirir.

Harmonik fonksiyonlar laplace eşitlik çözümlerinin bir sonucudur. Bu fonksiyonlar potansiyel alan metoduna problem yaratan lokal minimaları göstermez. Harmonik fonksiyonları, robotik uygulamalar için elverişli yapan özellikleri şunlardır:

a. Yüzey normalinin hızlı hesaplanması,

b. Tamlık (eksiksizlik),

c. Davranışların farklı modellerle ifade edilme yeteneği (sıyrılma vs. kaçınma),

ç. Umulmadık engeller ve hataların varlığında sağlam kontrol.

$\Omega \subset R^n$ uzayında laplace eşitliğini sağlayan bir harmonik fonksiyondur:

$$\nabla^2 \emptyset = \sum_{i=1}^{n} \frac{\partial^2 \emptyset}{\partial x_i^2} = 0 \qquad \textbf{(3.41)}$$

Robot yolu tasarımında alanın sınırları konfigürasyon uzayındaki hedef ve tüm engellerin sınırlarından ibarettir. Harmonik fonksiyonlar maksimum minimum prensibini sağlamaktadır. Eğer laplace eşitliği kullanılan fonksiyonlarda bir kısıt olarak uygulanırsa bölge içerisindeki lokal minimanın doğal oluşumu imkansızdır [64].

Denklem (3.41)'de yer alan x_i, i'inci kartezyen koordinatı ve n boyutu gösterir. Harmonik fonksiyonlar Ω sınır değerler içerisinde max-min prensibini sağladığı için yerel minimum problemi ile karşılaşılmazlar.

Harmonik fonksiyonların önemli bir özelliği, laplace eşitliğinin doğrusallığından oluşan süperpozisyon ilkesidir. Eğer $\emptyset_1$ ve $\emptyset_2$ harmonik ise, bu durumda $\emptyset_1$ ve $\emptyset_2$'nin lineer tüm kombinasyonları da harmoniktir ve laplace eşitliğinin bir çözümü olarak gerçekleşir. Harmonik fonksiyonların lokal minimumla ilgili diğer önemli özellikleri ise ortalama değer özelliği ile maksimum prensibi ve minimum prensibidir [63].

Laplace denklemi, sayısal veya analitik olmak üzere iki yolla çözülebilir. Sayısal çözümler, engel sınırları, çözüm parametreleri ve başlangıç ve hedef konumlarındaki sınır koşullarının ayarlanmasıyla elde edilir. Alanın kuvvet vektörlerinin hedef noktasını işaret edebilmesi için potansiyel, başlangıç noktasında düşük ve hedef noktasında yüksek olmalıdır.

Analitik çözümler başlangıç, hedef ve engellerle belirtilen basit harmoniklerin toplanmasını kapsar, analitik çözümler genellikle globaldir. Bu nedenle çözüm alanı mevcut değildir. Süperpozisyon ilkesi, temel öğeler harmonik olduğu sürece laplace eşitliklerinin çözümlerine uygulanır ve bunun sonucunda potansiyel alan da harmonik olacaktır. Çözüm yöntemi ne olursa olsun, engelleri belirtmede iki tür sınır koşulu vardır. Bunlar Dirichlet ve Neumann sınır koşullarıdır. Dirichlet sınır koşulun da engeller robotu kendilerinden uzaklaştırırlar. Bu koşul Khatib tarafından öne sürülen potansiyel alanla yakından benzerlik gösterir ve genellikle robotların kontrolünde kullanılan ifadedir. Neumann sınır koşulları değişken akışını ifade eder. Burada hız vektörü engel içinden geçemez fakat kenarından geçebilir. Neumann sınır koşulları tarafından oluşturulan yollar dirichlet sınır değerleri ile yapılanlardan

genellikle daha yumuşaktır. Çünkü doğrudan ondan uzaklaşmak yerine basitçe engelin etrafından dolaşır [84].

Açıklanan bu nedenlerden dolayı çalışmada potansiyel alanlar oluşturmak için, lokal minima problemini ortadan kaldıran harmonik fonksiyonlar kullanılmıştır. Lokal minimaların olmayışı, laplace eşitliğinin iki boyutlu versiyonu kullanılarak şu şekilde ispatlanabilir:

$$\frac{\partial^2 \emptyset}{\partial x_i^2} + \frac{\partial^2 \emptyset}{\partial y_i^2} = 0 \qquad \textbf{(3.42)}$$

P_0 noktasında $\emptyset$ fonksiyonunu kesen x ve y eğrilerini düşünelim. Eğer$\emptyset$'nin ikinci türevleri P_0 noktasında sıfır değilse bu iki eğrinin ikinci türevleri zıt işaretli olmak zorundadır.$\emptyset$'nin C^2'de bir fonksiyon olduğunu düşünürsek, bir eğri yukarıya doğru içbükeyken diğeri aşağıya doğru içbükey olmak zorundadır. Böylece $\emptyset$, P_0 noktasında düzlemselken $\emptyset$'nin azaldığı yerde P_0 dan dışarı doğru bir yön,$\emptyset$'nin arttığı yerde başka bir yön elde ederiz. Bu nedenle laplace eşitliğinin olduğu açık bölgelerde $\emptyset$'nin lokal ekstremumları mevcut değildir. Eğer bir fonksiyona, $\Omega \subset R^n$ de $\emptyset$ laplace eşitliği uygulanırsa, o zaman $\emptyset$ ancak Ω'nın kısmi türevindeki sınır değerinde minimum ve maksimum değerlerine ulaşır.

Bir fonksiyona herhangi bir bölgede laplace eşitliği uygulanırsa, bu bölge içerisinde fonksiyonun kritik noktaları, fonksiyonun lokal ekstrem değerlerinin olmadığı tepe noktaları olmak zorundadır. Rimon ve Kodistcek dört özelliği sağlayan bir navigasyon fonksiyonu tanımlamışlardır. Ω =d $\Omega \cup \Omega$ yoğun bir bölge üzerinde $\emptyset$ ile tanımlanan her harmonik fonksiyon navigasyon fonksiyonunun dört özelliğinden üçünü sağlar. Şu şekilde görülebilir [65];

a. *Analitiklik* : Her harmonik fonksiyon analitiktir.

b. *Kutupsallık* : Hedef noktası olarak bir qd noktası seçin, $\emptyset(qd) = 0$ sınırlı ve bütün engel sınırları p olarak belirlenen $\emptyset(p) = c$ sınırlı. Bütün harmonik fonksiyonlar min-max özelliğini sağladığında, $\emptyset$ polardır. Başka bir deyişle, $\bar{\Omega}$ da minimum değere ulaştığı nokta *qd* olacaktır.

c. *Kabul edilebilirlik*: Eğer sadece yukarıda $c = 1$ sabit ayarlanırsa, $\emptyset$ kabul edilebilir olacaktır. Bu $\emptyset$'nin basit normalizasyonudur.

ç. Dördüncü özellik, navigasyon fonksiyonunun Morse olmasıdır (örneğin bozuk kritik noktalar yoktur). Burada basitçe, $\emptyset$'nin Ω içerisindeki her kritik noktasının ayrı bir tepe noktasının olması gerektiğini ifade eder [65].

Lokal minimuma ilişkin harmonik fonksiyonların özellikleri aşağıda tanımlanmıştır.

a. *Süperpozisyon özelliği*: Potansiyel fonksiyonların bu özelliği, laplace eşitliğinin doğrusallığıyla ilgilidir. Eğer $\emptyset_1$ ve $\emptyset_2$ harmonik fonksiyonlarsa (laplace eşitliğini sağlayan), $\emptyset_1$ ve $\emptyset_2$'nin her lineer kombinasyonuda harmoniktir ve bir laplace eşitlik çözümüdür.

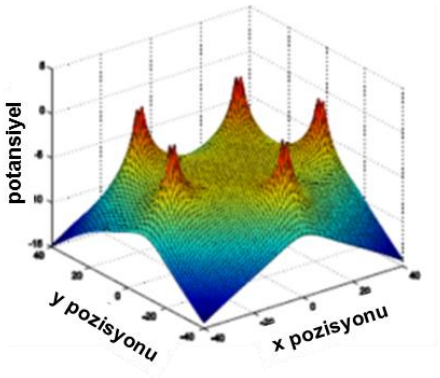
b. *Ortalama değer özelliği*: (x_0, y_0) merkezli bir daire içerisinde harmonik olan iki boyutlu potansiyel fonksiyon $\emptyset(x, y)$ için, $\emptyset$'nin ortalama değer özelliği vardır.

$$\emptyset(x_1, x_2) = \frac{1}{2\pi}\int \emptyset\,(x_1 + r\,cos\theta, x_2 + r\,sin\theta)d\theta \qquad \textbf{(3.43)}$$

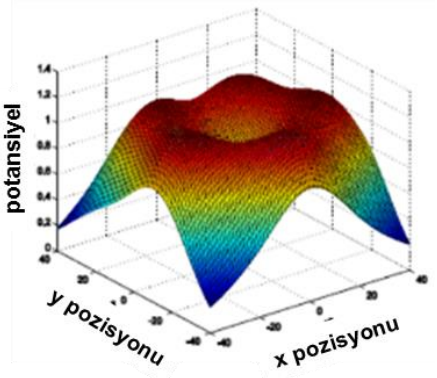
Başka bir deyişle, herhangi gelişigüzel bir dairenin merkezindeki potansiyelin değeri, dairenin çevresi üzerinde bütün potansiyelin ortalamasına eşittir. Denklem (3.43) ile gösterilen potansiyel fonksiyon $\emptyset$ daire içerisinde harmonikse, bu özellik dairenin yarıçapı r den bağımsızdır. Bu özelliğin tersi de doğrudur. Yani, eğer $\emptyset(x_1, x_2)$ sürekli ve bölgedeki her daire için ortalama değer özelliğine sahipse, o zaman $\emptyset$ harmoniktir. Bu özellik harmonik fonksiyonun minimum ve maksimum özelliğini ispatlamak için kullanılabilir.

Maksimum potansiyel özelliği açısından sabit olmayan bir harmonik fonksiyonun maksimumu, potansiyelin sonsuza eğilimi olduğu tekil sınırlarda ortaya çıkar. Minimum potansiyel özelliği açısından ise sabit olmayan bir harmonik fonksiyonun minimumu da, potansiyelin sonsuza eğilimi olduğu tekil sınırlar üzerinde ortaya çıkar. Harmonik fonksiyonun bu özellikleri engel kaçınma problemi için bir yapay potansiyel alan oluşturmakta çok faydalıdır, çünkü harmonik fonksiyon yerel minimumları tamamen ortadan kaldırır [63].

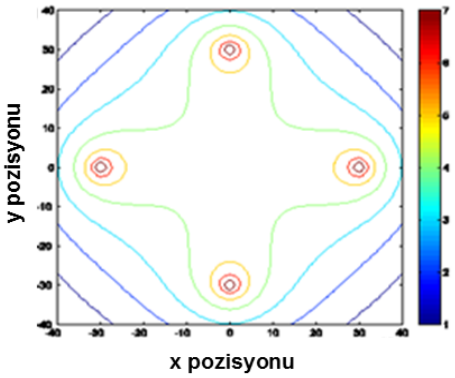
Harmonik potansiyel alan, harmonik fonksiyon tabanlı potansiyel alan yaklaşımının yerel minimum bağımsız çözüm sağlayan önemli bir sınıfıdır. Harmonik fonksiyonları güncelleme yeteneği statik ortamların yanı sıra dinamik ortamlarda da onların kullanılmasını sağlar.



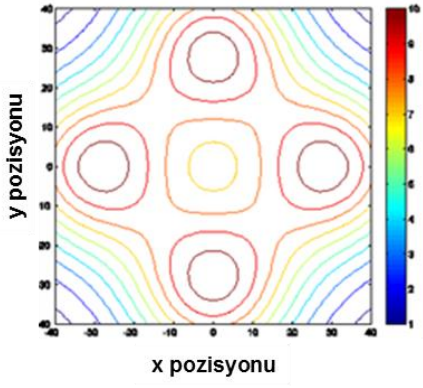
(a) Harmonik fonksiyonun üç boyutlu gösterimi.



(b) Harmonik olmayan fonksiyonun üç boyutlu gösterimi.



(c) Harmonik fonksiyonun iki boyutlu gösterimi.



(ç) Harmonik olmayan fonksiyonun iki boyutlu gösterimi.

Şekil 3.25: Harmonic ve bivariate fonksiyonların karşılaştırılması.

İki boyutlu durumlar için harmonik ve harmonik olmayan fonksiyonlar araç yol planlamasında kullanılmak üzere karşılaştırıldığında, harmonik fonksiyon olarak Denklem (3.44), Harmonik olmayan iki değişkenli üstel fonksiyon olarak da Denklem (3.45)'deki fonksiyon kullanılmıştır. Denklem (3.45)'deki fonksiyon harmonik fonksiyon olma koşullarını sağlamadığından dolayı lokal minimuma sahiptir [13].

$$g(x,y) = -\log(\alpha((x-x_h)^2 + \gamma(y-y_h)^2)) \quad \textbf{(3.44)}$$

$$f(x,y) = e^{\left(-\alpha((x-x_h)^2+\gamma(y-y_h)^2)\right)} \quad \textbf{(3.45)}$$

Harmonik fonksiyon ve harmonik olmayan fonksiyon kullanılarak oluşturulan potansiyel alanların karşılaştırılması sonucunda, potansiyel alanlarla yol planlamada karşılaşılan lokal minima probleminin harmonik fonksiyonlarda ortaya çıkmadığı MATLAB ortamında yapılan uygulamalar sonucunda elde edilen Şekil 3.25'de yer alan grafikler ile görülmektedir. Şekil 3.25'da, akışkan partiküllerinin hızının birden

sıfır olduğu yer olan başlangıç noktası *(0,0)* hidrodinamiklerde durağanlık noktası olarak adlandırılır. Ancak, bu durağanlık noktası kararsız boyun noktasıdır, lokal minimum değildir. Bu karşılaştırmayı genelleştirirsek, harmonik olmayan fonksiyonlar ile oluşturulan potansiyel alanlarda lokal minimumun kaçınılmaz olduğu sonucuna varırız.

Harmonik fonksiyon ve harmonik olmayan fonksiyonlarla oluşturulan alan içerisinde lokal minima probleminin oluşup oluşmadığı Şekil 3.25'de potansiyel alanların iki boyutlu gösteriminde daha açık bir şekilde görülmektedir. Şekil 3.25 (b)'de, Denklem (3.45)'deki harmonik olmayan (bivariate) fonksiyon ile oluşturulan alan içerisinde, dört engelin ortasında kalan alanda bir lokal minimum oluştuğu görülmektedir. Oysa ki Şekil 3.25 (a)'da, Denklem (3.44)'deki harmonik fonksiyon ile oluşturulan alanda lokal minimumun oluşmadığı görülmektedir. Harmonik fonksiyon kullanılarak oluşturulan potansiyel alan içerisinde dört adet engel olsun. Şekil 3.25 (a)'da bu alanın üç boyutlu gösterimi yer almaktadır. Şekil 3.25 (c)'de de görüldüğü üzere harmonik fonksiyon ile modellenen YPA içinde bir yerel minimum durumu gözlenmemektedir. Harmonik olmayan fonksiyon ile modellenen YPA ise üç boyutlu olarak Şekil 3.25 (b) gösterilmekte ve Şekil 3.25 (ç)'de de görüldüğü üzere alan içinde yer alan engellerin ortasında *(0,0)* noktasında bir yerel minimum durumunun oluştuğu gözlemlenmektedir. Bu noktaya ulaşan robotun hedefe ulaşmış gibi davranması beklenir ve bu istenmeyen bir durumdur.

Global ve yerel yol planlamasını birleştiren genelleştirilmiş yapay potansiyel alanlar metodu Krogh ve Thorpe [74] tarafından, gerçek sensor bilgisi içeren mobil robotlar üzerinde potansiyel alanların uygulanması Brooks [75] ve Arkın [76] tarafından uygulanmıştır. Grid ve potansiyel alanlar metotlarını birleştirerek kuvvet alan metodu Borenstein [77] tarafından önerilmiştir. Connolly, harmonik fonksiyonlara dayalı bir robot hareket planlama metodu amaçlamıştı [64]. Kazemi, bilinen sabit engeller arasında hareket eden bir hedefi izlemek için harmonik potansiyel yaklaşımını genelleştirmiştir [78].

Çizelge 3.5: Yerel ve genel potansiyel yaklaşımların karşılaştırılması.

	Yerel Potansiyel Yaklaşımları	**Genel Potansiyel Yaklaşımlar**
Özellikler	Toplam potansiyel alan yerine sadece robotun içinde olduğu belirli bir kesiti dinamik olarak hesaplanarak sonuca ulaşılır.	Toplam potansiyel alan bir arada hesaplanarak yol planlaması başlangıç durumundan hedefe kadar bir bütün olarak hesaplanır.
Avantajlar	**1.** Robotun hedefe ulaşması için izleyeceği yolun bulunduğu tüm çevrenin özellikleri planlama öncesi bilinmek zorunda değildir. **2.** Oluşturulan alt potansiyel alanlar belirli bir düzen içerisinde tekrar hesaplandığı için toplam potansiyele ilave hedef ve engellerin dinamik olarak eklenmesi mümkündür.	**1.** Sadece hedef ile temsil edilen genel minimum noktası ile potansiyel bir alan sağlar. **2.** Çıkmaz yol ve salınım problemlerinin tespiti ve çözümü daha kolay ve etkin şekilde gerçekleştirilebilir. **3.** Hedefe ulaşılamayacağı durumu başlangıç durumunda tespit edilebilir.
Dezavantajlar	**1.** İstenmeyen yerel minimum durumlarının oluşma ihtimali vardır. **2.** Çıkmaz yol probleminin oluşma ihtimali vardır. **3.** Salınım problemi ile karşılaşılabilir. **4.** Hiç bir zaman istenen hedefe ulaşılamaması durumu ile karşılaşılabilir.	**1.**Çalışma alanının tamamının bilgisinin başlangıçta bilinmesi gereklidir. **2.** Ek bir engel veya hedef eklenebilmesi için toplam potansiyelin tekrar hesaplanması gereklidir. **3.** Hesaplama alanı büyük olduğu için yol planlaması uzun sürecektir
Örnekler	**1.** GPF fonksiyonu [71] **2.** FIRAS fonksiyonu [12] **3.**Superquadratic Potansiyel [70] **4.**G&C potansiyel fonksiyonları [62][69] **5.**Harmonik potansiyel fonksiyonu[63][64] **6.**Çekici ve itici potansiyel[72]	Seyrüsefer Fonksiyonları [65][73]

Gerek performans açısından daha iyi sonuçlar elde etmek gerekse de YPA yaklaşımının eksik kaldığı noktaları tamamlamak için literatürde ortaya konulan bir diğer yol planlaması sınıfı olarak hibrit yaklaşımlardan söz edilebilir. Örneğin etkin

ve güvenli bir şekilde yol planlaması yapmak için yerel algılamalar ile çevrenin planlama öncesi elde edilen ön bilgisini entegre eden bir yaklaşım Kazemi tarafından ortaya konmuştur [78]. Bu tez çalışmasının sonuçları kısmen bilinmeyen dinamik bir ortamda dahi çıkmaz yol problemine maruz kalmadan yol planlaması yapılabileceğini göstermektedir. Her ne kadar faydalı çözümler üretse de, çalışmanın problemleri yerel algılama yapısına dayanması ve yerel yol planlaması için çözümler üretirken, genel yol planlamasında aynı verimliliği gösterememesi olarak sayılabilir. Bir diğer geliştirilmiş YPA tabanlı yol planlaması çözümü Zhang ve arkadaşları tarafından ortaya konulmuş ve çalışma kapsamında YPA yaklaşımı kuantum parçacık sürüsü optimizasyonu (quantum particle swarm optimization) kullanılarak farklı çevre koşullarına ve dinamik engellere karşı çalışabilir bir düzen oluşturulmak istenmiştir [79]. Klasik YPA tanımında karşımıza çıkan yerel minimum problemi [80][81] ile regresyon ve potansiyel alan doldurma teknikleri ile çözülmeye çalışılmıştır. Benzer problemlerin çözümü için engeller arasında bir kapalı bağlantı kurmak temeline dayalı çözümler geliştirilmiştir [82][83]. Hibrit yaklaşımlar literatürde, hesaplama performansı açısından daha iyi performans değerleri üretmek yerine daha çok temel YPA yaklaşımında karşılaşılan problemleri çözmek üzerine geliştirilmişlerdir

Tanımlanan YPA yaklaşımının karşılaştığı problemlerin bir kısmı genel ya da yerel yol planlaması yaklaşımına bağlı olarak görülürler. Yol planlaması açıklandığı şekilde yerel ve genel yol planlaması olarak ele alındığında ve yol planlamasının YPA yaklaşımları ile ortaya konulduğu değerlendirildiğinde Çizelge 3.5'de belirtilen sonuçlara ulaşılabilir.

3.5 YPA İle Otonom Araçlar İçin Yol Planlaması

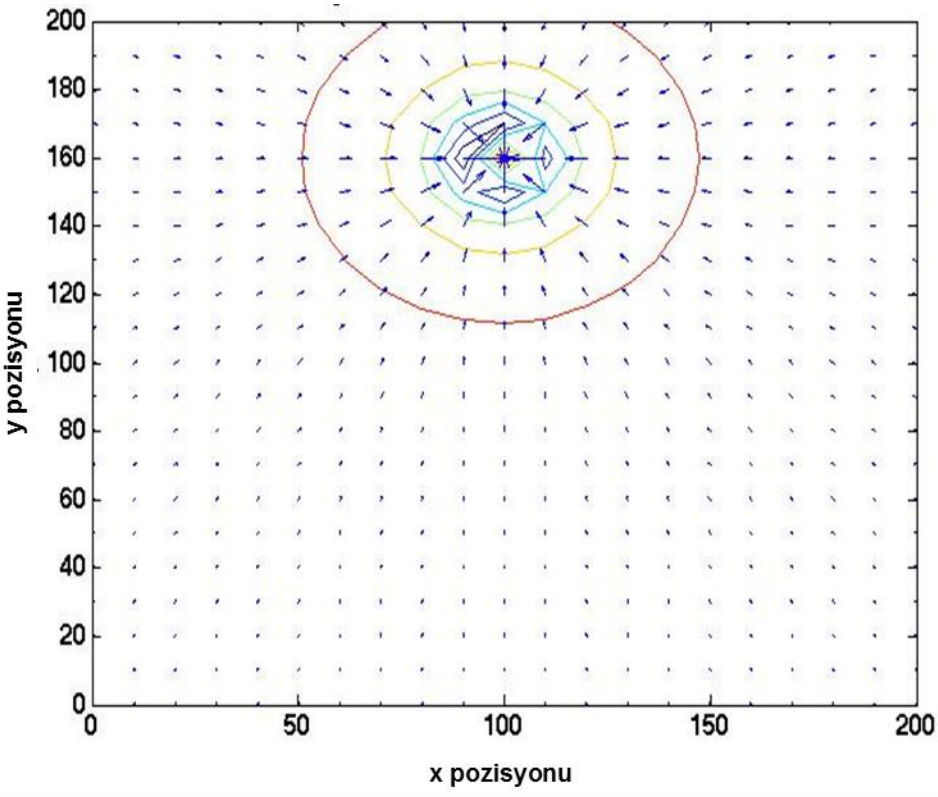
İki boyutlu uzayda açık hareket planlamasında ilk adım olan yol planlamasının çıktısı, YPA tabanlı yol planlaması ile gerçekleştirildiğinde, her bir elemanı bir vektör ile temsil edilen iki boyutlu bir dizidir. Bu dizi başlangıç konumundan hedef noktaya kadar platformun konumuna bağlı olarak yönlendirilmesi amacıyla kullanılır. Çizelge 3.1 ile ortaya konulan, x ve y eksenindeki değişimi gösteren fonksiyonlar ile potansiyel alandan vektörel alana dönüşüm Denklem (3.39) ile hesaplanmış ve hareket sahası içinde platforma etki eden vektör kuvvetlerin yön ve şiddet gibi bilgilerinin elde edilmesi mümkün olmuştur.

Çizelge 3.6: Engel içermeyen alan için simulasyon parametere değerleri.

Parametre	**Değer**
$size\ (Alan\ Boyut)$	200 x 200
$Res\ (Alan\ Çöznürlük)$	10
δ_c	1.0
γ_c	1.0
C_x, C_y	30,60
β_c	π

Bu durumu bir örnek uygulama ile açıklamak için Çizelge 3.6 ile verilen parametreler ile oluşturulmuş ve alan içinde yer alan ulaşılmak istenen hedef noktayı temsil eden çeken potansiyel alandan faydalanılmıştır.

Elde edilen örnek vektör alan Şekil 3.26 ile gösterilmiştir. Şekilden de görüldüğü üzere tanımlanan alanın her hangi bir noktasından hareketine başlayan mobil araç ilerledikçe bulunduğu konumdaki vektör ile yönlendirildiğinde hedef noktaya ulaşacaktır. Mobil platformun yönlendirilmesi için her ızgara hücresinde platformun davranışı şekillendirecek yön ve hız bilgisi edinilmelidir.



Şekil 3.26: Engel içermeyen örnek ortam için vektör alan gösterimi.

Bu kapsamda vektör alanın her bir $(x,\ y)$ noktasındaki vektör kuvvetin yönü $(Heading(x,y))$ bu noktadaki potansiyel kuvvetlerin toplamının x ve y bileşenlerinin Denklem (3.46)'deki şekilde hesaplanması ile elde edilir.

$$Heading(x,y) = arctan\left(\frac{SY(x,y)}{SX(x,y)}\right) \quad \textbf{(3.46)}$$

Toplam vektör alanın her bir $(x,\ y)$ noktasındaki vektör kuvvetin şiddeti $(Magnitude(x,y))$ ise bu noktadaki potansiyel kuvvetlerin toplamının x ve y bileşenlerinin Denklem (3.47)'deki şekilde hesaplanması ile elde edilir.

$$Magnitude(x,y) = 1 - \sqrt{SX(x,y)^2 + SY(x,y)^2} \quad \textbf{(3.47)}$$

Vektör alanın her bir $(x,\ y)$ noktasında yer alan her bir vektörün alan içerisinde hareket eden mobil robota o anda verilen komut olduğu değerlendirildiğinde, iki boyutlu alan içerisinde yer alan İHA platformu için bu vektör komutlar hız ve uçuş başı komutları haline getirilebilir.

	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	58,26	60,92	63,73	66,68	69,77	72,98	76,31	79,73	83,23	86,78	90	93,93	97,48	100,96	104,37	107,68	110,86	113,92	116,84	119,62	122,25
1	56,59	59,33	62,24	65,31	68,54	71,92	75,43	79,06	82,79	86,57	90	94,19	97,97	101,68	105,28	108,77	112,13	115,32	118,36	121,24	123,95
2	54,74	57,57	60,57	63,77	67,16	70,72	74,44	78,30	82,28	86,33	90	94,49	98,53	102,48	106,32	110,01	113,54	116,89	120,05	123,02	125,80
3	52,72	55,61	58,72	62,05	65,60	69,36	73,31	77,43	81,69	86,04	90	94,84	99,18	103,41	107,50	111,42	115,13	118,64	121,92	124,98	127,84
4	50,49	53,45	56,65	60,11	63,83	67,80	72,00	76,42	81,01	85,71	90	95,24	99,93	104,48	108,86	113,02	116,94	120,61	124,01	127,17	130,08
5	48,02	51,03	54,33	57,91	61,80	66,00	70,49	75,24	80,21	85,33	90	95,71	100,81	105,74	110,44	114,87	119,00	122,83	126,36	129,59	132,55
6	45,30	48,34	51,70	55,40	59,47	63,90	68,70	73,84	79,25	84,86	90	96,28	101,86	107,22	112,29	117,02	121,38	125,37	129,00	132,30	135,28
7	42,28	45,33	48,73	52,53	56,76	61,44	66,58	72,15	78,09	84,29	90	96,97	103,13	109,00	114,49	119,53	124,12	128,26	131,98	135,31	138,29
8	38,95	41,96	45,37	49,23	53,60	58,52	64,02	70,08	76,65	83,59	90	97,83	104,71	111,18	117,13	122,51	127,32	131,58	135,35	138,67	141,61
9	35,27	38,20	41,55	45,42	49,88	55,02	60,89	67,51	74,82	82,68	90	98,93	106,70	113,88	120,35	126,07	131,06	135,40	139,16	142,43	145,27
10	31,23	34,00	37,23	41,02	45,49	50,77	56,99	64,21	72,44	81,48	90	100,39	109,29	117,32	124,34	130,36	135,47	139,79	143,47	146,60	149,28
11	26,80	29,34	32,34	35,94	40,29	45,59	52,06	59,90	69,20	79,80	90	102,41	112,78	121,80	129,35	135,56	140,65	144,84	148,31	151,21	153,66
12	22,01	24,21	26,86	30,11	34,14	39,24	45,74	54,07	64,60	77,33	90	105,38	117,70	127,77	135,70	141,89	146,74	150,60	153,71	156,27	158,39
13	16,86	18,63	20,80	23,50	26,96	31,48	37,58	45,98	57,66	73,31	90	110,14	124,99	135,94	143,81	149,53	153,81	157,09	159,67	161,75	163,46
14	11,42	12,67	14,21	16,17	18,73	22,21	27,16	34,60	46,48	65,79	90	118,82	136,40	147,17	154,00	158,59	161,85	164,27	166,13	167,60	168,80
15	5,77	6,41	7,21	8,25	9,62	11,53	14,38	19,03	27,76	48,03	90	137,75	154,55	162,13	166,30	168,91	170,69	171,99	172,96	173,73	174,35
16	0,00	0,00	0,00	0,00	-0,01	-0,01	-0,01	-0,01	-0,02	-0,04	-90	-179,97	-179,98	-179,99	-179,99	-179,99	-179,99	-180,00	-180,00	-180,00	-180,00
17	-5,77	-6,42	-7,22	-8,25	-9,63	-11,55	-14,40	-19,05	-27,79	-48,06	-90	-137,71	-154,52	-162,11	-166,28	-168,90	-170,68	-171,98	-172,96	-173,72	-174,34
18	-11,43	-12,67	-14,22	-16,18	-18,74	-22,22	-27,17	-34,62	-46,50	-65,80	-90	-118,81	-136,39	-147,16	-153,99	-158,58	-161,84	-164,26	-166,12	-167,60	-168,79
19	-16,87	-18,64	-20,81	-23,51	-26,97	-31,49	-37,59	-45,99	-57,67	-73,32	-90	-110,14	-124,98	-135,93	-143,80	-149,52	-153,80	-157,08	-159,67	-161,75	-163,45
20	-22,01	-24,21	-26,87	-30,12	-34,15	-39,24	-45,74	-54,08	-64,61	-77,33	-90	-105,38	-117,69	-127,77	-135,70	-141,88	-146,73	-150,59	-153,71	-156,26	-158,39

Şekil 3.27: Vektörel alandan uçuş başı komut setinin üretilmesi.

Şekil 3.26 ile gösterilen ve Çizelge 3.6 ile parametre değerleri verilen çeken noktasal alana ait vektör alan içerisinde yer alan vektör komutları Denklem (3.46) ile hesaplanarak Şekil 3.27'daki uçuş başı komut setine dönüştürülebilir.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
20	99,98	99,89	99,80	99,71	99,63	99,56	99,49	99,45	99,41	99,39	99,38	99,39	99,42	99,45	99,51	99,57	99,64	99,73	99,81	99,91	100
19	99,82	99,71	99,61	99,50	99,41	99,33	99,25	99,19	99,15	99,12	99,11	99,13	99,15	99,20	99,27	99,34	99,43	99,52	99,63	99,73	99,84
18	99,65	99,52	99,40	99,28	99,17	99,07	98,98	98,90	98,85	98,82	98,81	98,82	98,86	98,92	98,99	99,09	99,19	99,30	99,42	99,55	99,67
17	99,46	99,32	99,17	99,03	98,90	98,77	98,66	98,57	98,51	98,47	98,46	98,47	98,52	98,59	98,68	98,80	98,92	99,06	99,20	99,35	99,49
16	99,27	99,10	98,93	98,76	98,59	98,44	98,31	98,19	98,11	98,06	98,04	98,07	98,12	98,21	98,33	98,47	98,62	98,79	98,96	99,13	99,30
15	99,06	98,86	98,65	98,45	98,25	98,06	97,89	97,75	97,64	97,58	97,56	97,59	97,66	97,78	97,93	98,10	98,29	98,49	98,69	98,90	99,09
14	98,83	98,60	98,36	98,11	97,87	97,63	97,41	97,23	97,09	97,00	96,98	97,01	97,11	97,26	97,45	97,68	97,91	98,16	98,41	98,65	98,88
13	98,60	98,32	98,03	97,73	97,43	97,13	96,85	96,61	96,42	96,30	96,26	96,32	96,45	96,65	96,90	97,19	97,49	97,79	98,09	98,38	98,65
12	98,35	98,03	97,68	97,32	96,94	96,55	96,18	95,85	95,59	95,42	95,37	95,45	95,63	95,91	96,25	96,63	97,01	97,39	97,76	98,10	98,41
11	98,10	97,72	97,31	96,86	96,38	95,88	95,38	94,92	94,55	94,30	94,23	94,34	94,61	95,01	95,48	95,98	96,48	96,96	97,40	97,80	98,17
10	97,85	97,41	96,92	96,37	95,77	95,11	94,43	93,76	93,20	92,82	92,70	92,87	93,30	93,89	94,56	95,24	95,89	96,49	97,03	97,51	97,93
9	97,61	97,11	96,53	95,86	95,10	94,23	93,27	92,29	91,40	90,76	90,56	90,86	91,56	92,49	93,47	94,41	95,26	96,00	96,65	97,21	97,70
8	97,38	96,82	96,15	95,35	94,39	93,25	91,91	90,40	88,90	87,74	87,36	87,93	89,19	90,71	92,19	93,49	94,60	95,52	96,29	96,94	97,49
7	97,19	96,56	95,80	94,86	93,70	92,22	90,35	88,01	85,32	82,91	82,03	83,31	85,87	88,52	90,76	92,55	93,95	95,07	95,96	96,70	97,30
6	97,04	96,36	95,52	94,46	93,09	91,27	88,75	85,18	80,13	74,13	71,39	75,28	81,27	86,00	89,33	91,68	93,40	94,69	95,70	96,51	97,16
5	96,94	96,23	95,33	94,19	92,67	90,56	87,46	82,47	73,50	55,68	39,85	60,24	75,82	83,70	88,19	91,05	93,01	94,44	95,53	96,38	97,07
4	96,91	96,18	95,27	94,09	92,52	90,30	86,94	81,27	69,62	32,07	99,80	45,11	72,85	82,70	87,75	90,81	92,88	94,36	95,47	96,34	97,04
3	96,94	96,23	95,33	94,19	92,67	90,56	87,46	82,48	73,51	55,71	39,92	60,26	75,82	83,70	88,19	91,05	93,01	94,44	95,53	96,38	97,07
2	97,04	96,36	95,52	94,46	93,09	91,27	88,75	85,18	80,14	74,15	71,41	75,30	81,28	86,01	89,33	91,68	93,40	94,69	95,70	96,51	97,16
1	97,19	96,56	95,80	94,86	93,70	92,23	90,35	88,02	85,33	82,91	82,04	83,32	85,88	88,52	90,77	92,55	93,96	95,07	95,96	96,70	97,30
0	97,38	96,82	96,15	95,35	94,39	93,25	91,91	90,41	88,91	87,75	87,37	87,93	89,19	90,72	92,19	93,49	94,60	95,52	96,29	96,94	97,49

Şekil 3.28: Vektörel alandan uçuş hızı komut setinin üretilmesi.

Aynı şekilde Şekil 3.26 ile gösterilen ve Çizelge 3.6 ile parametre değerleri verilen çeken noktasal alana ait vektör alan içerisinde yer alan vektör komutları Denklem (3.47) ile hesaplanarak Şekil 3.28'deki uçuş hızı komut setine dönüştürülebilir.

Bu iki komut seti tablosu ile alan içerisinde herhangi bir noktada yer alan İHA platformu ulaşılmak istenen hedef noktaya yönlendirilebilir ve yol planlaması gerçekleştirilebilir.

Potansiyel alan ile modellenmiş uçuş bölgesi içinde platform yönlendirmek için toplam vektör alandan faydalanılarak uçuş komutlarının üretilmesi gerekmektedir. Uçuş komutlarının üretilmesi sürecinin ortaya konulması maksadıyla bu bölüm kapsamında bir örnek üzerinden gidilecektir. Örnek uçuş alanına ait parametreler Çizelge 3.3 ile gösterildiği biçimdedir. Çizelge 3.3'de yer alan simülasyon parametrelerinden faydalanılarak hesaplanan vektör alan Şekil 3.19'da gösterilmektedir.

	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	58,26	60,92	63,73	66,68	69,77	72,98	76,31	79,73	83,23	86,78	90	93,93	97,48	100,96	104,37	107,68	110,86	113,92	116,84	119,62	122,25
1	56,59	59,33	62,24	65,31	68,54	71,92	75,43	79,06	82,79	86,57	90	94,19	97,97	101,68	105,28	108,77	112,13	115,32	118,36	121,24	123,95
2	54,74	57,57	60,57	63,77	67,16	70,72	74,44	78,30	82,28	86,33	90	94,49	98,53	102,48	106,32	110,01	113,54	116,89	120,05	123,02	125,80
3	52,72	55,61	58,72	62,05	65,60	69,36	73,31	77,43	81,69	86,04	90	94,84	99,18	103,41	107,50	111,42	115,13	118,64	121,92	124,98	127,84
4	50,49	53,45	56,65	60,11	63,83	67,80	72,00	76,42	81,01	85,71	90	95,24	99,93	104,48	108,86	113,02	116,94	120,61	124,01	127,17	130,08
5	48,02	51,03	54,33	57,91	61,80	66,00	70,49	75,24	80,21	85,33	90	95,71	100,81	105,74	110,44	114,87	119,00	122,83	126,36	129,59	132,55
6	45,30	48,34	51,70	55,40	59,47	63,90	68,70	73,84	79,25	84,86	90	96,28	101,86	107,22	112,29	117,02	121,38	125,37	129,00	132,30	135,28
7	42,28	45,33	48,73	52,53	56,76	61,44	66,58	72,15	78,09	84,29	90	96,97	103,13	109,00	114,49	119,53	124,12	128,26	131,98	135,31	138,29
8	38,95	41,96	45,37	49,23	53,60	58,52	64,02	70,08	76,65	83,59	90	97,83	104,71	111,18	117,13	122,51	127,32	131,58	135,35	138,67	141,61
9	35,27	38,20	41,55	45,42	49,88	55,02	60,89	67,51	74,82	82,68	90	98,93	106,70	113,88	120,35	126,07	131,06	135,40	139,16	142,43	145,27
10	31,23	34,00	37,23	41,02	45,49	50,77	56,99	64,21	72,44	81,48	90	100,39	109,29	117,32	124,34	130,36	135,47	139,79	143,47	146,60	149,28
11	26,80	29,34	32,34	35,94	40,29	45,59	52,06	59,90	69,20	79,80	90	102,41	112,78	121,80	129,35	135,56	140,65	144,84	148,31	151,21	153,66
12	22,01	24,21	26,86	30,11	34,14	39,24	45,74	54,07	64,60	77,33	90	105,38	117,70	127,77	135,70	141,89	146,74	150,60	153,71	156,27	158,39
13	16,86	18,63	20,80	23,50	26,96	31,48	37,58	45,98	57,66	73,31	90	110,14	124,99	135,94	143,81	149,53	153,81	157,09	159,67	161,75	163,46
14	11,42	12,67	14,21	16,17	18,73	22,21	27,16	34,60	46,48	65,79	90	118,82	136,40	147,17	154,00	158,59	161,85	164,27	166,13	167,60	168,80
15	5,77	6,41	7,21	8,25	9,62	11,53	14,38	19,03	27,76	48,03	90	137,75	154,55	162,13	166,30	168,91	170,69	171,99	172,96	173,73	174,35
16	0,00	0,00	0,00	0,00	-0,01	-0,01	-0,01	-0,01	-0,02	-0,04	-90	-179,97	-179,98	-179,99	-179,99	-179,99	-179,99	-180,00	-180,00	-180,00	-180,00
17	-5,77	-6,42	-7,22	-8,25	-9,63	-11,55	-14,40	-19,05	-27,79	-48,06	-90	-137,71	-154,52	-162,11	-166,28	-168,90	-170,68	-171,98	-172,96	-173,72	-174,34
18	-11,43	-12,67	-14,22	-16,18	-18,74	-22,22	-27,17	-34,62	-46,50	-65,80	-90	-118,81	-136,39	-147,16	-153,99	-158,58	-161,84	-164,26	-166,12	-167,60	-168,79
19	-16,87	-18,64	-20,81	-23,51	-26,97	-31,49	-37,59	-45,99	-57,67	-73,32	-90	-110,14	-124,98	-135,93	-143,80	-149,52	-153,80	-157,08	-159,67	-161,75	-163,45
20	-22,01	-24,21	-26,87	-30,12	-34,15	-39,24	-45,74	-54,08	-64,61	-77,33	-90	-105,38	-117,69	-127,77	-135,70	-141,88	-146,73	-150,59	-153,71	-156,26	-158,39

(a) Uçuş sürati komut tablosu gösterimi.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
20	99,98	99,89	99,80	99,71	99,63	99,56	99,49	99,45	99,41	99,39	99,38	99,39	99,42	99,45	99,51	99,57	99,64	99,73	99,81	99,91	100
19	99,82	99,71	99,61	99,50	99,41	99,33	99,25	99,19	99,15	99,12	99,11	99,13	99,15	99,20	99,27	99,34	99,43	99,52	99,63	99,73	99,84
18	99,65	99,52	99,40	99,28	99,17	99,07	98,98	98,90	98,85	98,82	98,81	98,82	98,86	98,92	98,99	99,09	99,19	99,30	99,42	99,55	99,67
17	99,46	99,32	99,17	99,03	98,90	98,77	98,66	98,57	98,51	98,47	98,46	98,47	98,52	98,59	98,68	98,80	98,92	99,06	99,20	99,35	99,49
16	99,27	99,10	98,93	98,76	98,59	98,44	98,31	98,19	98,11	98,06	98,04	98,07	98,12	98,21	98,33	98,47	98,62	98,79	98,96	99,13	99,30
15	99,06	98,86	98,65	98,45	98,25	98,06	97,89	97,75	97,64	97,58	97,56	97,59	97,66	97,78	97,93	98,10	98,29	98,49	98,69	98,90	99,09
14	98,83	98,60	98,36	98,11	97,87	97,63	97,41	97,23	97,09	97,00	96,98	97,01	97,11	97,26	97,45	97,68	97,91	98,16	98,41	98,65	98,88
13	98,60	98,32	98,03	97,73	97,43	97,13	96,85	96,61	96,42	96,30	96,26	96,32	96,45	96,65	96,90	97,19	97,49	97,79	98,09	98,38	98,65
12	98,35	98,03	97,68	97,32	96,94	96,55	96,18	95,85	95,59	95,42	95,37	95,45	95,63	95,91	96,25	96,63	97,01	97,39	97,76	98,10	98,41
11	98,10	97,72	97,31	96,86	96,38	95,88	95,38	94,92	94,55	94,30	94,23	94,34	94,61	95,01	95,48	95,98	96,48	96,96	97,40	97,80	98,17
10	97,85	97,41	96,92	96,37	95,77	95,11	94,43	93,76	93,20	92,82	92,70	92,87	93,30	93,89	94,56	95,24	95,89	96,49	97,03	97,51	97,93
9	97,61	97,11	96,53	95,86	95,10	94,23	93,27	92,29	91,40	90,76	90,56	90,86	91,56	92,49	93,47	94,41	95,26	96,00	96,65	97,21	97,70
8	97,38	96,82	96,15	95,35	94,39	93,25	91,91	90,40	88,90	87,74	87,36	87,93	89,19	90,71	92,19	93,49	94,60	95,52	96,29	96,94	97,49
7	97,19	96,56	95,80	94,86	93,70	92,22	90,35	88,01	85,32	82,91	82,03	83,31	85,87	88,52	90,76	92,55	93,95	95,07	95,96	96,70	97,30
6	97,04	96,36	95,52	94,46	93,09	91,27	88,75	85,18	80,13	74,13	71,39	75,28	81,27	86,00	89,33	91,68	93,40	94,69	95,70	96,51	97,16
5	96,94	96,23	95,33	94,19	92,67	90,56	87,46	82,47	73,50	55,68	39,85	60,24	75,82	83,70	88,19	91,05	93,01	94,44	95,53	96,38	97,07
4	96,91	96,18	95,27	94,09	92,52	90,30	86,94	81,27	69,62	32,07	99,80	45,11	72,85	82,70	87,75	90,81	92,88	94,36	95,47	96,34	97,04
3	96,94	96,23	95,33	94,19	92,67	90,56	87,46	82,48	73,51	55,71	39,92	60,26	75,82	83,70	88,19	91,05	93,01	94,44	95,53	96,38	97,07
2	97,04	96,36	95,52	94,46	93,09	91,27	88,75	85,18	80,14	74,15	71,41	75,30	81,28	86,01	89,33	91,68	93,40	94,69	95,70	96,51	97,16
1	97,19	96,56	95,80	94,86	93,70	92,23	90,35	88,02	85,33	82,91	82,04	83,32	85,88	88,52	90,77	92,55	93,96	95,07	95,96	96,70	97,30
0	97,38	96,82	96,15	95,35	94,39	93,25	91,91	90,41	88,91	87,75	87,37	87,93	89,19	90,72	92,19	93,49	94,60	95,52	96,29	96,94	97,49

(b) Uçuş başı komut tablosu gösterimi.

Şekil 3.29: Vektörel alandan uçuş komutalarının üretilmesi gösterimi.

Platformun davranışını biçimlendiren alan içerisinde iki adet iten ve bir adet çeken özellikte potansiyel kaynak yer almaktadır. Vektör alanın oluşturulması için ilk olarak her bir potansiyel alanın türevi ilgili denklemler ile hesaplanacaktır. Daha sonra her bir vektör alanın hesaplanmasından sonra toplam vektör alan elde edilecektir. Platformu yönlendirmek için, vektör alan uçuş başı ve sürat komutlarına dönüştürülecektir. Uçuş başı vektörün yönünden, sürat ise vektörün genliğinden faydalanılarak her bir (x, y) noktası için ayrı ayrı hesaplanır. Bu değerler sırasıyla Denklem (3.46) ve (3.47)'de gösterilen şekilde hesaplanır. Uçuş alanı içerisinde yer alan her bir (x, y) noktası için uçuş başı ve sürati komutları aynı prosedürden

faydalanarak hesaplanır ve Şekil 3.29 (a) ile gösterilen uçuş başı ve Şekil 3.29 (b) ile gösterilen uçuş sürati tabloları elde edilir.

4. YPA TABANLI ÜÇ BOYUTLU YOL PLANLAMASI

Bu tez çalışmasının amacı; önceki bölümlerde de açıklandığı üzere İHA platformlarından bir formasyon teşkil edilmesine imkan sağlayan ve bu esnada platformların birbirleri ve etraflarında yer alan engeller ile çarpışmasını engelleyebilen nitelikte yol planlaması çözümü üretmektir. İHA gibi yüksek süratli otonom platformların ihtiyacına cevap verebilecek özellikte gerçek zamana yakın sonuçlar üretebilen bir yol planlaması yaklaşımını ortaya koymaktır. Bu amaç doğrultusunda yol planlamasının literatürde yer alan örnekler arasında bulunan YPA yaklaşımı ile modellenmesine ve hesaplanmasına karar verilmiştir.

Ancak literatürde yer alan YPA yaklaşımı ile gerçekleştirilen yol planlaması çalışmaları incelendiğinde çoğunlukla iki boyutlu ve sınırlı ölçek değerindeki alanlar için faydalanıldığı, genellikle bu ortamların durağan olarak tanımlandığı görülmüştür. Fakat tez kapsamında ortaya konulan problem ise üç boyutlu, nispeten geniş ölçekli ve de dinamik alanların modellenmesine ihtiyaç duymaktadır. Çünkü oluşturulmak istenen formasyon yaklaşımı ve bu oluşum esnasında platformların birbirleri ile çarpışmasını önleme isteği modelin doğası gereği dinamik olmasını zorunlu kılmaktadır.

Yol planlaması yapılacak olan alan içinde hem ulaşılmak istenen hedef nokta hem de sakınılmak istenen engeller hareketlidir. Bu durum modellemede getirdiği zorlukların ötesinde gerçek zamana yakın hesaplama performansına olumsuz etki etmektedir. Bu bölüm kapsamında otonom İHA sistemlerinin YPA tabanlı yol planlaması kapsamında ortaya çıkan gerçek zamanlı çözüm ihtiyacı irdelenmiş ve faklı örnek modeller üzerinde tanımlanmıştır.

Bu durumun önüne geçilebilmesi amacıyla çalışmanın bu başlığı altında öncelikle ihtiyaç duyulan üç boyutlu dinamik modelleme ihtiyacı tanımlanmış ardında özellikle İHA benzeri otonom hava platformlarında halihazırdaki algılayıcı teknolojilerinden faydalanılarak etkin olarak kullanılabileceği değerlendirilen özgün çok katmanlı modelleme yaklaşımı ortaya konulmuştur.

4.1 YPA Tabanlı Üç Boyutlu Yol Planlaması

YPA tabanlı yol planlaması yaklaşımı iki veya üç boyutlu ortamlarda otonom yol planlaması yapılmasına imkan sağlayan bir yaklaşımdır [11][29]. Bunun yanı sıra YPA yaklaşımı ile durağan veya dinamik özellik arz eden ortamlar için yol planlaması gerçekleştirmek mümkündür.

İHA platformları üç boyutlu ortamda yol planlamasına ihtiyaç duyan otonom sistemlerdir. Ayrıca İHA platformlarından formasyon gibi icra etmesi beklenen görevler incelendiğinde ve uçuş emniyetinin etkin olarak sağlanabilmesi amacıyla çarpışmayı önleme gibi kazandırılmak istenen yetenekler göz önünde bulundurulduğunda, ihtiyacın sürekli değişiklik arz eden dinamik ortamlar için yol planlaması olduğu açıktır.

Ancak yol planlamasının üç boyutlu, geniş ölçekli ve dinamik olarak önceden kestirilemeyen değişiklikleri içeren bir alan dahilinde gerçekleştirilmesi her ne kadar YPA yaklaşımları ile mümkün olsa da, yüksek süratli ve agresif manevra yeteneğine sahip İHA benzeri platformların anlık ihtiyaçlarını karşılamak açısından yaklaşımın hesaplama performansı açısından sıkıntılar yaşanacağı değerlendirilmektedir. Özellikle önceki bölümler kapsamında açıklanan ve YPA yaklaşımlarında görülen temel problemlerin çözümü için genel yol planlaması icra edilmesine gerek duyulan durumlarda klasik yöntemler ile gerçekleştirilen hesaplama performansının yetersiz kalacağı bir gerçektir.

Çalışmanın bu bölümünde öncelikli olarak üç boyutlu ortamlar dahilinde İHA platformları için ideal yol planlaması yaklaşımının nasıl gerçekleştirilmesi gerektiğine değinilmiştir. Ardından bu başlık altında hesaplama yükünün azaltılması, dolayısıyla en iyi hesaplama performansının elde edilebilmesi amacıyla özgün olarak geliştirilen ve İHA sistemlerinde kullanılan mevcut algılayıcılar ile uygulanabileceği değerlendirilen çok katmanlı yaklaşıma değinilmiştir.

4.1.1 Üç boyutlu modelleme

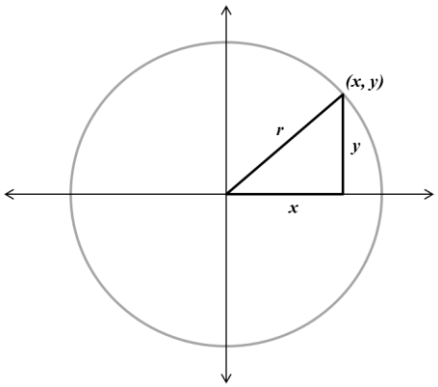
İster tek başına bir görev icra etsin, isterse de bir grup halinde oluşturulan bir formasyonun üyesi olsun İHA platformlarının yol planlaması, içinde bulundukları ortam ve platformların hareket kabiliyetleri gereği üç boyutlu olarak ele alınmalıdır.

Temel YPA modelleri arasında yer alan noktadan çeken ve iten alanlar çalışmanın bu bölümünde örnek olarak seçilmiş ve bunların üç boyutlu olarak nasıl modellendiğine değinilmiştir. İHA sistemleri için gerçekleştirilen YPA tabanlı yol planlamasında noktadan iten potansiyel alanlar ile engeller ve ortam içinde yer alan diğer platformlar modellenebilir. Çeken nitelikteki alanlar ile de platformun ulaşmak istediği konum noktası yani hedef modellenebilir.

YPA tabanı iki boyutlu yol planlaması durumunda iten ve çeken alanların modellenmesi esnasında Denklem (4.1) ile gösterilen temel daire denkleminden faydalanılmıştır.

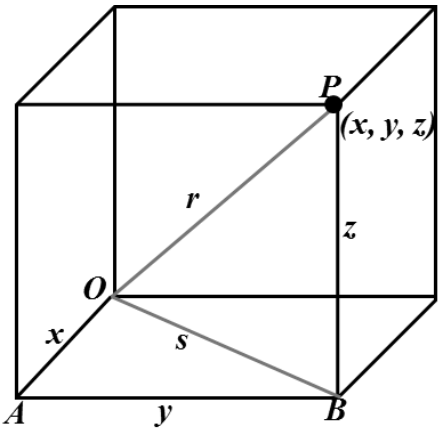
$$x^2 + y^2 = r^2 \tag{4.1}$$

Denklem (4.1) ile merkez noktası orijin ve yarıçapı r olarak kabul edilen daire elde edilebilir. Bu Pisagor teoreminin bir cebirsel ifadesidir. Şekil 4.1 ile gösterildiği biçimde, üçgen içinde hipotenüs uzunluğu, bu durumda yarıçap (r), üçgenin diğer bacaklarının uzunluğunun karesinin toplamının karekökünü eşittir.



Şekil 4.1: Daire için temel Pisagor teoremi gösterimi.

Esasında tüm yapay potansiyel alan yaklaşımı iki boyut için bu teorem üzerine oluşturulmuştur. Bir potansiyel kaynağın etkisi, etki ettiği ızgara hücresinin kaynağa olan mesafesi ve açısı ile elde edilir. Bu noktadan hareket etmeye devam edersek, üç boyutlu bir ortamda daire temsilinin yerine küre temsilinin kullanılabileceği söylenebilir. Bu durumda Pisagor teoreminden faydalanmaya devam edilirse iki boyutlu potansiyel ve vektör alan denklemlerini küre yapısına uyarlamak gerekecektir.



Şekil 4.2: Küre için temel Pisagor teoremi gösterimi.

Şekil 4.2'de görüldüğü üzere *O* noktası orijini, *P (x,y,z)* noktası küre yüzeyinde yer alan üç boyutlu bir noktayı temsil etmektedir. Kürenin merkez noktası olan *O* ile küre yüzeyinde yer alan *P* noktası arasındaki mesafe *r* ile gösterilmekte olup, aynı zamanda bu değer kürenin yarıçapıdır. Bu durumda oluşturulan *OAB* dik üçgeni için Pisagor denklemi uygulanabilir ve *s* uzunluğu elde edilir. Aynı şekilde oluşan *OBP* üçgeninden faydalanılarak *r* uzunluğu bu durumda hesaplanabilir. Bu iki denklemden şayet kürenin merkezi orijinde ise Denklem (4.2) ile ifade edilen bağıntı elde edilebilir.

$$x^2 + y^2 + z^2 = r^2 \tag{4.2}$$

Aynı şekilde şayet küre yüzeyinde yer alan üç boyutlu bir noktanın *P (x,y,z)* ile temsil edildiği ve kürenin merkezinin orijin yerine *M (h,i,j)* noktasında bulunduğu kabul edilir ise bu kez eşitlik Denklem (4.3) ile gösterildiği şekilde ifade edilir.

$$(x-h)^2 + (y-i)^2 + (z-j)^2 = r^2 \tag{4.3}$$

Bu temel noktadan hareket ile Denklem (3.14) ile verilen iki boyutlu potansiyel alan için çeken alan potansiyel modeli, Denklem (4.4) ile gösterildiği biçimde üç boyutlu alana uyarlanabilir.

$$F_{c_3}(x,y,z) = -log\big(\delta((x-x_h)^2 + \gamma(y-y_h)^2 + \varepsilon(z-z_h)^2)\big) \tag{4.4}$$

Robotun pozisyonunun üç boyutlu uzayda (x, y, z) noktası olduğu kabul edildiğinde Denklem (4.4)'deki küresel çeken alan fonksiyonun merkez noktası (x_h, y_h, z_h) ile temsil edilmiştir. Kontrol değişkleri γ ve ε kürenin formunu eksen oranları olarak gösterir. Eğer $\gamma = \varepsilon = 1$ olursa fonksiyon tam küre şekilde bir form aksi taktirde ellipsoid bir form oluşturacaktır.

Bu durumda üç boyutlu olarak çeken potansiyel alanları oluştururken kullandığımız Denklem (4.4) için her bir (x, y, z) noktasının gradienti (∇) Denklem (4.5)'de gösterildiği şekilde hesaplanabilmektedir:

$$d_x = \frac{\partial F_{c_{3rot}}}{\partial x} = -\frac{2(x - x_h)}{(x - x_h)^2 + \varepsilon(z - z_h)^2 + \gamma(z - z_h)^2}$$
$$d_y = \frac{\partial F_{c_{3rot}}}{\partial y} = -\frac{2\gamma(y - y_h)}{(x - x_h)^2 + \varepsilon(z - z_h)^2 + \gamma(z - z_h)^2} \quad \textbf{(4.5)}$$
$$d_z = \frac{\partial F_{c_{3rot}}}{\partial z} = -\frac{2\varepsilon(z - z_h)}{(x - x_h)^2 + \varepsilon(z - z_h)^2 + \gamma(z - z_h)^2}$$

Ellipsoid forma Denklem (3.16)'da iki boyutlu alan için gösterilen şekilde dönme etkisi verilebilir.

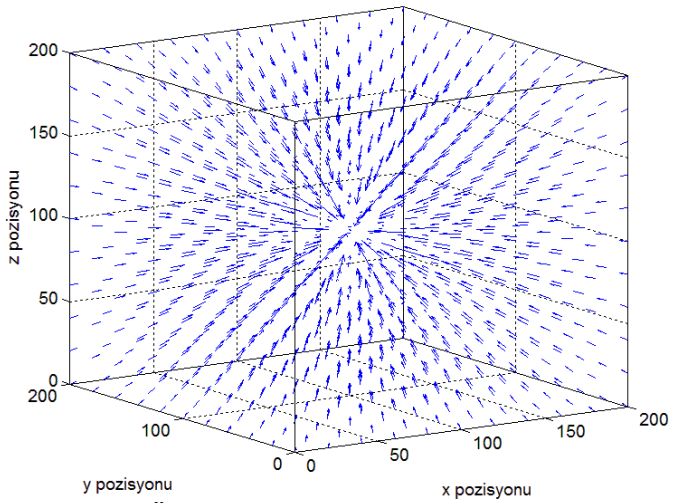
Sonuç olarak elde edilen potansiyel alan denklemlerinden Denklem (4.6) ile gösterildiği biçimde gradient operatörü vasıtasıyla iten ve çeken özelliklerdeki vektör alanlar üretilebilir.

$$\nabla F_{c_{3rot}}(x, y, z) = \frac{\partial F_{c_{3rot}}}{\partial x}\hat{\imath} + \frac{\partial F_{c_{3rot}}}{\partial y}\hat{\jmath} + \frac{\partial F_{c_{3rot}}}{\partial z}k \quad \textbf{(4.6)}$$

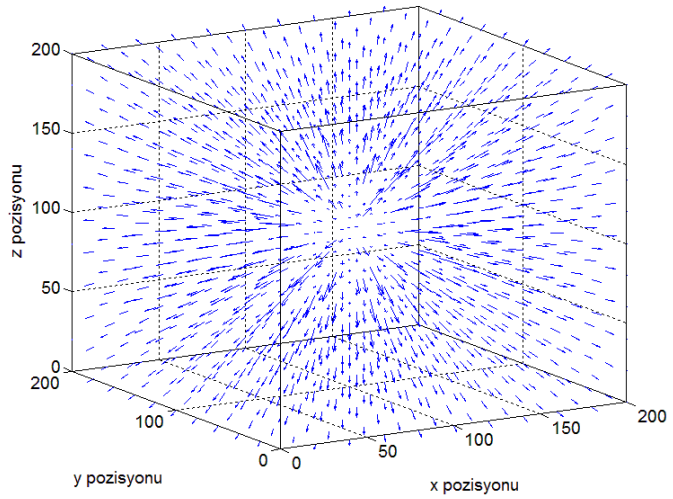
Bu durumda üç boyutlu ortamda yer alan ve parametre değerleri Çizelge 4.1 ile gösterilen çeken yönlü alan Şekil 4.3 (a) ile gösterildiği biçimde, iten yönlü alan ise Şekil 4.3 (b) ile gösterildiği biçimde elde edilir.

Çizelge 4.1: Üç boyutlu potansiyel alanda yer alan temel alan parametreleri.

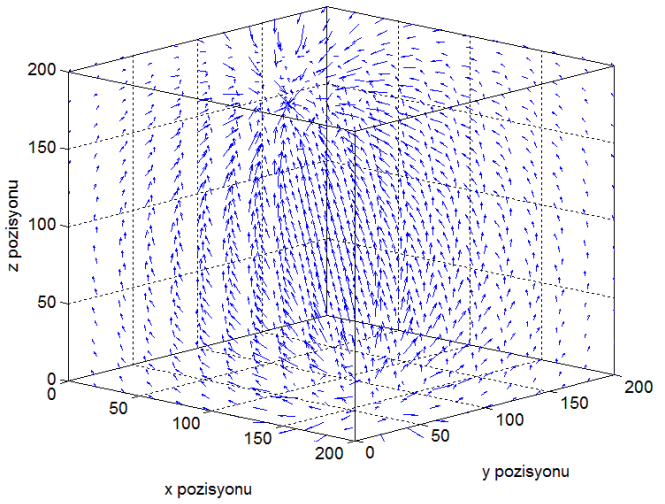
Parametre	**Değer**
$size\ (Alan\ Boyut)$	200x200
$Res\ (Alan\ \text{Çöznürlük})$	10
δ_c	1.0
γ_c, ε_c	1.0
C_x, C_y	30 60
β_c	π
δ_I	1.0
γ_I, ε_I	1.0
I_x, I_y	120,170
β_I	π



(a) Üç boyutlu çeken alan gösterimi.



(b) Üç boyutlu iten alan gösterimi.



(c) Üç boyutlu sınırlandırılmış toplam alan gösterimi.

Şekil 4.3: YPA ile üç boyutlu yol planlaması amaçlı vektör alan gösterimi.

Elde edilen potansiyel alan değerleri iki boyutlu alan için uygulandığı şekilde ikili veya sigmoid sınır fonksiyonlarından faydalanılarak sınırlandırılabilir ve Denklem

(4.6) ile ifade edildiği şekilde birden fazla temel üç boyutlu alandan faydalanılarak Şekil 4.3 (c) ile gösterilen biçimde vektör alan üretilebilir.

$$
\begin{aligned}
SX_s(x,y,z) = & \sum_{k=0}^{m} S_{C3(k)_s}(x,y,z).DX_{c3(k)}(x,y,z) \\
& + \sum_{k=0}^{n} S_{I3(k)_s}(x,y,z).DX_{I3(k)}(x,y,z) \\
& + \sum_{k=0}^{p} S_{Rp3(k)_s}(x,y,z).DX_{Rp3(k)}(x,y,z) \\
& + \sum_{k=0}^{r} S_{Rn3(k)_s}(x,y,z).DX_{Rn3(k)}(x,y,z) \\
SY_s(x,y,z) = & \sum_{k=0}^{m} S_{C3(k)_s}(x,y,z).DY_{c3(k)}(x,y,z) \\
& + \sum_{k=0}^{n} S_{I3(k)_s}(x,y,z).DY_{I3(k)}(x,y,z) \\
& + \sum_{k=0}^{p} S_{Rp3(k)_s}(x,y,z).DY_{Rp3(k)}(x,y,z) \\
& + \sum_{k=0}^{r} S_{Rn3(k)_s}(x,y,z).DY_{Rn3(k)}(x,y,z) \\
SZ_s(x,y,z) = & \sum_{k=0}^{m} S_{C3(k)_s}(x,y,z).DZ_{c3(k)}(x,y,z) \\
& + \sum_{k=0}^{n} S_{I3(k)_s}(x,y,z).DZ_{I3(k)}(x,y,z) \\
& + \sum_{k=0}^{p} S_{Rp3(k)_s}(x,y,z).DZ_{Rp3(k)}(x,y,z) \\
& + \sum_{k=0}^{r} S_{Rn3(k)_s}(x,y,z).DZ_{Rn3(k)}(x,y,z)
\end{aligned}
\tag{4.7}
$$

Bu durumu tüm temel alan modellemelerine uyarlanamak ve iki boyutlu hesaplama yerine doğrudan üç boyutlu potansiyel model ve denklemler kullanarak yol planlamasını gerçekleştirmek mümkündür.

4.1.2 Çok katmanlı modelleme yaklaşımı

Yol planlaması amacıyla üç boyutlu modellerin üretilmesinde ve hesaplanmasında özellikle uygulamada sıkıntılar yaşanacağı değerlendirilmektedir. Bunun temel iki sebebi vardır; ilki iki boyutlu modellemeye nazaran üç boyutlu modelleme

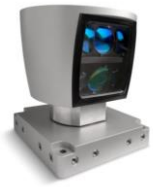
yaklaşımının hesaplama maliyetini aşırı derecede arttıracak ve geniş ölçekli ve yüksek çözünürlük değerine sahip dinamik alanların YPA ile modellenmesi çerçevesinde gerçek zamanlı yanıtlar üretilmesini günümüz hesaplama mimarileri ile mümkün kılmayacaktır. İkinci neden ise halihazırda İHA teknolojilerinde kullanılan engel ve diğer hava platformlarının konumlarını tespit etmek amacıyla faydalanılan algılayıcı sistemlerin üç boyutlu modelleme amacıyla yeterli bilgiyi üretmede yaşadıkları yetersizliktir.

Literatür incelendiğinde üç boyutlu ortamlar içerisinde engellerden sakınma amaçlı oluşturulmuş üç boyutlu farklı YPA modellemelerine rastlanabilir [29]. Ancak gerek tez kapsamında yer alan yaklaşım gerekse de literatürde yer alan diğer bu amaçlı yaklaşımlar engelin her noktasının üç boyutlu konumunun bilinmesini gerektirmektedir. Fakat insanlı ya da insansız hava araçlarında kullanılan algılayıcı teknolojileri incelendiğinde özellikle coğrafi engellerin tamamını üç boyutlu olarak algılamaya yönelik sensor sistemlerinin yaygın olmadığı, olanların ise oldukça kısıtlı veri üretme imkanına sahip oldukları değerlendirilmektedir.

Otonom bir sistem tarafından platformun çevresini algılayabilmesi, tanımlayabilmesi ve engelleri tespit edebilmesi amacıyla kullanılan algılayıcılar aktif ve pasif olmak üzere iki ana başlık altında gruplanabilirler. Pasif algılayıcılar; optik akış algılayıcılar, mono veya stereo kamera sistemleri veya kızılötesi kameralar şeklinde örneklenebilir. Bu tip algılayıcılar güneş ışığı ile üretilen ve engelden geri yansıyan veya engelin kendisi tarafından teşhir edilen işarları alarak çalışırlar. Bu algılayıcıların engel tespiti ve yol planlaması amacıyla kullanımları oldukça kısıtlıdır [16]. Bir diğer grup ise aktif algılayıcılar adını verdiğimiz kendi enerji kaynaklarına sahip olan ve bu enerjiyi yayarak geri toplama prensibi doğrultusunda hareket eden sistemlerdir. Radyo Sinyali ile Tespit ve Mesafe Ölçme Sistemi (Radio Detection & Ranging System – RADAR), Ses Sinyali ile Tespit ve Mesafe Ölçme Sistemi (Sound Navigation & Ranging System – SONAR) ve Lazer ile Tespit ve Mesafe Ölçme Sistemi (Laser Detection & Ranging System – LIDAR) gibi sistemleri aktif algılayıcı sistemlere örnek olarak saymak mümkündür.

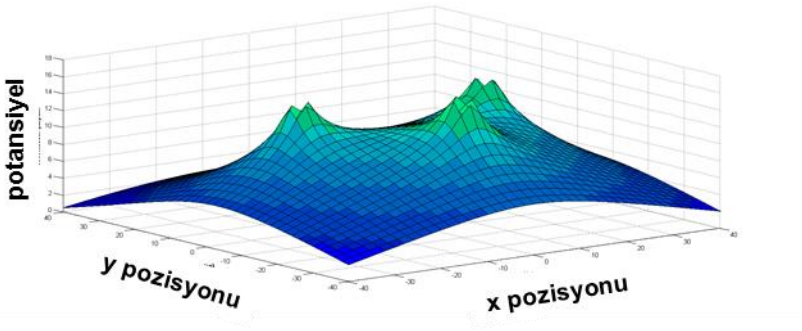
İnsanlı hava platformlarında uçuş doğrultusunda yer alan engelleri algılayabilmek ve bunlara bağlı olarak pilotu yol planlamasını değiştirmesi hususunda uyarmak amacıyla tasarlanmış Yüzey Takip Radarı (Terrain Following Radar – TFR) adı verilen radar sistemleri mevcuttur [85].

Çizelge 4.2: İHA platformlarında kullanılan engel algılayıcı sistemler [16].

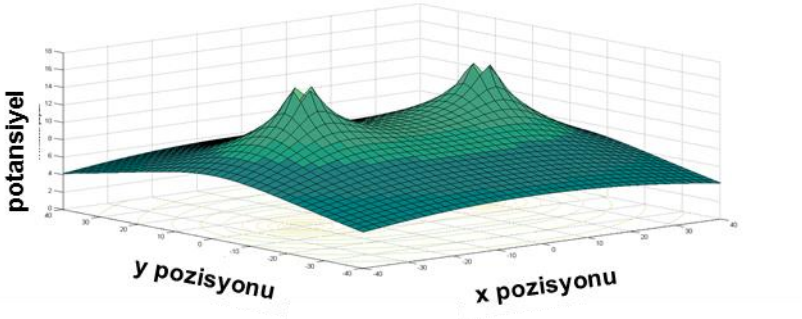
Algılayıcı	Ağırlık ve Güç Tüketimi	Algılama açısı (Yatay/Dikey Derece)	Algılama menzili (m.)	Görünüm
SONAR Range Finder [16]	30-40 gr <1 W	0^o / 0^o	15 m.	
Sick Laser [16]	4,5 kg. 20 W	160^o / 0^o	~70 m.	
Velodyne HDL-64E Laser [16]	13 kg. 25 W	120^o / $\pm 3^o$	120 m.	
Velodyne HDL32E [16]	1.2 kg 12 W	360^o / ($+10^o$ -30^o)	~160 m.	
AN/AAQ-33 [87]	200 kg.	160^o / -21^o	~60 km.	

Ancak İHA sistemlerinde uçuş doğrultusunda yer alan engelleri tespit edebilecek algılayıcı mimarileri oldukça sınırlıdır. Halihazırda bu sistemler üzerine gerçekleştirilen çok sayıda çalışma mevcuttur [16]. İHA platformlarında kullanılan sensor mimarileri incelendiğinde ortaya Çizelge 4.2 ile gösterilen durum çıkmaktadır.

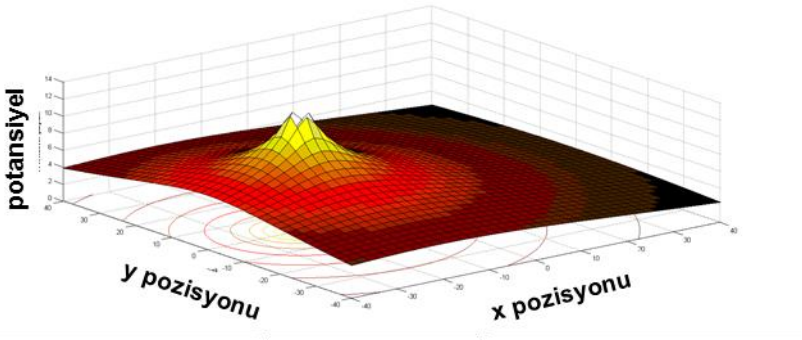
İHA platformlarında kullanılanılan fiziki engellerin tespiti amaçlı algılayıcılar incelendiğinde bu algılayıcıların iki boyutlu çalışma performanslarının üç boyutlu algılama yeteneklerine nazaran çok üstün olduğu ve üçüncü boyutu oluşturan dikey algılama açılarının çok kısıtlı olduğu görülmüştür.



(a) Birinci seviye modelleme gösterimi.



(b) İkinci seviye modelleme gösterimi.



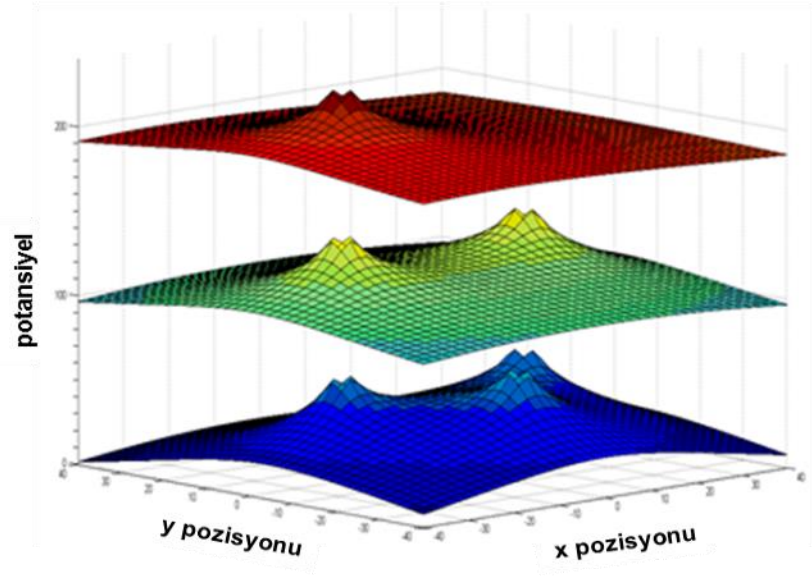
(c)Üçüncü seviye modelleme gösterimi.

Şekil 4.4: Farklı irtifalarda tespit edilen engellerin YPA yaklaşımı ile modelleme gösterimi.

İnsanlı hava araçlarından biri olan ve son güncel çalışmalar ile insansız hale getirilmeye çalışıldığı değerşendirilen F-16 Falcon muharip savaş uçağının radarı AN/APG-68, uçuş hattı doğrultusunda en fazla yatayda 60 derece, dikeyde 30 derece açıyla diğer hava araçlarını yaklaşık 150 deniz mili mesafeden algılayabilmekteyken [86], gece uçuşlarında faydalanılan AN/AAQ-33 Lantırn Podu ise en fazla yatayda 28 derece dikeyde 21 derece açıyla coğrafi engelleri en fazla 60 km. mesafeden algılayabilmektedir [87]. Şu ana kadar geliştirilen İHA sistemlerinde bu kadar gelişmiş bir radar kullanımının gerek ağırlık, ebat gibi fiziksel nedenlerden gerekse de diğer aviyonik destek sistemlerinin olmayışı gibi nedenlerden dolayı kullanımının

söz konusu olmadığından, hali hazırda İHA sistemlerinde coğrafi engelleri uçuş hattı doğrultusunda ve özellikle de hattın üzerindeki irtifalarda üç boyutlu olarak engellerden sakınmak amacıyla tespit edebilecek bir algılayıcının yaygın kullanımından söz edememekteyiz.

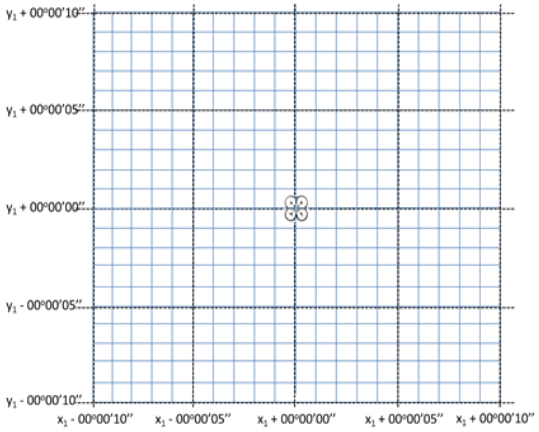
Bu veriler ile ortaya konulduğu üzere uygulanabilir biçimde üç boyutlu bir potansiyel alan tanımı yapılması ve hatta engellerden otonom şekilde kaçınma amacıyla kullanılması pratikte mümkün olarak değerlendirilmemektedir. Bu nedenle problemin pratik ve mevcut yaygın teknolojilerden faydalanılarak çözümü için tez kapsamında çoklu katman YPA modeli adını verdiğimiz bir yaklaşım üzerinde durulacaktır. Şekil 1.2'de gösterildiği gibi İHA platformunun uçuş irtifasında sensoru ile iki boyutlu bir tarama gerçekleştirdiği kabul edilecek ve farklı irtifalarda tespit ettiği engelleri dinamik olarak Şekil 4.4 ile gösterilen biçimde YPA ile modelleyecektir. Şayet anlık olarak oluşturulan hedef doğrultusundaki uçuş rotasından farklı bir yol planlaması ortaya çıkıyor, yani uçuş doğrultusunda engel bulunuyorsa ve platform bu engele ulaşmadan önce irtifa alabilme kabiliyetine sahip ise, anlık sürat değerine bağlı olarak hesap edilen minimum dönüş yarıçapı içerisine girmeden platforma irtifa alması komutu verilecektir. Belirli irtifa aralıklarında oluşturulan iki boyutlu YPA modeli algılayıcılardan gelen engel verilerine bağlı olarak tırmanış esnasında güncellenecektir. Böylece üst üste bindirilmiş dikey derinliği olan bir YPA modeli oluşturulması sağlanacaktır. Örneğin 100 metre irtifada bir modellenerek üst üste bindirilmesi ön görülen çok katmanlı YPA modelinin bir tasarımı Şekil 4.5'de gösterilmektedir. Şekil incelendiğinde görüleceği üzere irtifa arttıkça karşılaşılan coğrafi engellerin sayısında azalma olacağı ve de formlarının değişikliği göstereceği beklenmektedir. Şayet İHA platformu engele dönüş kütürü içerisine girecek kadar yaklaşmış veya platformun tavan irtifasına yakın bir değere ulaşılmış ise iki boyutlu olarak modellenen YPA baz alınarak erişilen irtifada engelden kaçınma manevrası icra edilecektir. Engel geçildikten sonra aynı şekilde çok katmanlı yapıdan faydalanılarak rota irtifasına geri dönülecektir.



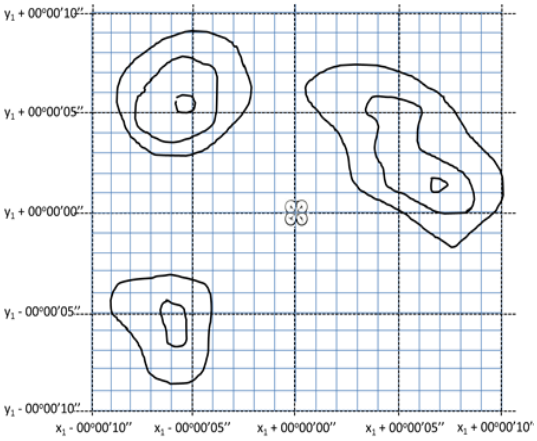
Şekil 4.5: Çok katmanlı YPA yaklaşımı ile modelleme gösterimi.

Bu yaklaşım hali hazırda İHA sistemlerinde kullanılan mevcut SAR, optik ya da lazer tipi algılayıcılar ile üç boyutlu engellerden bir kol düzeni şeklinde uçulurken başarı ile sakınma sağlamaya yetecek nitelikte bir veri alınmasını amaçlamakta ve yaklaşımın gerçekleştirilebilirliğini güçlendirmektedir. Bunun yanında hali hazırda yüksek çözünürlük ve de dinamik olarak değişen ortam koşulları nedeniyle fazla işlemci gücü gerektiren YPA hesaplanmasına bir karmaşıklık faktörü daha eklemekte ve hesaplanması gereken vektör büyüklük sayısının daha da artmasına ve böylece işlemci gücü ihtiyacının daha da büyümesine neden olmaktadır. Ancak üç boyutlu hesaplama ihtiyacına istinaden oldukça tasarruf edilmektedir.

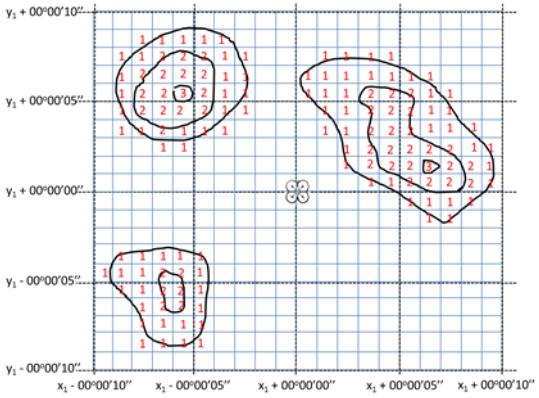
Çalışma kapsamında ortaya konulacak olan araştırmanın amacı coğrafi özellikleri önceden bilinmeyen bir alan içerisinde GPS tabanlı seyrüsefer esnasında engellerden kaçınacak şekilde oluşturulan formasyonu muhafaza edecek bir kontrol algoritması geliştirmek ve de ortaya koymaktır. Formasyonun oluşturulması ve de muhafazası amacıyla kullanılacak olan yaklaşım potansiyel alan tabanlı olacaktır. Formasyon düzeni içinde yer alan her bir platformun hareketini doğrudan etkileyen iki önemli unsur yer almaktadır. Bu unsurlar formasyonu teşkil eden diğer platformlar ve de seyrüseferin icra edildiği alan içerisinde yer alan fiziki engel teşkil eden unsurlar olarak ikiye ayrılabilir. Seyrüseferi icra etmek için çalışma kapsamında faydalanılan konumlandırma sistemi GPS tabanlı konumlandırma sistemi olarak belirlenmiştir. Her bir platform kendi konumunu üzerinde yer alan GPS alıcısı vasıtasıyla belirleyecektir.



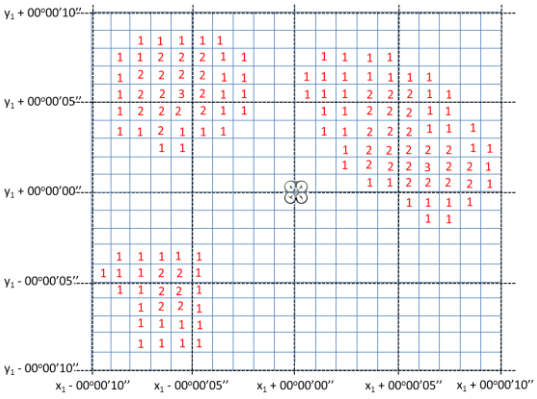
(a) GPS bazlı platform göre konumlandırma yapısı.



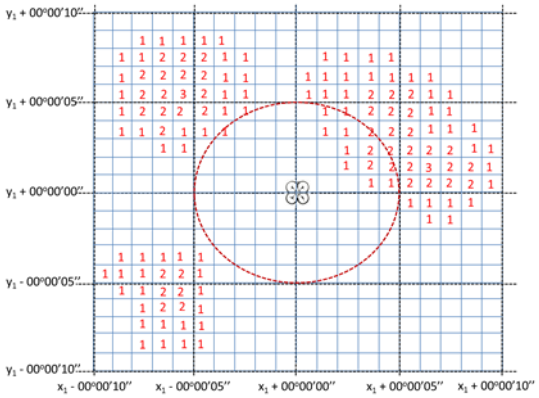
(b) İzohips (eş yükselti eğrileri) ile engellerin temsili.



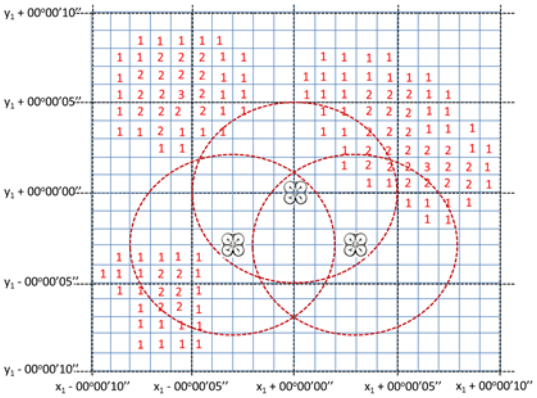
(c) Grid tabanlı engellerin temsili.



(ç) Platform konumuna bağlı coğrafi engellerin grid alan üzerinde temsili.



(d) Platform sensorunun algılayabildiği coğrafi engel alanı temsili.



(e) Birden fazla platformun sensorları ile algılayabildiği coğrafi engel alanı temsili.

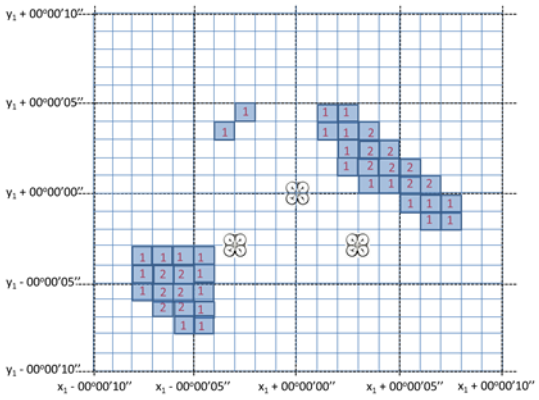
Şekil 4.6: GPS tabanlı konumlandırma ortamında grid tabanlı engellerin platform sensorları ile tespit edilmesi.

Bu bilgi diğer platformlar ile kablosuz iletişim sistemi üzerinden paylaştırılabilir. Platformların konum bilgilerinin gerçekçiliği GPS konumlandırma sisteminin

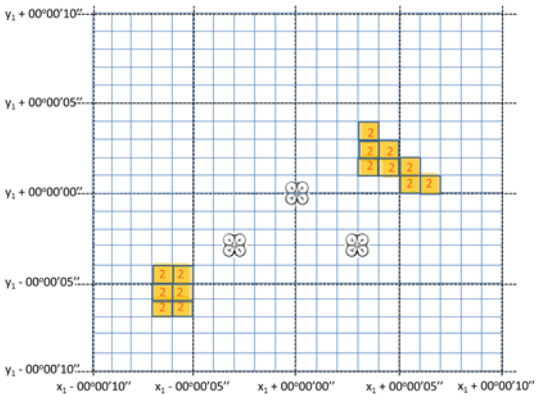
hassasiyeti ile sınırlıdır. Bu hassasiyet en iyi durumda yaklaşık 2 ile 8 metre arasındadır. GPS tabanlı konumlandırma sistemine göre platformun konumunu (x_1,y_1) olarak kabul ederek, GPS çözünürlüğünü baz alacak biçimde Şekil 4.6 (a) ile gösterilen yapıda bir ızgara (grid) alan ortaya konulabilir. Bu yapı üzerine eş yükselti eğrileri (izohips) ile temsil edilen Şekil 4.6 (b) ile gösterildiği biçimde engeller yerleştirilebilir. Bu engellerin platform tarafından tespit edilebilirliği dışında var olan yapıyı temsil etmek amacıyla oluşturulmuştur. Eş değer yükselti eğrileri baz alınarak ızgara içerisinde belirli bir dikey çözünürlük aralığı baz alınarak (örneğin 100 metre gibi) her bir hücre içerisindeki en yüksek değer temsilen Şekil 4.6 (c)'de gösterildiği biçimde gösterilebilir. Bu yapı engellerin ızgara tabanlı bir yapıda temsili kolaylaştıracaktır. Şekil 4.6 (ç) ile gösterildiği biçimde izohips çizgileri kaldırıldığında geriye sadece ızgara hücreleri ile temsil edilebilecek nitelikte engelleri temsil eden yükseklik değerleri kalacaktır. Platform üzerinde çevresindeki engelleri belirli bir mesafeye kadar tespit edebilecek nitelikte Şekil 4.6 (d) ile temsil edilen biçimde bir algılayıcı olduğu kabul edildiğinde önceden coğrafi özellikleri bilinmeyen bir alan içerisinde hareket eden bir yapı modellemesi geliştirilebilir.

Bu noktada önemli olan husus coğrafi engellerin statik olduğunun kabul edilmesi ve dinamik özelliklere sahip mobil platformun bu yapı içinde hareket etmesidir. Birbirlerine göre konumları GPS bazlı konumlandırma sistemine göre ifade edilebilen birden fazla platformun tanımlanan alan içerisinde bir arada yer aldığı ve her bir platformun benzer coğrafi engel algılayabilen sensorlar ile donatıldığı kabul edildiğinde Şekil 4.6 (e) ile gösterilen biçimde engellerin tespit edilmesi olanağı elde edilmiş olacaktır.

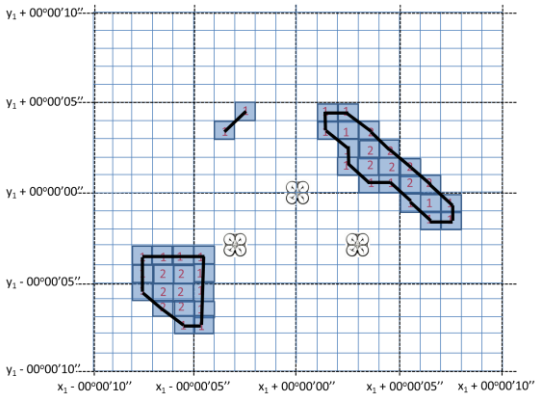
Bu noktada belirtilmesi gereken bir diğer husus da farklı platformlar tarafından tespit edilen engellerin diğer platformlar için de engel teşkil edebilmesi için her bir platformun tespit ettiği engel bilgilerini diğerleri ile paylaşmasına olanak sağlayacak nitelikte bir haberleşme mimarisi ile desteklenmesi gerekmektedir. Bu mimariye raporun ilerleyen bölümlerinde ayrıca değinilmiştir.



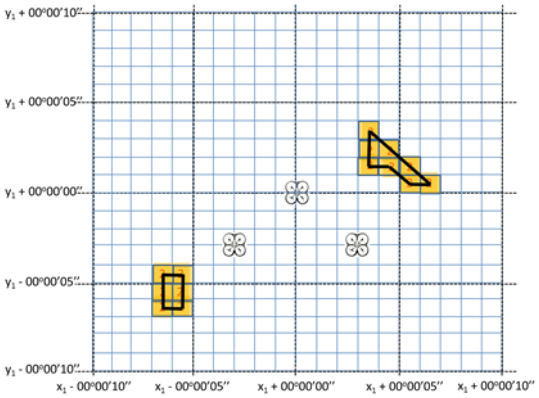
(a) Basit birinci katman (layer-1) engel gösterimi.



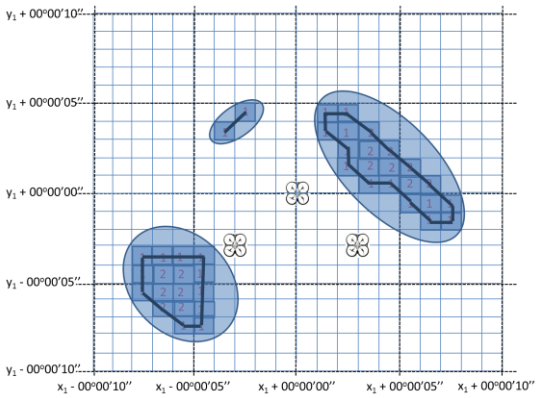
(b) Basit ikinci katman (layer-2) engel gösterimi.



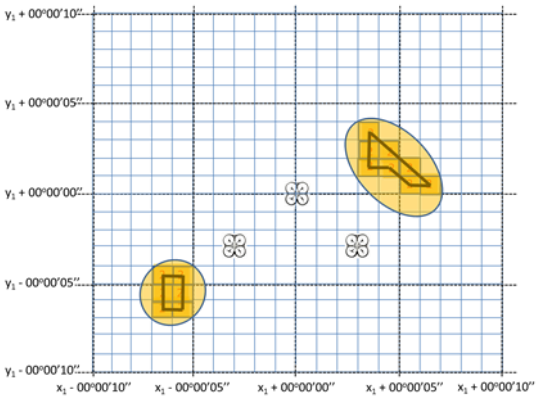
(c) Basit birinci katman engel tespiti.



(ç) Basit ikinci katman engel tespiti.



(d) Basit eliptik potansiyel alan ile engel temsili.



(e) Basit eliptik potansiyel alan ile engel temsili.

Şekil 4.7: GPS tabanlı konumlandırma ortamında grid tabanlı engellerin çok katmanlı potansiyel alan ile temsil edilmesi.

Şekil 4.7 (a) ile belirlenen coğrafi engel temsil alanı içerisinde yer alan üç adet platform tarafından tespit edilebilen birinci katman (layer-1) yüksekliğindeki engeller gösterilmiştir. Aynı biçimde Şekil 4.7 (b) ile de ikinci katman (layer-2)

yüksekliğinde tespit edilebilen engeller gösterilmiştir. Katmanlar arasındaki fark engellerin yüksekliğidir. Izgara şeklindeki GPS tabanlı konumlandırma sistemi içinde temsil edilen coğrafi engellerin doğrular ile çevrelerinin belirlenmesi ile kaba şekilleri Şekil 4.7 (c) ve (ç) ile farklı katmanlar için gösterilmiştir. Potansiyel alanlar ile engellerin temsil edilmesinde faydalanılabilincek en kolay temsil şekli eliptik engel potansiyel alanlar kullanımıdır.

Bu şekilde her ne kadar coğrafi engeller birebir modellenemese dahi etkin bir engel temsili dolayısıyla engelden uzaklaştıran iten nitelikte potansiyel bir alan oluşturulabilir. Bu yapı farklı katmanlar için Şekil 4.7 (d) ve (e) ile gösterilmektedir.

GPS tabanlı konumlandırma sistemi kapsamında tanımlanan tüm coğrafi engeller durağan (statik) olarak tanımlanmaktadır. Ancak alan içerisinde platformlar için engel teşkil eden tek yapılar coğrafi engeller değildir. Belirli bir formasyon dahilinde hareket edebilen mobil platformlar göz önüne alındığında kendisi dışındaki formasyonu oluşturan diğer tüm platformlar çarpışmayı önlemek maksadıyla değerlendirilen mobil engellerdir. Bundan sonraki bölümde mobil engellerin ve dolayısıyla hareketli potansiyel alanların modellenmelerine değinilecektir.

4.2 Gerçek Zamanlı Yol Planlaması İhtiyacı

Bu bölüm kapsamında formasyon uçuşunu teşkil eden yukarıda örnekleri verilen İHA platformları için ve yine yukarıda örnekleri verilen algılayıcılar ile geniş ölçekli yüksek çözünürlüklü önceden özellikleri bilinmeyen bir alan içerisinde engellerden sakınarak ve çarpışmayı önleyici özelliklerde YPA tabanlı bir yol planlaması gerçekleştirildiğinde, platformların yol planlaması ihtiyacının ne derece karşılanabileceği, hesaplama performansının gerçek zamanlı hesaplama ihtiyacına yanıt verebilmesi için ne olması gerektiği irdelenmiştir.

Formasyon oluşumunda toplam platform sayısı (P) dört platform olduğu kabul edildiğinde, potansiyel alan her platform için ayrı olarak hesaplanır ve toplam dört farklı potansiyel alanın hesaplanması gerekir. Çarpışmadan kaçınmayı destekleyen formasyon kontrolünü sağlamak için her biri dört farklı temel potansiyel modelinden oluşan dört farklı potansiyel alan oluşturulur ve gerçek zamanlı dinamik olarak hesaplanması gerekmektedir.

Çözünürlük doğrudan hesaplama performansını ve manevraların hassasiyetini etkileyen bir parametredir. Bu nedenle çözünürlük değeri ($\propto metre$) olarak GPS konumlandırma sisteminin sapmalarında göz önünde bulundurularak sağladığı en düşük değerden yüksek olan 2 metre değeri çözünürlük değeri olarak belirlenmiştir.

Çok katmanlı mimariden faydalanıldığı değerlendirildiğinde 100 ft. dikey katman aralığı ve algılayıcı dikey açıları göz önünde bulundurularak dikey uçuş alanı olarak 100m/sn altında hızla uçan platformlar için 500 ft. (alttan ve üstten toplam), üstünde bir hızla uçan platfomlar için ise 1.000 ft. alan belirlendiği, bu durumda toplam katman sayısının (Γ) 5 ile 10 katman arasında olduğu kabul edilmiştir.

Önceki bölümlerde açıklandığı üzere YPA tabanlı yol planlamasının karşılaştığı sorunlara çözüm olması açısından genel yol planlaması yapılacağına karar verilmiştir. Ayrıca tespit ettiği engelleri sürekli olarak hafızasında tutacağı ve de yol planlamasına dahil edeceği kabul edilmiştir. Bu durumda en kötü senaryo dahilinde engel içeren ve hedefe ulaşmak isteyen platfom için engellerin algılanabildiğinin ötesinde görevin icra edileceği tüm alanın modellemesinin yapılması gerekmektedir. Bu nedenle platformun menzili (*M metre*) dahilinde yol planlamasının yapılacağı toplam alan (*A metrekare*) örnek olarak 64 km^2 olarak belirlenmiştir.

Formasyon dahilinde yer alan platformların birbirlerine eşit mesafede olduklarını kabul ederek bu mesafeyi (d) parametresi ile metre cinsinden belirleyebiliriz. Her bir platformun maksimum sürat değerini (V) parametresi ile m/sn cinsinden temsil edebiliriz.

Yukarıda yazdığımız koşullar altında YPA tabanlı yol planlamasının icra edileceği üç boyutlu alanın durağan olduğu kabul edildiğinde bir hücrenin hesaplanması için gereken maksimum süre (*HS*) Denklem (4.8) ile gösterildiği şekilde hesaplanır.

$$HS = \frac{(d/V)}{\left(\frac{M}{\alpha}\right).\Gamma.P} \qquad \textbf{(4.8)}$$

Bu durumda Çizelge 2.2 kapsamında tanımlanan ve özellikleri belirtilen platfromlar arasında yer alan en yavaş İHA olan Aerosonde platformunun, maksimum sürati (V) 40 m/sn., uçuş alanının (A) 8.000 m^2. olduğu kabul edildiğinde yatay çözünürlüğü ($\propto$) 2 m. ve dikey çözünürlüğü 100 ft. dolayısıyla katman sayısı (Γ) 5 olan bir alan içinde formasyonu teşkil eden dört platform (P) olduğu kabul edildiğinde saniyede

hesaplanması gereken hücre sayısı 128.000.000 olarak ortaya çıkacaktır. Bu durumda dört adet Aerosonde platformundan oluşan İHA formasyonunun merkezi bir birim tarafından YPA tabanlı yol planlamasının icra edildiği değerlendirildiğinde, her bir hücre hesaplamasının ~8 nanosaniye içinde, her platformun ayrı işlemciler için hesaplandığı kabul edildiğinde ise ~32 nanosaniyede tamamlanması gerekecektir. Aksi taktirde bu platformların aralarının 100 m.'den az olması ve bu esnada genel yol planlaması icra etmeleri mümkün değildir.

5. YPA TABANLI DINAMIK YOL PLANLAMASI

Çalışmanın bu bölümünde YPA tabanlı dinamik yol planlaması yaklaşımına çalışma kapsamında belirlenen örnek problemlere istinaden senaryolar geliştirilmiştir. Bu senaryolar İHA sistemleri tarafından uygulanması istenen formasyon davranışlarına örneklerdir.

Havada Yakıt İkmali (HYİ) senaryosu kapsamında insanlı ve insansız hava araçlarının oluşturduğu formasyona değinilmiştir. HYİ görevi insanlı tanker uçak ile insansız yakıt alan platform tarafında oluşturulan bir formasyon örneği olup dinamik bir yol planlaması çözümüne ihtiyaç duymaktadır. Bu kapsamında İHA sistemleri için HYİ görevinin gerçekleştirilmesinde kullanılabilecek nitelikte dinamik olarak değişkenlikler gösterebilen YPA tabanlı otonom HYİ yol planlamasının YPA ile modellenerek ortaya konulmasına değinilmiştir.

Bir diğer örnek senaryo ise birden fazla otonom İHA platformunu bir araya getirerek formasyon teşkil etmeyi, bu formasyonu bir seyrüsefer süresince muhafaza etmeyi, platformların birbirleri ile çarpışmasını engellemeyi ve formasyonun karşılaştığı engelleri değerlendirerek dinamik yol planlaması ihtiyacını ortaya koymaktadır.

5.1 Otonom Havada Yakıt İkmali Yol Planlaması

Havada kalış süresi gibi platform tabanlı yeteneklerin geliştirilmesi amaçlı farklı araştırma çalışmaları mevcuttur. Bu çalışmalardan biriside İHA sistemleri için HYİ çalışmasıdır. Amaçlanan minimum insan katkısı ile İHA platformunun görev etkinliğini arttıracak şekilde otonom HYİ görevini gerçekleştirmesidir. Esasında HYİ görevleri yaklaşımı İHA platformları için yeni ele alınan bir araştırma konusu değildir [26][27]. İlk yaklaşımlar simülasyon ortamı dışında yaklaşımlarını uygulama imkanı bulamamışlardır. Fakat geçen yıllarda, İHA teknolojilerindeki gelişmeler paralelinde bu yaklaşımlar uygulanabilir hale gelmiştir.

Bu alanda gerçekleştirilen çalışmaların başlıcaları arasında ABD Hava Kuvvetleri Araştırma Laboratuvarı ve Boeing tarafından 2007 yılındaki çalışma gelmektedir. Çalışma kapsamında Deniz ve Hava Kuvvetleri tarafından kullanılan HYİ teknikleri

ile tam otonom biçimde HYİ görevi Şekil 2.9 (b)'de gösterildiği şekilde gerçekleştirilmiştir. Birkaç yıl sonra, 2012'de, İHA sisteminin otonom şekilde tanker uçağı ile bulaşabildiğini göstermişlerdir [26]. İHA platformları insanlı hava araçlarına nazaran oldukça sınırlı faydalı yük taşıma kapasitesine sahip platformlardır. Dolayısıyla İHA platformları ile ilgili yürütülen çalışmaların başında havada kalış sürelerini arttırmaya yönelik araştırmalar gelmektedir. Bu amaç doğrultusunda gerçekleştirilen önemli yatırımlar mevcuttur. Bunlardan bir tanesi de ABD hükümeti tarafından desteklenen Otomatik Yakıt İkmali projesidir ve amacı İHA platformunun güvenli şekilde yakıt tanker uçağına yaklaşması ve boom ile bağlantıya geçerek ayrılmasının gerçekleştirilmesidir.

Bir diğer HYİ ile ilgili önemli çalışma 2012 yılında National Aeronautics and Space Administration (NASA) tarafından gerçekleştirilmiştir [27]. İki çalışma arasındaki en önemli fark, tanker uçak olarak da otonom Global Hawk İHA sistemlerinin Şekil 2.9 (a)'da gösterilen şekilde kullanılmasıdır. Dolayısıyla görevin karmaşıklık düzeyi bir seviye daha arttırılmış ve insan faktörü ve riski azaltılmıştır.

Temel olarak HYİ çalışmalarının amacı aşağıdaki şekilde listelenebilir;

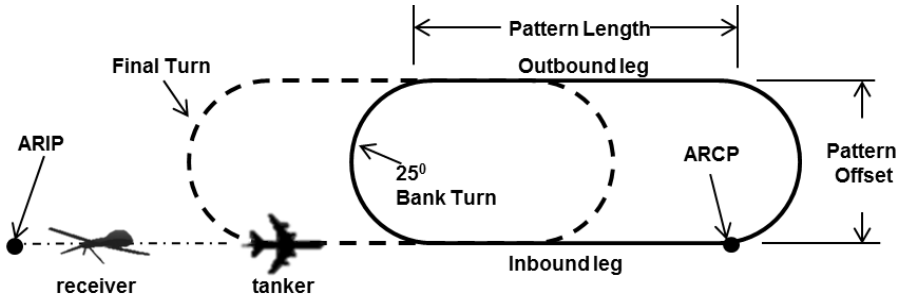
a. İHA platformlarına HYİ kabiliyeti kazandırılması ile görev yarıçapı ve de havada kalış süresi önemli ölçüde arttırılabilir,

b. Otonom sistemlerin kullanımı esnasında özellikle güvenli yaklaşma ve manevralar açısından HYİ görevlerinin başarısını önemli ölçüde etkileyen insan faktörü minimize edilmiş olacaktır,

c. Platformların havada kalış süreleri insan gereksinimlerinden bağımsız olarak arttırılmış olacaktır.

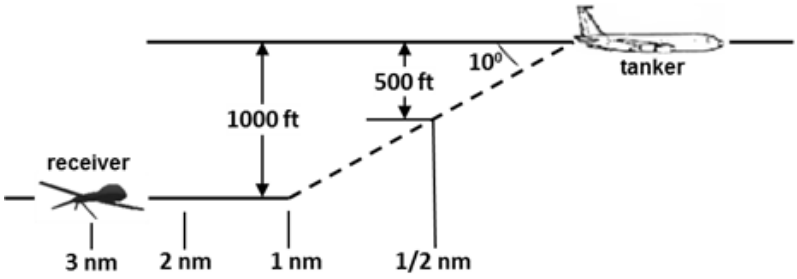
Bu çalışmalar göstermiştir ki başarılı bir HYİ görevini gerçekleştirmek için seyrüsefer algoritmaları ve kontrol yaklaşımları ve donanımları açısından sistem entegrasyonu, süreklilik ve tutarlılık önemli araştırma alanlarıdır. Tanker etrafında güvenli olarak manevra desteği sunan seyrüsefer algoritmaları ve otonom patern planlaması bu tez çalışması kapsamında ele alınacak olan başlıca konulardır.

Havada yakıt ikmali sürecinde kritik görevlerden birisi de görev planlamasıdır. Tanker ve de alıcı tüm yönleriyle görevi doğru planlamalı ve planın her aşamasına hakim olmalıdır.



Şekil 5.1: Havada yakıt ikmali patern ve görev planlaması.

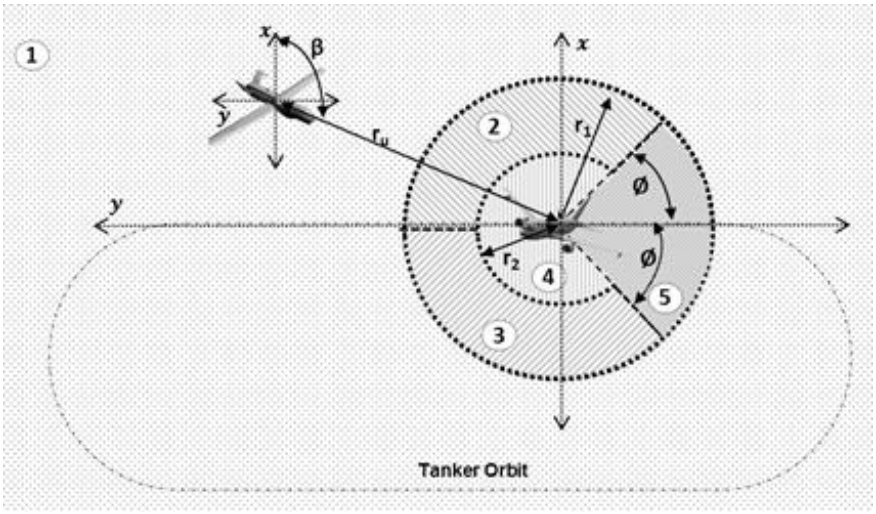
Şekil 5.1'de gösterildiği gibi, tanker ve alıcı havada yakıt ikmali kurallarına uymalı ve de önceden belirlenen havada yakıt ikmali başlangıç noktası (Air Refueling Initiation Point – ARIP), kontrol noktası (Air Refueling Control Point – ARCP), zaman kontrol noktası (Air Refueling Control Time – ARCT) ve bitiş noktası (End of Air Refueling Point - EA/R) noktası gibi referanslara koşulsuz riayet etmelidir. Temel HYİ paterni incelendiğinde, alıcı önceden belirlenmiş elips şeklinde hareket eden tankeri takip etmeli ve de yaklaşmalı şeklinde özetlenebilir. Bu yaklaşım oldukça zamana bağlı ve yüksek koordinasyon gerektirmektedir.



Şekil 5.2: Havada yakıt ikmali dikey yaklaşma planlaması.

Bir diğer kritik husus da dikey yaklaşmadır. Şekil 5.2'de gösterildiği gibi, alıcı tanker ile dikey ayırmayı (yaklaşık 1000 ft.) yatay konumunu sağlayana kadar muhafaza eder. Yatay konum elde edildikten sonra göz temasını kaybetmeyecek şekilde dikey yaklaşmaya başlar.

Birinci bölümde açıklandığı şekilde, insanlı sistemler için tasarlanmış olan HYİ paternlerini, başarıyla uygulama yeteneğine sahip nitelikte İHA platformları geliştirilmiştir. Bunun ötesinde insanlı HYİ görevlerinin gerçekleştirilmesi zamanlama ve yüksek koordinasyon gibi bir takım problemleri beraberlerinde getirmektedir.



Şekil 5.3: İHA platformları için havada yakıt ikmal paterni.

Özellikle yatay yaklaşma açısından İHA sistemleri için HYİ görev planlaması daha basit şekilde ele alınabilir. Yatay konumun alınması esnasında öne çıkan etkenler, hassas konum bilgileri ve de alıcının dönüş yarıçapıdır. Genellikle tanker Şekil 5.3'de gösterildiği biçimde sabit eliptik bir patern üzerinde uçar. İHA platformu alıcı olarak "*1*" numara olarak temsil edilen ve Yakıt ikmali Bölgesi olarak adlandırılan bölgeye her hangi bir noktadan giriş yapar. Bu bölge balistik bir bölge şeklide bir görev üstlenir ve alıcının tankere doğru yönlendirilmesini amaçlar. Alıcı, tankerin uçuş başının aksi istikametinden alıcı ve tankerin süratlerine bağlı olarak sınırlandırılmış dinamik olarak belirlenen bir açıdan (2x∅) tankere yaklaşmalıdır. Bu Yaklaşma Bölgesi olarak adlandırılır ve Şekil 5.3'de 5 numara olarak gösterilmiştir. İHA platformunu Yaklaşma Bölgesi içine getirebilmek için Geçiş Alt Bölgesi adı verilen Şekil 5.3'de "*2*" ve "*3*" numaralar ile gösterilen iki farklı bölgeden yararlanılır. Yaklaşma Bölgesi'ne erişmek için "*2*" numara ile işaretlenmiş olan Geçiş Alt Bölgesi saat yönünde, "*3*" numara ile işaretlenen ise saat yönünün tersine bir hareket sağlar. Geçiş Alt Bölgesi, göreceli sürate ve alıcının minimum süratine bağlı olarak belirlenen r_1 parametresi ile sınırlandırılmıştır. Son bölge olan ve Şekil 5.3'de "*4*" numara ile belirtilen bölge Ayırma Bölgesi'dir. Bu bölge tanker ve alıcının çarpışmasını engellemek amacıyla güvenlik için ve alıcının minimum dönüş yarıçapına istinaden Yaklaşma Bölgesi içinde tankere yönlendirilebilmesi için oluşturulmuştur. Son yaklaşma evresinde İHA ve tanker aynı uçuş (β) başında uçmalıdır. Gereksinim duyulan uçuş başına erişebilmek için ve bu başı elde etmek için gerekli baş değişikliğini elde edebilme imkanı sunması açısından Yaklaşma Bölgesi İHA'nın o anki süratindeki minimum dönüş açısına bağlı olarak, Şekil 5.3 ile

gösterilen r_2 parametresi ile alan "*4*" içinde dinamik olarak limitlenmektedir. İHA yatayda uygun pozisyonu elde ettikten sonra dikey olarak yaklaşımını insanlı sistemlerdekine benzer bir planlama ile gerçekleştirir.

Başarılı ve güvenli olarak tanker uçak etrafında manevra yapabilmek için bir takım koşullar mevcuttur;

a. HYİ görevi süresince tanker ve alıcının (İHA) kesin konumları birbirleri tarafından bilinmelidir,

b. İHA'nın anlık hava sürati ve de buna bağlı olarak minimum dönüş açısı sürekli olarak anlık hesaplanmalıdır,

c. Tanker görev süresince önceden tanımlı olan paterni birebir uygulamalıdır.

HYİ görevinin planlanması esnasında önemli bir nokta yaklaşımın dinamik bir yapıda oluşudur. Dinamik olma özelliği ile yaklaşım esnek ve kullanılabilir olmaktadır. Fakat bu durum aynı zamanda problemin karmaşıklık düzeyini arttırmakta ve çözüm daha fazla işlemci gücüne ihtiyaç duymaktadır. Bu noktada üç farklı planlama yaklaşımı oluşmaktadır;

a. Tanker tarafından yönetilen yaklaşım: Bu durumda tanımlanan tüm otonom yapı, hem tanker hem de alıcı için tanker tarafından ele alınır ve hesaplanır, gerekli komutlar alıcı İHA'ya gönderilir.

b. Alıcı tarafından yönetilen yaklaşım: Bu yaklaşım esnasında tanker önceden tanımlanan eliptik rotayı izlemek dışında hiç bir otonom işleme karışmaz ve tüm planlama alıcı tarafından gerçekleştirilir ve uygulanır.

c. Hibrit yaklaşım: Tanker otonom HYİ görevi için gerekli olan tüm otonom yapıyı kendisi ve de alıcı için hesaplar ve alıcıya gerekli komutları gönderir. Alıcı gönderilen komutlar arasından doğru manevraya pozisyon, sürat gibi parametrelerine bağlı olarak karar verir.

Tanker tarafından yürütülen yaklaşım birden fazla alıcının yakıt ikmali noktasına yaklaşması durumunda etkindir. Her bir alıcı için süreç aynıdır ve dolayısıyla aynı hesaplamanın her bir alıcı için tekrarlanmasına gerek yoktur. Ancak alan içerisinde yer alan birden fazla alıcı platformun birbirleri ile çarpışmasını önlemek amacıyla platformların birbirlerini dinamik birer engel olarak görmelerine imkan sağlayacak

şekilde potansiyel alan güncellenmelidir. Bu durumda her platform için alan ayrı ayrı hesaplanmak zorunda kalacaktır.

Bu yaklaşım doğru zaman aralığında tankerin her bir alıcının kesin pozisyon bilgisini bilmeyi gerektirmektedir ve bunu uygulamak zordur. Özellikle HYİ gibi gerçek zamanlı görevlerde iletişim ayrı bir darboğaz oluşturmaktadır.

Tanımlanan tüm potansiyel alanlar önceki bölümlerde tanımlanmış olan temel potansiyel alanlardan oluşturulmuştur. Açıklandığı şekilde bir arada kullanılabilmeleri için sınır fonksiyonların tanımlanması ve birleştirilmiş toplam potansiyel alanın üretilebilmesi gerekmektedir.

İki boyutlu koordinat sisteminde İHA'nın pozisyonu $q_u = x_u, y_u$ ve tankerin pozisyonu $q_t = x_t\ , y_t$ olarak kabul edilebilir. Geçiş Alt Bölgesi, S_1 parametresi ile ve Ayırma Bölgesi S_2 parametresi ile sınırlandırılmıştır. Bu alanlar tam bir daire formunda kabul edilebilecekleri gibi γ parametresi ile eliptik formlarda da temsil edilebilirler. HYİ planlamasında bahsedildiği üzere tanker eliptik bir rota üzerinde uçar ve bu rotanın merkezi $q_c = x_{tc}\ , y_{tc}$olarak temsil edilebilir. Elipsin uzun ve de kısa eksenleri ise $ax_{maj}\ , ax_{min}$ olarak tanımlanabilir. ω parametresi tankerin uçtuğu eliptik rotanın kısa eksenin koordinat sisteminin *y*-ekseni ile yaptığı açıyı göstermek için kullanılır. Tankerin rotası üzerinde bulunduğu nokta ile merkez noktası arasındaki açı θ açısı olarak belirtilmiştir. Belirtilen notasyondan faydalanılarak tankerin zamana bağlı konumu Denklem (5.1)'deki şekilde ifade edilebilir.

$$\begin{aligned} & q_t(t) \\ & = [\ (x_t(t-1) + (\ ax_{maj}.\cos\omega\ .\cos\theta \\ & - ax_{min}.\sin\omega\ .\sin\theta))\ , (\ y_t(t-1) \\ & + (ax_{maj}.\cos\omega\ .\sin\theta \\ & + ax_{min}.\sin\omega\ .\cos\theta\ ))] \end{aligned} \qquad \textbf{(5.1)}$$

Belirtilen notasyon çerçevesinde tankerin anlık uçuş başı (h_t) Denklem (5.2)'de gösterildiği biçimde, tanker ve İHA arasındaki mesafe (r_u) ise Denklem (5.3) ile gösterildiği şekilde hesaplanabilir.

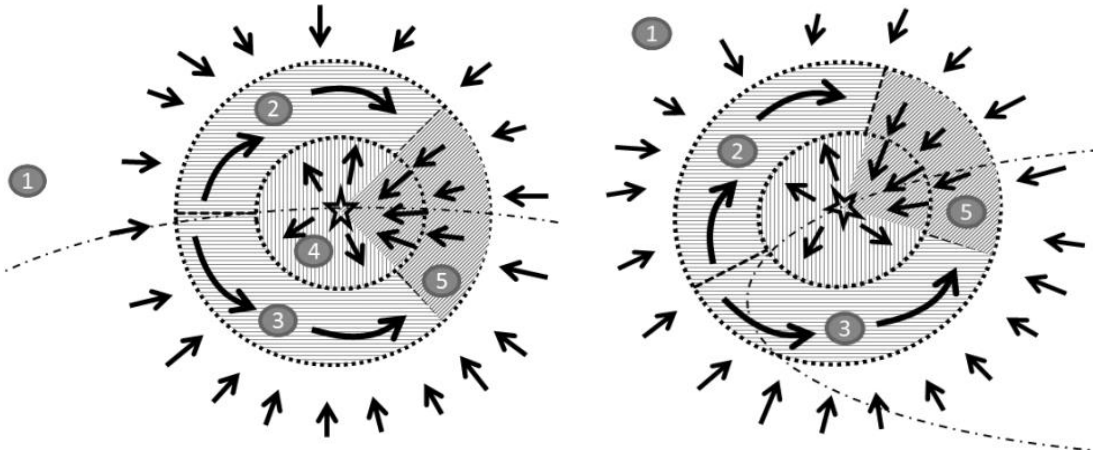
$$h_t = mod(90 + atan2(q_t), 360) \qquad \textbf{(5.2)}$$

$$r_u = \sqrt{(x_u - x_t)^2 + \gamma(y_u - y_t)^2} \qquad \textbf{(5.3)}$$

İHA platformu alıcı rolünde tanker platforma Yaklaşma Bölgesi olarak tanımlanan ve (τ) açısı ile temsil edilen bir aralıktan yaklaşmalıdır. Ayrıca bu aralık dinamik olarak τ_s ve τ_f açısal değerleri ile Denklem (5.4)'de gösterildiği şekilde hesaplanarak sınırlandırılmıştır. Ayrıca tanker ve de İHA arasındaki açı da (ε) iki boyutlu koordinat sisteminde Denklem (5.5) ile gösterildiği şekilde hesaplanır.

$$\tau_s = mod(h_t + 180 - \tau, 360)$$
$$\tau_f = mod(h_t + 180 + \tau, 360) \quad \textbf{(5.4)}$$

$$\varepsilon = atan2(y_t - y_u, x_t - x_u) \quad \textbf{(5.5)}$$



(a) Birleştirilmiş Potansiyel Alan (t_0). (b) Birleştirilmiş Potansiyel Alan (t_n).

Şekil 5.4: Havada yakıt ikmali potansiyel alan tabanlı yol planlaması modeli.

Potansiyel alanlar vasıtasıyla Şekil 5.3'dekine benzer bir patern üretmek için, en azından Şekil 3.10 (a) ile gösterilen noktasal çekim, Şekil 3.10 (c) ile gösterilen noktasal iten ve dairesel döndüren (Şekil 3.11 (a) ile gösterilen saat yönüne ve Şekil 3.11 (b) ile gösterilen saat yönü tersine) gibi 3 temel model kullanılmalıdır. Bu alanların doğru olarak sınırlandırılmış bileşkesi Şekil 5.3'dekine benzer şekilde davranabilir. Bir önceki bölümde tanımlanmış olan her bir alan, potansiyel alanlar ile temsil edilmelidir. Örneğin, Yaklaşma Bölgesi bir sınırlı noktasal çeken alan şeklinde, Geçiş Alt Bölgesi sınırlı dairesel döndüren alan şeklinde ve Ayırma Bölgesi ise sınırlı noktasal iten alan şeklinde modellenebilir. Tüm bu alan sınırlamaları ve de tanımlamaları dinamik şekilde gerçekleştirilmedir, çünkü tanker uçak belirtilen şekilde görev süresince eliptik bir yörünge üzerinde hareket etmektedir ve bu alanların durumunu değiştirir. Örneğin t_0 anı için alanın durumu Şekil 5.4 (a)'daki gibi iken t_n anı için Şekil 5.4 (b)'de gösterilen duruma gelir.

HYİ görevinin gerçekleştirilmesi için 5 farklı alan tanımlaması yapılmıştır. Alıcı bir anda bu beş bölgeden yalnızca birisinde bulunabilir. Bu bölümün devamında her bir bölge için bir üyelik fonksiyonu (S) tanımlaması yapılacak olup, bu fonkisyon İHA'nın bu bölge (S_x, x bölge numarasını gösterir) içinde olup olmadığını ortaya koymak amacıyla kullanılacaktır [0||1]. Yakıt ikmali Bölgesi sınırlı kare şeklinde tanımlanmış bir bölge olup, x_{s1}, x_{s2} ve y_{s1}, y_{s2} koordinatları arasında yer almaktadır. Yakıt ikmali Bölgesi için üyelik fonksiyonu (S_1) Denklem (5.6) ile gösterildiği şekilde ortaya konulabilir.

$$S_1 = (x_{s1} \geq x_u \geq x_{s2}) \wedge (r_u \geq r_1) \wedge (y_{s1} \geq y_u \geq y_{s2}) \quad \textbf{(5.6)}$$

Yaklaşma Bölgesi için üyelik fonksiyonu (α_5) üst (α_{5_s}) ve alt (α_{5f}) şeklinde Denklem (5.7)'de gösterildiği şekilde sınırlandırılmıştır. Denklem (5.8), tanker uçuş başı ile yaklaşma aralığı arasındaki açısal ilişkiyi göstermektedir (rot_5). Yaklaşma Bölgesi üyelik fonksiyonun (α_5) son şekli Denklem (5.9) ile ifade edildiği şekildedir.

$$\begin{aligned} S_{5_s} &= (\varepsilon \geq \tau_s) \\ S_{5_f} &= (\varepsilon \leq \tau_f) \end{aligned} \quad \textbf{(5.7)}$$

$$rot_5 = \big(mod(180 - \tau, 360)\big).\big(h_t < mod(180 + \tau, 360)\big) \quad \textbf{(5.8)}$$

$$S_5 = (1 - S_1).\Big(\Big(rot_5.\big(S_{5_s} + S_{5_f}\big)\Big) + \Big((1 - rot_5).\big(S_{5_s}.S_{5_f}\big)\Big)\Big) \quad \textbf{(5.9)}$$

Ayırma Bölgesi üyelik fonksiyonu (S_4) Denklem (5.10) ile gösterilmiştir.

$$S_4 = (1 - S_5)(r_u \leq r_2) \quad \textbf{(5.10)}$$

(rot_2) ve (rot_3) notasyonları Geçiş Alt Bölgesi 2 ve 3 için sınır değerlerini temsil etmek ile birlikte doğrudan tankerin uçuş başına bağlı olarak (h_t) Denklem (5.11) ile gösterildiği şekilde hesaplanırlar.

$$\begin{aligned} rot_2 &= (1 - rot_5).(h_t > mod(180 - \tau, 360)) \\ rot_3 &= (1 - rot_5).(h_t > mod(180 + \tau, 360)) \end{aligned} \quad \textbf{(5.11)}$$

Saat istikametine olan Geçiş Alt Bölgesi Denklem (5.12) ile gösterilen parametreler ile saat yönünün tersine olan Geçiş Alt Bölgesi ise Denklem (5.13) ile gösterilen parametreler ile sınırlandırılmıştır.

$$S_{2_s} = (\varepsilon \leq h_t)$$
$$S_{2_f} = (\varepsilon \geq \tau_s) \quad \textbf{(5.12)}$$

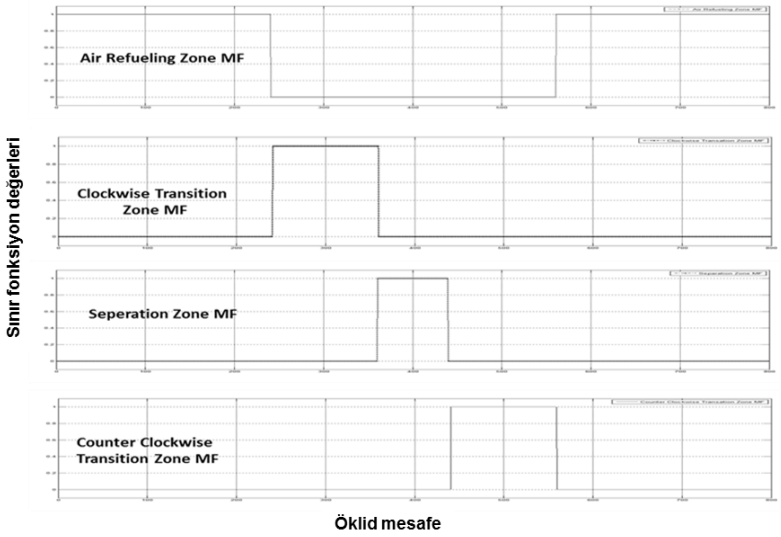
$$S_{3_s} = (\varepsilon \geq \tau_s)$$
$$S_{3_f} = (\varepsilon \leq \tau_f) \quad \textbf{(5.13)}$$

HYİ görev planlamasında tanımlanan 2 no.lu Geçiş Alt Bölgesi üyelik fonksiyonu Denklem (5.14) ile gösterilmiştir. Önceki bölümlerde planlama kapsamında yer alan iki Geçiş Alt Bölgesi arasındaki fark dönüş yönü olarak belirtilmiştir. Bir diğer fark da sınır değerlerinin belirlenmesinde ortaya çıkmaktadır. Saat yönünün tersine olarak tanımlamış Geçiş Alt Bölgesi üyelik fonksiyonu Denklem (5.15)'de gösterildiği şekilde ortaya çıkmaktadır.

$$S_2 = (1 - S_1).(1 - S_4).(1 - S_5).\Big(\Big(rot_3.\Big(S_{3_s} + S_{3_f}\Big)\Big) + \Big(rot_2.\Big(S_{3_s}.S_{3_f}\Big)\Big) + \Big(rot_5.\Big(S_{3_s}.S_{3_f}\Big)\Big)\Big) \quad \textbf{(5.14)}$$

$$S_3 = (1 - S_1)(1 - S_4)(1 - S_5).\Big(\Big(rot_3.\Big(S_{3_s}.S_{3_f}\Big)\Big) + \Big(rot_2.\Big(S_{3_s}.S_{3_f}\Big)\Big) + \Big(rot_5.\Big(S_{3_s} + S_{3_f}\Big)\Big)\Big) \quad \textbf{(5.15)}$$

İHA platformları için ortaya konulan yaklaşımın doğru nitelikte HYİ paterni üretebilmesi için her bir potansiyel alan tabanlı alt bölgenin doğru şekilde bir sınır fonksiyonu [9] ile sınırlandırılmasına ihtiyaç vardır.

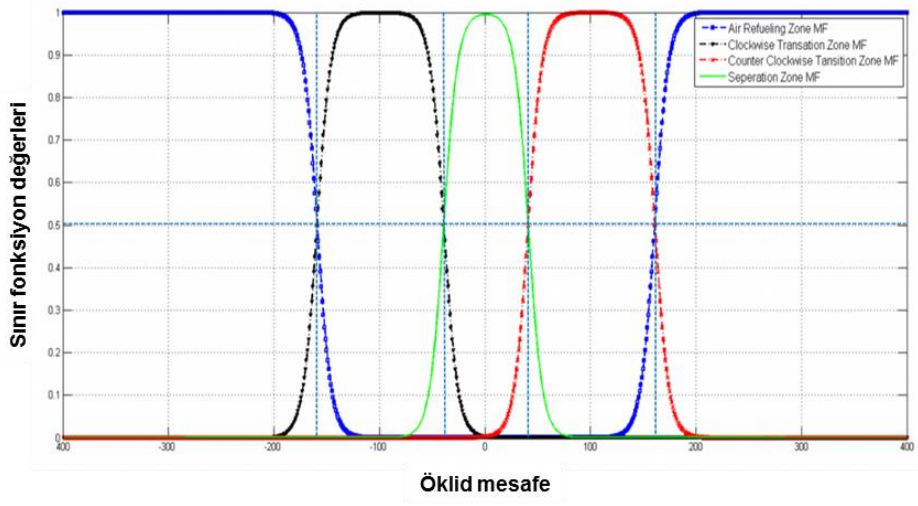


Şekil 5.5: İkili (Binary) değerler üreten nitelikte mantıksal denklemlerden üretilmiş sınır fonksiyon çıktıları.

Şekil 5.5'da gösterilen şekilde ikili (binary) değerler üreten nitelikte mantıksal denklemlerden sınır fonksiyonları olarak faydalanılmıştır [1]. Bu yaklaşım uygulanması kolay ve hızlı hesaplanabilir bir çözümdür. Ancak aynı zamanda bu yöntem ile üretilen yol planlamaları İHA platformları için uygulanması zor, keskin dönüş komutları gerektiren sonuçlar üretmiştir.

Alt bileşenleri oluşturan potansiyel alanları dinamik olarak sınırlandırabilmek için her alt alana ait sınır fonksiyonları (ε) açısına ve tanker ile alıcı platform arasındaki (r_u) mesafeye bağlı olarak değişkenlik göstermelidir. Her bir ızgara noktasındaki toplam sınır fonksiyon değerleri "*1*" olmalıdır. Sınır değer fonksiyonları S_i notasyonu ile temsil edilmek ile birlikte *i* parametresi sınır fonksiyonun Şekil 5.3'de gösterilen hangi alt alana ait olduğunu temsil etmektedir ve ürettiği değerler "*0-1*" aralığında değerlerdir.

Bir diğer sınırlandırma fonksiyonu ise daha önceki bölümler kapsamında açıklandığı biçimde "sigmoid" fonksiyonların kullanımıdır. Bu kapsamda sigmoid fonksiyonların "*Slope*" değeri tüm sınır fonksiyonları için aynı ve sabit bir değer olarak belirlenebilir. Örneğin belirli bir çözünürlük değerine sahip alan için her bir bölgede yaklaşık 50 piksel değerinde bir alanı etkileyecek şekilde sabit bir değer olarak seçilebilir. Sınır fonksiyonlarının tamamı Şekil 5.6'de de gösterildiği şekilde sigmoid fonksiyonlar kullanılarak da hesaplanabilir. Ancak Yaklaşma ve Geçiş Bölgeleri için sigmoid fonksiyonlar kullanarak limit fonksiyon değerleri hesaplanmasına gerek yoktur, bu alanlar daha az hesaplama gücü ihtiyacı duyan ve dolayısıyla daha hızlı hesaplanabilen ikili fonksiyonlar kullanılarak sınırlandırılabilir.



Şekil 5.6: Sigmoid fonksiyon tabanlı tüm alt alanlar için sınır fonksiyonları.

Şekil 5.6'de de görüldüğü üzere, alan dahilinde yer alan her bir noktadaki toplam sınır değer fonksiyonlarının toplam değeri "*1*" dir.

Yakıt İkmali alt bölgesinde yer alan her bir (x, y) noktası için sınır değer fonksiyonu Denklem (5.16)'da yer alan eşitlik ile hesaplanabilir.

$$S_1 = \left(\frac{1}{1 + e^{-Slope*(r_u + r_1)}}\right) * \left(1 - \left(\frac{1}{1 + e^{-Slope*(r_u + r_2)}}\right)\right) \quad \textbf{(5.16)}$$

Geçiş Alt Bölgeleri (2 ve 3) ve Yaklaşma Bölgesi için sırasıyla Denklem (5.9) ve Denklem (5.14) ile (5.15) ile tanımlanan ikili fonksiyonlardan faydalanılarak sınırlandırma yapmak mümkündür. Sigmoid fonksiyonlar yerine ikili fonksiyonların kullanılmasının iki temel nedeni vardır; ilki bu bölgeler arasındaki geçişlere platformun keskin tepkiler üretmesinin istenmesi, ikincisi ise ikili fonksiyonların hesaplanmasının sigmoid fonksiyonlara nazaran dahi iyi hesaplama performansı göstermesidir.

Bir diğer sınır fonksiyonu olan (S_4) değeri Ayırma Bölgesi için kullanılan fonksiyondur. Denklem (5.17)'den de görüldüğü üzere eşitlik sigmoid bir sınır fonksiyonudur.

$$S_4 = (1 - S_5)\left(\left(\frac{1}{1 + e^{-Slope*(r_u + r_2)}}\right) * \left(1 - \left(\frac{1}{1 + e^{-Slope*(r_u - r_2)}}\right)\right)\right) \quad \textbf{(5.17)}$$

Şu ana kadar bölgelere ait üyelik (sınır) fonksiyonlarının tanımlaması yapılmıştır. HYİ görevi kapsamında tanımlanan her bir bölge S_x fonksiyonu ile sınırlandırılmıştır. Bu tanımlamalar doğrudan potansiyel alanların oluşturulmasında ve toplam potansiyel alan değerlerinin hesaplanmasında kullanılacaktır. Bundan sonraki kısımda ise her bir bölge için sınırlı potansiyel alan değerlerinin nasıl oluşturularak hesaplandığına değinilecektir.

Daha önceki bölümlerde açıklandığı gibi yerel minimum durumundan sakınmak için potansiyel alanların tanımlanmasında harmonik fonksiyonlardan faydalanılmıştır [24]. HYİ kapsamında tanımlanan her bir p ile tanımlanan bölgenin (δ_p) olarak temsil edilen kuvvet çarpanı tanımlaması olacaktır. Şayet tankerin elips şeklinde bir yörüngede uçtuğu kabul edildiğinde ve tanker ile alıcı arasındaki mesafenin Denklem (5.3)'te ifade edildiği şekilde (r_u) olarak hesaplanması durumunda alan içerisindeki her bir noktanın potansiyel değeri Denklem (5.18) ile gösterildiği biçimde

hesaplanabilir. (q_u) noktasındaki potansiyel değer $F_p(q_u)$ notasyonu ile ifade edilmektedir.

$$F_p(q_u) = \log\left(\delta_p\left[\sqrt{(x_u - x_t)^2 + \gamma(y_u - y_t)^2}\right]\right) = \log\left(\delta_p\, r_u\right) \quad \textbf{(5.18)}$$

Denklem (5.19)'da gösterildiği şekilde tankerin uçuş başının etkisi denkleme dahil edildiğinde potansiyel alan değerinin hesaplanması için eşitlik Denklem (5.20) ile ifade edilen formata dönüşecektir.

$$\begin{aligned} Y_{rot} &= \sin(h_t)(x_u - x_t) + cos(h_t)(y_u - y_t) \\ X_{rot} &= \cos\,(h_t)(x_u - x_t) - sin(h_t)(y_u - y_t) \end{aligned} \quad \textbf{(5.19)}$$

$$r_{rot} = \sqrt{(X_{rot})^2 + \gamma(Y_{rot})^2} \quad \textbf{(5.20)}$$

HYİ bünyesinde tanımlanan her bir bileşke alan için potansiyel kuvvet ayrı ayrı hesaplanabilir. Yakıt ikmali bölgesi için her bir (x, y) noktasındaki potansiyel alan kuvveti değeri $F_1(x, y)$, Denklem (5.21) ile ifade edildiği şekilde hesaplanabilir. Eşitlik $F_1(x, y)$ çeken balistik etkiyi modelleyebilmek için negatif değerler üretecektir.

$$F_1(x, y) = -\log(\delta_1[r_{rot}]) \quad \textbf{(5.21)}$$

HYİ için gerek duyulan ikinci alan tanımlaması saat yönündeki Geçiş Alt Bölgesi olarak adlandırılan alan olup, Denklem (5.22) ile ifade edildiği şekilde dönen yapıda bir alan tanımlamasıdır. Bir diğer HYİ alan tanımlaması da saat yönünün tersine dönen şeklinde ifade edilen Geçiş Alt Bölgesi tanımlaması olup Denklem (5.23) ile tanımlanmıştır.

$$F_2(x, y) = 1 - 2\,(\log(\delta_2(r_u)^2)\,) \quad \textbf{(5.22)}$$

$$F_3(x, y) = -(1 - 2\,(\log(\delta_3(r_u)^2)\,)) \quad \textbf{(5.23)}$$

HYİ kapsamında tanımlanan iten kuvvet vektörlerinden oluşan dördüncü alan Ayırma Bölgesi olarak adlandırılmıştır. Ayırma Bölgesi içinde yer alan her bir noktanın potansiyel alanı Denklem (5.24) ile gösterildiği şekilde hesaplanabilir.

$$F_4(x, y) = \log(k_4[r_{rot}]) \quad \textbf{(5.24)}$$

HYİ kapsamında ele alınan son alan tanımlaması olan Yaklaşma Bölgesi için potansiyel alan değerleri Denklem (5.25) ile gösterilen şekilde hesaplanabilir. Yaklaşma Bölgesi bir tür çeken yapıdaki potansiyel alan tanımlamasıdır.

$$F_5(x,y) = -\log(\delta_5[r_{rot}]) \quad \textbf{(5.25)}$$

Vektör hesaplamasında, vektör potansiyeli olarak tanımlanmış bir vektör alan, vektör alanın bir eğimidir. Sayısal potansiyel için bu durum verilen vektör alanın gradienti olarak ifade edilir. Potansiyel bir değerin gradienti vektör bir alan üretmek için kullanılabilir. Formal olarak ($\vec{F}$) ile tanımlanan bir vektör alan için vektörel potansiyel (*F*) Denklem (5.26)'dan faydalanılarak üretilebilir.

$$\vec{F} = \nabla F \quad \textbf{(5.26)}$$

Temel işlem gradient fonksiyonudur. Potansiyel kuvvet değerlerinden vektör alanın üretilebilmesi için Denklem (5.27)'ten faydalanılmıştır

$$\vec{F}_t(x,y) = \nabla F_t(x,y) \quad \textbf{(5.27)}$$

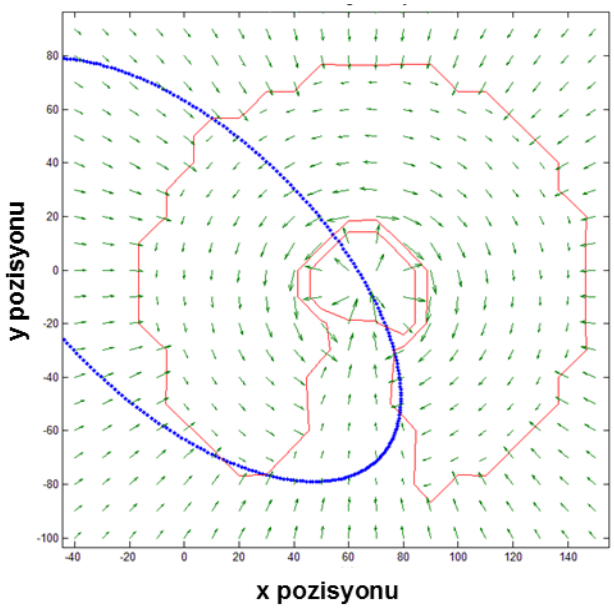
HYİ görevi için tanımlanmış vektör alanın üretilmesi için üretilen potansiyel alanların toplamı gereklidir. HYİ alanı içerisindeki her bir nokta bir potansiyel alandan etkilenmektedir. Bu sınırlı tanımlama alanların limitli üyelik fonksiyonlarından faydalanılarak türetilebilir. (q_u) noktasındaki vektörel kuveet $\vec{F}_t(q_u)$ notasyonu ile ifade edilir ve de Denklem (5.28) ile hesaplanmaktadır.

$$\vec{F}_t(x,y) = \sum_{j=1}^{5} S_j \nabla F_j(x,y) \quad \textbf{(5.28)}$$

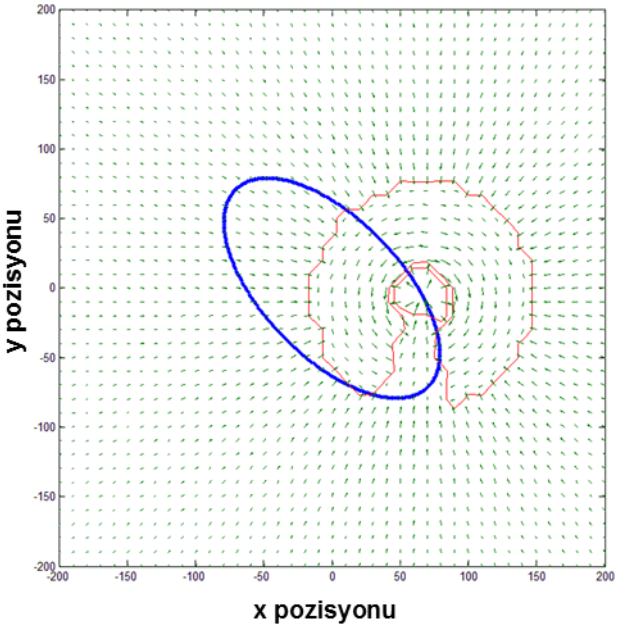
Vektör alan içerisinde alıcının izleyeceği patern (U) Denklem (5.28) ile üretilebilir. Alıcının başlangıç pozisyonu $\mathrm{q_u}(0)$ olarak ifade edilmiştir.

$$U = \int_{q_u(0)}^{q_t} \vec{F}_t . dl \quad \textbf{(5.29)}$$

Toplam vektör alanı Şekil 5.8'de gösterilmektedir. (t_n) anındaki HYİ görevi için ikili fonksiyonlar ile sınırlandırılmış YPA tabanlı yerel yol planlaması için vektör alan gösterimi Şekil 5.8 (a)'da, genel vektör alan gösterimi ise Şekil 5.8 (b)'de gösterildiği şekilde üretilir.



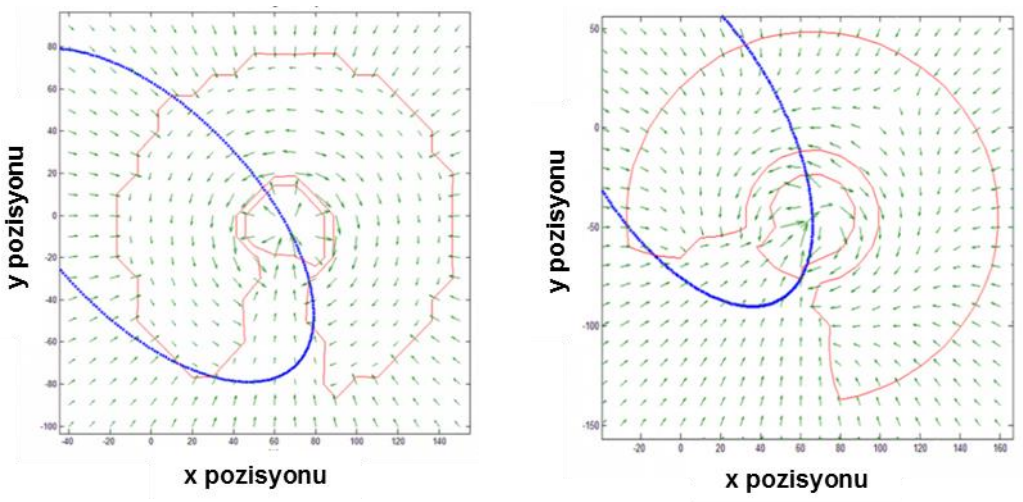
(a) Yerel potansiyel alan tanımlaması (t_n).



(b) HYİ için genel vektör alan tanımlaması (t_n).

Şekil 5.7: HYİ patern planlaması için vektör alan.

HYİ bölgesi içindeki her bir nokta üzerinde belirlenen her bir alt bölgenin etkisinin toplamı vardır. Bu durum özellikle sınır fonksiyonları olarak alt bölgelerin sınırlandırılmasında ikili fonksiyonlar yerine sigmoid fonksiyonlar kullanılmasının bir sonucudur.

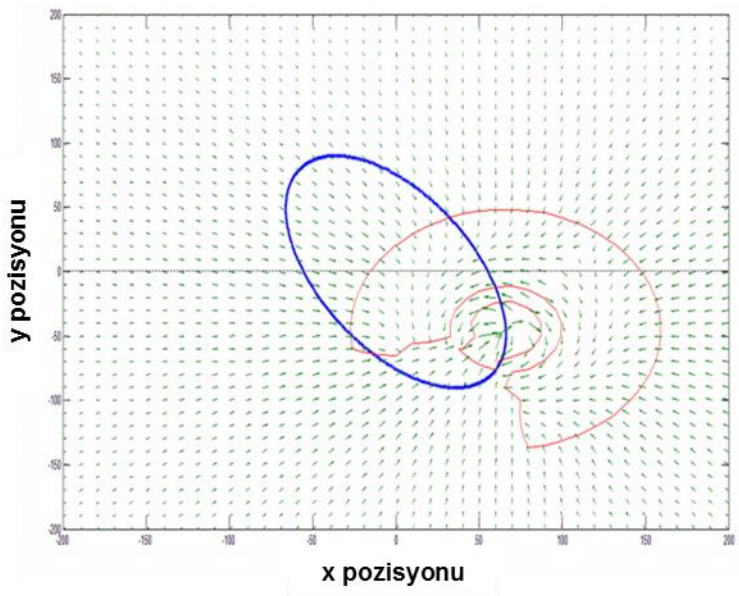


(a) İkili sınır fonksiyonları kullanarak elde edilen vektörel alan gösterimi.

(b) Sigmoid fonksiyonlar kullanarak elde edilen vektörel alan gösterimi.

Şekil 5.8: HYİ görevi için otonom yol planlaması yaklaşımında sınır fonksiyonların kullanımının karşılaştırılması.

Şekil 5.8 ile sigmoid fonksiyonlar kullanılması ile ikili fonksiyonlar kullanılmasının farkları ortaya konulmuştur. Şekil 5.8 (b)'den de görüldüğü üzere sigmoid fonksiyonlar kullanılması neticesinde daha yumuşak ve İHA platformları tarafından daha kolay uygulanabilir platformun hareketini belirleyen komutlar üretilmiştir. Özellikle alt bölgelerin geçişleri esnasında bu fark daha iyi görülmektedir.



Şekil 5.9: Sigmoid fonksiyonların sınır fonksiyonları olarak kullanıldığı YPA tabanlı genel yol planlaması gösterimi.

Sigmoid fonksiyonların sınır fonksiyonlar olarak kullanıldığı potansiyel alan tabanlı yol planlaması için ortaya konulan vektörel alanın gösterimi Şekil 5.9'da gösterildiği biçimdedir. Şekilden de görüldüğü üzere özellikle alt bölge geçişleri İHA

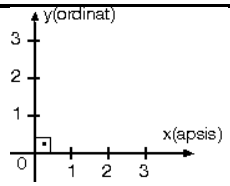
platformlarının alıcı olarak daha kolay uygulayabilecekleri nitelikte sonuçlar üretmiştir.

5.2 Otonom Formasyon Uçuşu Yol Planlaması

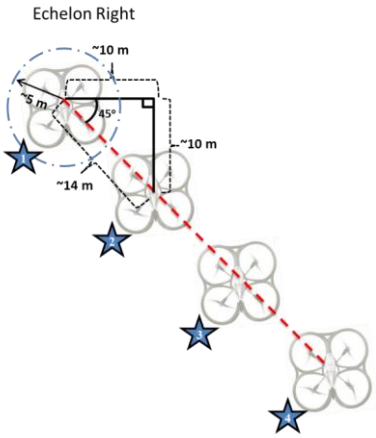
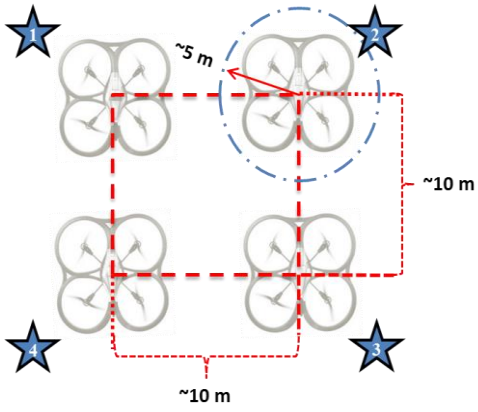
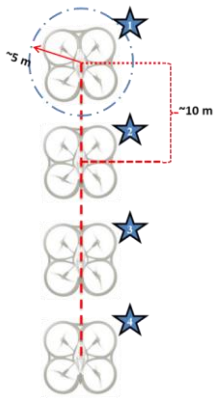
Formasyon uçuşunda amaç, bir arada bulunan platformların lider platforma göre belirlenen konumda seyrüsefer süresince kalmalarını sağlamaktır. Bu nedenle formasyon kontrolünün matematiksel olarak modellenmesinden önce farklı formasyon uçuş yapıları için sanal olmayan lider düzenine göre lider dışındaki platformların bulunması gereken konumlarının belirlenmesi gerekmektedir. Bu tez kapsamında uygulanacak olan formasyon düzenleri ve de formasyon dahilindeki platformların konum planlaması Çizelge 5.1 ile gösterilmiştir.

Potansiyel alan içerisinde yer alan her İHA platformu için diğer platformlar birer iten nitelikte engel potansiyel alt alan niteliğindedir. Bu yaklaşım platformların birbirleri ile çarpışmasını önlemek için uygulanacaktır. Çarpışmayı önleyici güvenlik alanı Çizelge 5.1'deki örnek değerler için 5 m. olarak belirlenmiş ve dolayısıyla bu alt potansiyel bölgenin sınır fonksiyonu bu değer göz önünde bulundurularak değerlendirilecektir. Ayrıca her platform için formasyon düzeninde olmasını istediğimiz lidere göre konumu hedef oluşturacak şekilde çeken nitelikte bir potansiyel alan niteliğindedir. Hedef alt alanın etkisi çarpışma önleme güvenlik alanı aksine sınırsızdır. Ancak platformun kendisi dışındaki platformların çarpışmayı önleme alanları içinde etkisiz kabul edilecektir.

Çizelge 5.1: Otonom formasyon kontrolü konumlandırma yaklaşımı.



Alanın 500m.x500m. (size=500 px) ve Çözünürlük değerinin 1 px (res=1) seçildiği kabul edilmiştir.

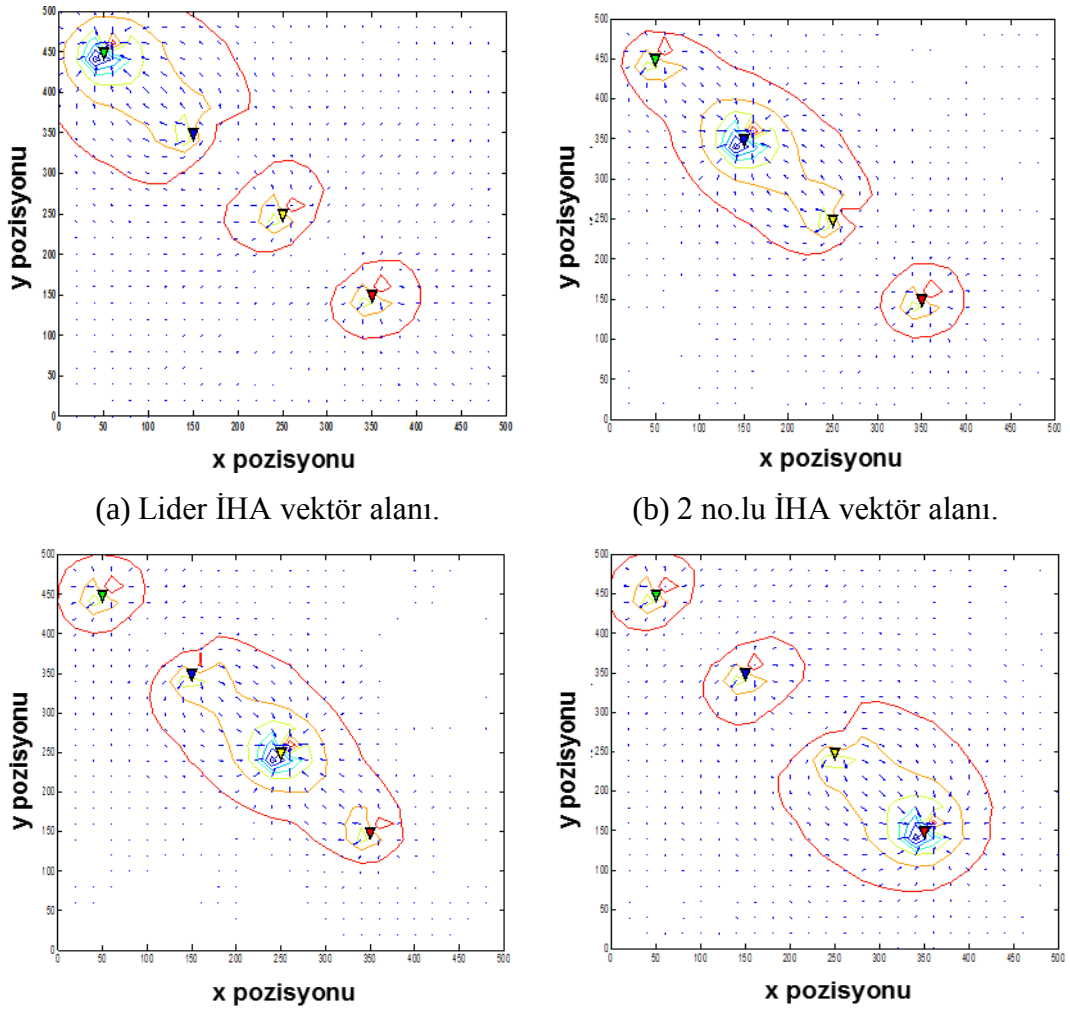
Formasyon	İHA Konumları	Tanım
 **Echelon Right**	***İHA₁***: x_1, y_1 ***İHA₂***: $x_2 = x_1 + \left(\frac{size}{10 * res}\right)$ $y_2 = y_1 - \left(\frac{size}{10 * res}\right)$ ***İHA₃***: $x_3 = x_1 + 2 * \left(\frac{size}{10 * res}\right)$ $y_3 = y_1 - 2 * \left(\frac{size}{10 * res}\right)$ ***İHA₄***: $x_4 = x_1 + 3 * \left(\frac{size}{10 * res}\right)$ $y_4 = y_1 - 3 * \left(\frac{size}{10 * res}\right)$	Lider: $İHA_1$ (referans) Her İHA için yaklaşık 5 m. Çarpışma Önleme Güvenlik Alanı $= r_u = \left(\frac{size}{5 * res}\right)$
 **Box**	***İHA₁***: x_1, y_1 ***İHA₂***: $x_2 = x_1 + \left(\frac{size}{10 * res}\right)$ $y_2 = y_1$ ***İHA₃***: $x_3 = x_1 + \left(\frac{size}{10 * res}\right)$ $y_3 = y_1 - \left(\frac{size}{10 * res}\right)$ ***İHA₄***: $x_4 = x_1$ $y_4 = y_1 - \left(\frac{size}{10 * res}\right)$	Lider: $İHA_1$ (referans) Her İHA için yaklaşık 5 m. Çarpışma Önleme Güvenlik Alanı $= r_u = \left(\frac{size}{5 * res}\right)$
 **Close Trail**	***İHA₁***:x_1, y_1 ***İHA₂***: $x_2 = x_1$ $y_2 = y_1 - \left(\frac{size}{10 * res}\right)$ ***İHA₃***: $x_3 = x_1$ $y_3 = y_1 - 2 * \left(\frac{size}{10 * res}\right)$ ***İHA₄***: $x_4 = x_1$ $y_4 = y_1 - 3 * \left(\frac{size}{10 * res}\right)$	Lider: $İHA_1$ (referans) Her İHA için yaklaşık 5 m. Çarpışma Önleme Güvenlik Alanı $= r_u = \left(\frac{size}{5 * res}\right)$

Çizelge 5.2'de gösterildiği gibi lider platformun konumu manuel olarak verilecek ve diğer platformların konumu Çizelge 5.1 ile tanımlandığı şekilde lider platrormun konumuna ve formasyon şemasına bağlı olarak hesaplanacaktır.

Çizelge 5.2: Otonom formasyon kontrolü konumlandırma uygulaması parametreleri.

Simülasyon No	Lider Konumu	Formasyon Düzeni
1	(50,450)	Echelon Left
2	(100,450)	Box
3	(250,450)	Trail

Bu noktadan hareket ile Şekil 5.10 (a) ile lider platform için Echelon Right uçuş formasyonu düzenine göre platformların birbirleri ile çarpışmasına engel olacak şekilde düzenlenmiş güvenlik alan modeli gösterilmektedir.



(a) Lider İHA vektör alanı.

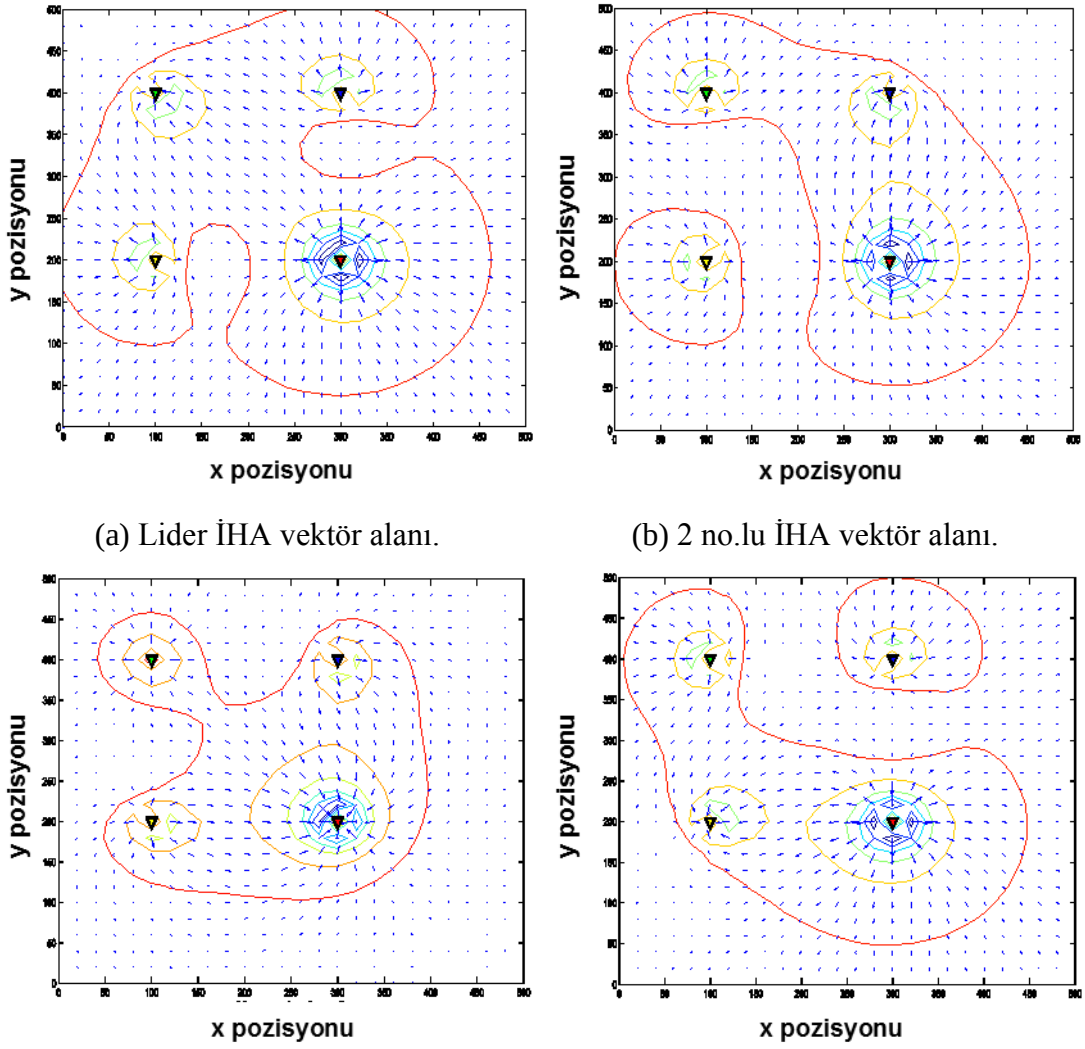
(b) 2 no.lu İHA vektör alanı.

(c) 3 no.lu İHA vektör alanı.

(ç) 4 no.lu İHA vektör alanı.

Şekil 5.10: Echelon right formasyonunda yer alan her bir platform için çarpışmayı önleyici güvenlik alanı modeli gösterimi.

Lider araç için diğer üç araç iten karakterde noktasal potansiyel alan olarak tanımlanmıştır. Alan sınırları Çizelge 5.1'deki örneğe uygun olarak olarak belirlenmiştir. Lider dışındaki diğer platformlar için de çarpışmayı önleyici potansiyel alanlar Şekil 5.10'dan görülebilir. Dinamik olarak hareket eden formasyonu teşkil edecek İHA'ların formasyon düzenini almaları esnasında ya da engel gibi nedenlerden dolayı formasyon değişimine gereksinim duyulması sırasında aralarında oluşabilecek çarpışma durumunun yaşanmamasını her bir platform için tasarlanan ve hesaplanan çarpışmayı önleyici potansiyel alan garanti etmektedir.



(a) Lider İHA vektör alanı. (b) 2 no.lu İHA vektör alanı.

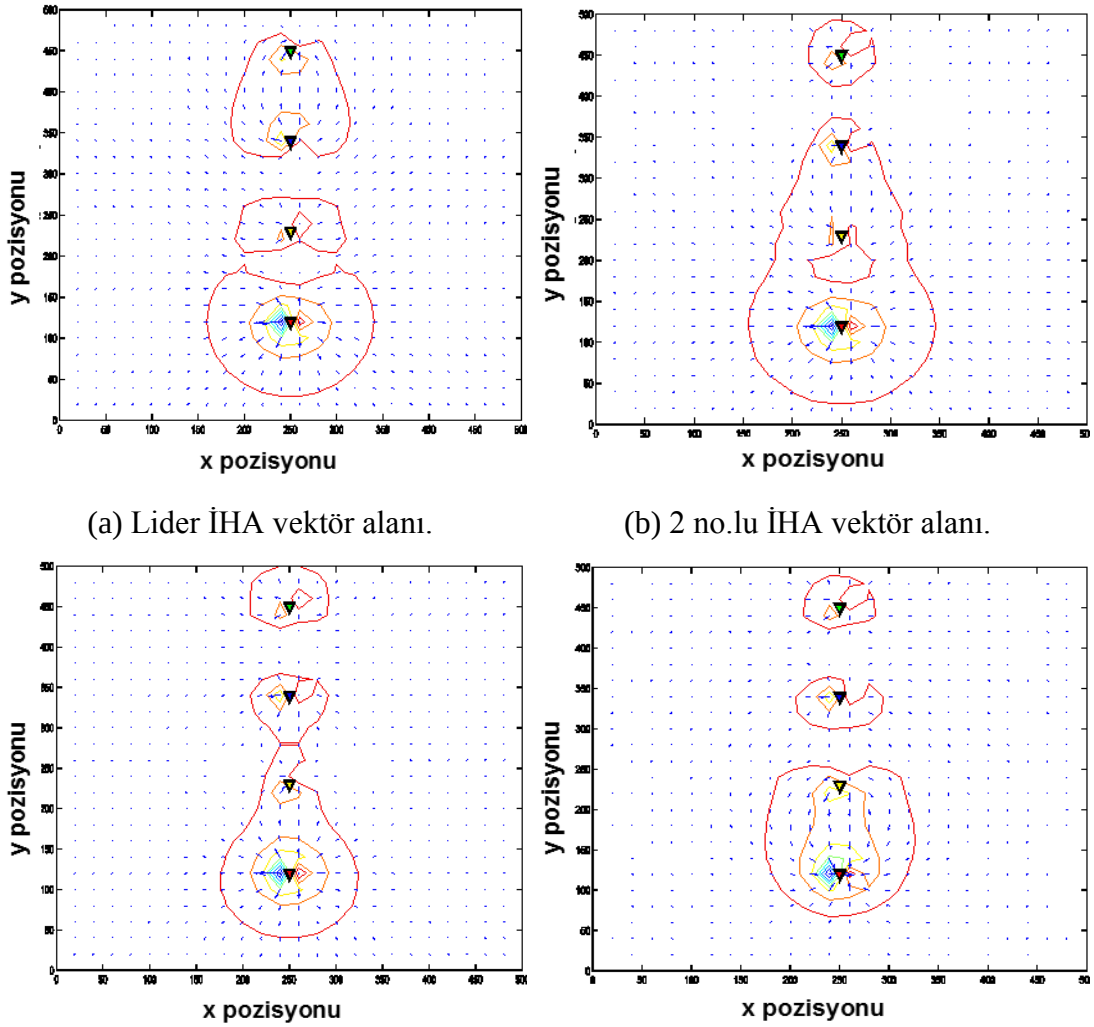
(c) 3 no.lu İHA vektör alanı. (ç) 4 no.lu İHA vektör alanı.

Şekil 5.11: Box formasyonunda yer alan her bir platform için çarpışmayı önleyici güvenlik alanı modeli gösterimi:

Şekil 5.11'de her bir İHA'nın Çizelge 5.1 ile belirlenen box formasyon düzenine göre bulunması gereken noktanın noktasal kaynağa çeken potansiyel alan olarak modellendiği ve buna göre ortaya çıkan vektör alan da gösterilmiştir. HYİ görevi için

otonom yol planlamasına nazaran formasyon kontrolünün YPA ile tanımlanması sırasında görülen bir diğer fark, formasyon dahilinde otonom kontrolü planlanan her bir araç için ayrı ayrı potansiyel alanların hesap edilerek vektörel alanların oluşturulmasıdır.

Şekil 2.10 (d) ile gösterilen ve Çizelge 5.1 ile tanımlanan Trail formasyonunda yer alan İHA'ların her birisine ait çarpışmayı önleyici potansiyel alan tanımlamaları Şekil 5.12 ile sırasıyla gösterilmiştir.



(a) Lider İHA vektör alanı.

(b) 2 no.lu İHA vektör alanı.

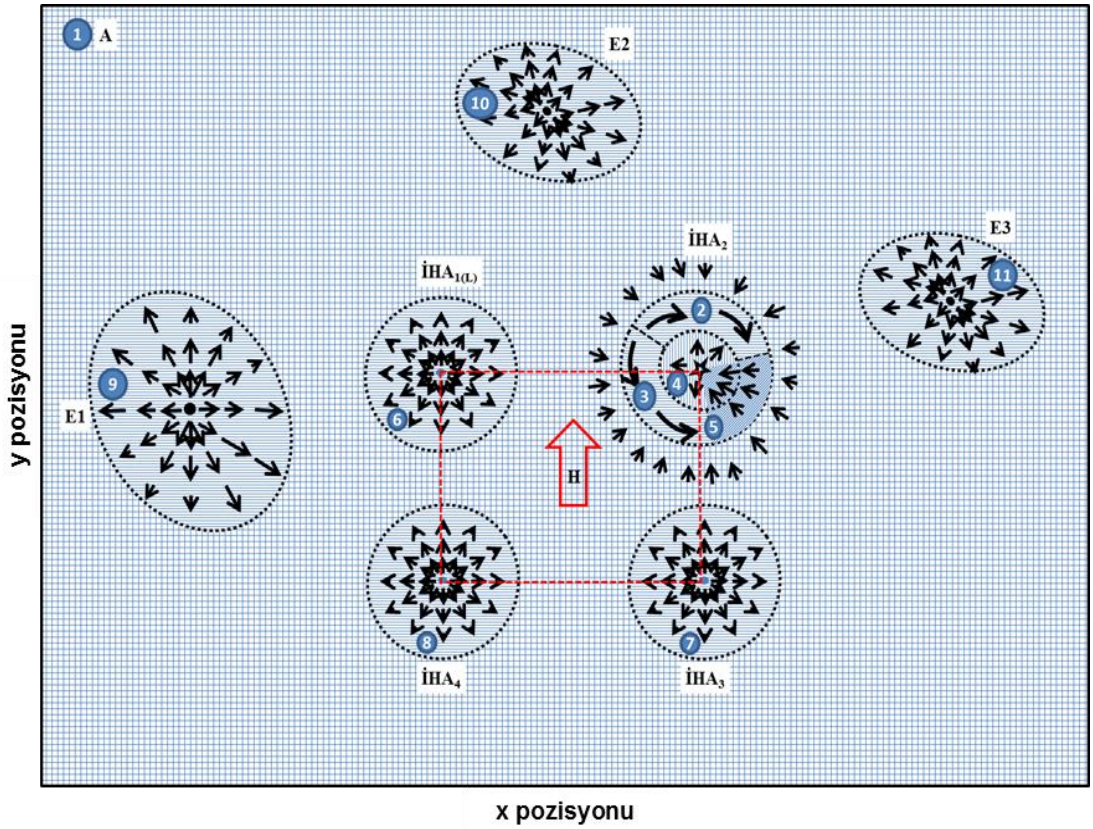
(c) 3 no.lu İHA vektör alanı.

(ç) 4 no.lu İHA vektör alanı.

Şekil 5.12: Trail formasyonunda yer alan her bir platform için çarpışmayı önleyici güvenlik alanı modeli gösterimi.

Formasyonun gruptan farkı, formasyon dahilinde yer alan platformların birbirlerinin konumu alamamasıdır. Bir başka ifade ile grup oluşumu dahilinde yer alan platformlardan hangisinin istenen konumu aldığı önemli değildir, genellikle rastlantısal bir durum olarak değerlendirilebilir. Ancak formasyon dahilinde yer alan

her bir platforma tahsis edilmiş özel bir şema dahilinde konum vardır. Formasyon şemasına bağlı olarak formasyon dahilinde yer alan platformların platform düzeninde doğru yeri almaları ve bu esnada birbirleri ile kesişme ihtimali olan rotalarda birbirleri ile çarpışmalarını önleyecek biçimde bir yol planlaması modeli kurulması ihtiyacı kaçınılmazdır. Bu durumda dinamik olarak modellenen HYİ görevindeki platformun hareketinin potansiyel alanlar ile modellenmesine benzer bir yaklaşımdan faydalanmak mümkündür. Ayrıca formasyon dahilinde konumu almak isteyen platforma ortam içinde bulunan diğer engeller de etki edecektir.



Şekil 5.13: Engeller içeren alan içinde kare formasyonda lidere göre konumunu alan platforma etki eden sınırlandırılmış potansiyel alanların gösterimi.

Şekil 5.13 ile, "*A*" olarak tanımlanan alan içinde yer alan ve hareketinin doğrultusu "*H*" ile temsil edilen kare formasyonu dahilindeki dört platformdan $İHA_2$ olarak adlandırılan platformun $İHA_1$ olarak adlandırılan lider platforma göre konumunu alması esnasında etkisinde kalabileceği değerlendirilen vektör alanlar gösterilmektedir. HYİ görevindeki benzer olarak temel beş alan platformun konumunu alması için faydalanılmış olup, "*5*" no.lu alanın y ekseni ile açısı formasyonun "*H*" ile ifade edilen hareket yönü baz alınarak özellikle $İHA_3$ ile kesişmeden harekete devam edebilecek nitelikte konum alınması için

oluşturulmuştur. Temsili olarak "*E*" ile gösterilen üç engelin alan dahilinde yer aldığı kabul edilmiş olup, bu sayı daha fazla ya da az olabilir.

5.3 YPA Tabanlı Otonom Yol Planlaması Algoritması

Alan içerisinde bulunan "*1*" adet noktasal çeken, "*n*" adet noktasal iten alt alan olduğu kabul edilen YPA fonksiyon çağrısı incelendiğinde girdi parametreleri olarak sırasıyla alan bilgileri, hedef bilgileri ve engel bilgileri verilir. Çıktı olarak da vektörel alan bilgileri geri döndürülür. Bu yapı aşağıda gösterilmiştir.

[step, dist_X, dist_Y, SX, SY, Magnitude, Heading]=ypa ([size res], [δ_c γ_c β_c C_x C_y], [δ_{cm} γ_{cm} β_{cm} C_{xm} C_{ym}], [δ_{I1} γ_{I1} β_{I1} I_{x1} I_{y1}], […],[δ_{In} γ_{In} β_{In} I_{xn} I_{yn}])

Vektörel alanın hesaplaması için tasarlanan algoritma *Algoritma 1* ile gösterilmiştir. *1.3* numaralı döngü iki boyutlu alanın *x*-eksenindeki her bir noktanın, *1.4* numaralı döngü ise *y*-eksenindeki her bir noktanın potansiyel değerinin hesaplanması içindir. *1.8* numaralı döngü ise her bir engel için iten potansiyel kuvvetin hesaplanması için kullanılan döngüdür. Algoritma her bir (*x, y*) noktası ve engel sayısı kadar tekrar eden bir yapıdadır. Alan içerisinde en az bir çeken alan hedef özellikleri gösterecek şekilde tanımlanmalıdır. Dolayısıyla *YPA* fonksiyon çağrısının en az iki girdi parametresi olmalıdır. Alan içinde olması istenen engeller ise opsiyon olarak girdi şeklinde tanımlanabilir. Algoritmanın *1.16* ve *1.17* adımında potansiyel alanın her bir (*x, y*) noktasındaki toplam potansiyel kuvvetlerin değeri, *1.18* adımında potansiyel alanın her bir (*x, y*) noktasında yer alan vektörün genliği ve *1.19* adımında potansiyel alanın her bir (*x, y*) noktasında yer alan vektörün yön değeri hesaplanır.

Algoritma 1 sadece tek tip potansiyel alan tanımlaması olduğunda ya da çeken potansiyel alana ilave olarak tek tip bir diğer potansiyel alan tanımlaması hesaplanmak istediğinde etkin bir yaklaşımdır. Ancak Çizelge 3.1 ile gösterilen farklı potansiyel alanlar bir arada kullanılarak hesaplanmak istendiğinde yetersiz kalacaktır. Bu nedenle temel algoritma üzerinde koşul değerlerine bağlı olarak değişiklikler yapmak zorundayız. Bu nedenle Çizelge 3.1'deki farklı alan modelleri bir parametre ile algoritmaya eklenmiş ve *Algoritma 2* şeklinde yeniden düzenlenmiştir.

Algoritma 1. YPA_Hesapla

girdi: ([size res], [δ_c γ_c β_c C_x C_y], [δ_{cm} γ_{cm} β_{cm} C_{xm} C_{ym}], [δ_{I1} γ_{I1} β_{I1} I_{x1} I_{y1}], […],[δ_{In} γ_{In} β_{In} I_{xn} I_{yn}])

çıktı: [step, dist_X, dist_Y, SX, SY, Magnitude, Heading]

1.1: step değerini çözünürlük (res) ve boyut (size) parametrelerine göre hesapla
1.2: $dist\_X$ ve $dist\_Y$ iki boyutlu mesafe matrislerini step değerlerine göre hesapla
1.3: **for** i ← 1 to step **do**
1.4: **for** j ← 1 to step **do**
1.5: $DX_c(i,j)$ hesapla
1.6: $DY_c(i,j)$ hesapla
1.7: $SX(i,j)$ ve $SY(i,j)$ değerlerini sıfırla
1.8: **for** *k* ← 1 to *(nrhs-2)* **do** //nrhs : girdi parametre sayısı
1.9: $DX_{I(k)}(i,j)$ ve $DY_{I(k)}(i,j)$ değerlerini hesapla
1.10: $DX_{I(k)}(i,j)$ ve $DY_{I(k)}(i,j)$ düzeltmesini hesapla
1.11: (i, j) noktası için engellerin oluşturduğu düzeltilmiş potansiyel toplamını hesapla:
1.12: $SX(i,j) = SX(i,j) + DX_{I(k)}(i,j)$;
1.13: $SY(i,j) = SY(i,j) + DY_{I(k)}(i,j)$;
1.14: **end for**
1.15: Engellerin oluşturduğu toplam potansiyel değerini hedef potansiyel değerinden çıkar ve noktanın potansiyel değerini bul:
1.16: $SX(i,j) = DX_c(i,j) + SX(i,j)$;
1.17: $SY(i,j) = DY_c(i,j) + SY(i,j)$;
1.18: (i,j) noktasındaki vektörün genliğini (magnitude) hesapla $magnitude(i,j)$
1.19: (i,j) noktasındaki vektörün yönünü (heading) hesapla $heading(i,j)$
1.20: **end for**
1.21: **end for**

Alan içerisinde bulunan "*m*" adet noktasal çeken, "*n*" adet noktasal iten, "*p*" adet saat yönünde dönen, "*r*" adet saat yönü tersine dönen ve "*t*" adet panel tarafından itilen alt alan olduğu kabul edilen YPA fonksiyon çağrısı incelendiğinde girdi parametreleri olarak sırasıyla alan bilgileri, hedef bilgileri ve/veya engel bilgileri verilir. Çıktı olarak da vektörel alan bilgileri geri döndürülür. Bu yapı aşağıda gösterilmiştir.

[step, dist_X, dist_Y, SX, SY, Magnitude, Heading]=ypa ([size res],
[ptype δ γ β x y], […])

Fonksiyon çağrısı incelendiğinde hedef ve engel alanlar için farklı parametre tanımlaması yapmaya gerek yoktur. Çünkü Çizelge 3.1 incelendiğinde hedef (çeken) ve engel (iten) potansiyel alanlar modellemek için kullanılabilen yapılar farklı olarak

görülmektedir. Bu nedenle alan içerisinde yer alan her bir alt potansiyel alan "*ptype*" olarak belirtilen parametreye istinaden farklı şekilde hesaplanacaktır. Ancak algoritmanın bu yapısı da istenilen otonom yol planlamasının yapılması için yeterli değildir. Çünkü farklı tipte alt potansiyel modellerin bir arada kullanıldığı otonom YPA tabanlı yol planlaması modellerinde istenilene yakın paternler üretilebilmesi için alt potansiyel bölgelerin doğru sınır fonksiyonları ile (bu tez kapsamında ikili ve sigmoid fonksiyonlar kullanılmıştır) sınırlandırılması gerekmektedir. Bu nedenle sınır fonksiyon değerleri de parametre olarak eklenmelidir. Dolayısıyla YPA fonksiyon çağrısı aşağıdaki şekilde güncellenebilir.

[step, dist_X, dist_Y, SX, SY, Magnitude, Heading]=ypa2 ([size res],
[ptype δ γ β x y stype range], […])

Gerek ikili gerekse de sigmoid sınır fonksiyonları kullanıldığında "*range*" parametresine ihtiyaç vardır. Alt potansiyel bölgenin merkez noktasından ne kadar nokta sayısı kadar etkin olacağını gösterir. Ayrıca sigmoid sınır fonksiyonları kullanıldığında bir diğer ihtiyaç duyulan parametre de "*slope*" parametresidir ve bu tez kapsamında bu değer sabit değer olarak alınmıştır. Bu durumda algoritmayı *Algoritma 2*'de görüldüğü şekilde güncellemek mümkündür.

Algoritma 2. YPA_Hesapla_2

girdi: ([size res], [ptype δ γ β x y stype range], […])
çıktı: [step, dist_X, dist_Y, SX, SY, Magnitude, Heading]

2.1: step değerini çözünürlük (res) ve boyut (size) parametrelerine göre hesapla
2.2: $dist\_X$ ve $dist\_Y$ iki boyutlu mesafe matrislerini step değerlerine göre hesapla
2.3: **for** $i \leftarrow$ 1 to step **do**
2.4: **for** $j \leftarrow$ 1 to step **do**
2.5: $SX(i,j)$ ve $SY(i,j)$ değerlerini sıfırla
2.6: **for** $k \leftarrow$ 1 to *(nrhs-1)* **do** //nrhs : girdi parametre sayısı (number of right hand side)
2.7: **Select Case of** ptype
2.8: **Case:** 5 //Kaynağa Dik Potansiyel Alan Modeli
2.9: **if** (stype == binary)
2.10: $DX_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ değerlerini ikili s.f. ile hesapla
2.11: **else**
2.12: $DX_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ değerlerini sigmoid s.f. ile hesapla
2.13: **end if**
2.14 : **Case:** 4 // Noktasal Kaynağa Teğet Alan Modeli (saat yönü tersine)
2.15: **if** (stype == binary)
2.16: D$X_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ değerlerini ikili s.f. ile hesapla
2.17: **else**

2.18: $DX_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ değerlerini sigmoid s.f. ile hesapla
2.19: **end if**
2.20: **Case:** 3 // Noktasal Kaynağa Teğet Alan Modeli (saat yönünde)
2.21: **if** (stype == binary)
2.22: D$X_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ değerlerini ikili s.f. ile hesapla
2.23: **Else**
2.24: $DX_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ değerlerini sigmoid s.f. ile hesapla
2.25: **end if**
2.26: **Case:** 2 // Noktasal Kaynaktan İten Potansiyel Alan Modeli
2.27: **if** (stype == binary)
2.28: D$X_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ değerlerini ikili s.f. ile hesapla
2.29: **else**
2.30: $DX_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ değerlerini sigmoid s.f. ile hesapla
2.31: **end if**
2.32: **Default** // Noktasal Kaynağa Çeken Potansiyel Alan Modeli
2.33: **if** (stype == binary)
2.34: D$X_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ değerlerini ikili s.f. ile hesapla
2.35: **else**
2.36: $DX_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ değerlerini sigmoid s.f. ile hesapla
2.37: **end if**
2.38: **end Case**
2.39: D$X_{a(k)}(i,j)$ ve $DY_{a(k)}(i,j)$ düzeltmesini hesapla
2.40: (i, j) noktası için bölgelerin oluşturduğu düzeltilmiş potansiyel toplamını hesapla
2.41: $SX(i,j) = DX_{a(k)}(i,j) + SX(i,j)$;
2.42: $SY(i,j) = DY_{a(k)}(i,j) + SY(i,j)$;
2.43: **end for**
2.44: (i,j) noktasındaki vektörün genliğini (magnitude) hesapla $magnitude(i,j)$
2.45: (i,j) noktasındaki vektörün yönünü (heading) hesapla $heading(i,j)$
2.46: **end for**
2.47: **end for**

Algoritma 2'de satır 2.7'de yer alan "*case*" yapısı ile farklı alt potansiyel alanların farklı hesaplamalarına imkan verilmiştir. Ayrıca her bir alt potansiyel bölgenin "*if*" koşuluna bağlı olarak iki farklı sınır fonksiyonundan birisi ile tanımlanmasına imkân sağlanmıştır. Doğal olarak *Algoritma 2*'nin karmaşıklığı *Algoritma 1*'e göre artmış ancak bellek kullanımında kayda değer değişim olmamıştır.

Çalışmanın bu bölümüne kadar temel YPA tabanlı yol planlaması algoritmaları ele alınmıştır. Ancak YPA tabanlı yol planlaması algoritmaları tanımlanan probleme özgün olarak özelleştirilebilir.

Algoritma 3. YPA_HYİ

girdi: ([size res], $[x_t\ y_t\ \gamma\ r_1\ r_2\ k_1\ k_2\ k_3\ k_4\ k_5]$)
çıktı: [step, dist_X, dist_Y, SX, SY, Magnitude, Heading, S_1, S_2, S_3, S_4, S_5]

3.1: step değerini çözünürlük (res) ve boyut (size) parametrelerine göre hesapla
3.2: $dist\_X$ ve $dist\_Y$ iki boyutlu mesafe matrislerini step değerlerine göre hesapla
3.3: **for** i ← 1 to step **do**
3.4: **for j** ← 1 to step **do**
3.5: $S_1(i,j)$ değerini değerini sigmoid fonksiyon kullanarak hesapla
3.6: $S_5(i,j)$ değerini değerini sigmoid fonksiyon kullanarak hesapla
3.7: $S_4(i,j)$ değerini değerini sigmoid fonksiyon kullanarak hesapla
3.8: $S_3(i,j)$ değerini değerini ikili fonksiyon kullanarak hesapla
3.9: $S_2(i,j)$ değerini değerini ikili fonksiyon kullanarak hesapla

3.10: $DX_{s1}(i,j)$ ve $DY_{s1}(i,j)$ değerlerini hesapla (x, y noktasındaki birinci bölge vekörel bileşen değerleri)
3.11: $DX_{s2}(i,j)$ ve $DY_{s2}(i,j)$ değerlerini hesapla (x, y noktasındaki ikinci bölge vekörel bileşen değerleri)
3.12: $DX_{s3}(i,j)$ ve $DY_{s3}(i,j)$ değerlerini hesapla (x, y noktasındaki üçüncü bölge vekörel bileşen değerleri)
3.13: $DX_{s4}(i,j)$ ve $DY_{s4}(i,j)$ değerlerini hesapla (x, y noktasındaki dördüncü bölge vekörel bileşen değerleri)
3.14: $DX_{s5}(i,j)$ ve $DY_{s5}(i,j)$ değerlerini hesapla (x, y noktasındaki beşinci bölge vekörel bileşen değerleri)

3.15: $SX(i,j) = \left(k_1 * S_1(i,j) * DX_{s1}(i,j)\right) + \left(k_2 * S_2(i,j) * DX_{s2}(i,j)\right) + \left(k_3 * S_3(i,j) * DX_{s3}(i,j)\right) + \left(k_4 * S_4(i,j) * DX_{s4}(i,j)\right) + \left(k_5 * S_5(i,j) * DX_{s5}(i,j)\right)$

3.16: $SY(i,j) = \left(k_1 * S_1(i,j) * DY_{s1}(i,j)\right) + \left(k_2 * S_2(i,j) * DY_{s2}(i,j)\right) + \left(k_3 * S_3(i,j) * DY_{s3}(i,j)\right) + \left(k_4 * S_4(i,j) * DY_{s4}(i,j)\right) + \left(k_5 * S_5(i,j) * DY_{s5}(i,j)\right)$

3.17: (i,j) noktasındaki vektörün genliğini (magnitude) hesapla $magnitude(i,j)$
3.18: (i,j) noktasındaki vektörün yönünü (heading) hesapla $heading(i,j)$
3.19: **end for**
3.20: **end for**

Bundan sonraki bölümde örnek olarak raporun 5.1 bölümde "Otonom havada yakıt ikmali yol planlaması" başlığı altında açıklanan şekilde otonom yol planlamasının yapılabilmesi için raporun 3.2 başlığı altında tanımlanan temel YPA modellerinin bir araya getirilerek çalışmanın 3.3.1 bölümünde açıklanan temel potansiyel alanların sınırlandırılması başlığı altında tanımlanan sınır fonksiyonlardan faydalanarak elde edilmesi için gereksinim duyulan algoritma tasarımına değinilmiştir. Böylece,

otonom havada yakıt ikmalinin gerçekleştirilmesine imkan sağlayacak nitelikte yol planlaması formasyon dahilinde yer alan otonom platform için üretilebilmiştir.

Algoritma 3 için fonksiyon çağrısı aşağıda görüldüğü şekildedir:

[step, dist_X, dist_Y, SX, SY, Magnitude, Heading, S_1, S_2, S_3, S_4, S_5*] = apf_hyi ([size res],* $[x_t\ y_t\ \gamma\ r_1\ r_2\ k_1\ k_2\ k_3\ k_4\ k_5]$*)*

İlk olarak bu kapsamda *Algoritma 2* üzerinde özellikle sınır değerlerin hesaplanmasına yönelik değişiklikler yapılarak *Algoritma 3* elde edilmiştir. Fonksiyon çağrısından da görüldüğü üzere bu kez girdi parametresi sayısı fonksiyon için sabittir. Bunun nedeni HYİ görevi için alanı içerisinde tanımlı sayıda (toplam beş adet) alt bölge tanımlaması yapıldığı içindir. Bu alt bölgelerin hangi tip potansiyel alanlardan teşkil edileceği dolayısıyla hareketi şekillendirmek açısından hangi tip temel potansiyel alana uygun oldukları da bellidir. Bu nedenle tek parametre olarak alınması gereken tanker platforma ait konum bilgileri ve alan sınırları ile alan şiddetlerinin parametrelerinin belirtildiği parametrelerdir.

Algoritma *3.5* ve *3.9* adımları arasında her bir alt bölgenin her bir (x, y) noktasındaki sınır fonksiyon değerleri Denklem (5.6) ve Denklem (5.17) arasındaki denklemler kullanılarak hesaplanabilir. Algoritma 3'de de görüldüğü üzere iki döngü içerisinde alan dahilinde yer alan her bir (x, y) noktası için beş alt potansiyel alan değeri hesaplanır. Bu değerlerin etkisi her bir alan için hesaplanan alfa kat sayısı hesaplanarak bununla çarpılarak düzeltilir ve en son kuvvet çarpanı (k) ile düzeltme verilerek o noktadaki toplam potansiyel değer hesaplanabilir. Bu kapsamda potansiyel değer üzerinden işlem yapmak yerine hesaplamalar doğrudan vektörel değerin x ve y bileşenlerine ayrılarak hesaplanması yoluna gidilmiştir.

Toplam beş alt bölgenin kısmi türevleri potansiyel fonksiyonlarına göre alınmış ve hesaplamalar sabit olan bu alan fonksiyonlarının üzerinden hesaplanmıştır. Alt bölgelere ait kısmı türev fonksiyonları Denklem (5.30) ile (5.33) arasında gösterilmiştir. Bu adımlar algoritmanın *3.10* ve *3.14* adımları arasında gerçekleştirilen işlemleri kapsamaktadır.

$$DXs_1 = -\frac{\left(2cos(h_t) * \left(cos(h_t) * (x - x_t) - sin(h_t) * (y - y_t)\right) + 2\gamma * sin(h_t) * \left(cos(h_t) * (y - y_t) + sin(h_t) * (x - x_t)\right)\right)}{\left(2\left(\gamma * \left(cos(h_t) * (y - y_t) + sin(h_t) * (x - x_t)\right)^2 + \left(cos(h_t) * (x - x_t) - sin(h_t) * (y - y_t)\right)^2\right)\right)}$$

$$DYs_1 = \frac{2 * sin(h_t) * \left(cos(h_t) * (x - x_t) - sin(h_t) * (y - y_t)\right) - 2\gamma * cos(h_t) * \left(cos(h_t) * (y - y_t) + sin(h_t) * (x - x_t)\right)}{\left(2\left(\gamma * \left(cos(h_t) * (y - y_t) + sin(h_t) * (x - x_t)\right)^2 + \left(cos(h_t) * (x - x_t) - sin(h_t) * (y - y_t)\right)^2\right)\right)} \quad \textbf{(5.30)}$$

$$DXs_2 = -\frac{\left(4 * (x - x_t)\right)}{(x - x_t)^2 + \gamma * (y - y_t)^2}$$

$$DYs_2 = -\frac{\left(4 * \gamma * (y - y_t)\right)}{((x - x_t)^2 + \gamma * (y - y_t)^2)} \quad \textbf{(5.31)}$$

$$DXs_3 = \frac{\left(4 * (x - x_t)\right)}{((x - x_t)^2 + \gamma * (y - y_t)^2)}$$

$$DYs_3 = \frac{\left(4 * \gamma * (y - y_t)\right)}{((x - x_t)^2 + \gamma * (y - y_t)^2)} \quad \textbf{(5.32)}$$

$$DXs_4 = \frac{\left(2cos(h_t) * \left(cos(h_t) * (x - x_t) - sin(h_t) * (y - y_t)\right) + 2\gamma * sin(h_t) * \left(cos(h_t) * (y - y_t) + sin(h_t) * (x - x_t)\right)\right)}{(2 * (\gamma * \left(cos(h_t) * (y - y_t) + sin(h_t) * (x - x_t)\right)^2 + \left(cos(h_t) * (x - x_t) - sin(h_t) * (y - y_t)\right)^2))}$$

$$DYs_4 = -\frac{\left(2sin(h_t) * \left(cos(h_t) * (x - x_t) - sin(h_t) * (y - y_t)\right) - 2\gamma * cos(h_t) * \left(cos(h_t) * (y - y_t) + sin(h_t) * (x - x_t)\right)\right)}{\left(2 * \left(\gamma * \left(cos(h_t) * (y - y_t) + sin(h_t) * (x - x_t)\right)^2 + \left(cos(h_t) * (x - x_t) - sin(h_t) * (y - y_t)\right)^2\right)\right)} \quad \textbf{(5.33)}$$

Her bir alt bölgenin (x, y) noktasındaki potansiyel değerinin ve sınır fonksiyon değerinin hesaplanması sonucunda algoritmanın *3.15* ve *3.16* ile numaralandırılan adımlarda gösterildiği şekilde her bir (x, y) noktasına etki eden toplam potansiyel kuvvet değeri elde edilir. (x, y) noktasında elde edilen vektörel büyüklüğün bileşen kuvvetlerinden faydalanılarak alan içindeki her noktada İHA platformuna etki eden komuta ait yön ve genlik verileri algoritmanın *3.17* ve *3.18* adımlarında hesaplanabilir.

6. GPGPU TABANLI PARALEL YPA HESAPLAMASI

Bölüm 2 kapsamında ortaya konulan ve açıklanan şekilde otonom İHA platformlarının formasyon uçuşu dahilinde kontrolü dinamik bir yaklaşım olup çok hızlı değişiklik gösterebilen koşullara bağlıdır. Aslında uçuş ortamı üç boyutlu olduğu halde, aynı irtifada uçuş gerçekleştiren platformlar için Bölüm 4 kapsamında açıklandığı şekilde problem iki boyutlu olarak kabul edilebilir.

Çalışma kapsamında yol planlaması amacıyla Bölüm 3 kapsamında nedenleri ile açıklandığı şekilde potansiyel alan yaklaşımı belirlenmiştir. Potansiyel alan yaklaşımı gradient fonksiyonun bir sonucu olan ızgara tabanlı bir yol planlaması sunar. Potansiyel alan tabanlı kontrol, skalar bir alanın gradientinin hesaplanması ile davranışları modelleyen vektör alanın üretilmesi ile elde edilir. Skalar bir alanın gradienti vektör alanı oluşturur ve onun genliği değişimin şiddetini, yönü ise en yüksek değişimin alan içerisinde yönünü temsil etmektedir. Şayet vektör bileşenlerine ayrılır ise, her bir alan eksenindeki değişim miktarları elde edilebilir. Gradient sonucunda elde edilen genlik bileşenleri her bir eksendeki değişimin miktarını temsil edecektir. Şayet skalar fonksiyon türevi alınabilir bir fonksiyon ise, fonksiyonun gradienti bir vektör alan oluşturur ve bileşen durumundaki her bir vektör değişimin yönünü ve şiddetini temsil eder.

Bölüm 5 kapsamında ortaya konulan havada yakıt ikmali ve formasyon uçuşu gibi örnek senaryolara ait otonom sistemlerin yol planlaması ihtiyacı, Bölüm 3 kapsamında açıklanan temel YPA tabanlı yol planlaması yapılarından faydalanılarak modellenmiştir. Ancak her ne kadar elde edilen sonuçlar başarılı olsalar dahi ortaya konulan sonuçlar özellikle üç boyutlu ortamda yüksek hızlı hareket kabiliyetine sahip İHA platformları için hesaplama performansı açısından yetersiz olmaktadır. Bu nedenle çalışmanın bu başlığı altında YPA tabanlı yol planlaması hesaplama performansının paralel işlemciler üzerinde koşturmak üzere nasıl bir yöntem ile paralel algoritmalar hale getirileceğine değinilmiş ve özellikle SIMD mimarisinde paralel uygulamaların koşturulması amacıyla son yıllarda etkin olarak kullanılan

GPU tabanlı mimariler üzerinde uygulamanın hesaplama performansının nasıl arttırılacağı irdelenmiştir.

Bu tez kapsamında özgün olarak ortaya konulmak istenen yaklaşım, YPA yaklaşımının eksik kaldığı durumları giderecek şekilde düzenlenmiş (harmonik fonksiyonlar ile modellenen genel yol planlaması fonksiyonları gibi) YPA algoritmasını, uygulanabilir sonuçlar üretecek nitelikte paralel işlemciler üzerinde hesaplayarak gerçek zamana en yakın sonuçlar üretmektir. Gerçek zamanlı hesaplama ile amaçlanan performans değerleri otonom platformun yeteneklerine bağlı olarak değişiklik göstermektedir. Amaçlanan hesaplama performansı yol planlamasına ihtiyaç duyan otonom platformların gerçekleştirmeyi amaçladığı görev çerçevesinde uygulanabilir en hızlı sonuçları üretmektir. Bu durum çalışmanın Bölüm *4.2* başlığı altında detaylı olarak açıklanmıştır.

Özellikle geniş ölçekli tanımlanmış ve hassas kontrol imkânı sağlayan dinamik potansiyel alanlar için gerçek zamanlı hesaplama ihtiyacı karşılanması zor bir problemdir. Bunun yanında hesaplama performansı doğrudan skalar modelin boyutlarına, ölçeğine ve de karmaşıklığına bağlıdır. Geniş ölçekli ve üç boyutlu modellenmiş dinamik ve karmaşık problemlerin geleneksel işlemciler ile hesaplanması gerçek zamanlı çözümlerin üretilmesi açısından yeterli değildir. Çok boyutlu skalar alanların gradientinin hesaplanması işlemi aslında alan dahilinde yer alan her bir nokta için kısmı türev komutlarından oluşan aynı komut setinin tekrarlanmasıdır. Bu komut setinin girdi değerleri her bir nokta için farklıdır, çünkü her noktanın potansiyel değeri farklıdır. Bu tanımdan da anlaşıldığı üzere bu yaklaşım SIMD tipindeki paralel programlama mimarisinde uygundur [53].

Bölüm 2.5.2 kapsamında açıklanan biçimde, çalışma kapsamında geliştirilen paralel algoritmaların koşturulabileceği bellek paylaşımlı çok işlemcili donanım mimarisinin, İHA sistemleri gibi hareketli ve uzun süreli görev yapan ortamlarda etkin olarak kullanılabilmesi amacıyla;

a. Düşük güç tüketimleri,

b. Fiziksel olarak küçük boyutta olmaları,

c. Hafif olmaları,

ç. Çok sayıda çekirdek ile SIMD tipinde paralel uygulamaların koşturulmasına imkan vermesi gibi gereksinimleri vardır.

Bu noktada, hem fiziksel ihtiyaçları karşılayabilecek hem de istenen hesaplama performans değerlerini sağlayabilecek paralel mimariler olarak grafik işlemcilerin uygun olduğu değerlendirilmiştir [54].

Çalışmanın bu bölümü kapsamında, geniş ölçekli skalar alanlar için gradient fonksiyonun hesaplama performansının SIMD yapısına uygun olarak GPGPU mimarisinden faydalanılarak arttırılmasına değinilecektir [49]. GPU tabanlı akış işlemcilerin tek bir merkezi işlemci birimi yerine kullanılması amacıyla paralel algoritmalar geliştirilecek ve vektör veriler ile modellemeler ortaya konularak paralel gradient hesaplama ve yol planlaması algoritmaları geliştirilecektir. GPU donanımları üzerinde CUDA desteği ile GPGPU paralel programlama yapısına uygun olarak potansiyel alan tabanlı vektör alanların hesaplanması amacıyla algoritmaların geliştirilmesine bu bölüm kapsamında değinilecektir.

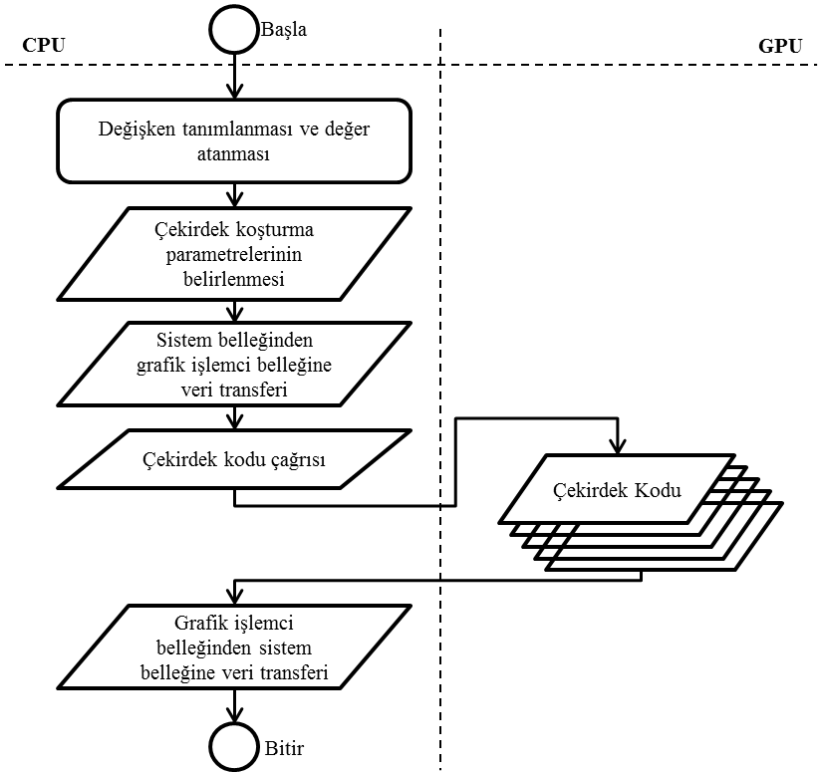
Bu bölüm kapsamında tasarlanan paralel algoritmalar, çalışmanın ilerleyen aşamalarında bellek paylaşımlı işlemciler üzerinde koşturulmuş ve gerçek zamanlı çözüm ihtiyaçlarına üç boyutlu ortamlar için karmaşık yol planlaması hizmetinin (formasyonu destekleyen, çarpışmayı engelleyen ve engellerden sakınma özelliğini barındıran) ne derece cevap verebileceği sınanmıştır.

6.1 GPGPU Tabanlı Algoritma Geliştirme

Yüksek çözünürlüklü ve dinamik parametrelere bağlı olarak değişiklik gösterebilen vektör alanın hesaplama ihtiyaçlarını gerçek zamanlı olarak karşılayabilmek için paralel hesaplama tekniklerinden faydalanmak çözüm yöntemlerinden birisi olarak kabul edilebilir. Grafik işlemciler günümüzde genel maksatlı paralel programlama amacıyla kullanılan güçlü yapılar arasındadır. Genel maksatlı programlama amacıyla kullanılan GPU donanımları SIMD tipinde problemlerin paralel olarak hesaplanmasına yönelik olarak kullanılabilir mimarilerdir. Çalışmanın önceki bölümlerinde tanımlanan vektör alanın hesaplanması amaçlı seri algoritmalar incelendiğinde, alan içerisinde yer alan her bir noktanın hesaplanması aynı fonksiyonun veya komut setinin tekrar edilmesi ile gerçekleştirilmektedir.

Grafik işlemciler ham veri işleme güçlerini programlama imkanları ile birleştirerek paralel bir işlemci mimarisi sunmaktadırlar [53]. Özellikle geniş veri setleri üzerinde çok sayıda birlikte çalışan çekirdek imkanı sunan GPU mimarileri yüksek hesaplama

performanslarına ulaşabilmektedir. CUDA yaygın olarak kullanılan GPU üzerinde paralel programlama imkanı sağlayan NVIDIA firması tarafından desteklenen paralel yazılım geliştirme uygulamalardan birisidir [54]. NVIDIA Fermi GPU mimarisi çok sayıda akış işlemciyi bir arada bulunduran bir grafik donanımı mimarisidir. Her bir SM 32 adet birbirleri ile köprülenmiş çekirdeği ve iki bağımsız komut işleme ünitesini bünyelerinde bulundururlar. Çalışma süresince her bir komut işleme ünitesi SIMD yapısına uygun olarak 16 adet olmak üzere SM üzerinde toplam 32 komutu bir arada koştururlar [54]. Her ne kadar CUDA altyapısı GPU mimarisinin genel maksatlı paralel programlanmasına imkan sağlasa da en yüksek hesaplama performansına erişilebilmesi için programcının dikkat etmesi gereken ve tabi olduğu bazı sınırlamalar da vardır.



Şekil 6.1: Temel CUDA program akış diyagramı.

Uygulamalara özel paralel programlama stratejisinden bağımsız olarak bir CUDA programı Şekil 6.1 ile gösterilen genel akış diyagramına uygun olarak koşturulur. Şekilde yer alan akış diyargramından da görüleceği üzere programın koşturulduğu iki farklı mimari söz konusudur. Bunlardan ilki CPU tarafındaki işlemleri, diğeri ise GPU tarafındaki işlemleri temsil eder. GPU tarafında koşturulan kod bloğu çekirdek kodu (kernel) olarak adlandırılır ve bu kod bloğundan aynı anda çok sayıda üretilerek paralel bir koşturma sağlanır. Çekirdek kod bloğundan kaç adet oluşturulacağı ve

verilerin nasıl bir organizasyon ile çekirdekler üzerine dağıtılarak sonuçların geri döndürüleceği CPU tarafındaki çekirdek koşturma parametrelerinin belirlenmesi ile tespit edilir ve GPU tarafındaki kod bloklarının koşturulması sonlanana kadar sabit kalır.

Bu parametreler iki ana unsura bağlı olarak belirlenir. İlk unsur GPU donanımının özellikleridir. Kaç iplik oluşturulabilineceği, ne kadar boyutta veri transfer edilebileceği gibi sorulara donanım özelliklerine bağlı olarak karar verilir. Diğer bir unsur ise paralel olarak koşturulmak istenen uygulamanın sistem gereksinimleridir. Her bir çekirdek kodunun ne kadar bellek alanına ihtiyaç duyduğu, çekirdeklerin birbirleri ile senkronizasyon gereksinimi gibi kısıtlar göz önünde bulundurulur ve paralel programlama algoritması buna uygun olarak geliştirilir. Dolayısıyal bir CUDA paralel programı ev sahibi (host) ve çekirdek (kernel) adı verilen iki algoritma ile temsil edilir. Ev sahibi algoritmanın içinden parametreleri belirlenmiş bir paralel çekirdek çağrısı gerçekleştirilerek paralel hesaplama icra edilir.

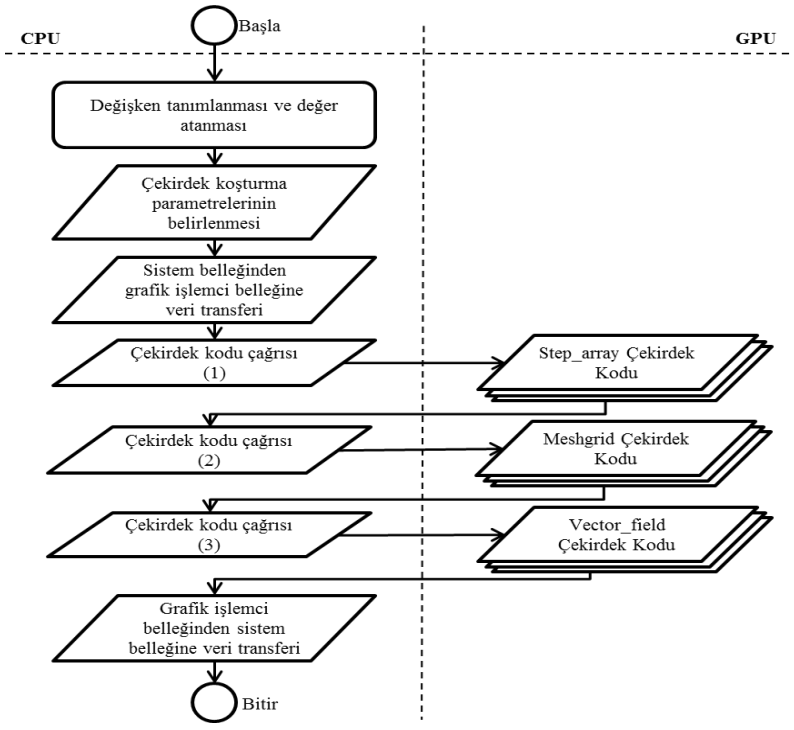
Çalışma kapsamında geliştirilecek olan çekirdek algoritmalarının geliştirilmesinde farklı paralel stratejiler belirlenebilir. Bu nedenle öncelikle SIMD paralel programlama mimarisinin nasıl probleme özgü olarak ele alınacağı irdelenmelidir. Yol planlaması dahilindeki temel hareketler; doğru konuma yönlendirilme, diğer platformlardan uzaklaşma ve diğer engellerden sakınarak çevresinden dolaşma şeklinde Çizelge 3.1 ile gösterildiği biçimde özetlenebilir. Bu kapsamda YPA tabanlı yol planlaması amacıyla faydalanılan temel yapılar ve bu yapıların elde edilmesi amacıyla kullanılan matematik modeller Çizelge 3.1 ile kapsamında ortaya konulmuştur. Sonuç olarak istenilen senaryo gereği özetle YPA tabanlı yol planlaması kapsamında ortaya konulan yaklaşım, Bölüm *3.3.1*'de yer alan sınır fonksiyonların uygun formlarının Çizelge 3.1 ile açıklanan temel modellere uygulanarak Bölüm *3.3.2*'de açıklandığı şekilde bir arada kullanılması şeklinde tanımlanabilir. Bu durumda ele alınması gereken temel fonksiyonlar Denklem (3.39) ile ifade edilen ve vektör değişimlerin iki boyutlu uzaydaki matematiksel temsilidir. Şayet alt bileşen bölgelerden oluşmuş bir potansiyel toplam alan içerisinde "*m*" adet çeken temel alt bölge, "*n*" adet iten temel alt bölge, "*p*" adet saat yönünde dönen alt bölge ve de "*r*" adet saat yönünün tersinde dönen alt bölge olduğunu kabul edersek ve toplam "*t*" adet sınırlanmış alt bölgenin kısmi türevlerinin toplamlarının bir araya

getirerek oluşturdukları alanın her bir eksendeki değişimlerini Denklem (3.39)'dan faydalanarak hesaplayabiliriz.

Bu durumda SIMD tipinde paralel programlama yapısına uygun iki farklı paralel programlama strateji izleyebiliriz. Bunlardan ilki her bir (x, y) noktasına etki eden farklı potansiyel değerlerin vektör bileşenlerini tek bir iplik ile hesaplamak ve mümkün olduğunca her bir (x, y) noktasının değerini hesaplamak üzere bir iplik oluşturarak (şayet alandaki nokta sayısı üretilebilen iplik sayısında ya da daha az ise) tayin etmek şeklinde açıklanabilir. Bir diğer yaklaşım ise her bir ipliğe bir temel potansiyel alan modelini hesaplattırarak akabinde toplam potansiyel değeri hesaplamak olarak tanımlanabilir. Bundan sonraki bölümler dahilinde öncelikle bu paralel hesaplama yaklaşımları açıklanacak akabinde ise karşılaştırmaları ile hesaplama performansına etkileri tartışılacaktır.

6.1.1 Hücre bazlı paralel hesaplama yaklaşımı

İkinci paralel hesaplama stratejisi her bir çekirdek kodunda ızgara alan içerisinde her alan her bir hücre değerine etki eden toplam potansiyel alan değerini bularak, akabinde bu hücre içinde yer alan vektör büyüklüğün bileşenlerinin hesaplanmasını ön görmektedir. Bu yaklaşım *Algoritma 2* ile ortaya konulan seri hesaplama yaklaşımının doğrudan paralel şekilde uyarlanması olarak açıklanabilir.



Şekil 6.2: GPGPU tabanlı paralel YPA hesaplaması akış diyagramı.

Tüm bu bilgiler temelinde çok sayıda sınırlı temel potansiyel alan modelini bir arada barındıran potansiyel alanın hesaplanması ve vektör alanın elde edilerek uçuş komutlarının gerçek zamanlı olarak nispeten büyük ölçekli ve de yüksek çözünürlük değerine sahip alanlar için üretilebilmesi amacıyla GPGPU tabanlı paralel programlama yapısından faydalanılmıştır. Bu kapsamda kullanılan hesaplama modeli Şekil 6.2 ile gösterilen akış diyagramı ile gösterilmiştir. Şekil 6.2'den de görüleceği üzere GPU üzerinde koşturan paralel mimaride üç çekirdek kod bloğu tanımlaması yapılmıştır. Bu çekirdek kodları bir arada potansiyel alanın hesaplanmasına ve de uçuş komutlarının üretilmesine imkân sağlamaktadır.

Algoritma 4 ile Şekil 6.1 ile gösterilen akış diyagramındaki CPU tarafından gerçekleştirilen çekirdek koşturma parametrelerinin belirlenmesi işlemleri ortaya konulmuştur.

Algoritma 4. Çekirdek Koşturma Parametrelerinin Belirlenmesi Algoritması

Girdi **:** *x, y* iki boyutlu alan büyüklüğü, *res* çözünürlük değeri
Çıktı **:** *threadsPerBlock, blocksPerGrid* çekirdek koşturma parametreleri

4.1 İki boyutlu alan içindeki hücre sayısı değerini bul ve max değerine ata
max = (x/res).(y/res)

4.2 Grafik işlemci donanım özelliklerine bağlı olarak koşturulacak paralel iplik ve blok sayısını belirle
if *(max<MaxThreadsPerBlock)*
{threads = max;}
else
{threads = MaxThreadsPerBlock; }
end if
blocks = (max + threads – 1) / threads;

4.3 Çekirdek fonksiyon koşturma parametrelerini belirle ve fonksiyon çağrılarını gerçekleştir
dim3 threadsPerBlock (threads,1,1);
dim3 blocksPerGrid (blocks,1,1);

4.4 *step_array<<< blocksPerGrid, ThreadsPerBlock>>>(x, y, res)*
meshgrid<<< blocksPerGrid, ThreadsPerBlock>>>(x, y, A)
vector_field <<<blocksPerGrid, ThreadsPerBlock>>>($A_{type}[\,]$, $A_{\alpha}[\,]$, $A_x[\,], A_y[\,], A_{\varphi}[\,], A_{\gamma}[\,]$, *X[ ], Y[ ],* $A_{slope}[\,]$, $A_{\rho}[\,]$)

GPU mimarisi üzerinde paralel olarak koşturacak olan ilk iki çekirdek kod bloğu, potansiyel alanın üzerinde yer aldığı ızgara yapısının GPU belleği üzerinde problem kapsamında tanımlanan alan ve çözünürlük parametrelerine bağlı olarak oluşturulması amacıyla tasarlanmıştır. Izgara yapısı kapsamında YPA

hesaplamasının temelini oluşturan ve Bölüm 4 kapsamında açıklanan Öklid mesafesine bağlı uzaklık ölçekleri GPU belleği üzerinde tüm alan için paralel olarak hesaplanmıştır. Böylece CPU belleğinden GPU belleğine transfer edilen veri miktarının azaltılarak, bellek transferinin performansa etkileri minimize edilmeye çalışılmıştır.

Algoritma 5. Step_array Çekirdek Algoritması

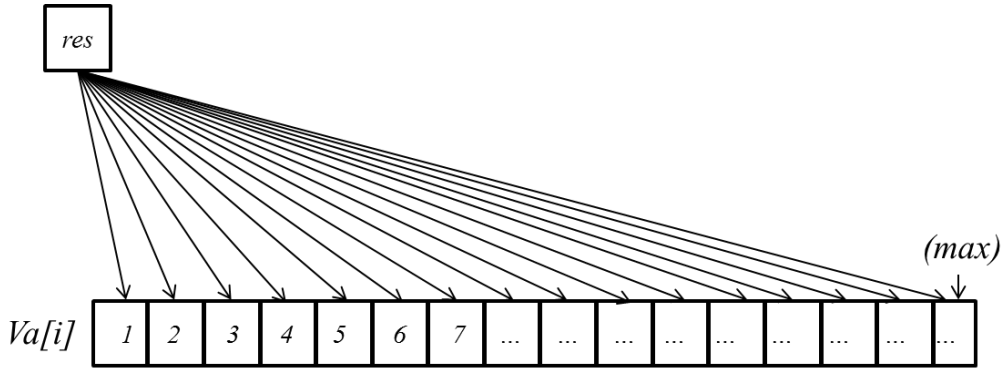
Girdi : : *x, y, res*
Çıktı : *Va* [ ]

5.1 İplik numarası değerini hesaplayarak bul
*int const i = blockDim.x * blockIdx.x + threadIdx.x;*

5.2 İplik numarası ve çözünürlük değerine bağlı olarak tek boyutlu A dizisinin değerlerini hesaplayarak ata

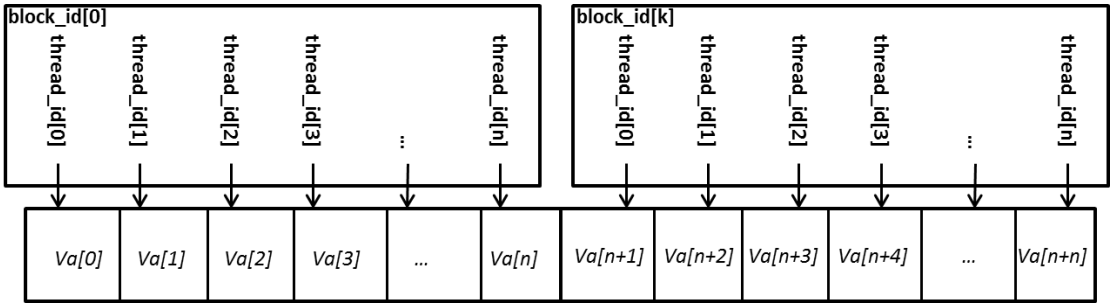
```
if  x < y
        max = y
else
        max = x
end if

if i<max
        Va[i] = i * res;
end if
```

Bu kapsamda ilk çekirdek kod bloğu olan *step_array* adı verilen kod bloğunun algoritması *Algoritma 5* ile gösterilmektedir. Algoritmadan da anlaşılacağı üzere belirlenen alan boyutları içerisinde tek boyutlu bir çözünürlük değerine bağlı mesafe dizisinin çok sayıda iplik ile üretilerek paralel biçimde ortaya konulması amaçlanmıştır.



Şekil 6.3: *step_array* çekirdek kodunun bellek erişimi gösterimi.

Şekil 6.3'den de görüleceği üzere algoritmanın doğal formu her ne kadar tüm iplikler ortak bellek alanı üzerinden okuma yapsa da, alanın çözünürlüğü değerinin bir tek boyutlu dizi şeklinde (örneğin iplik sayısına bağlı olarak) tekrar eden bir temsil ile bellekte saklanması neticesinde performans artışları kolayca elde edilecektir.



Şekil 6.4: *step_array* çekirdek kodunun yazma bellek erişimi.

step_array çekirdek algoritmasının yazma amaçlı bellek erişimi Şekil 6.4 ile gösterildiği biçimde olup, belleğe yazma biçimi GPGPU tabanlı paralel programlama mimarisi açısından algoritmanın temel formunda dahi düzenli olarak görülmektedir.

GPU donanımı üzerinde yapay potansiyel alanın hesaplanması amacıyla geliştirilen bir diğer çekirdek fonksiyonu ise *meshgrid* algoritmasıdır. İsminden de anlaşılacağı üzere bu algoritma GPU belleği üzerinde tanımlanan alan parametrelerine bağlı olarak iki boyutlu ızgara veri yapısının oluşturulması amacıyla geliştirilmiş ve *Algoritma 6* ile gösterilmiştir.

Algoritma 6. meshgrid Çekirdek Algoritması

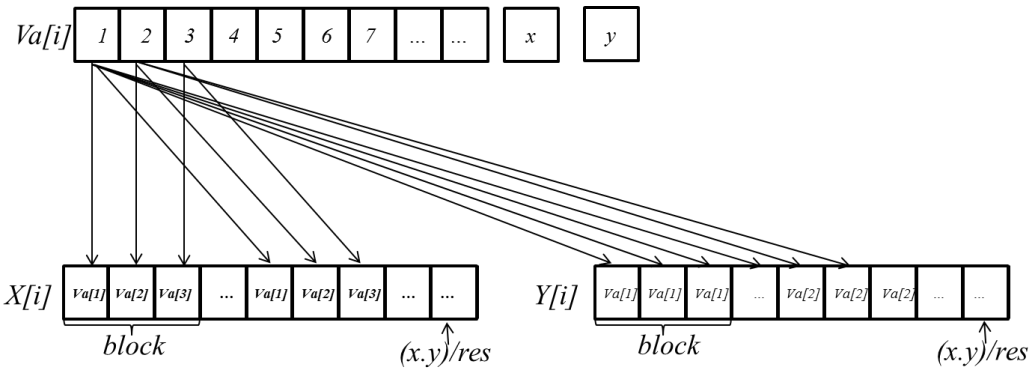
Girdi **:** *x, y, Va [ ]*
Çıktı **:** *X[ ], Y[ ]*

6.1 İplik numarası değerini hesaplayarak bul
*int const i = blockDim.x * blockIdx.x + threadIdx.x;*

6.2 *X,Y* iki boyutlu mesgrid dizilerini hesapla
X[round_lower((i-mod(i,x))/x)+1, mod(i,x)] = Va [mod(i,x)];
Y[round_lower((i-mod(i,x))/x)+1, mod(i,x)] = Va [round_lower((i-1)/x)+1];

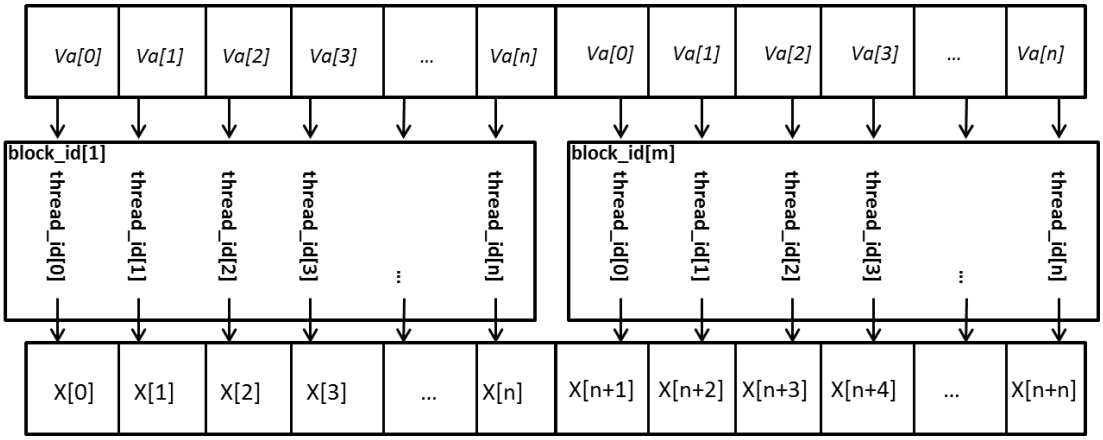
Otonom hareketin icra edileceği alanın çözünürlük ve ölçek değerlerine bağlı olarak ızgara yapısını ortaya koyan çekirdek kodu algoritması yukarıda gösterilmiştir. Izgara yapısı çözünürlük değerine bağlı olarak iki boyutlu mesafe matrisinin üretilmesi için kullanılır. Bu durum ızgara üzerinde yer alan iki konumum gerçekte

birbirleri ile ne mesafede yer aldığının kolaylıkla hesaplanabilmesi amacıyla geliştirilmiştir.



Şekil 6.5: *meshgrid* çekirdek algoritmasının bellek erişimi.

Blok ve blok içinde yer alan iplik sayısına bağlı olarak okuma ve yazma amaçlı belleğe erişim yapısı Şekil 6.5 ile gösterilmiştir. Paralel olarak koşturulan iplik sayısı her bir ipliğin bir hücre değeri hesapladığı göz önünde bulundurulduğunda alan dahilinde yer alan hücre sayısına eşit olacaktır. Ancak alan içerisinde yer alan hücre sayısı donanıma bağlı olarak oluşturulabilecek iplik sayısını geçmesi durumda algoritma benzer şekilde her bir ipliğin birden fazla hücre değerini hesaplayabileceği şekilde güncellenebilir.



Şekil 6.6: *meshgrid* algoritmasının memory coalsing biçimde bellekten okuyup yazmasına olanak sağlayan bellek mimarisi.

meshgrid çekirdek algoritmasının memory coalesing biçimde belleğe erişim sağlaması amacıyla Şekil 6.6 ile gösterilen biçimde güncellenmesi mümkündür. Ancak bu durum öncelikle GPU donanımı üzerinde yer alan bellek miktarına bağlıdır. Zira bu tip bir erişim için blok sayısı kadar giriş veri dizinin parçalanmasına ihtiyaç duyulacaktır.

Algoritma 7'de yer alan *vector_field* algoritmasının girdi değerleri sırasıyla alt bileşenleri oluşturan ve Çizelge 3.1 ile açıklanan temel modellere ait temel alan tipi, temel alan parametreleri ve sınır değer parametre dizileridir.

Algoritma 7. vector_field Hücre Bazlı Hesaplama Çekirdek Algoritması

Girdi :$x,y,res,A_{type}[\,]$, $A_{\alpha}[\,]$, $A_{x}[\,]$, $A_{y}[\,]$, $A_{\varphi}[\,]$, $A_{\gamma}[\,]$, $X[\,]$,$Y[\,]$, $A_{slope}[\,]$, $A_{\rho}[\,]$

Çıktı : *SX[ ], SY[ ], Heading[ ], Magnitude[ ]*

7.1 İplik numarası değerini hesaplayarak bul
*int const i = blockDim.x * blockIdx.x + threadIdx.x;*

7.2 *i* iplik değeri ile temsil edilen noktanın vektör alan bilşen değerlerini sıfırla
x=ceil(x/res);
y=ceil(y/res);
a=floor((i-mod(i,x))/x)+1;

7.3 *b=mod(i,x);*
SX(a,b) = 0; SY(a,b) = 0;

7.4 İplik tarafından hesaplanan (a,b) noktasındaki toplam sınırlandırılmış vektör

7.5 değeri hesapla

7.6 ***for*** *s = 1:length*(A_{type})

if ($A_{type}(s)$) $= C_{rot}$

$DX_{C_{rot}(s)}(a,b)$ değerini Denklem (3.21) ile hesapla
$DY_{C_{rot}(s)}\ (a,b)$ değerini Denklem (3.22) ile hesapla
$S_C^+(a,b)$ değerini Denklem (3.37) ile hesapla
$SX(a,b) = SX(a,b) + DX_{C_{rot}(s)}(a,b).S_C^+(a,b)$

7.7 $SY(a,b) = SY(a,b) + DY_{C_{rot}(s)}(a,b).S_C^+(a,b)$

elseif ($A_{type}(s)$) $= I_{rot}$

$DX_{I_{rot}(k)}(a,b)$ değerini Denklem (3.22) ile hesapla
$DY_{I_{rot}(k)}\ (a,b)$ değerini Denklem (3.24) ile hesapla
$S_I^-(a,b)$ değerini Denklem (3.37) ile hesapla
$SX(a,b) = SX(a,b) + DX_{I_{rot}(s)}(a,b).S_I^-(a,b)$

7.8 $SY(a,b) = SY(a,b) + DY_{I_{rot}(s)}(a,b).S_I^-(a,b)$

elseif ($A_{type}(s)$) $= R_p$

$DX_{R_p(s)}(a,b)$ değerini Denklem (3.29) ile hesapla
$DY_{R_p(s)}\ (a,b)$ değerini Denklem (3.31) ile hesapla
$S_R^-(a,b)$ değerini Denklem (3.37) ile hesapla
$SX(a,b) = SX(a,b) + DX_{R_p(s)}(a,b).S_R^-(a,b)$

7.9 $SY(a,b) = SY(a,b) + DY_{R_p(s)}(a,b).S_R^-(a,b)$

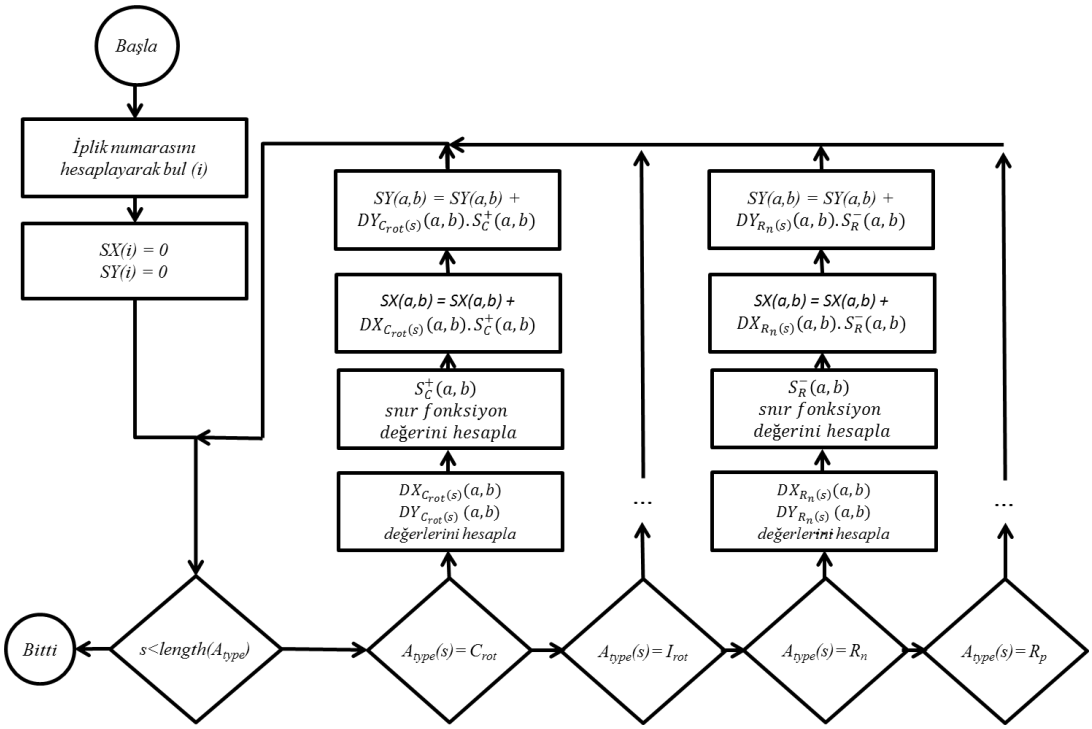
elseif ($A_{type}(s)$) $= R_n$

$DX_{R_n(s)}(a,b)$ değerini Denklem (3.30) ile hesapla
$DY_{R_n(s)}\ (a,b)$ değerini Denklem (3.32) ile hesapla
$S_R^-(a,b)$ değerini Denklem (3.37) ile hesapla
$SX(a,b) = SX(a,b) + DX_{R_n(s)}(a,b).S_R^-(a,b)$
$SY(a,b) = SY(a,b) + DY_{R_n(s)}(a,b).S_R^-(a,b)$

7.10 end if
7.11 end for

7.12 İplik tarafından hesaplanan (a, b) noktasındaki sürat ve yön tablo değerlerini hesapla
Heading (a,b) değerini Denklem (3.41) ile hesapla
Magnitude (a,b) değerini Denklem (3.42) ile hesapla

Potansiyel alanının hesaplanması amacıyla kullanılan son çekirdek kodu algoritması da *Algoritma 7* ile gösterilen *vector_field* algoritmasıdır. Kod bloğundan da görüldüğü üzere bu paralel algoritmanın amacı *meshgrid* sonucunda ortaya konulan ızgara yapısındaki her bir hücrede oluşan toplam potansiyel alanı, tanımlanan "*t*" sayıdaki temel potansiyel kaynak modelinin toplam etkisi biçiminde hesaplamaktır. Ayrıca gradient dönüşümü ile elde edilen vektörel değerlerin bileşenlerinden hücre içerisinde üretilecek olan uçuş komutalarının üretilebilmesine olanak sağlamaktır.

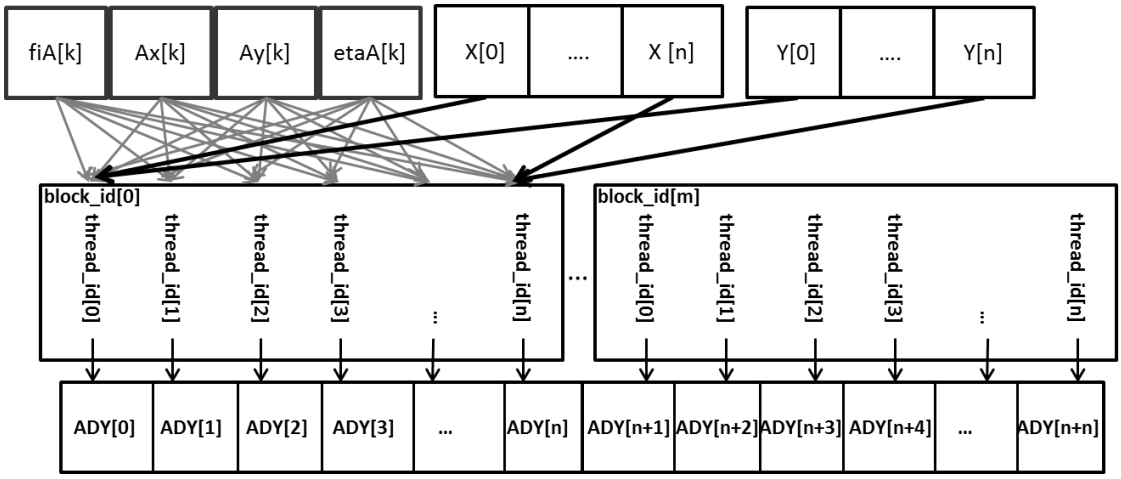


Şekil 6.7: *Vector_field* algoritması akış diyagramı

Şekil 6.7 ile de gösterildiği üzere, *vector_field* çekirdek kodu içerisinde yer alan döngü ile hücre üzerinde etki eden sınırlı temel potansiyel alan model sayısı kadar tekrar eden bir yapı elde edilmiştir. Bu döngü yapısının ardından temel potansiyel alan model tipine bağlı olarak dallanan bir karar noktası oluşturulmuştur.

Akabinde ilgili hücre üzerinde etki eden her bir sınırlı temel potansiyel alan modelinin vektörel değeri hesaplanarak toplam vektörel değere eklenmiş ve hücre

üzerindeki toplam vektörel değer elde edilmiştir. Algoritmanın sonuna Denklem (3.46) ve (3.47) eklenerek doğrudan aynı çekirdek kod yapısı üzerinde her bir hücrede oluşan uçuş komutalarının üretilmesi sağlanabilir. Şayet hücre sayısı algoritmanın koşturulduğu donanımın imkân verdiği sistemde üretilebilen toplam iplik sayısından az ise her bir çekirdek bir hücreyi, şayet fazla ise bir çekirdek birden fazla hücre değerini hesaplayacak şekilde güncellenebilir.



Şekil 6.8: *vector_field* çekirdek algoritmasının bellek erişimi gösterimi.

Şekil 6.8 ile *vector_field* algoritmasının bellek erişim paternleri gösterilmiştir. Algoritmanın okuma ve yazma bellek erişiminde performans açısından dar boğaz yaratacak bir etken görülmemektedir.

6.1.2 Temel alan bazlı paralel hesaplama yaklaşımı

Platformların otonom olarak hareket etmek istediği ortam içerisinde, platformların hareketine dolayısıyla doğrudan istenilen hedef noktaya ulaşmalarına engel teşkil edebilecek çok sayıda engel yer alabilir. Bu engeller ortam içinde yer alan tabii sütreler veya alan dahilinde hareket eden diğer araçlar olabilir. Tüm bunların ötesinde görev HYİ görevinde olduğu gibi özel bir patern icra edilmesini ve dolayısıyla bu hareketi modelleyebilmek için birden fazla temel potansiyel alan modelinin bir arada doğru şekilde sınırlandırılması sonucu kullanılmasına gereksinim duyabilir. Bu durumda toplam potansiyel alan içinde "*m*" adet çeken temel alt bölge, "*n*" adet iten temel alt bölge, "*p*" adet saat yönünde dönen alt bölge ve de "*r*" adet saat yönünün tersinde dönen alt bölge olduğu kabul edilebilir. Bunların toplam miktarının "*t*" alan içinde yer alan ızgara nokta sayısı ile kıyaslandığında yakın bir değer olduğu kabul edildiğinde her bir temel potansiyel alan değeri ve üretilen vektör

değerleri bir çekirdek vasıtasıyla hesaplanabilir. Çekirdeklerin ürettikleri sonuçlar tekrar toplam vektör değerlerin elde edilmesi amacıyla Denklem **(3.39)**'den faydalanarak hesaplanabilir.

Örneğin Şekil 5.13 ile gösterilen senaryo dahilinde alan içerisinde üç adet engel modellemesi olmak üzere toplam *11* adet temel potansiyel alan modelinden faydalanılmıştır. Bu durumda hiç engel bulunmayan alan içerisinde en az sekiz adet temel potansiyel alan modelinden faydalanılması gerekmektedir. Bir başka açıdan durum ele alınır ise alan içerisinde yer alan çok sayıda engel olduğu kabul edildiğinde, yüzlerce temel potansiyel alan modelinden aynı anda faydalanılması gerekecektir. Bu durumda temel potansiyel alan bazlı paralel hesaplama yaklaşımının etkin olarak kullanılabileceği değerlendirilmektedir. Ancak tam aksine hiç engel bulunmayan bir alan içinde ise, temel alt modellerin sayısı formasyon dahilinde yer alan platform sayısı ile sınırlı olacağından yaklaşım GPU kaynaklarının israfı ile sonuçlanabilir.

Hücre bazlı paralel hesaplama yaklaşımında olduğu gibi *Algoritma 4* ile belirlenen çekirdek koşturma organizasyonuna uygun şekilde çağrı yapılır ve *Algoritma 5* ve 6'da yer alan algoritmalar koşturularak step vektörü ile meshgrid üretilir.

Algoritma 8. Çekirdek Koşturma Parametrelerinin Belirlenmesi Algoritması

Girdi **:** *x, y* iki boyutlu alan büyüklüğü, *res* çözünürlük değeri
Çıktı **:** *threadsPerBlock, blocksPerGrid* çekirdek koşturma parametreleri

8.1 İki boyutlu alan içindeki temel alan sayısı değerini bul ve max değerine ata
max = t

8.2 Grafik işlemci donanım özelliklerine bağlı olarak koşturulacak paralel iplik ve blok sayısını belirle
if *(max<MaxThreadsPerBlock)*
{threads = max;}
else
{threads = MaxThreadsPerBlock; }
end if
blocks = (max + threads – 1) / threads;

8.3 Çekirdek fonksiyon koşturma parametrelerini belirle ve fonksiyon çağrılarını gerçekleştir
dim3 threadsPerBlock (threads,1,1);
dim3 blocksPerGrid (blocks,1,1);

8.4 *vector_field2 <<<blocksPerGrid, ThreadsPerBlock>>>(*A_{type}[ ]**,** A_{α}[ ], A_x[ ], A_y[ ], A_{φ}[ ], A_{γ}[ ], *X[ ],Y[ ],* A_{slope}[ ] , A_{ρ}[ ]*)*

Bu durumda hücre bazlı paralel hesaplamada vektör alanının hesaplanması için çekirdek koşturma organizasyonun belirlenmesi amacıyla faydalanılan *Algoritma 4* yerine aşağıda yer alan *Algoritma 8*'den faydalanılacaktır. Çünkü koşturulacak olan çekirdek sayısı toplam alan dahilinde yer alan temel alan sayısına bağlı olarak yani "t" parametre değeri ile bulunabilir.

Algoritma 9. Temel Alan Bazlı Hesaplama Çekirdek Algoritması

Girdi : *x, y, res*, alt alan parametreleri
Çıktı : *SX(i), SY(i)*

İplik numarası değerini hesaplayarak bul
9.1 *int const i = blockDim.x * blockIdx.x + threadIdx.x;*

i iplik değeri ile temsil edilen temel sınırlandırılmış vektör alan bilşen
9.2 değerlerini hesapla
for *a* ← *1 to (x/res)*
9.3 ***for*** *b* ← *1 to (y/res)*
9.4 ***if*** $(A_{type}(i)) = C_{rot}$
9.5 $DX_{C_{rot}(k)}(a,b)$ değerini Denklem (3.21) ile hesapla
$DY_{C_{rot}(k)}(a,b)$ değerini Denklem (3.22) ile hesapla
$S_C^+(a,b)$ değerini Denklem (3.37) ile hesapla
$SX_i(a,b) = DX_{C_{rot}(s)}(a,b).S_C^+(a,b)$
$SY_i(a,b) = DY_{C_{rot}(s)}(a,b).S_C^+(a,b)$
elseif $(A_{type}(i)) = I_{rot}$
9.6 $DX_{I_{rot}(k)}(a,b)$ değerini Denklem (3.22) ile hesapla
$DY_{I_{rot}(k)}(a,b)$ değerini Denklem (3.24) ile hesapla
$S_I^-(a,b)$ değerini Denklem (3.37) ile hesapla
$SX_i(a,b) = DX_{I_{rot}(s)}(a,b).S_I^-(a,b)$
$SY_i(a,b) = DY_{I_{rot}(s)}(a,b).S_I^-(a,b)$
elseif $(A_{type}(i)) = R_p$
9.7 $DX_{R_p(k)}(a,b)$ değerini Denklem (3.29) ile hesapla
$DY_{R_p(k)}(a,b)$ değerini Denklem (3.31) ile hesapla
$S_R^-(a,b)$ değerini Denklem (3.37) ile hesapla
$SX_i(a,b) = SX(a,b) + DX_{R_p(s)}(a,b).S_R^-(a,b)$
$SY_i(a,b) = SY(a,b) + DY_{R_p(s)}(a,b).S_R^-(a,b)$
elseif $(A_{type}(i)) = R_n$
9.8 $DX_{R_n(k)}(a,b)$ değerini Denklem (3.30) ile hesapla
$DY_{R_n(k)}(a,b)$ değerini Denklem (3.32) ile hesapla
$S_R^-(a,b)$ değerini Denklem (3.37) ile hesapla
$SX_i(a,b) = SX(a,b) + DX_{R_n(s)}(a,b).S_R^-(a,b)$
$SY_i(a,b) = SY(a,b) + DY_{R_n(s)}(a,b).S_R^-(a,b)$
9.9 ***end if***
9.10 ***end for***
9.11 ***end for***

Temel alan bazlı paralel hesaplama yaklaşımı ile hücre bazlı yaklaşımdan farklı olarak ilave iki çekirdek algoritmasından faydalanılmıştır. Bu durumda hesaplama için faydalanılan algoritma *vector_field2* olarak adlandırılmış ve *Algoritma 9* ile ortaya konulmuştur. "*k*" parametresi alt alan indeks değerini temsil etmektedir.

Toplam alan vektör bileşen değerlerinin hesaplanması ve bu değerlerden de heading ve magnitude tablolarının üretilmesi yine *Algoritma 4* ile ortaya konulan bir iplik organizasyonu dahilinde ikinci bir *Algoritma 7*'de yer alan *7.12* ile *7.13* adımlarında yer alan komutların benzer biçimde koşturulması ile gerçekleştirilebilir.

6.2 Paralel Hesaplama Yaklaşımlarının Karşılaştırılması

GPGPU tabanlı YPA hesaplanmasına yönelik paralel hesaplama yaklaşımlarının karşılaştırmasına geçmeden önce grafik işlemci üzerinde gerçekleştirilen hesaplamalarda karşılaşılacağı değerlendirilen sistem kısıtlarına değinmekte fayda vardır.

CUDA ile programlanmış ve Bölüm *2.5.2* kapsamında özellikleri tanımlanın Compute Capability değeri 2.0'dan büyük NVIDIA grafik işlemciler üzerinde koşturulacak bir uygulama için blok başına tanımlanabilecek iplik sayısı (*maxThreadsPerBlock*) en fazla 1024 olacaktır. Bu durumda şayet bir blok içinde yer alan her bir ipliğin bir potansiyel alan hücre değerini hesapladığı ön görülür ise bir blok ile hesaplanabilecek alan büyüklüğü en fazla 32x32 ölçeğinde olacaktır.

Bu durumda bu ölçekten büyük alanların hesaplanması istendiğinde blok sayısını arttırmak gerekecektir. CUDA programlama ortamında ve yine Bölüm *2.5.2* kapsamında özellikleri tanımlanan Compute Capability değeri 2.0'dan büyük NVIDIA grafik işlemciler üzerinde bir uygulama geliştirilmek istendiğinde üç boyutlu değerleri en fazla [1024 1024 64] şeklinde tanımlanabilecek en fazla blok sayısı 65535 olacaktır. Bu durumda şayet bir blok içinde yer alan her bir hücrenin bir potansiyel alan hücre değerini hesapladığı ve her bir blok içerisinde en fazla iplik değeri olarak belirlenen 1024 iplik olduğu ön görülür ise bir grid ile hesaplanabilecek alan büyüklüğü en fazla 4.194.240 x 4.194.240 ölçeğinde (32x32 x 65535) olacaktır.

Ancak bu durum maalesef mümkün değildir. Çünkü GPGPU tabanlı programlama esnasında ihtiyaç duyulacak bir diğer kaynak ise bellek miktarıdır. CUDA programlama ortamında ve yine Bölüm 2.5.2 kapsamında özellikleri tanımlanan

Compute Capability değeri 2.0'dan büyük NVIDIA grafik işlemciler üzerinde bir uygulama geliştirilmek istendiğinde blok başına düşen bellek miktarı (paylaşılan bellek miktarı) 48 KB ile sınırlıdır. Bu durumda 32 x32 ölçeğinde bir alanın yani 1024 YPA ızgara hücresinin hesaplanabilmesi için ihtiyaç duyulan bellek miktarının incelenmesi gerekmektedir. Bu durumda Algoritma 7 için girdi ve çıktı değerlerinin yorumlanması gerekir. Şayet bellekteki veri temsilini single float tipinde yaptığımızı kabul edersek bu durumda her bir temsil için 4 byte alan ihtiyacı oluşacaktır. Bu durumda dört adet alt potansiyelden oluşan vektör alanın hesaplanması için girdi verilerinin toplam bellekte kaplayacakları alan yaklaşık 0,5 KB olacaktır. Aynı şekilde çıktı verilerinin bellekte kaplayacağı alan ise bu ölçekte bir vektör alan için toplam 16 KB olarak gerçekleşecektir. Bu durumda toplam bellek ihtiyacının 16,5 KB ile yeterli olacağını söylemek mümkündür.

Ancak bir diğer durum grid için ihtiyaç duyulan bellek alanıdır ki bu da grafik işlemci üzerindeki global bellek miktarı ile sınırlanmaktadır. Bu durum çalışmanın ilerleyen kısımlarında ayrıntılı olarak ele alınmıştır.

Çizelge 6.1: Alan ölçeğine bağlı olarak iplik ve blok planlaması.

Alan	Maks. BlockDim	Maks. BlockId.x	Maks. ThreadId.x
500	1	245	1024
1000	1	980	1024
2000	4	978	1024
3000	9	977	1024
10000	96	1018	1024

Bu bilgiler ışığında hücre bazlı ve temel alan bazlı hesaplama karşılaştırmasına geri dönersek her bir iplikte toplam alan dahilinde yer alan bir temel alanın etkisini ölçek için paylaşılan bellekte girdi parametrelerinin kapladığı alanı göz ardı ettiğimizde sadece çıktı parametresi olarak alan ölçeğinin yaklaşık dört katı bellek ihtiyacı oluşacaktır. Bu durumda 48 KB ile sınırlı olan paylaşılan bellek alanını yani 6*(32x32) = (192x192) ölçeğinin üzerindeki alanların hesaplanması için bir üst seviye bellek alanına (global bellek) erişim ihtiyacı oluşacak ve de performans kayıpları yaşanacaktır. Bu durumda bellek erişimi açısından hücre bazlı hesaplamanın temel alan bazlı hesaplamaya göre oldukça avantajlı olduğunu söylemek mümkündür.

Bir başka problem oluşturulacak iplik sayısından dolayısıyla sistem kaynaklarının ne derece verimli kullanıldığından kaynaklanmaktadır. Çizelge 6.1 ile hücre bazlı hesaplama uygulandığında farklı ölçeklerdeki ve çözünürlük değeri "1" olarak kabul edilen vektör alan değerlerinin hesaplanması için faydalanılacak iplik, blok ve grid parametreleri gösterilmektedir. Her bir blok içinde yer alan iplik sayısı 1024 olarak sabit tutulmuştur. Bunun nedeni warp (32 iplik) şeklinde bir arada planlanan iplik sayısının katı olması ve en fazla paylaşılan bellekten faydalanılacak blok içindeki en yüksek belirlenebilen değer olmasıdır. Blok içinde tanımlanabilen iplik sayısına bağlı olarak blok miktarı ve boyut değeri alan ölçeğine bağlı olarak belirlenmiştir. Blok parametreleri şu şekilde hesaplanmıştır:

```
Alan içindeki hücre sayısı = Alan x Alan
Maks_BlokId.x = Alan içindeki hücre sayısı /Maks_ThreadId.x
if Maks_BlokId.x>1024
        Maks_BlokDim = ceil(Maks_BlokId.x/1024)
        Maks_BlokId.x = Maks_BlokId.x/Maks_BlokDim
end if
```

Yol planlaması yapılmak istenen alan içinde yer alan engel sayısı geçerli yolların üretilebilmesi için alan içindeki hücre sayısına kıyasla daha küçük bir rakam olacaktır. Ayrıca alan içinde yer alan engel sayısı arttıkça ihtiyaç duyulan bellek miktarı artacaktır.

Sonuç olarak GPGPU tabanlı paralel programlama performansının belirlenmesinde donanım özelliklerine bağlı olarak öne çıkan iki etken vardır; iplik organizasyonu ve bellek organizasyonu. Amaçlanan mümkün olan en fazla sayıda iplik ile mümkün olan en alt seviyedeki erişimi en hızlı olan (on-chip) bellek mimarisinden faydalanarak hesaplamayı gerçekleştirmektir. Bu durumda uygun hesaplama algoritmasının hücre bazlı hesaplama algoritması olduğu ortadadır. Her ne kadar hücre bazlı paralel hesaplama yaklaşımında alan içinde yer alan temel potansiyel alan sayısı hesaplama performansını etkilese dahi, önemli derecede performansı etkileyen yapı alan ölçek değeridir. Bir başka deyim ile hücre bazlı hesaplama yaklaşımında alan ölçeği performansın belirleyici ana unsurudur. Çünkü paralel hesaplama performansını etkileyen esas faktör bellek erişim maliyetidir. Bu durum çalışmanın ilerleyen bölümlerinde farklı senaryolar dahilinde ele alınmıştır

7. ENGELLER İÇEREN ALAN İÇİNDE YPA TABANLI YOL PLANLAMASI

Formasyonu teşkil eden platformlar çok sayıda engel içeren bir alan içinde seyrüsefer icra edebilirler. Çalışma kapsamında ortaya konulan yaklaşım platformların alan içinde yer alan engellerden sakınarak formasyonu olabildiğince muhafaza etmelerine imkan sağlayacak özellikte olmalıdır. Bu nedenle çalışmanın bu bölümünde alan içinde yer alan engellerin nasıl modelleneceğine, YPA tabanlı yol planlaması kapsamında nasıl engel olarak temsil edileceklerine ve yol planlamasının gerçek zamanlı olarak ele alınabilmesi için grafik tabanlı işlemciler vasıtasıyla nasıl paralel olarak hesaplanabileceklerine değinilmiştir.

7.1 Engellerin Modellenmesi

Gerek formasyon yapısı içerisinde gerekse de yalnız uçuş gerçekleştiren bir otonom İHA platformunun içinde bulunduğu coğrafi ortam dahilinde, görevinin icrasına engel olabilecek nitelikte doğal veya yapay engeller bulunabilir. Etkin bir otonom yol planlaması yaklaşımı bu engeller karşısında tepkiler üretebilmeli ve de bu engellere rağmen en optimum şekilde bir çözüm üretebilmelidir. Bu kapsamda ortaya konulmak istenen otonom mimari gerçek zamanda engellerden sakınarak görevin icrasına imkan tanıyacak özellikleri bünyesinde barındırmalıdır.

Özellikle formasyon uçuşu gibi yalnız görev icra eden otonom platformlara nazaran daha yüksek karmaşık derecesinde bir yol planlamasına ihtiyaç duyan sistem içinde üç önemli amaç tanımlanabilir;

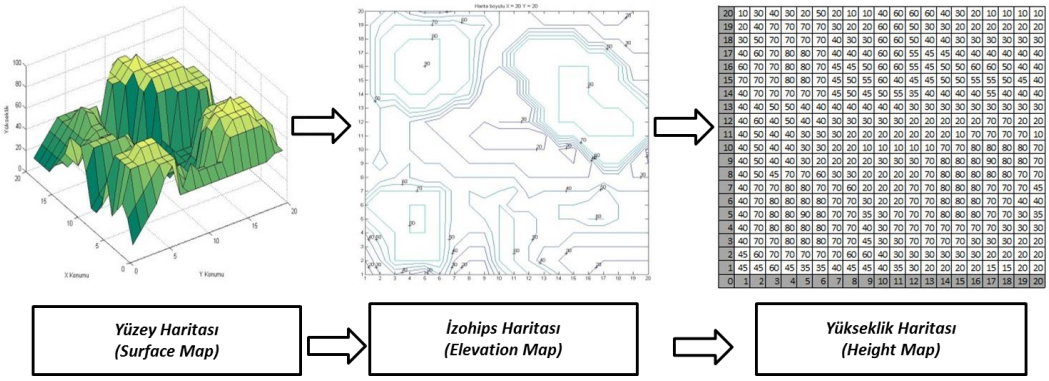
a. Görevin icrası amacıyla gerçek zamanlı formasyon dahilinde doğru konumlandırmaya imkân tanıyacak;

b. Ortam içerisinde yer alan doğal veya yapay engellerden kaçınacak;

c. Formasyon dahilinde yer alan veya almayan ancak varlığı teşhis edilebilen ve çarpışma potansiyeli olduğu değerlendirilen hareketli, dinamik diğer platformlar ile çarpışmaya engel olacak gerekli önlemleri alacak nitelikte olmalıdır.

Her ne kadar bu tez kapsamında amaçlanan daha önceden özellikleri bilinmeyen bir alan içinde formasyon uçuşunun yapılması olsa dahi, çalışma kapsamında platformlar üzerinde engellerin tespiti amacıyla kullanılan algılayıcı donanımlarının modellenmesine ve yeteneklerinin uygulanmasına değinilmemiştir. Durum araştırma kısıtı olarak çalışmanın Bölüm 1.5 başlığı altında belirtilmiştir. Bu ihtiyaç en temel şekliyle, platforma belirli bir mesafade yer alan engellerin, araç üzerinde yer alan algılayıcılar vasıtasıyla iki boyutlu olarak algılanabildiği kabulü üzerine temsilen oluşturulmuştur.

Çalışmanın bundan sonraki bölümlerinde engel algılanması ile kastedilen, sayısal harita üzerinde önceden varolduğu bilinen engellerin, sadece platforma belirli bir mesafede olanlarının yol planlamasını doğrudan etkilediği kabulü üzerine kurulmuştur. Platform alan içinde hareket ettikçe yol planlamasına etki eden engeller buna göre dinamik olarak değişiklik göstermektedir.



(a) Yüzey haritası gösterimi. (b) İzohips (eş yükselti) haritası gösterimi. (c) Sayısal yükseklik haritası gösterimi.

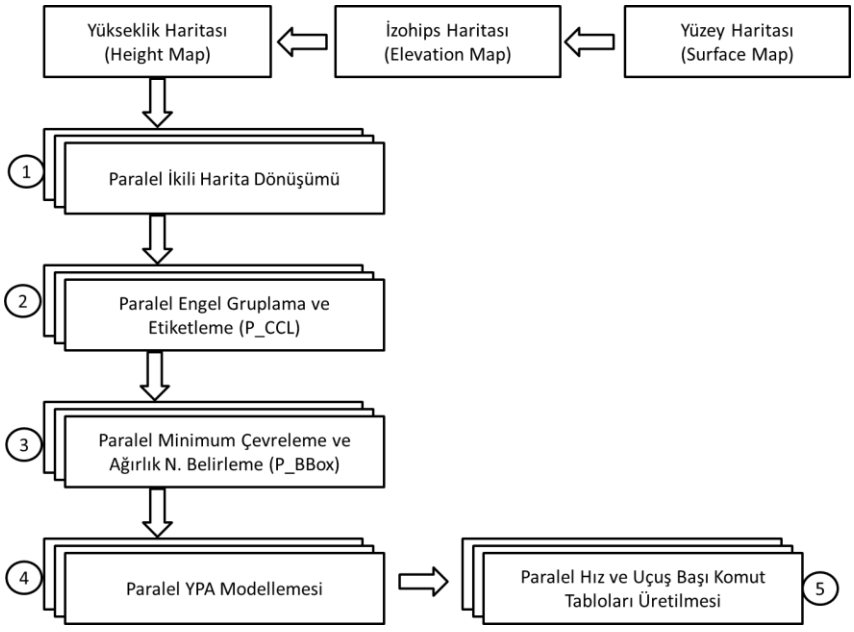
Şekil 7.1: Üç boyutlu sayısal engel temsil yapıları ve birbirleri arasında dönüşümü.

Şekil 7.1 ile gösterildiği gibi görevin icrası esnasında içinde bulunulan sınırlı alan dahilindeki doğal engeller değişik şekilde sayısal ortamda temsil edilebilir. Bu engeller genellikle sınırlandırılmış üç boyutlu görev alanı içerisinde yer alan ve platformun icra ettiği görevin gereksinim duyduğu uçuş irtifasının veya platformun yetenekleri dahilinde erişebildiği uçuş tavan irtifasının üzerinde yer alan tepe ve dağlar olarak isimlendirilebilir. Doğal olmayan üç boyutlu engellerin de (bina, kule, girilmesi istenmeyen tehdit alanı vb.) doğal engellerin temsiline benzer şekilde sayısal olarak ifade edilebilir.

Engel sistemleri sayısal ortama Şekil 7.1 (a) ile gösterilen biçimde üç boyutlu yüzey haritaları olarak aktarılabilir. Bu sayısal haritalar, eş yükselti eğrileri yardımıyla iki

boyutlu olarak Şekil 7.1 (b) ile gösterildiği şekilde bir dönüşüme tabi tutulabilirler. Ancak genellikle bu iki gösterimde görsel olarak engellerin daha iyi kavranmasına imkân sağlarken sayısal işlemlerin icrası açısından etkin değildirler. Bu nedenle izohips haritalar iki boyutlu sayısal diziler olarak ifade edilen ve Şekil 7.1 (c) ile gösterilen yükseklik haritalarına dönüştürülürler. İki boyutlu yükseklik haritaları tıpkı izohips ya da yüzey haritalarında olduğu gibi üç boyutlu bir çözünürlük değerine bağlı olarak temsil edilirler. Enlem ve boylam derece çözünürlüğüne bağlı olarak hücrelere bölünmüş bir alan içerisinde yer alan deniz seviyesinden yüksekliğin ortalama değerini veya hücre alanı içerisinde yer alan en yüksek değeri temsil edebilirler.

Özellikle İHA benzeri otonom platformlar, bu tip sayısal haritaları kullanarak görev icra ettikleri ortam dahilinde yer alan ve kendilerine engel teşkil eden yükseltilere karşı şayet olası bir yaklaşım mevcut ise çözüm üretebilirler. Bu noktada yapılması gereken işlem öncelikle ortam dahilinde engel teşkil eden yapıların tespit edilmesi ve belirli standartlar dahilinde temsil edilmek üzere modellenmeleridir.



Şekil 7.2: Üç boyutlu sayısal engellerin tespit ve temsili amaçlı gerçekleştirilen işlemlerin akış diyagramı.

Şekil 7.2 ile gösterilen akış diyagramında görüldüğü üzere çalışma kapsamında geliştirilen otonom yol planlaması mimarisinin giriş verilerinden birisi de yükseklik haritasıdır. Engellerin tespit edilmesi ve modellenebilmeleri amacıyla bir dizi işlem sırasıyla sayısal haritalar üzerinde uygulanır. Bu işlemler gerçek zamanlı çözüm

şeklinde ortaya konulmak amacıyla akış diyagramında da gösterildiği şekilde paralel algoritmalar kullanılarak gerçekleştirilecektir.

7.1.1 Paralel ikili harita dönüşümü

Gerçekleştirilen ilk işlem paralel ikili harita dönüşümüdür. İkili harita ile kasıt ikili (binary) sayılar ile temsil edilen haritalardır. İkili haritalar için iki karar verisi mevcuttur;

a. Coğrafi koordinatları verilen konum içerisinde yer alan yükseklik otonom sistem için engel teşkil ediyor ise bu noktanın değeri "*1*" olarak,

b. Şayet engel teşkil etmiyor, görevin icrasına etkisi yok ise bu noktanın değeri "*0*" olarak temsil edilir.

Yükseklik haritasının ikili haritaya dönüştürülebilmesi için bilinmesi gereken bilgi hangi yükseklik değeri limitinin platform için engel teşkil ettiğidir. Bu değere "*treshold*" eşik değeri adı verilir. Özetle yükseklik haritası içinde yer alan yükselti verilerinden eşik değerine eşit ve üzerinde yer alan hücreler "*1*", bu değerin altında kalan hücreler ise "*0*" ile temsil edilir. İkili haritaları kullanmanın avantajları şu şekilde sıralanabilir;

a. Sayısal bellekte temsil edilebilmeleri diğer harita yapılarına göre daha etkindir ve daha az bellek alanına ihtiyaç duyarlar,

b. Otonom yol planlamasında problemin çözümü için bir yol olup olmadığının anlaşılması daha kolaydır,

c. Sparse ("*sıfır*") diziler şeklinde gösterilmeleri nedeniyle matematik işlemlerin uygulanması açısından daha hızlı sonuçlar üretebilirler.

Sayısal bir yükselti haritasının ikili bir harita şekline dönüştürülmesi basit ancak harita boyutlarına ve çözünürlük değerlerine bağlı olarak oldukça zaman alabilecek bir işlemdir. Gerçek zamanlı olarak bu dönüşümün yapılması için paralel işlem imkânı sağlayan donanımlar üzerinde paralel algoritmalar geliştirilebilir.

Algoritma 4 benzer şekilde engellerin modellenmesi amacıyla paralel koşturacak iplik ve blokların parametrelerinin belirlenmesi amacıyla kullanılabilir. "*H*" parametresinin tek boyutlu temsil edilen giriş verisini temsil ettiği, bu vektör değişkenin uzunluğunun yükseklik haritasının içinde yer alan hücre sayısına bağlı

olarak değiştiği kabul edilebilir. GPGPU tabanlı paralel programlama mimarisi içinde oluşturulacak iplik ve blok sayısına karar vermek amacıyla Algoritma 4'de faydalanılan alan ölçeğine benzer şekilde "*H*" verisinin kullanılabileceği ortadır. Bilindiği üzere bu değerler donanıma özgü değerler olup, kodun koşturulduğu işlemci platformuna bağlı olarak maksimum iplik ve blok sayıları değişkenlik gösterebilirler.

Algoritma 10'da yer alan çekirdek kodu algoritması incelendiğinde, her bir sayısal harita üzerinde yer alan hücre için bir iplik üretildiği kabul edildiğinde üretilen her bir ipliğin koşturduğu çekirdek kodu içinde ise basit bir mantıksal karşılaştırma ile ikili harita "*B*" dönüşümünün yapıldığı görülmektedir.

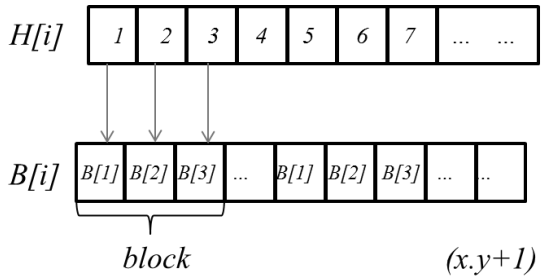
Algoritma 10. Paralel ikili harita dönüşümü çekirdek kodu

girdi: *H[ ], threshold*
çıktı: *B[ ]*

10.1: *thread id* değerini hesapla
*int const i = blockDim.x * blockIdx.x + threadIdx.x;*

10.2: Yükselti haritasını ikili haritaya dönüştür
10.3: ***if*** *H[i] < threshold*
B[i] = 0
10.4: ***else***
B[i] = 1
10.5: ***end if***

Algoritma 10, GPGPU tabanlı paralel programlama uygulamalarının performansını etkileyen unsurlar arasında yer alan bellek erişimi açısından değerlendirildiğinde Şekil 7.3 ile temsil edildiği biçimde her bir çekirdek kodu farklı bir veri alanından okuma yapar ve de sonucunu farklı bir veri alanına yazar.



Şekil 7.3: Paralel ikili harita dönüşümü çekirdek kodunun okuma-yazma amaçlı bellek erişimi temsili.

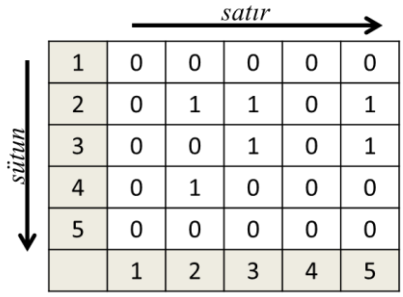
Okunan sayısal harita verileri "*H*" olarak adlandırılan tek boyutlu akış tipi işlemlere uygun veri mimarisi olan vektör üzerinde tutulmaktadır. Bu diziye çekirdek kod

bloğu okuma amacıyla erişir. Çekirdek kodu sonucu yine aynı şekilde "*B*" olarak adlandırılmış farklı bir vektör üzerine yazar. Okuma ve yazma amaçlı vektör üzerindeki hangi bellek alanına erişeceğine eşsiz bir değer olan iplik numarasına (*thread_id*) bağlı olarak karar verir.

Algoritma 10'da yer alan çekirdek kodunun paralel şekilde "*H*" sayısal harita verisi üzerinde koşturulması neticesinde ikili değerler ile temsil edilen "*B*" sayısal harita verisi üretilecektir.

7.1.2 Paralel engel gruplama ve etiketleme

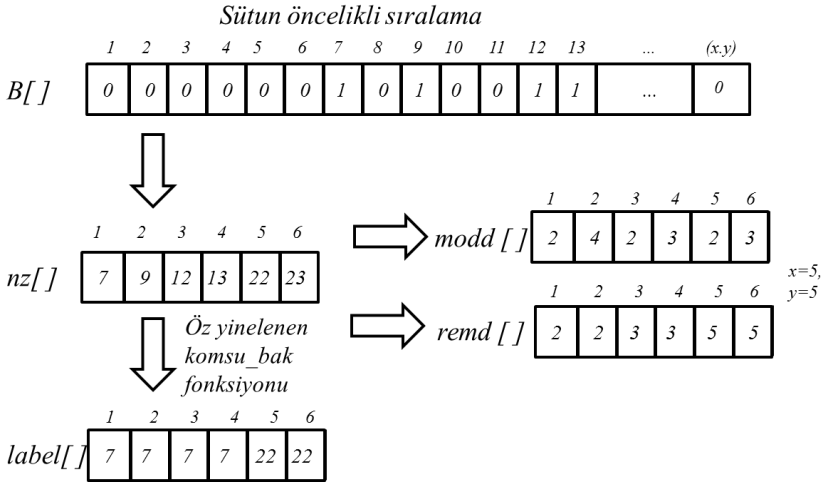
Geniş ölçekli ve de yüksek çözünürlüklü bir alanı temsil eden ikili bir sayısal harita üzerinde çok sayıda eşik değeri üzerinde yer alan hücre ortaya çıkabilir. Bu engelleri temsil eden her bir hücreyi bağımsız olarak ele almak ve modellemek hesaplama performansı açısından sıkıntı oluşturacaktır. Bu nedenle özellikle engel olarak temsil edilen hücreleri gruplamak ve geometrik bir form ile temsil etmek, hesaplama performansı açısından daha uygun bir yaklaşımdır. Ayrıca birbirleri ile ilişkili olmayan eşik değeri üzerindeki hücreleri farklı farklı ele almak gerekir. Eşik değeri üzerinde yer alan ve otonom bir sistem için sınırlı bir alan içerisinde yer alan engeli temsil eden hücreleri gruplamak ve de her bir gruba ayrı bir temsil imkânı sağlayan eşsiz bir etiket değeri vermek gerekir. Bu işlemleri gerçek zamanlı olarak uygulamak açısından paralel olarak ortaya koymak amacıyla paralel engel gruplama ve etiketleme mimarisi geliştirilmiştir.



Şekil 7.4: Örnek ikili harita temsili.

Paralel engel gruplama ve etiketleme yapısı Şekil 7.4 ile gösterilen örnek sayısal ikili haritadan faydalanarak açıklanacaktır. Örnek haritadan da görüldüğü üzere ikili bir sayısal harita iki boyutlu bir ikili dizidir. Bu temsil aynı zamanda bilgisayar ortamında iki renkli sayısal görüntülerin saklanmasında kullanılan veri yapıları ile benzerlik göstermektedir. Görüntü işleme algoritmaları incelendiğinde görüntü

içerisinde yer alan nesneleri ortamdan ayırt ederek ayrı şekilde ele almak amacıyla kullanılan Bağlı Bileşen Etiketleme (Connected Component Labeling - CCL) algoritmasına rastlamak mümkündür [88]. CCL algoritması belirli bir objeyi sayısal olarak temsil edilen bir görüntünün arka planından ayırmak amacıyla kullanılan algoritmalardan birisidir. Bu noktada engeller içeren ikili bir harita üzerinde bir grup temsil eden çok sayıdaki eşik değerinin üzerindeki noktanın gruplanması amacıyla da kullanılabilir. Genel olarak sıralı işlem gerçekleştiren donanım mimarileri üzerinde koşturan CCL algoritması, bir görüntü üzerinde yer alan her bir piksel olarak adlandırılan ve renk değeri olan hücrenin, bu renk değerine bağlı olarak komşusu olan diğer hücrelerin renk değerlerini kontrol ederek bir ilişki kurmayı amaçlar. Dolayısıyla ortamdan ayırt edilmek istenen nesnenin ilişkili komşu hücreleri gruplandırarak temsil edilmesine olanak sağlayan bir yapısı vardır.



Şekil 7.5: CCL algoritması ile engel gruplama ve etiketleme işlemi esnasında kullanılan veri yapılarının gösterimi.

Şekil 7.5 ile gösterildiği şekilde, Şekil 7.4 de yer alan örnek ikili sayısal harita verisi *B[ ]* ile temsil edilen tek boyutlu bir vektör veri yapısında akış tipi işlemcilerin performansını olumlu şekilde etkileyecek biçimde bellekte saklanabilir. İkili sayısal haritaların bellekteki temsili amacıyla kullanılan tek boyutlu vektör yapısı, elemanlarının büyük çoğunluğunun "0" değeri aldığı sparse veri yapılarına benzerlik göstermektedir. Tek boyutlu vektörün bellek alanında saklanması amacıyla "1" olan değerlerinin indeks değerlerinin bir vektör üzerinde tutulmasına imkân sağlayan bir dönüşüm ile temsili mümkündür. Bu yapı Şekil 7.5'de yer alan "*nz*" dizisi ile gösterilebilir. Şayet grafik işlemci belleğinde saklanan bir ikili harita temsili bir vektör veri yapısı ile mümkün ise bu dönüşümün paralel olarak yapılmasına imkân

sağlayan çok sayıda algoritma literatürde mevcuttur [90]. Bu algoritmalardan birisi de Algoritma 8 ile ortaya konulmuştur. Bu algoritmada performansı etkileyen en önemli nokta atomik bir operasyon şeklinde tanımlanan sayaç değerinin güncellenmesidir. "*nz*" vektörünün indeks değeri sayaç ile tutulmakta ve vektör içinde yer alan değeri "*0*" dan farklı ikili haritada yer alan elemanların vektör indis değerlerinin bu vektör üzerinde hangi sırayla yer alacağı bilinmemektedir. Bu işlem paralel biçimde GPU üzerinde gerçekleştirilmiştir.

Algoritma 11. PCCL çekirdek kodu

Girdi *: B[ ], counter, x*
Çıktı *: nz[ ], modd[ ],remd[ ],labe[ ]*

11.1: *thread id* değerini hesapla
*int const i = blockDim.x * blockIdx.x + threadIdx.x;*

11.2: *sayaç değerini sıfırla*
counter =0;

11.3: *syncthreads*

11.4: şayet hücre engel ise engel temsil vektör değerlerini hesapla
11.5: ***if*** *(B[i]==1)* ***then***
11.6: *index = atomicInc(Counter);*
11.7: *nz[index] = i;*

11.8: *temp = mod(i,x);*
11.9: ***if*** *(temp<>0) then*
11.10: *modd[index]= temp;*
11.11: ***else***
11.12: *modd[index]=x;*
11.13: ***end if***

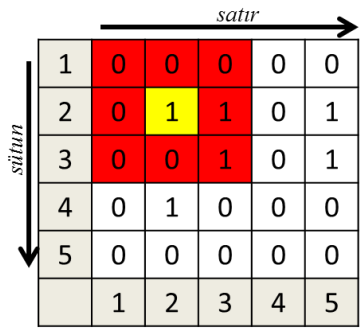
11.14: *remd[index]=ceil(i/x);*

11.15: *syncthreads*

11.16: *min = komsu_bak(i,nz,x);*
11.17: *label(index) = min;*
11.18: ***end if***

Ortaya çıkan "*nz[ ]*" vektörünün değerleri "*B[ ]*" ile temsil edilen ikili sayısal harita değerlerinin yer aldığı vektör üzerindeki "*1*" değerlerinin, yani sıfırdan faklı (non-zero, nz) elemanların "*B[ ]*" vektörü üzerindeki konum bilgisi ve "*nz*" vektörünün uzunluğu "*B[ ]*" vektöründeki değeri "*1*" olan hücrelerin sayısıdır. *Algoritma 11* ile

ortaya konulduğu şekilde şayet her bir ipliğin "*B[ ]*" ikili vektörü üzerinde yer alan bir hücre elemanını işlediği ön görülür ise, hücre eleman değeri "*0*" olduğunda gerçekleştirdiği bir işlem bulunmamaktadır. Şayet hücre eleman değeri "*1*" ise öncelikle "*nz*" vektörüne bu elemanın "*B[ ]*" vektöründeki indis değerini, bir başka deyim ile *thread_id* değerini aktarır. Akabinde bu değeri mod işlemine tabi tutar ve sonucu "*modd*" vektöründeki elemanının "*nz*" vektöründe konumuna karşılık gelen konuma yazar. Ardından elde ettiği değeri iki boyutlu yükseklik haritasının sütun sayısı değerinden faydalanarak bir bölme işlemine tabi tutar ve sonucu yukarı yuvarlayarak "*remd*" vektöründe yine aynı karşılık konuma yazar. Bu işlemleri gerçekleştiren yapı ***Algoritma 11*** ile ortaya konulmuştur.



Şekil 7.6: İkili boyutlu sayısal harita üzerinde komşuluk ilişkisi içinde yer alan hücrelerin gösterimi.

Böylece Şekil 7.5'de ortaya konulan veri yapılarından "*label*" vektörü haricindekiler elde edilmiş olur. "*label*" vektörü ise bir komşuluk ilişkisi içinde ikili sayısal harita üzerinde yer alan ve değerleri "*1*" olan hücrelerin aynı eşsiz etiket değeri ile etiketlenmelerini sağlayan özyinemeli bir fonksiyon ile sonucu ile oluşturulur. Özyinemeli olarak koşturan "*komsu_bak*" fonksiyonunun çağrılması sonucunda iplik numarasına bağlı olarak üzerinde çalışılan hücrenin Şekil 7.6 ile gösterilen sekiz komşusundan değeri "*1*" olanlar arasında ve bunlarında komşuları arasında değeri "*1*" olanlar arasında şeklinde en küçük konuma sahip hücre değeri elde edilmeye çalışılır. Böylece en küçük konuma sahip hücrenin konum değeri komşuluk ilişkisi içinde bulunan tüm hücrelerin etiket değeri olarak tespit edilir. Bu özyinemeli fonksiyon algoritması *Algoritma 12* ile gösterilmiştir.

Algoritma 12. komsu_bak fonksiyonu

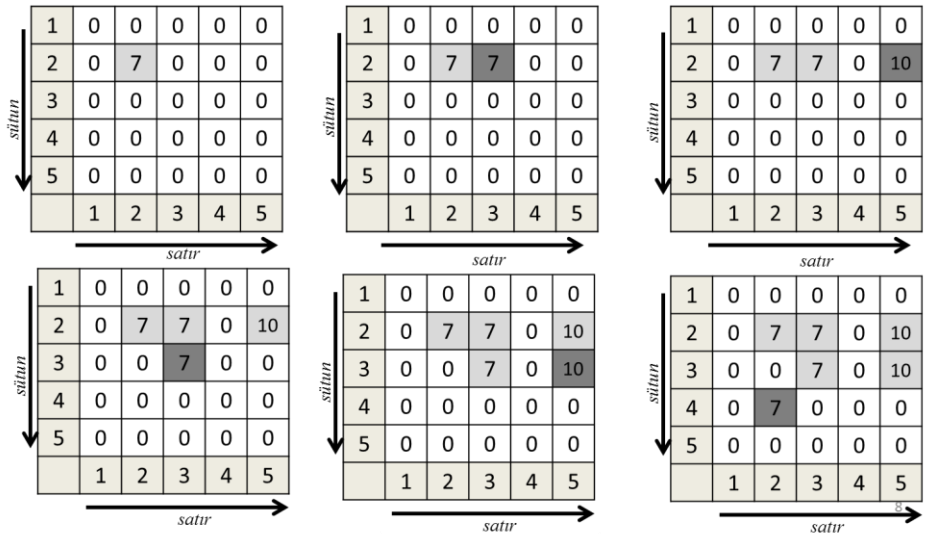
Girdi ***:*** *i,nz,x*
Çıktı ***:*** *min*

12.1: öncelikle min değeri olarak komşularına bakılın hücre değerini ata
12.2: *min = i;*

12.3: konumu kendinden küçük olan komşu hücrelere bak
12.4: if *find (i-x-1, nz)*
min = komsu_bak(i-x-1, nz, x)
12.5: elseif *find (i-x, nz)*
min = komsu_bak(i-x, nz, x)
12.6: elseif *find (i-x+1, nz)*
min = komsu_bak(i-x+1, nz, x)
12.7: elseif *find (i-1, nz)*
min = komsu_bak(i-1, nz, x)
12.8: endif

12.9: return *min*

İkili bir sayısal harita üzerinde yer alan hücrelerin gruplanması amacıyla yapılması gereken temel işlem, her bir hücrenin komşuluk ilişkisi içinde bulunduğu diğer hücrelerin değerlerini kendi değeri ile karşılaştırarak bir grup oluşturması ve ilişkili hücrelere ortak bir etiket değeri verilmesidir. Komşusu ile aynı değere sahip hücre aynı zamanda komşunun komşusu olan ve de aynı değere sahip diğer bir hücre ile de aynı grubun üyesidir. Bu ilişkinin ortaya konulabilmesi için *Algoritma 12* ile gösterilen şekilde öz yinelenen bir yapı geliştirilmiştir.



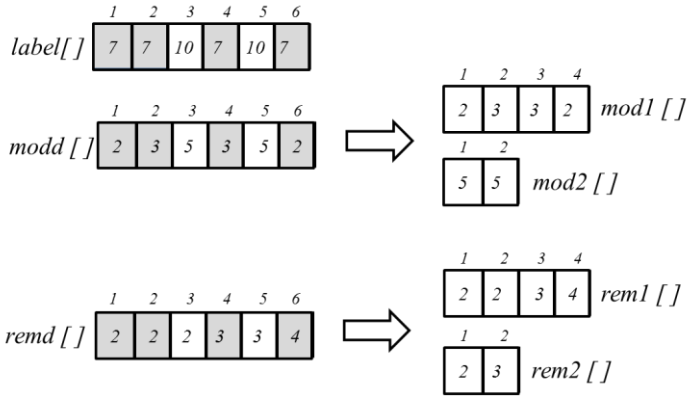
Şekil 7.7: Algoritma koşturma anında adım adım label vektörünün iki boyutlu gösterimi.

Algoritma 12 incelendiğinde bir eleman değerinin bir vektör içeriğinde bulunup bulunmadığına bakılmasına ihtiyaç olduğu görülmektedir. Bu işlemin gerçekleştirilebilmesi için MATLAB *Paralel Computing Toolbox* Kütüphanesi içinde yer alan "*find*" komutundan faydalanılmıştır. "*find*" komutu geriye "*true*" veya "*false*" şeklinde bir sonuç döndürmektedir ve bu sonuca istinaden algoritmadan da görüldüğü üzere dallanma sağlanarak en küçük konuma sahip hücre değerine ulaşılmaya çalışılmıştır.

Algoritma 11 ve *Algoritma 12*'de yer alan programlama yapısının adım adım örnek uygulama üzerinde koşturulması Şekil 7.7 ile temsil edilmiştir. Sonuç olarak görüldüğü üzere örnek sayısal harita verisi üzerinde iki adet farklı sayısal etiket değeri (7 ve 10) ile temsil edilen iki engel ortaya çıkmıştır.

7.1.3 Paralel minimum çevreleme ve ağırlık noktası belirleme

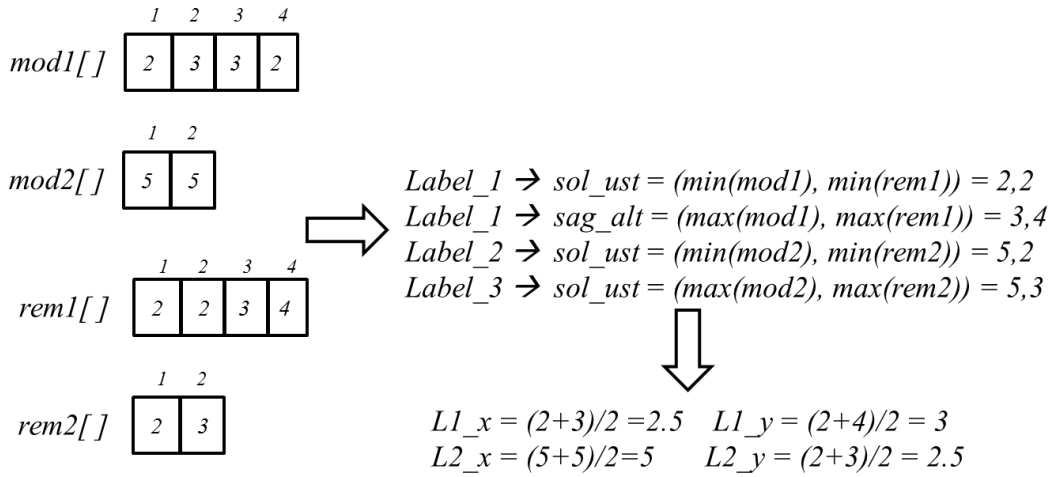
İkili sayısal harita üzerinde yer alan eşik değeri üzerindeki belirli bir komşuluk ilişkisine sahip olduğundan dolayı gruplanan ve eşsiz bir sayısal değer ile etiketlenen hücreler belirgin bir geometrik form ile çevrelenebilir.



Şekil 7.8: Etiket değerlerine bağlı *rem* ve *mod* veri yapılarının üretilmesi.

Bu işlemin temel amacı sayısal harita üzerinde yer alan ve engel olarak değerlendirilen ilişki halindeki hücrelerin tek tek modellenmesi problemi yerine bir grup olarak ele alınmasıdır. Bu nedenle modellemeye temel oluşturabilecek elips veya dörtgen gibi standart bir geometrik form belirlenebilir. Potansiyel alan yapısına uygun olarak modellemeye olanak sağlayacak nitelikte bu tez çalışması kapsamında standart geometrik form olarak dörtgen belirlenmiştir. Aynı etiket değerine sahip hücrelerin en küçük dörtgen geometrik form ile çevrelenmesi işlemine literatürde "*Minimum Bounding Box Problem*" adı verilmiştir [91]. Farklı algoritmalar ile

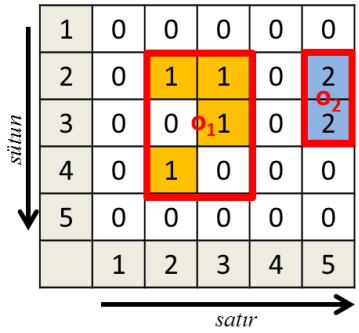
uygulanması görülmektedir [91]. Çalışma kapsamında geliştirilen yaklaşım ise paralel hesaplama mimarisi içinde etkin olarak kullanılabilecek bir yöntem olarak belirlenmiştir. Bu noktadan hareket ile "*B[ ]*" ve "*label*" vektörlerinin paralel olarak hesaplanması esnasında Şekil 7.8 ile gösterilen ilave iki veri yapısı daha ortaya konulmuştur. Bunlar indis değerlerin yer aldığı vektör üzerindeki her bir veri değerinin mod işlemine tabi tutularak üretilen "*modd*" vektörü ve de aynı değerlerin sayısal haritanın satır sayısına bölümünden kalan değerleri tutan "*remd*" vektörüdür. "*modd*" ve "*remd*" vektörleri, "*label*" vektörü temel alınarak her bir eşsiz etiket değeri için birer adet olacak şekilde örnek problem için "*mod1*" ve "*mod2*" veya "*rem1*" ve "*rem2*" şeklinde adlandırılan vektörlere parçalanabilir. Böylece her engel grubu için ayrı birer mod ve rem vektörü elde edilir.



Şekil 7.9: Algoritma ile dörtgen geometri sınırları ile ağırlık merkezinin belirlenmesi.

Bu diziler her bir etiket değeri için dolaysıyla her bir engel grubu için ayrı ayrı birer adet olacak şekilde hesaplanır. Şekil 7.9 ile gösterilen biçimde her bir etiket değerine sahip gruba ait "*mod"* dizisinin en küçük elemanı ile "*rem*" dizisine ait en küçük elemanı bu gruba ait hücreleri çerçeveleyen en küçük dörtgenin sol üst köşe nokta koordinatını temsil eder. Aynı şekilde "*mod*" dizisinin en büyük elemanı ile "*rem*" dizisinin en büyük elemanı ise bu grubu çevreleyen en küçük dörtgenin aynı düzlem koordinat sistemindeki sağ alt köşe noktasının konum bilgisini verir.

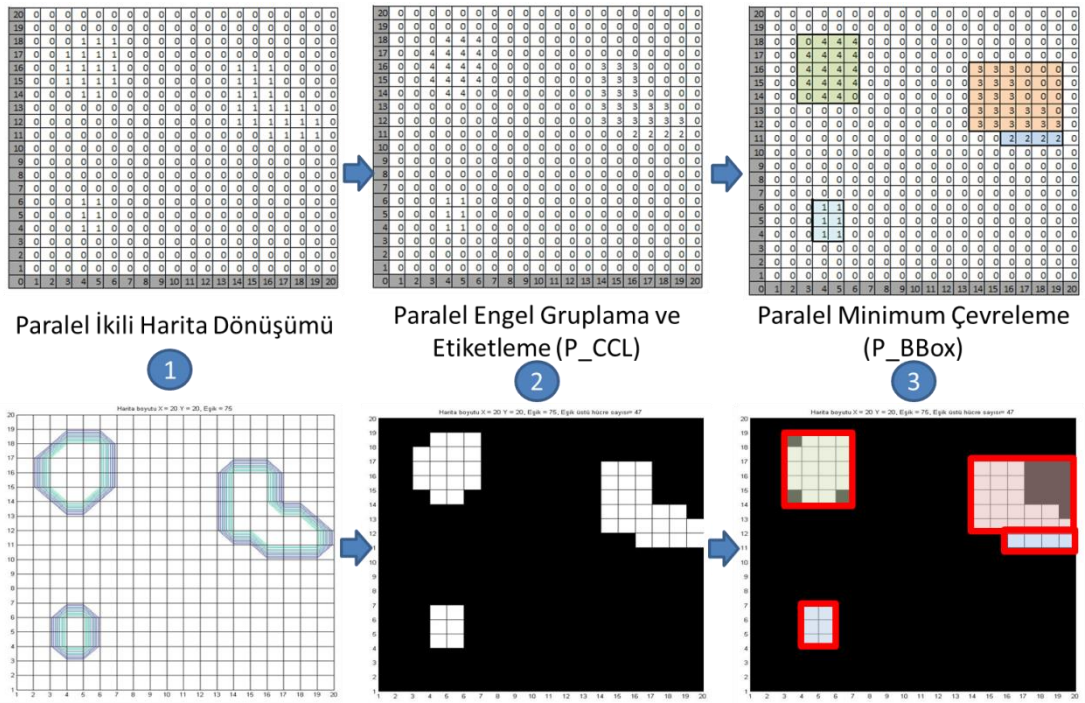
Bu iki değerden faydalanarak aynı etiket değerine sahip grubu çevreleyen en küçük dörtgen ve bu dörtgenin merkez koordinatları dolayısıyla ağırlık merkezi hesaplanabilir. Örnek ikili sayısal harita için elde edilen sonuçlar Şekil 7.10 ile gösterilmiştir.



Şekil 7.10: Örnek ikili harita için engel çevreleme ve ağırlık merkezlerinin tespitinin gösterimi.

7.1.4 Engelleri parçalama ve olası çözüm uzayını genişletme

Şekil 7.11 ile paralel ikili harita dönüşüm işlemine tabi tutulan bir sayısal harita üzerinde paralel engel gruplama ve minimum çevreleme algoritmalarının sonuçları farklı bir örnek için adım adım gösterilmiştir.

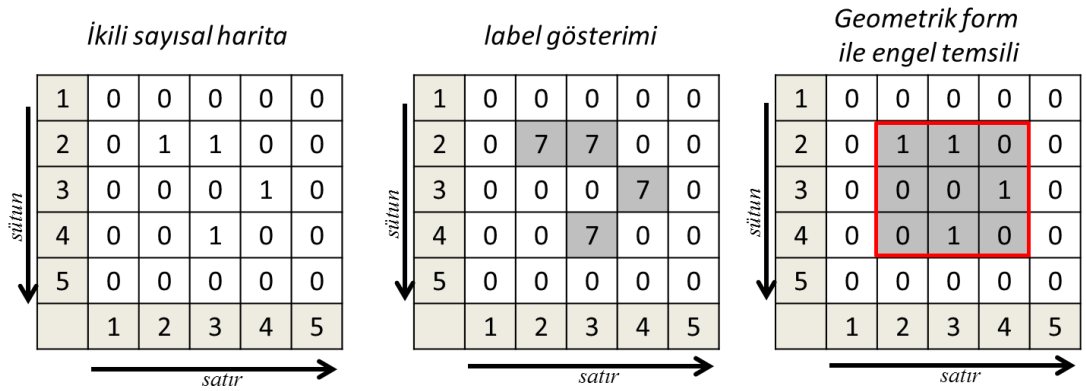


Şekil 7.11: Sayısal ikili harita üzerinde engel gruplama ve etiketleme ile minimum çevreleme işlemlerinin gösterimi.

Şekil 7.11'den görüldüğü üzere aynı komşuluk ilişkisi içinde bir grup oluşturabilecek nitelikte olan aynı etiket değerine sahip hücreler birden fazla geometrik form ile temsil edilerek bağımsız engeller olarak ele alınabilir. Bunun nedeni potansiyel kaynak olarak modellenmek istenen engellerin modellenmesi esnasında karşılaşılacak temsil problemlerini en aza indirgemektir. Böylece bir kısım olası çözüme giden yollardan mahrum kalınmamış olunabilir. Bir başka ifade ile engel

olmadığı halde engel olarak seçilen geometrik form gereği engel olarak temsil edilen hücrelerin geriye kazanımı sağlanabilir.

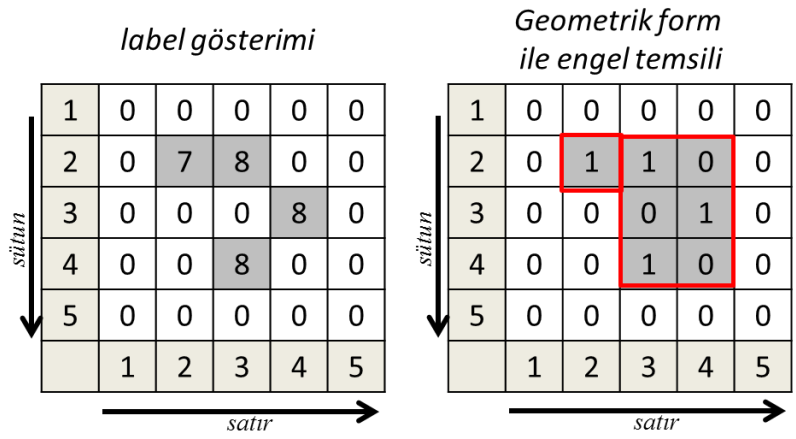
Bunu sağlamak için iki yöntemden faydalanmak mümkündür. Bunlardan ilki Şekil 7.11'de de görüldüğü üzere belirli bir ölçek değerinin üzerinde nispeten daha büyük şekilde ortaya çıkan engel grupları rastgele bir noktadan parçalanarak her biri için ayrı bir form ve ağırlık noktası teşkil edilebilir. Böylece aynı etiket değerine sahip olsalar dahi farklı engeller olarak modellemek mümkündür. Engelleri birbirinden ayırt etmek için ayrılan gruba ayrı etiket numarası vermek kolaylık sağlayacaktır.



Şekil 7.12: Sayısal ikili harita üzerinde engel gruplama ve etiketleme ile minimum çevreleme işlemlerinin örnek uygulama için gösterimi.

Diğer bir yöntem ise geometrik form ile temsil edilen hücrelerin değerlerine bakarak belirlenecek bir sabit orana bağlı şekilde geometrik form içinde yer alan ve engel teşkil eden hücre sayısının bu oranın altında kalana kadar ortadan parçalanması ile sağlanabilir. Bu yaklaşım örnek olarak Şekil 7.12 ile gösterilen sonuç üzerinde uygulanmıştır. Şekilden de görüleceği üzere tespit edilen ve dörtgen geometrik form ile temsil edilen engel içinde toplam dokuz hücre bulunmaktadır. Ancak bunlardan dördü engel, diğer kalan beşi ise engel oluşturmamaktadır. Engel oluşturmadığı halde seçilen dörtgen geometrik form gereği engel olarak kabul edilmişlerdir. Bunun sebebi her bir hücreyi engel olarak temsil etme maliyetinden kurtulmaktır. Ancak bu durum bir takım çözüm olasılıklarından feragat etmeye, hatta olası bir çözüm olduğu halde yokmuş sonucuna varılmasına neden olabilir. Şekilde gösterilen örnek durum için temsil edilen geometrik form içindeki engel teşkil eden hücrelerin oranı toplam hücre sayısının yaklaşık %44'ünü oluşturmaktadır. Şayet örnek olarak seçilen eşik değerinin %50 olduğu kabul edilir ise bu durumda bu engeli parçalamak ve birden fazla geometrik form ile temsil etmek gerekecektir. Bu bölme işlemini geometrik

formun uzun kenarının ortadan ikiye bölünmesi şeklinde gerçekleştirebiliriz. Şayet dörtgen formun kenar uzunlukları eşit ise formu yatayda ikiye böleriz.



Şekil 7.13: Sayısal ikili harita üzerinde engel parçalama işlemlerinin gösterimi.

Şekil 7.12 ile gösterilen örnek ikili sayısal harita üzerindeki etiket değerlerini Şekil 7.13 ile gösterildiği şekilde ve açıklanan biçimde ikiye bölerek engel temsilinde görüldüğü üzere iki ayrı engel şeklinde modelleyebiliriz. Bu durumda birinci engel için %100, ikinci engel için ise %50 bir engel kaplama oranı elde ederiz.

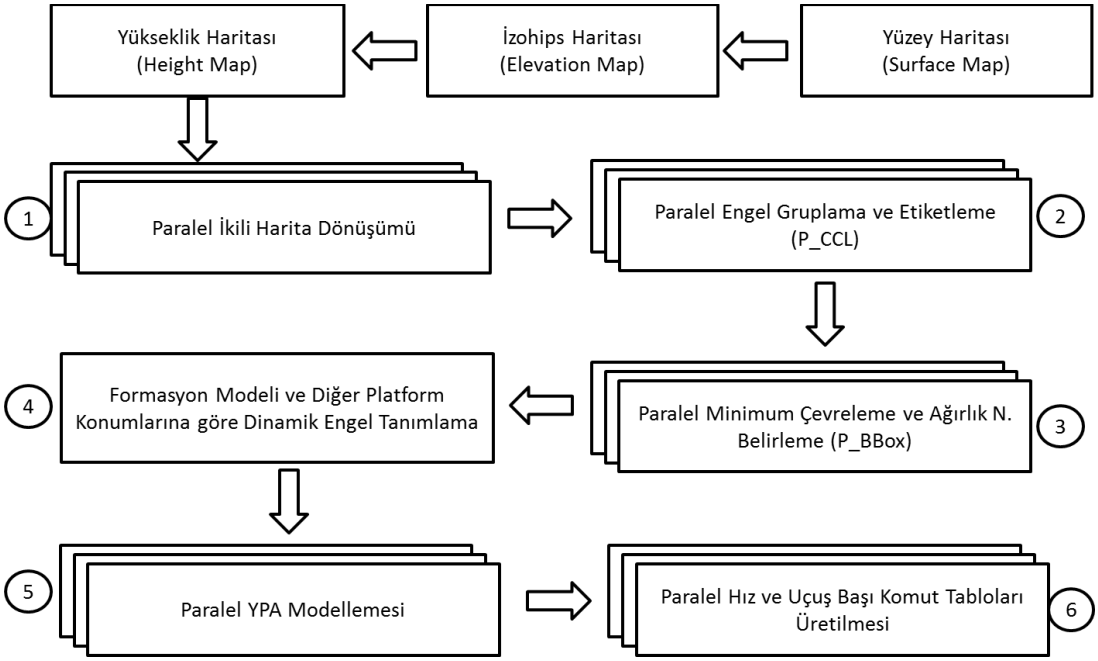
Belirlenen eşik değer yüzdesinin yüksek seçilmesi olası çözüm yollarının kaybedilmemesi açısından önem arz etmektedir. Ancak bu yüzde değer yüksek seçildikçe aynı sayısal harita üzerinde yer alan engel sayısı da o derece artacaktır. Hatta eşik yüzde derecesine ve sayısal harita verilerine bağlı olarak her bir engel teşkil eden hücrenin bağımsız olarak temsil edilmesine dolayısıyla engel gruplama işlemine gerek kalmamasına sebebiyet verilebilir.

Alan içerisinde çok sayıda engel modellemek hesaplama yükünü ve gerçek zamanlı modelleme ihtiyacını karşılamada problemlere neden olabilir. Ortaya çıkan her bir engel alan içerisinde çalışmanın ilerleyen bölümlerinde açıklanacağı şekilde YPA ile modellenecek ve yol planlamasına etki edecektir. Tüm bu nedenlerden dolayı bu bir optimizasyon problemidir. Olası çözüm yollarının arttırılması ve modelleme sayısının arttırılarak yol planlaması gerçekleştirilmesi performansı ters orantılıdır.

7.2 Engellerin YPA Tabanlı Yol Planlamasına Dahil Edilmesi

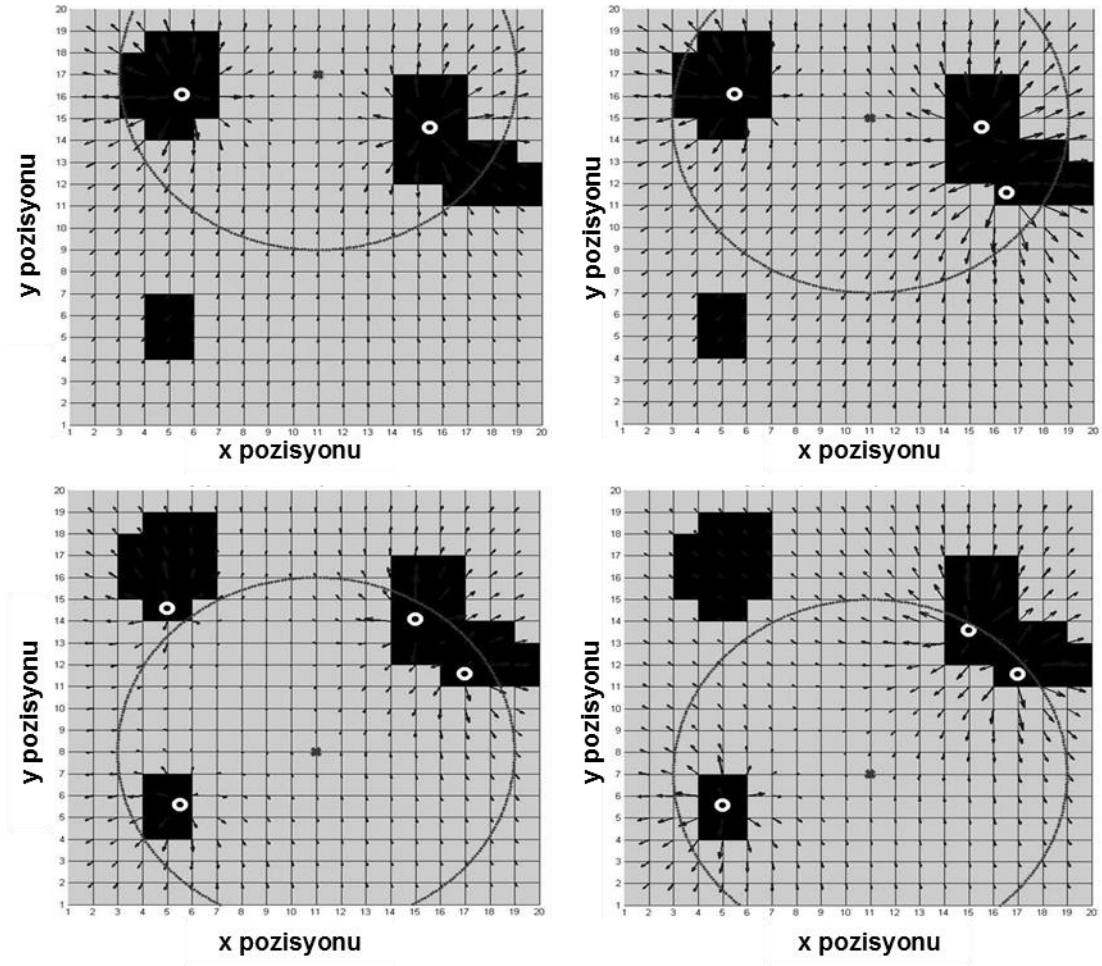
Bu ihtiyaca istinaden Şekil 7.2 ile verilen akış diyagramına Şekil 7.14 ile de gösterildiği biçimde "*Formasyon Modeli ve Platform konumlarına göre dinamik*

engel tanımlama" şeklinde ayrı bir kontrol mekanizması tanımlaması eklenmiştir. Bu yapının amacı belirli bir formasyon modelinde (V, box, Trail vb.) platformların birbirlerine göre göreceli konumlarını muhafaza etmelerine imkân sağlamak ve de bu esnada çarpışmayı önleyici bir yapı oluşturmaktır.



Şekil 7.14: Engellerin tespit ve temsili amaçlı gerçekleştirilen işlemlere çarpışma önleme yapısının eklenmesi.

Sayısal harita üzerinde yer alan engellerin platform üzerinde yer alan algılayıcı ile tespit edilmesi sonucunda modellenmesi neticesinde üretilen statik engellere ek olarak platformun hareketini sürdürdüğü müddetçe veya alan içinde yer alan diğer mobil platformlardan her hangi birisinin her hareketinde tüm potansiyel alanın yeniden hesaplanmasına ihtiyaç duyulacaktır. Çünkü hareketin etkisinin sınırları bilinmediğinden alan içinde tanımlanan tüm çözüm uzayı da durumdan etkilenecektir. Alan geneline dağılmış olan potansiyel değerler değişiklik göstereceğinden, vektör alanın yeniden hesaplanmasına ihtiyaç duyulacaktır.

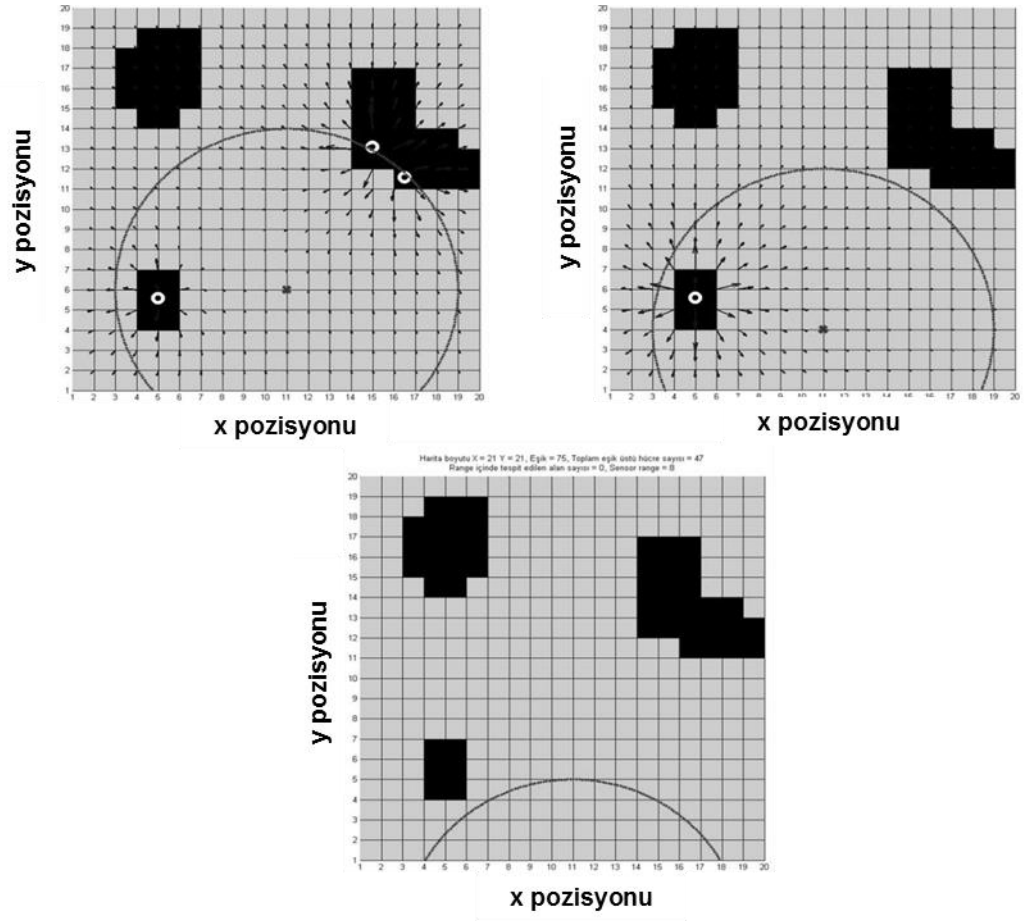


Şekil 7.15: Engeller içeren alan içerisinden platform algılayıcıları ile engellerin dinamik olarak tespit edilmesi.

Şekil 7.15 ile gösterilen örnek senaryo dahilinde platformun içinde yer aldığı bölge aslında çok daha büyük ölçekte olmasına rağmen yaklaşımın daha kolay açıklanabilmesine istinaden 400 x 400 metre olarak belirlenmiş ve alan çözünürlük değeri 20 metre olarak seçilmiştir. Alan içinde yer alan platformun Çizelge 2.2 ile özellikleri açıklanan *Aerosonde Mark I* platformu olduğu ve üzerinde yer alan engel algıyıcı sensorün Çizelge 4.2 ile özellikleri açıklanan *Velodyne HDL32E* olduğu kabul edilebilir. Bu durumda sensorun engel algılama yarıçapı 160 metre olacağından maksimum 8 birim hücre mesafedeki engelleri algılayabilecektir. Platformun hareketine etki eden engeller algılayıcının 360 derecelik çevresini tarayabilmesine istinaden algılanabilen sekiz birim yarıçapındaki daire içinde kalan engeller olacaktır.

Bölüm 7.1 kapsamında gerçekleştirilen sayısal harita verisi üzerindeki işlemler sadece platform üzerinde yer alan algılayıcının modellenmesi ve gerçek bir uygulama anında davranışlarının temsil edilebilmesi amacıyla gerçekleştirilmiştir.

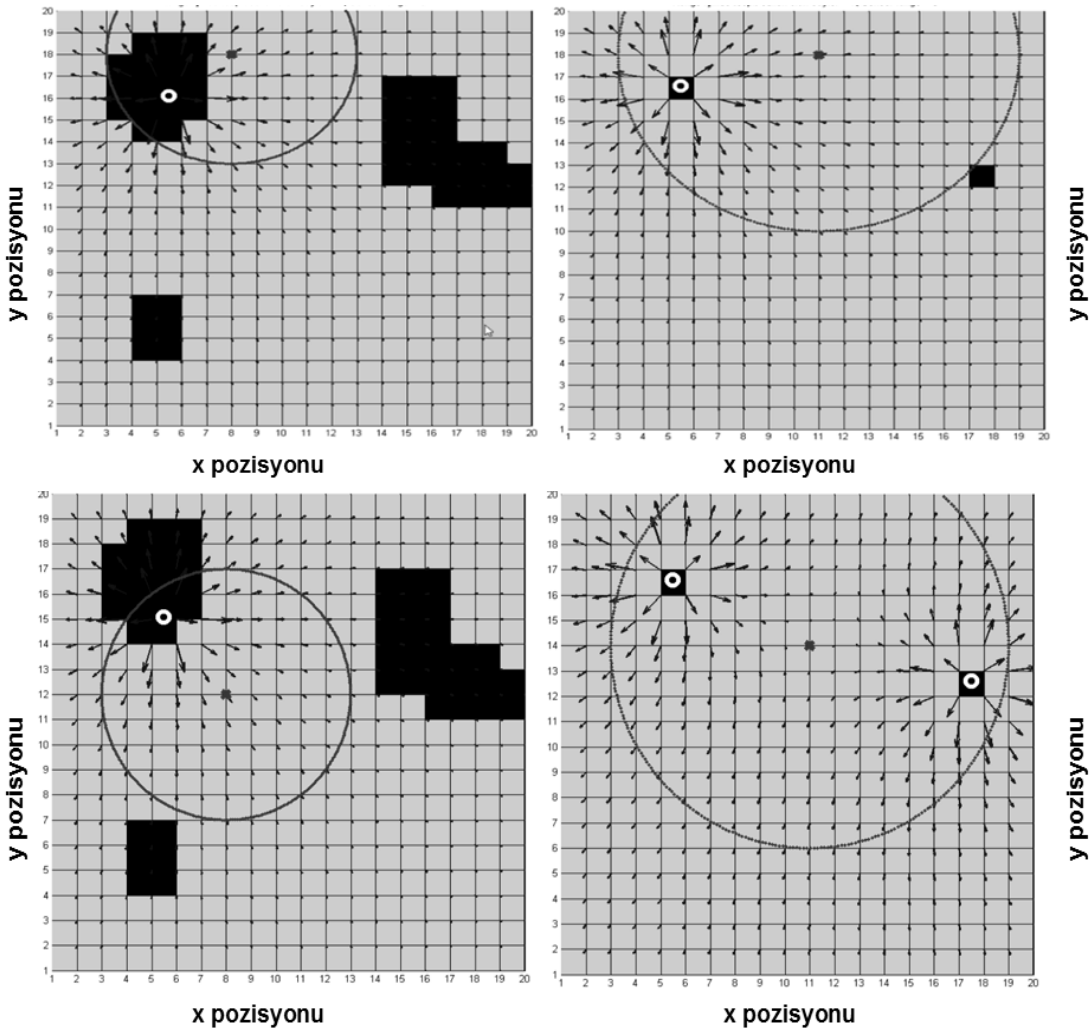
Normal bir uygulamada, bir başka tanım ile üzerinde bir algılayıcının olduğu platformun yol planlamasının icrası esnasında, algılayıcı tarafından tespit edilen engeller ikili sayısal harita yapısına benzer şekilde modellenecek ve de aynı önceki kısımlarda açıklandığı şekilde gruplanarak belirli geometrik formlar ile temsil edilecektir.



Şekil 7.16: Engellerden etkilenmeden potansiyel alan içinde ilerleyen mobil platformun gösterimi.

Şayet platformun içinde bulunduğu alan hakkında önceden bilgi sahibi olmadığını kabul edersek, problemin boyutunu bir derece daha arttırmış oluruz. Örneğin Şekil 7.15 ile gösterilen alan içerisinde mobil platformun hareketi öncesi bilmediği bir takım engellerin olduğunu ve platformun engelleri daire ile temsil eden bölge içinde platform üzerinde yer alan algılayıcılar ile tespit edebildiğini kabul edersek; platform alan içerisinde gerçekleştirdiği hareketi süresince yeni engeller tespit edecektir. Tespit ettiği engellere bağlı olarak da hareketini tekrar planlayacaktır. Şekil 7.15 ve Şekil 7.16 ile gösterilen örnek uygulamalardan da görüleceği üzere tespit edilen engellerin her hareket adımında geometrik formları dolayısıyla da ağırlık merkezleri değişecektir.

Bu nedenle her ne kadar tespit edilen engeller statik olsa dahi potansiyel alanın ve vektör alanın tekrar hesaplanmasına ihtiyaç duyulacaktır. Platformun hızına ve de alanın çözünürlüğüne bağlı olarak hesaplama süresi belirli bir zaman aralığında bitirilmesi gerekecek ve de bir sonraki hareket için planlama yapılmasına imkân sağlanacaktır. Ayrıca platformun sadece üzerinde yer aldığı ya da belirli bir çevre için potansiyel ve vektörel değerlerin hesaplanması yeterli olmayacaktır. Çünkü öneri kapsamında da açıklandığı gibi yerel minimum veya robot tuzağı gibi potansiyel alanlara özgü problemler ile karşılaşılmaması için yol planlamasının genel olarak ele alınmasına ihtiyaç duyulacaktır.

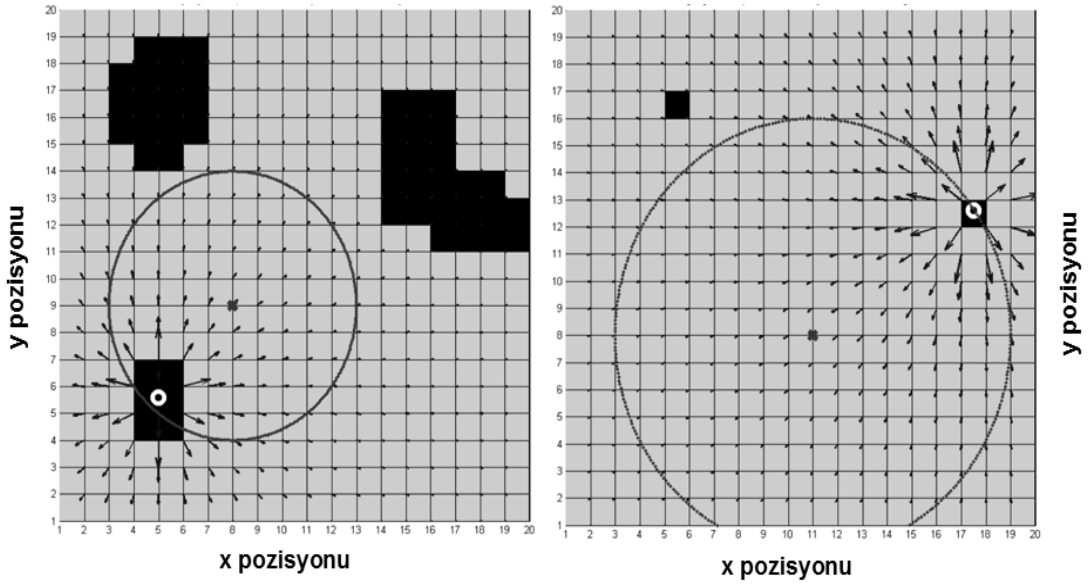


Şekil 7.17: Engellerin oluşturduğu potansiyel alan değişimlerinden etkilenmeden ilerleyen mobil platformun gösterimi.

Şekil 7.15 ve Şekil 7.16 ile gösterilen uygulama kapsamında alan içerisindeki platformun engellere istinaden üretilen vektörel kuvvet bileşenlerinin haricinde bağımsız olarak hareket ettiği düşünülmüştür. Bu nedenle mobil platform alan içinde doğrusal bir rota izlemiştir. Bu yaklaşım bize her *t* anında alan içinde oluşan ve

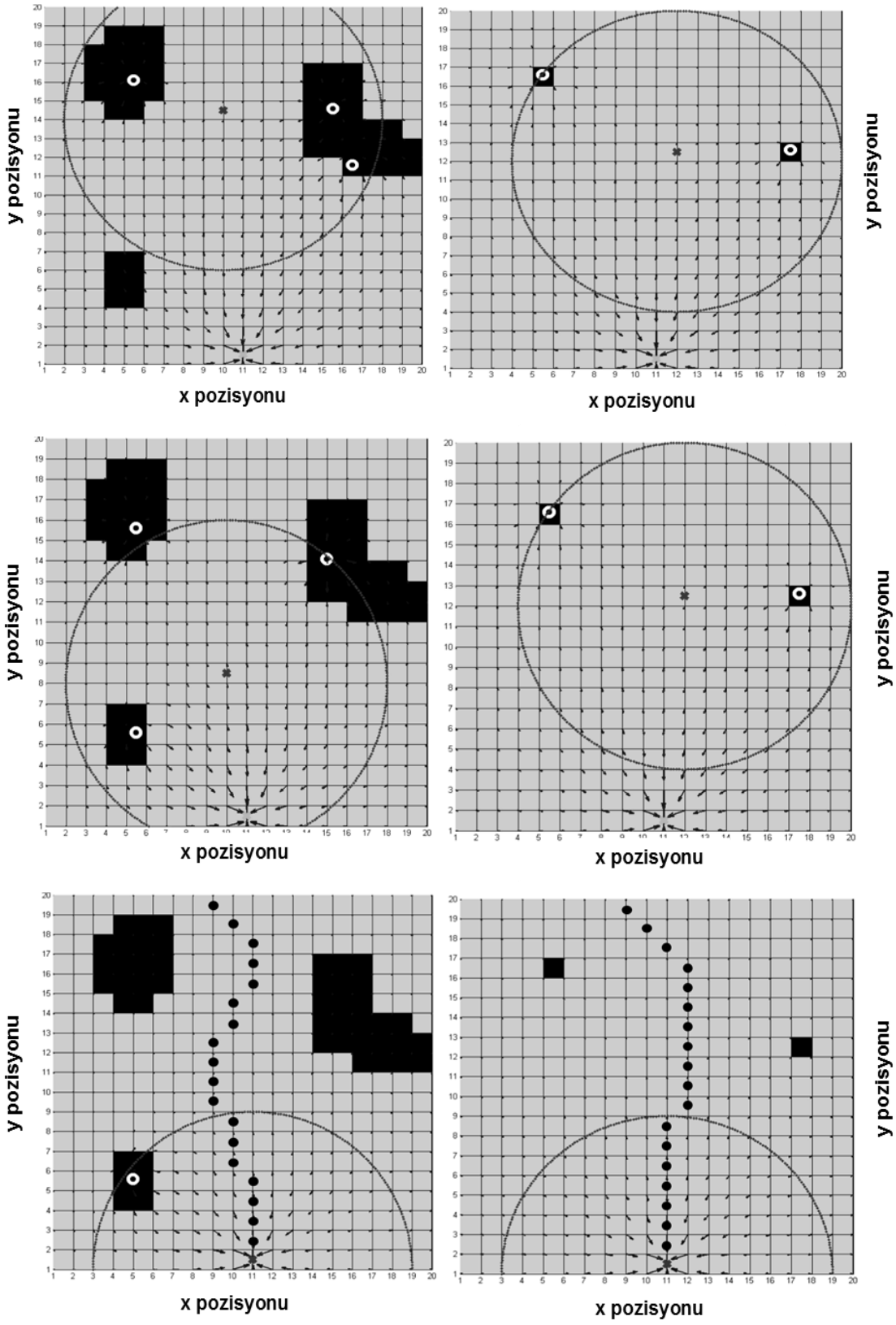
platform tarafından gerçek zamanlı tespit edilen engellerin dinamik olarak potansiyel alan değerlerini nasıl etkilediğini göstermektedir. Bir diğer husus platforma üçüncü boyuttaki hareket özelliğini kazandıran dikey çözünürlük değeri parametresidir. Dikey çözünürlük parametresi daha önceden üç boyutlu alanların iki boyutlu katmanlar şeklinde modellenmesi amacıyla çalışmanın 4.1.2 bölümünde açıklanmıştır.

Şekil 7.17 (a) ve (b) aynı sayısal haritaya sahip iki farklı dikey çözünürlük değerleri için engel modellemesini göstermektedir. Şekil 7.17 (c) ve Şekil 7.18 (a) daha düşük bir dikey çözünürlük değerini (75) temsil ederken, Şekil 7.17 (ç) ve Şekil 7.18 (b) daha yüksek bir dikey çözünürlük değerini (85) göstermektedir.



Şekil 7.18: Farklı algılayıcı menzillerine sahip farklı eşit değerli alanlar içinde hareket eden platformların gösterimi.

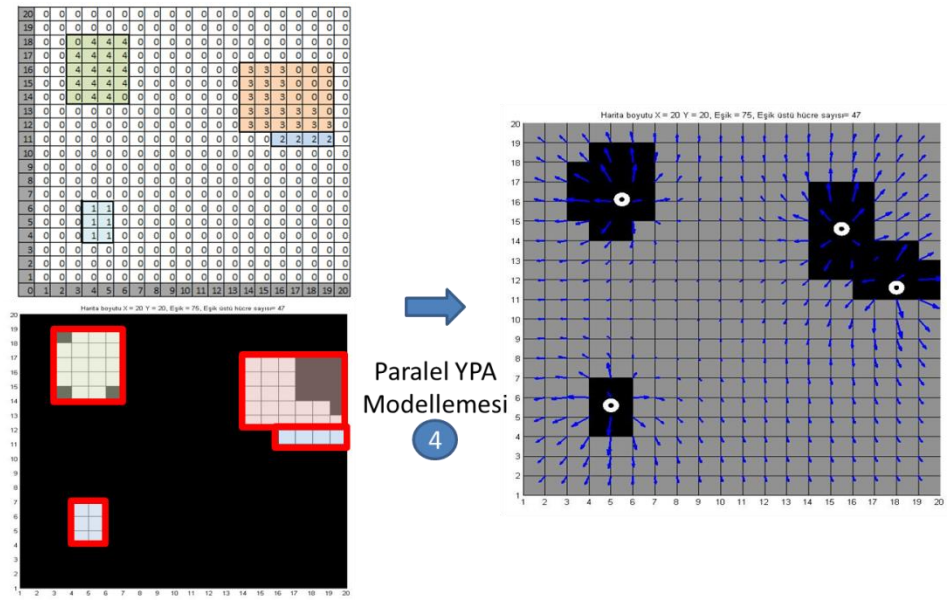
Şekil 7.19 ile gösterilen örnek uygulama da ise bir önceki örnekte anlık olarak üretilen potansiyel alana ait vektörel kuvvetlerin etkisinden bağımsız şekilde hareket eden platformun, bu defa vektör alan yönlendirmesi altında hareketi benzetilmiştir. Otonom platformun hareket modeli çalışmanın ilerleyen kısımlarında açıklanan noktasal yüklü parçacığın modellenmesi şeklinde gerçekleştirilmiştir.



Şekil 7.19: Engellerden kaynaklanan potansiyel alan değişimlerinin platformun hareketlerine etkisinin gösterimi.

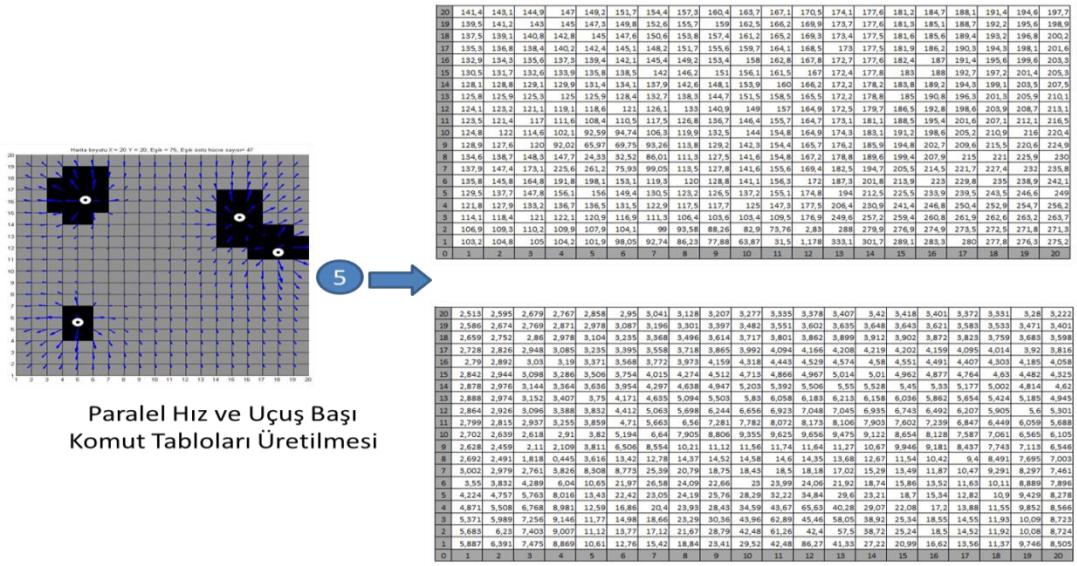
Şekil 7.19 (a) ve (b) ile gösterilen başlangıç noktasından hareketine başlayan platform sürekli olarak varmak istediği hedef noktadan kaynaklı çeken bir potansiyel

alanın etkisindedir. Dolayısıyla bir önceki örnek uygulamadaki izlediği doğrusal rotanın aksine bu kez her tespit edilen engelden uzaklaşma yönünde bir tepki ortaya koyacak şekilde davranışını değiştirecektir. Farklı eşik değerlerine sahip aynı sayısal harita üzerinde icra edilen ve aynı algılayıcı menzili ile tespit edilen iki uygulama neticesinde ortaya konulan rotalar tamamen farklı olmuştur. Bu sonuç Şekil 7.19 (c) ve (ç)'den görülmektedir.



Şekil 7.20: YPA modellemesi sonucunda elde edilen vektör alanın gösterimi.

Şekil 7.20 ile örnek ikili harita için engel çevreleme ve ağırlık merkezlerinin tespitinin ardından engel teşkil eden alanlar için vektörel alanın elde edilmesi gösterilmiştir. Örnek sayısal harita üzerinde dört adet engel grubu tespiti yapılmış ve de bu engel gruplarının ağırlık merkezleri iten potansiyel alanların kaynak noktasını oluşturacak şekilde alan içinde temsil edilmiştir. Alan içinde oluşturulan çözünürlük değerine bağlı ızgara yapısı içinde yer alan her bir hücre için paralel olarak vektör değeri hesaplanmıştır.



Şekil 7.21: Vektör alandan türetilen uçuş başı ve sürat komut tablolarının gösterimi.

Şekil 7.21 ile hesaplanan toplam vektör alan içindeki her bir ızgara hücresinde platforma komut niteliğinde olan uçuş başı ve sürat komutlarının üretilmesi gösterilmiştir. Böylece Şekil 7.14 ile gösterilen akış diyagramındaki işlem adımlarının tümü tanımlanmıştır.

Bu bölüm kapsamında ortaya konulan örnek uygulamalarda, alan içinde tek bir platform olduğu kabul edilmiş ve platformun sayısal haritalar ile belirtilen statik engelleri algılayıcıları ile tespit ederek ardından yol planlamasını gerçekleştirdiği ve bunun neticesinde oluşan hareketin sonuçlarının ortaya çıktığı görülmektedir. Oysaki çalışma kapsamında ortaya konulmak istenen bir diğer husus platformun statik bir hedef noktaya ulaşmasından çok bir formasyon dahilinde dinamik bir hedef noktada kalabilmesini sağlamak ve de bu noktaya ulaşma esnasında ya da sonrasında statik ve dinamik (diğer platformlar gibi) diğer engellere istinaden yol planlamasını tekrarlamasını sağlamak olarak açıklanabilir. Bu nedenle çalışmanın bu bölümü dahilinde ortaya konulan engel tanımlama ve modelleme yaklaşımı Bölüm 5 kapsamında açıklanan modelleme yapıları ile entegre edilebilinecek şekilde tasarlanmıştır.

8. GPGPU DESTEĞİ İLE PARALEL OLARAK YPA TABANLI OTONOM YOL PLANLAMA YAKLAŞIMI HESAPLANMA PERFORMANSI

Çalışmanın bu kısmına kadar farklı görevler doğrultusunda ortaya konulan otonom yol planlaması ihtiyacının nasıl karşılanacağına dair YPA tabanlı modellemeler geliştirilmiş ve bu modellemelerin hayata geçirebilmesi amacıyla seri ve GPGPU tabanlı paralel algoritmalar ortaya konulmuştur. Problemin zorluk derecesini arttıracak nitelikte çarpışmayı önleme ve engellerden sakınma opsiyonları formasyon uçuşu kriterleri arasına eklenmiş ve yol planlamasının icra edilmesinde dikkate alınması sağlanmıştır.

Ancak ızgara tabanlı YPA ve benzeri yol planlamaları yaklaşımlarının nispeten büyük ölçekli ve düşük çözünürlük değerine sahip karmaşık ortamlarda yol planlaması ihtiyacını özellikle yüksek süratli otonom araçlar için zamanında karşılayamamalarından dolayı yaklaşımın paralel donanımlar üzerinde daha yüksek hesaplama performansı ile icra edilmesinin uygun olacağı değerlendirilmiştir. Özellikle İHA platformlarında GPU tabanlı donanımların paralel programlama amacıyla maliyet etkin ve kullanımı uygun donanımlar olmasından dolayı etkinlikle kullanılabileceği değerlendirilmiştir.

Ancak GPGPU tabanlı paralel uygulamalar daha önceden açıklandığı üzere hesaplama performansları açısından oldukça donanım özelliklerine bağlı olup bu bölüm içerisinde farklı donanımlar üzerinde yaklaşımın başarısı sınanmış ve uygun optimizasyonlar ile hesaplama performansı en iyilenmeye çalışılmıştır.

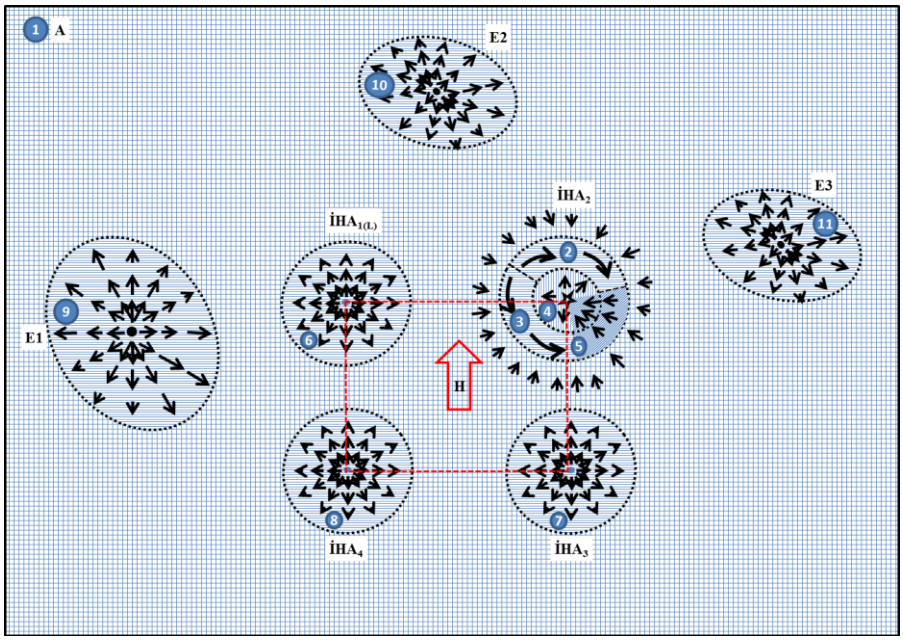
8.1 Hesaplama Performansının Ölçülmesi için Örnek Senaryo

Bu bölüm kapsamında, daha önceden açıklanan problem tanımları, gerçek zamanlı hesaplama ihtiyacı, geliştirilen seri ve paralel YPA tabanlı otonom yol planlaması algoritmaları doğrultusunda yaklaşımın ne derece ihtiyaca cevap verebildiği farklı donanım mimarileri üzerinde geliştirilen algoritmaların koşturulması sonuçları ile karşılaştırmalı biçimde ortaya konulacaktır. Bu nedenle iki temel yapı oluşturulmuştur:

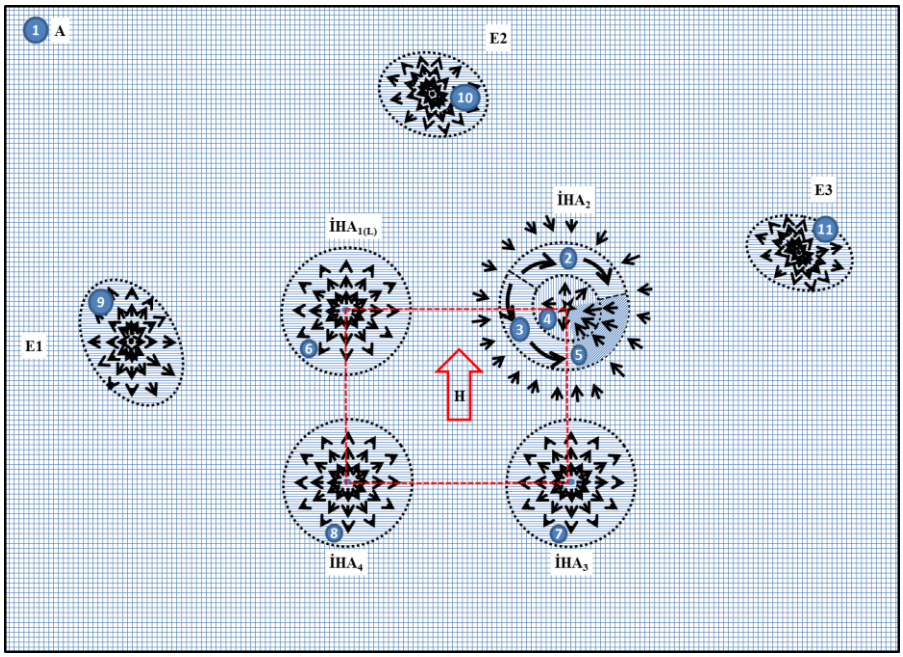
a. Bunlardan ilki bir ekseni 500 ile 10000 arasında değişen ölçek değerine sahip (çözünürlük değeri bir olacak şekilde) kare şeklindeki alan içinde yer alan tek kaynak çeken potansiyel alan değerinin hesaplanması performansıdır.

b. Bir diğeri ise örnek bir senaryo tanımlaması kapsamında performans karşılaştırmasının yapılmasıdır.

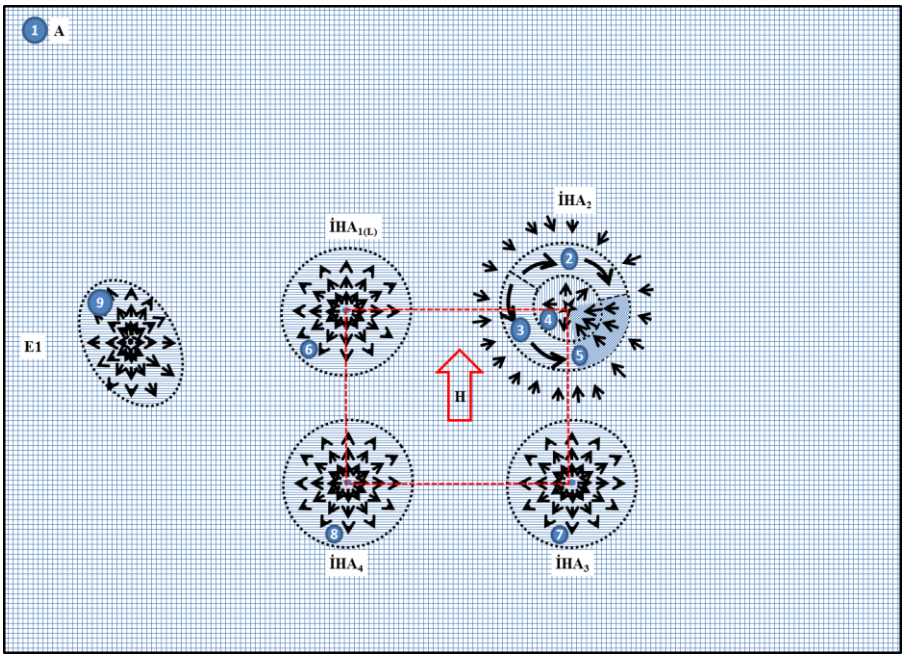
Örnek senaryo ile ortaya konulmak istenen potansiyel alan içindeki araç sayısının, alan ölçek ve çözünürlük değerinin, engel ile katman sayısının performansı nasıl etkilediğinin gösterilmesidir.



(a) 1000 ft. katman görünümü.



(b) 2000 ft. katman görünümü.



(c) 3000 ft. katman görünümü.

Şekil 8.1: Örnek hesaplama performansı senaryosu kapsamında dörtlü kare formasyon uçuşu platformlarının ve çevre engellerinin gösterimi:

Örnek senaryo, Bölüm 5.2 kapsamında tanımlanan otonom formasyon uçuşu yol planlaması ihtiyaçlarına emsal oluşturabilecek, Bölüm 4.1.2 ile açıklanan çok katmanlı modelleme yaklaşımı ile temsil edilebilen ve Şekil 5.13 ile gösterilen engeller içeren alan içerisinde kare formasyonun oluşturulması problemine benzer bir yapıda oluşturulmuştur.

Şekil 8.1 ile örnek senaryo gösterilmiştir. Örnek senaryo kapsamında dört adet otonom platformun kare formasyonu çarpışmadan teşkil etmesi ve üç boyutlu ortamda formasyonu muhafaza ederek engellerden sakınacak şekilde seyrüsefere imkan sağlayan yol planlamasının ortaya konulması amaçlanmaktadır. Çok katmanlı modelleme yaklaşımına uygun olarak üç katman oluşturulmuş ve her bir katman aralığı 1000 ft. olacak şekilde kabul edilmiştir. Engellerin dağ veya tepe gibi doğal sütreler olduğu kabul edilmiştir. Üçten sonraki katmanlarda yani 4000 ft. üzerinde engel olmadığı kabul edilmiştir.

Üst irtifada yer alan katmanlara çıkıldıkça engellerden bir kısmının yok olduğu gözlemlenmektedir. Örneğin Şekil 8.1 (a) dahilinde *E1*, *E2* ve *E3* olarak gösterilen engelleri çevrelediği kabul edilen vektör alanlar bir üst katman olan ve Şekil 8.1 (b) ile gösterilen katmanda daha dar vektörel alanlar ile temsil edilebilirken, Şekil 8.1 (c) ile gösterilen katmanda ise *E2* ve *E3* etkisini kaybedecek şekilde ortadan kalkmıştır.

Çizelge 8.1: Hesaplama performansı karşılaştırması için örnek senaryo parametreleri.

Parametre	Değer
$size\ (Alan\ Boyut)$	8000x8000
$Res\ (Alan\ Çöznürlük)$	2
δ_c	1.0
γ_c	1.0
C_x, C_y	4200, 4200
β_c	Π
$\delta_{I(1)}$	1,3
$\gamma_{I(1)}$	0,8
$I_{x(1)}, I_{y(1)}$	4000, 7600
$\beta_{I(1)}$	Π/12
$\delta_{I(2)}$	1,2
$\gamma_{I(2)}$	0,8
$I_{x(2)}, I_{y(2)}$	600, 4000
$\beta_{I(2)}$	Π/8
$Slope$	0.5
ρ	10
$I_{x(3)}, I_{y(3)}$	7400, 5000
$\beta_{I(3)}$	Π/8
$\delta_{I(3)}$	1,1
$\gamma_{I(3)}$	0,8

Bu durumda Şekil 8.1 kapsamında ele alınan üç katmandaki oluşumları YPA ile yol planlaması açısından irdeleyebiliriz. İkinci katman dahilinde yer alan $İHA_2$ için gerçekleştirilecek yol planlamasının hesaplanabilmesi için üç adet sabit ve üç adet hareketli (diğer platformlar, çarpışmayı önleme amaçlı) olmak üzere altı adet engel modellemesine gereksinim duyulacaktır. Ayrıca beş adet de doğru hareketi planlayabilmek için faydalanılmak üzere toplam 11 adet alt potansiyel alan değerinin uygun sınır fonksiyonlar ile sınırlandırılarak toplam vektör alanın hesaplanmasına ihtiyaç duyulmaktadır. Bu kapsamda örnek senaryoyu oluşturmak için faydalanılan sistem parametreleri Çizelge 8.1 ile gösterilmiştir.

Çalışmanın bundan sonraki kısmında parametre değerleri Çizelge 8.1 ile gösterilen ve Şekil 8.1 (b) ile temsil edilen vektör alının hesaplanma performansı karşılaştırmalı olarak farklı mimariler ile ortaya konulmuştur.

8.2 Seri Hesaplama Performansı

Paralel hesaplama performanslarına geçilmeden önce referans oluşturulabilmesi ve karşılaştırmanın sağlıklı olarak yapılabilmesi amacıyla farklı ölçeklerdeki alanlar içinde yer alan (çözünürlük değeri bir kabul edilmiştir) tek bir çeken nitelikteki

potansiyel kaynağın oluşturduğu vektör alanının hesaplanması performansı irdelenmiştir.

Çizelge 8.2: Tek bir çeken potansiyel kaynak içeren alanın vektör alan değerlerinin seri hesaplama performansı.

Alan	Toplam Hesaplama Süresi (s.)	Bir Hücrenin Hesaplama Süresi (ns.)
500	0,017	68
1000	0,068	68
2000	0,264	66
3000	0,563	63
4000	1,087	68
5000	1,490	60
6000	1,833	51
7000	2,176	44
8000	2,620	41
9000	2,963	37
10000	3,307	33

Denklem (3.17)'de yer alan harmonik fonksiyon ile ortaya konulan çeken yönlü potansiyel alanın Denklem (3.19)'da yer aldığı şekilde iki boyutlu gradient işlemine tabi tutularak alanın vektör alana dönüştürülmesi sağlanmaktadır. Denklem (3.46) ve (3.47)'den faydalanılarak alanda yer alan mobil platformları yönlendirebilecek nitelikte yön ve sürat komut tablolarının üretilebilmesi için Çizelge 2.4 ile açıklanan donanım ve yazılımlar vasıtasıyla üretilen yol planlaması seri hesaplama performans değerleri Çizelge 8.2 ile gösterilmiştir. Çizelge 8.2'den de görüleceği üzere hesaplanmak istenen vektör alan büyüdükçe, bir başka deyiş ile hesaplanan hücre sayısı arttıkça alanın tümünün hesaplanması için ihtiyaç duyulan süre artmaktadır. Örneğin 10.000 x 10.000 ölçeğinde içerisinde bir alanın hesaplanabilmesi için yaklaşık 3,3 saniye bir süreye ihtiyaç duyulmaktadır.

İkinci performans karşılaştırma amaçlı ortaya konan ve parametre değerleri Çizelge 8.1 ile gösterilen Şekil 8.1 (b) ile temsil edilen vektör alının hesaplanma performansı irdelenmiştir. Bölüm 2.5.1 kapsamında tanımlanan ve Çizelge 2.4 ile gösterilen seri hesaplama mimarileri üzerinde Bölüm 5.3 kapsamında ortaya konulan algoritmalar ile hesaplanmıştır.

Örnek senaryolar tek bir noktasal çeken kaynağın yer aldığı potansiyel alanın hesaplanmasına göre oldukça karmaşıktır. Örneğin senaryo S_2 olarak adlandırılmış

olup, 4000 x 4000 ölçeğinde birim çözünürlük değerine sahip bir alanda yer alan dört adet platformdan birisinin, içinde durağan olduğu kabul edilen üç adet engelden sakınarak formasyon şemasına uygun şekilde konumunu alması olarak tanımlanabilir. Platformun konumunu sağlıklı olarak alabilmesi istendiğinde YPA tabanlı yol planlaması genel yol planlaması şeklinde icra edildiğinden tüm alan harmonik fonksiyonlar ile tanımlanmış ve sigmoid fonksiyonlar ile sınırlandırılmış 11 adet alt temel potansiyel alan temsil edilmiştir.

Senaryo S_2 de dahil olmak üzere çalışma kapsamında tanımlanan ihtiyaçlar doğrultusunda farklı ölçek ve çözünürlük değerlerinde, değişik sayıda engel teşkil eden farklı sayılarda platformların oluşturdukları senaryoların (S_x) Çizelge 2.4 ile gösterilen seri hesaplama mimarileri üzerinde hesaplanması sonucunda elde edilen hesaplama performans değerleri Çizelge 8.3 ile gösterilmektedir.

Çizelge 8.3: Farklı senaryolar için seri hesaplama performansı.

Örnek Senaryo (S_n)	Alan Büyüklüğü ($size$)	Alan Çözünürlük Değeri (res)	Toplam Hücre Sayısı	Formasyon Şeması	Platform Sayısı	Durağan Engel Sayısı ($\#E$)	Çeken Potansiyel Sayısı ($\#F_{c_{rot}}$)	İten Potansiyel Sayısı ($\#F_{I_{rot}}$)	Saat Yön. Dönen Potansiyel Sayısı ($\#F_{rp_{rot}}$)	Saat Yön. Tersine Dönen Potansiyel Sayısı ($\#F_{rn_{rot}}$)	Katman Sayısı ($\#L$)	Alanın Toplam Hesaplama Süresi (t_A) (µs.)	Bir hücre Hesaplama Süresi (t_h) (µs.)
S_1	1000x1000	1	1.000.000	Box	4	0	1	3	1	1	1	781.600	0,78
S_2	4000x4000	1	16.000.000	Box	4	3	1	6	1	1	1	14.015.741	0,88
S_3	4000x4000	1	16.000.000	Box	4	0	1	3	1	1	1	13.865.910	0,87
S_4	4000x4000	4	1.000.000	Box	4	0	1	3	1	1	1	777.260	0,78
S_5	4000x4000	20	40.000	Box	4	0	1	3	1	1	3	92.015	0,77
S_6	1000x1000	1	1.000.000	Trail	4	0	1	3	1	1	1	781.840	0,78
S_7	1000x1000	1	1.000.000	Trail	6	0	1	5	1	1	1	770.140	0,77
S_8	4000x4000	20	40.000	Trail	4	0	1	3	1	1	1	29.926	0,75
S_9	4000x4000	20	40.000	Trail	4	10	1	11	1	1	5	155.100	0,78
S_{10}	4000x4000	1	16.000.000	Trail	4	25	1	28	1	1	1	18.865.910	1,18
S_{11}	10000x10000	1	100.000.000	Box	4	1	1	4	1	1	1	77.965.946	0,78
S_{12}	10000x10000	20	250.000	Box	4	40	1	43	1	1	5	1.041.525	0,83
S_{13}	10000x10000	1	100.000.000	Box	4	40	1	43	1	1	1	269.616.305	2,70

Çizelge **8.3** kapsamında toplam 13 adet farklı senaryonun Çizelge 2.4 ile gösterilen seri hesaplama mimarileri üzerinde hesaplanma performansları gösterilmektedir. Alanın toplam hesaplama süresi ile verilen süre (t_A) alan içinde yer alan her bir formasyon dahilindeki platforma ait anlık (durağan) vektör alanın üretilmesi ve uçuş sürat ile baş komut tablolarının elde edilmesi için gereken süredir. Elde edilen performans sonuçları irdelendiğinde hesaplama performansına etki eden en büyük faktörün alan içinde hesaplanması gereken toplam hücre sayısı olduğu gözlemlenmektedir. Bir diğer performansa etki eden faktör ise alan içinde yer alan toplam alt potansiyel alan sayısıdır. Yüksek ölçekli ve düşük çözünürlük değerine sahip, dolayısıyla nispeten fazla hücre hesaplanmasına gereksinim duyan ve içeriğinde çok sayıda alt potansiyel alan barındıran çok katmanlı bir alanın hesaplanması en uzun süren alandır. Bu noktadan hareket ile S_3 ile S_{10} ve S_{13} arasında ortaya konulan örnek alanların hesaplama süresi saniye mertebesinin üzerinde sürmektedir.

Örneğin S_2 olarak adlandırılan senaryonun Çizelge 2.2 ile gösterilen tabloda özellikleri belirtilen Aerosonde platformları tarafından icra edilen bir senaryo olduğunu kabul edelim. Birim uzunluğun 20 metre dolayısıyla 4000 x 4000 ölçeğindeki bir alanın 80 km. x 80 km. olduğu kabul edildiğinde, alanın hesaplanması için gerekli süre yaklaşık 14 sn. olacaktır. Bu süre zarfında platformların hareketine devam ettiği gerçeğinden hareket edilir ise hesaplama için geçen sürede her bir platform yaklaşık 560 m. yer değiştirecektir. Bu durumda Çizelge 5.1 ile ifade edilen formasyon şemalarından her hangi birini platform aralığı 560 m. den az olacak şekilde uygulayamayız. Ayrıca 560 m. mesafeden yakın engellerden sakınmak amacıyla gereksinim duyulan yol planlamasının istenen zaman aralığında üretilmesi seri hesaplama yaklaşımı ile Çizelge 2.4 ile gösterilen seri hesaplama mimarileri üzerinde mümkün değildir.

8.3 Paralel Hesaplama Performansı

Izgara üzerinde yer alan her bir noktadaki vektörel kuvvetleri etkileyen farklı dinamik değişkenlerin tanımlaması yapılmıştır. Bu parametrelerin her bir değişiminde bütün vektörel alanın değerleri etkileneceğinden tekrar hesaplanması gereksinimi vardır. Bu durum YPA yaklaşımının hesaplanması yönünden ortaya

çıkan dar boğazlardan birisidir. Ayrıca HYİ bölgesi içinde alıcı İHA platformunun manevralarının daha hassas gerçekleştirilmesi istenildiğinde ızgara alanının çözünürlüğü mümkün olan en yüksek değerde seçilmelidir ki, yüksek çözünürlük değeri seçildiğinde hassaslaşan manevra yeteneğinin yanı sıra daha fazla hesaplama gücüne ihtiyaç duyulacaktır. Ayrıca dinamik özelliklere sahip formasyon amaçlı yol planlaması hesaplamaları değişik parametrelere bağlı olduğundan düzenli periyodik olarak tekrarlanan bir süreç değildir. Saha içinde yer alan bir diğer platformun hareketi veya hareketli bir engelin yer değiştirmesi ve hatta platformun yer değişikliği bile yol planlaması amacıyla tüm potansiyel alanın güncellenmesine neden olabilir.

S_2 olarak adlandırılan senaryonun Çizelge 2.2 ile gösterilen tabloda özellikleri belirtilen Aerosonde platformları tarafından icra edilen bir senaryo olduğu kabul edilebilir. Bu durumda formasyon uçuşunu icra eden platformlar arasındaki mesafe 200 m. ve platformların maksimum sürati 40 m/s olarak belirlenebilir. Yol planlamasının dört platform için en iyi ihtimal ile toplam 5 sn. içinde, her bir platforma ait yol planlamasının ise gerekli emniyet tedbirleri de düşünüldüğünde 1 sn. den kısa bir sürede tamamlanması gerekmektedir. Hesaplama performansının arttırılması için bu nedenle paralel hesaplama mimarilerinden yararlanılması ön görülmüştür. Vektör alanının paralel olarak hesaplanması amacıyla üç farklı programlama ortamından faydalanılmıştır.

8.3.1 MATLAB ortamında paralel hesaplama performansı

Vektör alanının paralel olarak hesaplanması amacıyla faydalanılan ilk programlama ortamı MATLAB tarafından sunulan Paralel Programlama Araç Takımı (Parallel Computing Toolbox) kütüphanesidir [92]. Bu kütüphane GPGPU programlama imkanını kütüphane kapsamında bulunan ve önceden geliştirilmiş paralel fonksiyonlar ile sağlamaktadır.

MATLAB Paralel Programlama Araç Takımı içinde yer alan ve grafik işlemciler üzerinde paralel olarak koşturma imkanı sağlanmış olan fonksiyonlardan birisi de gradient fonksiyonudur [92]. Gradient fonksiyonu vasıtasıyla paralel olarak vektör alanların hesaplanması gerçekleştirilebilir.

MATLAB ortamında paralel hesaplamanın icra edilebilmesi için Çizelge 2.4 ile gösterilen seri hesaplama mimarileri üzerinde Şekil 2.23 ile gösterilen, donanım

özellikleri Çizelge 2.10 ile ortaya konulan NVIDIA Tesla K20c grafik donanımından paralel programlama amacıyla faydalanılmıştır.

Çizelge 8.4: Tek bir çeken potansiyel kaynak içeren alanın vektör alan değerlerinin MATLAB ortamında paralel hesaplama performansı.

Alan	Toplam Hesaplama Süresi (s.)	Bir Hücrenin Hesaplama Süresi (ns.)	Seri Hesaplamaya Kıyasla Hızlanma Oranı
500	0,014	57	1,21
1000	0,029	29	2,37
2000	0,066	17	3,98
3000	0,134	15	4,20
4000	0,226	14	4,80
5000	0,345	14	4,32
6000	0,490	14	3,74
7000	0,668	14	3,26
8000	0,907	14	2,89

Çizelge 8.4 ile gösterilen sonuçlar daha önceden Çizelge 8.2 ile gösterilen tek bir çeken potansiyel alanın vektör alan değerlerinin seri olarak hesaplanması değerleri ile karşılaştırıldığında hesaplama performansının 4000 x 4000 ölçeğinde bir alan için yaklaşık 4,8 kat arttığını göstermektedir.

Çizelge 8.5: Vektör alan değerlerinin hesaplanabilmesi için ihtiyaç duyulan bellek miktarları.

Alan Ölçeği	Toplam Hücre Sayısı	Giriş Verisi Büyüklüğü (MB)	Çıktı Verisi Büyüklüğü (MB)	Toplam Bellek İhtiyacı (MB)
500	250.000	0,95	15,26	16,21
1000	1.000.000	3,81	61,04	64,85
2000	4.000.000	15,26	244,14	259,40
3000	9.000.000	34,33	549,32	583,65
4000	16.000.000	61,04	976,56	1037,60
5000	25.000.000	95,37	1525,88	1621,25
6000	36.000.000	137,33	2197,27	2334,59
7000	49.000.000	186,92	2990,72	3177,64
8000	64.000.000	244,14	3906,25	4150,39
9000	81.000.000	308,99	4943,85	5252,84
10000	100.000.000	381,47	6103,52	6484,99

Çizelge 8.5 ile farklı ölçeklerdeki iki boyutlu potansiyel alanlardan vektör alanların üretilmesi ve de bu vektör alanlar vasıtasıyla otonom platformun yol planlamasının icra edilmesine imkan sağlayacak hız ve yön tablolarının elde edilmesi için ihtiyaç

duyulan toplam bellek kapasiteleri gösterilmiştir. Toplam bellek ihtiyacı Denklem seti **(8.1)** ile gösterildiği şekilde hesaplanmıştır.

$$T\text{Bİ} = (THS\ x\ SFS\ x\ PS\ x\ TS) + (THS\ x\ SFS) \quad \textbf{(8.1)}$$

Denklem **(8.1)**'de yer alan "$T\text{Bİ}$" terimi ihtiyaç duyulan toplam bellek miktarını byte olarak, "THS" terimi hesaplanan alanda yer alan toplam hücre sayısını, "SFS" terimi her bir hücrenin potansiyel alan ve vektör alan değerlerinin single float veri tipinde saklanacağı kabul edildiğinden, dört byte değerini, "PS" terimi alanda yer alan ve yol planlaması icra edilecek platform sayısını ve "TS" terimi Algoritma 7 ile gösterildiği şekilde hız, uçuş başı, SX ve SY olmak üzere dört tablo tutulacağından dört değerini temsil etmektedir.

Toplam bellek ihtiyacı ile gösterilen değerler aynı zamanda farklı grafik işlemcilerin sınır değerlerini temsil etmektedir. Örneğin Çizelge 2.11 ile özellikleri ortaya konan NVIDIA GTX 480 grafik işlemcisinin bellek kapasitesi 1536 MB ile sınırlı olup sistem belleğine gidip gelinmeden sadece tek bir sistem belleği, grafik belleği transfer işlem icra edilerek gerçekleştirilebilinecek hesaplama en büyük 4000 x 4000 ölçeği için mümkündür. Aynı şekilde GTX 670 MX donanımı için bu alan 5000 x 5000 ile sınırlıdır.

NVIDIA Tesla K20c grafik donanımının grafik işlemci belleği Çizelge 2.10 ile gösterildiği şekilde 5120 MB'dır. Bu nedenle grafik işlemcinin belleği 8000 x 8000 ve üzerindeki alanların hesaplanması için yeterli olmamaktadır. Bu problem PCI veri yolu üzerinden haberleşerek eksik bellek alanının sistem belleği ile kaşılanması şeklinde çözülebilir. Ancak hiç şüphe yoktur ki EK A'da yer alan PCI veri yolu bağlantı hızı göz önünde bulundurulduğunda hesaplama veri transfer maliyetleri ile birlikte oldukça artacaktır.

Çizelge 8.3 kapsamında ortaya konulan örnek problemlerden bazıları MATLAB Paralel Hesaplama Araç Takımı vasıtasıyla çalışmanın Bölüm 6'da açıklanan paralel algoritmalardan faydalanarak hesaplanmış ve Çizelge 8.4 ile gösterilen hesaplama performansları elde edilmiştir.

Çizelge 8.6: Farklı senaryolar için MATLAB paralel hesaplama araç takımı ile paralel hesaplama performansı.

Örnek Senaryo (S_n)	Alan Büyüklüğü ($size$)	Alan Çözünürlük Değeri (res)	Alanın Toplam Hesaplama Süresi (t_A) (µs.)	Bir hücre Hesaplama Süresi (t_h) (µs.)	Seri Hesaplamaya Kıyasla Hızlanma Oranı
S_1	1000x1000	1	365.233,64	0,37	2,14
S_2	4000x4000	1	3.591.542,96	0,22	3,98
S_3	4000x4000	1	3.194.910,14	0,20	4,34
S_4	4000x4000	4	368.369,67	0,37	2,11
S_5	4000x4000	20	63.458,62	0,53	1,45
S_6	1000x1000	1	361.962,96	0,36	2,16
S_7	1000x1000	1	381.257,43	0,38	2,02
S_8	4000x4000	20	20.084,56	0,50	1,49
S_9	4000x4000	20	117.500	0,59	1,32
S_{10}	4000x4000	1	5.004.220,16	0,31	3,77
S_{11}	10000x10000	1	0,00	0,00	0
S_{12}	10000x10000	20	49.128,54	0,20	4,24
S_{13}	10000x10000	1	0,00	0,00	0

Seri hesaplama performansı neticesinde ortaya konulan ve S_2 olarak adlandırılan senaryonun Aerosonde platformları tarafından icra edilen bir senaryo olduğu kabulüne geri dönerek MATLAB Paralel Hesaplama Araç Takımı vasıtasıyla elde edilen sonuçları tekrar değerlendirebiliriz. Birim uzunluğun 20 metre dolayısıyla 4000 x 4000 ölçeğindeki bir alanın 80 km. x 80 km. olduğu kabul edildiğinde, alanın hesaplanması için gerekli süre yaklaşık 3,59 sn. olacaktır. Bu süre zarfında platformların hareketine devam ettiği gerçeğinden hareket edilir ise hesaplama için geçen sürede her bir platform yaklaşık 144 m. yer değiştirecektir. Bu durumda Çizelge 5.1 ile ifade edilen formasyon şemalarından her hangi birinin, platformlar arasındaki mesafe 144 m. den az olacak şekilde uygulanması mümkün değildir. 144 m. mesafeden yakın engellerden sakınmak amacıyla gereksinim duyulan yol planlamasının istenen zaman aralığında üretilmesi seri hesaplama yaklaşımı ile Çizelge 2.4 ile gösterilen sistem üzerinde MATLAB Paralel Hesaplama Araç Takımı vasıtasıyla ve NVIDIA Tesla K20c grafik işlemcisinin paralel programlama aracı olarak kullanıldığı durumlarda gerçekleştirilememektedir.

8.3.2 CUDA ile paralel hesaplama performansı

MATLAB ortamında programlama yapmak her ne kadar etkin ve kolaylıkla görsel grafik tarzı çıktıların üretilmesine imkan sağlasa dahi bir takım hesaplama performansı sorunlarını da bir arada getirmektedir. Çünkü MATLAB Paralel Programlama Araç Takımı kütüphanelerinde yer alan önceden geliştirilmiş paralel fonksiyonlar genel maksatlı programlama amacıyla hazırlanmış ve farklı uygulamaları destekleyebilecek nitelikteki hazır fonksiyonlardır. Ayrıca ne yazık ki bu fonksiyonların problemlere özgün olarak uygulama bazında özelleştirilme imkanı yoktur. Bu noktadan hareket ile hesaplama performansının etkilenebileceği değerlendirilmiş ve başta gradient fonksiyonu olmak üzere uygulamada ihtiyaç duyulan fonksiyonların CUDA ile özgün olarak geliştirilmesine karar verilmiştir.

Bu noktada uygulamaya özgün bir çözüm ortaya koyarak daha etkin hesaplama performansı elde etmek için doğrudan Bölüm 3.2 kapsamında ortaya konulan ve potansiyel alanların DX ile DY ile temsil edilen birinci dereceden türevleri ile vektör alanların hesaplanması yoluna gidilmiştir. Böylece Bölüm 6 kapsamında ele alınan paralel algoritmalar birebir uygulanabilir hale gelmiştir.

Çizelge 8.7: Tek bir çeken potansiyel kaynak içeren alanın vektör alan değerlerinin CUDA ile paralel hesaplama performansı.

Alan	Toplam Hesaplama Süresi (s.)	Bir Hücrenin Hesaplama Süresi (ns.)	Seri Hesaplamaya Kıyasla Hızlanma Oranı
500	0,007	29	2,36
1000	0,015	15	4,6
2000	0,042	10	6,34
3000	0,073	8	7,75
4000	0,098	6	11,1
5000	0,130	5	11,42
6000	0,162	5	11,31
7000	0,205	4	10,64
8000	0,246	4	10,67

CUDA ve C++ programlama dillerinden faydalarınılarak Microsoft Visual Studio 2010 yazılım geliştirme ortamında paralel hesaplamanın icra edilebilmesi için Çizelge 2.4 ile gösterilen seri hesaplama mimarileri üzerinde Şekil 2.23 ile gösterilen, donanım özellikleri Çizelge 2.10 ile ortaya konulan NVIDIA Tesla K20c grafik donanımından paralel programlama amacıyla faydalanılmıştır. Hesaplamanın

icra edilebilmesi için Bölüm 6.1.1 kapsamında ortaya konulan hücre bazlı paralel programlama yaklaşımından faydalanılmıştır.

Çizelge 8.7 ile gösterilen sonuçlar daha önceden Çizelge 8.2 ile gösterilen tek bir çeken potansiyel alanın vektör alan değerlerinin seri olarak hesaplanması değerleri ile karşılaştırıldığında hesaplama performansının 4000 x 4000 ölçeğinde bir alan için yaklaşık 11,1 kat arttığını göstermektedir.

Çizelge 8.3 Çizelge 8.3 kapsamında ortaya konulan örnek problemlerden bazıları özgün olarak CUDA ve C++ programlama ortamında geliştirilen fonksiyonlar vasıtasıyla çalışmanın Bölüm 6.1.1'de açıklanan hücre bazlı paralel hesaplama yaklaşımına uygun algoritmalardan faydalanarak hesaplanmış ve Çizelge 8.8 ile gösterilen hesaplama performansları elde edilmiştir.

Çizelge 8.8: Farklı senaryolar için CUDA ile paralel hesaplama performansı.

Örnek Senaryo (S_n)	**Alan Büyüklüğü ($size$)**	**Alan Çözünürlük Değeri (res)**	**Alanın Toplam Hesaplama Süresi (t_A) (µs.)**	**Bir hücre Hesaplama Süresi (t_h) (µs.)**	**Seri Hesaplamaya Kıyasla Hızlanma Oranı**
S_1	1000x1000	1	173.303,77	0,17	4,51
S_2	4000x4000	1	1.009.779,61	0,06	13,88
S_3	4000x4000	1	986.195,59	0,06	14,06
S_4	4000x4000	4	173.883,67	0,17	4,47
S_5	4000x4000	20	13.374,27	0,11	6,88
S_6	1000x1000	1	171.456,14	0,17	4,56
S_7	1000x1000	1	182.497,63	0,18	4,22
S_8	4000x4000	20	6.665,03	0,17	4,49
S_9	4000x4000	20	30.292,97	0,15	5,12
S_{10}	4000x4000	1	1.593.404,56	0,10	11,84
S_{11}	10000x10000	1	0,00	0,00	0,00
S_{12}	10000x10000	20	13.579,20	0,05	15,34
S_{13}	10000x10000	1	0,00	0,00	0,00

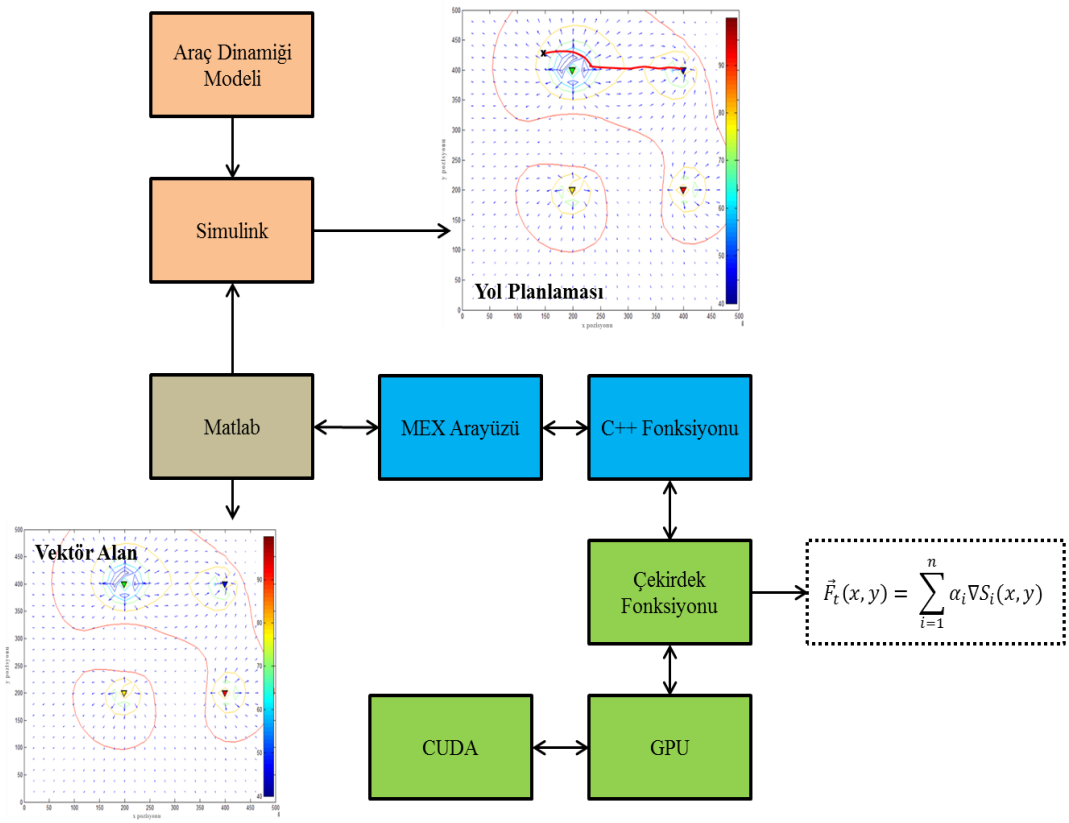
Seri hesaplama performansı neticesinde ortaya konulan ve S_2 olarak adlandırılan senaryonun Aerosonde platformları tarafından icra edilen bir senaryo olduğu kabulüne geri dönerek CUDA ve C++ programlama dilleri vasıtasıyla ortaya konulan özgün hesaplama neticesinde elde edilen sonuçları tekrar değerlendirebiliriz. Birim uzunluğun 20 metre dolayısıyla 4000 x 4000 ölçeğindeki bir alanın 80 km. x 80 km.

olduğu kabul edildiğinde, alanın hesaplanması için gerekli süre yaklaşık 1 sn. olacaktır. Bu süre zarfında platformların hareketine devam ettiği gerçeğinden hareket edilir ise hesaplama için geçen sürede her bir platform yaklaşık 40 m. yer değiştirecektir. Bu durumda Çizelge 5.1 ile ifade edilen formasyon şemalarından her hangi birinin platformlar arasındaki mesafe 40 m. den az olacak şekilde uygulamak mümkün değildir. 40 m. mesafeden yakın engellerden sakınmak amacıyla gereksinim duyulan yol planlamasının istenen zaman aralığında üretilmesi, seri hesaplama yaklaşımı ile Çizelge 2.4 ile gösterilen sistem üzerinde CUDA ve C++ programlama dillerinden faydalanılarak geliştirilen özgün hesaplama yaklaşımı vasıtasıyla ve NVIDIA Tesla K20c grafik işlemcisinin paralel programlama aracı olarak kullanıldığı durumlarda gerçekleştirilememektedir.

8.3.3 MATLAB ile CUDA programlama dilinde geliştirilen fonksiyonların bir arada kullanımı ve hesaplama performansı

Çalışmanın bundan önceki kısımlarında paralel hesaplama performansının MATLAB Paralel Hesaplama Araç Kiti kütüphanesinde yer alan ve GPU üzerinde gradient işleminin paralel şekilde koşturulmasına imkan sağlayan hazır fonksiyonlar yardımıyla ve de CUDA ile C++ programlama dillerinden faydalanılarak özgün olarak ortaya konulan hesaplama fonksiyonu ile ortaya konulmasına değinilmiştir. Paralel programlamanın MATLAB ortamında gerçekleştirilmesi elde edilen sonuçların görsel çıktılar ile temsil edilmesi, simülasyon ve uygulama çalışmalarında kullanılması gibi nedenlerden dolayı büyük kolaylıklar sağlamaktadır. Ancak bu esnada potansiyel alanın MATLAB fonksiyonlarının kullanımı ile hesaplanması özgün geliştirilen paralel hesaplama fonksiyonlarına göre Çizelge 8.4 ve Çizelge 8.7 ile ortaya konulduğu biçimde hesaplama performans kayıpları yaşanmaktadır.

Bir önceki bölüm kapsamında performans değerleri ortaya konulan ve CUDA ile geliştirilen fonksiyon MATLAB MEX ara yüzü yardımıyla MATLAB ortamında Şekil 8.2 ile gösterildiği biçimde kullanılabilir. Bu bölüm kapsamında MEX ara yüzünün nasıl tasarlandığına ve MATLAB ortamından CUDA fonksiyonun nasıl erişildiğine değinilmiştir [93].



Şekil 8.2: MATLAB ile CUDA programlama dilinde geliştirilen fonksiyonların bir arada kullanımı ve yol planlaması hesaplanması.

Şekil 2.18 ile gösterildiği biçimde, bu tez çalışması kapsamında C++ ve CUDA programlama dili ile geliştirilen çekirdek kodu bir üst seviye kod ile çevrelenmiştir. MATLAB MEX arayüzü adı verilen bir erişim yazılımı CUDA ve C++ programlama dilinde geliştirilerek düzenlenmiş ve çekirdek dahilinde yer alan fonksiyonlara erişim imkânı sağlanmıştır. MEX fonksiyonları MATLAB scriptleri ile çağrılarak koşturulmaktadır. Bu yazılım geliştirme ortamının sağlanması için faydalanılan yapılar aşağıda teker teker açıklanmıştır:

MATLAB script: Yazılım geliştirme mimarisi içindeki MATLAB script yapısı dıştaki tabaka olarak ele alınacaktır. Aşağıdaki sahte kod bloğu içinde gösterildiği gibi, ilk olarak potansiyel alanın parametreleri belirlenmiştir ve daha sonra MEX fonksiyon çağrısı bu parametrelerle gerçekleştirilmiştir. Fonksiyonun dönüş değerleri MATLAB standart fonksiyonları kullanarak görsel çıkışların hesaplanması için kullanılmaktadır.

Algoritma 13. MATLAB Script

13.1 Workspace ve diğer değişkenleri temizle

13.2 Potansiyel alan parametrelerini tanımla ve ilk değerlerini ata

area_dimension, area_res, x1, y1, alfa1,…

13.3 Girdi parametreleri ile birlikte MEX fonksiyon çağrısını gerçekleştir

[A,X,…]=Potential_field([area_dimension,area_size],[x1,y1,alfa1,…],…]

13.4 Fonksiyon geri dönüş değerlerini al ve çıktı değişkenlerine ata

13.5 Çıktı değişkenlerinden faydalanarak görsel çıktılar üret

quiver (A, X, Y, SX, SY)

Kod bloğunda gösterildiği gibi, MEX fonksiyonu "*Potential_field*" olarak adlandırılan ve giriş/çıkış parametreleri olan bir fonksiyondur. Giriş çıkış parametre değerleri önceden tanımlanmış MATLAB değişken uzayında tutulan değişkenlere atılırlar.

MATLAB MEX Fonksiyon Kodu: Aslında MATLAB MEX kodu, özel bir notasyon ile tanımlanmış giriş ve çıkış parametrelerine sahip C++ programlama dilinde yazılmış bir fonksiyonudur. Fonksiyon MATLAB scriptleri ile gerçekleştirilen komut çağrılarını destekler. CUDA derleyicisi (nvcc) ile MEX kodunun derlemesinden ardından PTX uzantılı derlenmiş program elde edilir. Derleyici donanıma özgü olmayan bir derlenmiş PTX programı üretir. Koşturma anında derlenmiş PTX programının belirli bir hedef GPU üzerinde yürütülmesi sürücünün sorumluluğundadır. PTX programı GPU üzerinde koşturulan kodların yanı sıra CPU üzerinde çalıştırılabilen birçok işlevi de içerebilir.

MEX fonksiyonu özel bir fonksiyon adına sahiptir ve fonksiyon parametreleri aşağıdaki sahte kod bloğunda gösterildiği gibi özel bir notasyon ile tanımlanır. Fonksiyon tanımlamasından sonra gerçekleştirilen ilk işlem grafik işlemcinin tanımlanması komutudur. Bu yapı, aşağıdaki kod bloğu ile gösterilmiştir:

Algoritma 14. Potential_field MEX Fonksiyonu

14.1 *void mexFunction(int nlhs, mxArray *plhs[], int nrhs, mxArray const *prhs[])*

14.2 Hesaplamanın yapılacağı GPU'yu belirle ve aktif et

mxInitGPU()

…

"*nlhs*" parametresi fonksiyonun kaç adet çıktı parametresi değeri olduğunu belirten değişken olup, çıktı parametrelerine ait değişkenlerin bellekteki adres alanları "*plhs*" işaretçi dizisi ile tutulmaktadır. "*nrhs*" parametresi ise fonksiyonun kaç adet girdi

parametresi olduğu bilgisini tutar ve bu parametreler dizi şeklinde "*prhs*" içinde işaretçi olarak saklanır.

CUDA Ana Kod: Temel CUDA programı süreci Şekil 6.1 ile gösterilmektedir. Bu yapı altı ana rutinden oluşur ve bunların beşi CPU tarafında çalışır. SIMD yapısındaki çekirdek kodu, GPU işlemci üzerinde koşturulan tek paralel bloğu oluşturur. Aşağıdaki sahte kod bloğu MEX yapısını ve CUDA ile uygun tanımlarını gösterir.

Algoritma 14. Potential_field MEX Fonksiyonu

14.2 Sistem belleğinde değişkenleri tanımla

```
alpha_CPU  = mxCreateDoubleMatrix(nrhs-1, 1, mxREAL);
...
```

14.3 Başlangıç parametre değerlerini al ve değişkenlere ata

```
in_area = mxGetPr(prhs[0]);
double a_size=in_area[0];
...
```

14.4 GPU belleğinde değişkenleri tanımla

```
Dp_alpha_GPU  = mxGPUCreateFromMxArray(alpha_CPU);
...
```

14.5 Sistem belleğinde yer alan değişken adreslerini işaretçilere ata

```
Hp_alpha_CPU   = mxGetPr(alpha_CPU);
Hp_ptype_CPU   = mxGetPr(ptype_CPU);
...
```

14.6 GPU belleğinde yer alan değişken adreslerini işaretçilere ata

```
double *p_kX_GPU , *p_kY_GPU, *p_eta_GPU, *p_fi_GPU;
p_kX_GPU = (double*)(mxGPUGetDataReadOnly(Dp_kX_GPU));…
...
```

Veri girişi gibi MEX yapısı ile belirtilen değerler CPU bellek değişkenlerine atanır. Bu değişkenlerin bellek adresi değerleri farklı değişkenler tarafından temsil edilmektedir. Daha sonra aynı şekilde GPU değişken tanımını yaparak, bellek veri GPU belleği sistem belleğinden transfer edilmiş olur.

Algoritma 14. MEX Fonksiyonu

14.7 GPU belleğinde başlangıç parametre değerlerini tutacak değişkenler oluştur

```
mwSize element_size1[1];
element_size1[0] = size;
...
```

14.8 GPU Belleğinde değişken dizi adreslerini işaretçilere ata

```
mxGPUArray *A_GPU, *X_GPU, *Y_GPU, *DX_GPU, … ;
A_GPU  = mxGPUCreateGPUArray(1, element_size1,
        mxDOUBLE_CLASS, mxREAL, MX_GPU_INITIALIZE_VALUES);
```

...

13.9 GPU belleğinde yer alan değişken dizi adreslerini işaretçilere ata

*Dp_A_GPU = (double *)(mxGPUGetData(A_GPU));*

*Dp_X_GPU = (double *)(mxGPUGetData(X_GPU));*

...

Üstteki kod bloğunda görüldüğü üzere bir sonraki işlem GPU bellek alanında tanımlanan ancak henüz dış giriş parametrelerinden değer alamayan değişkenlerin tanımlanmasıdır. GPU üzerinde doğal olarak değişken tanımları SIMD yapısına uygun olarak genellikle dizi şeklinde tanımlanır. Çekirdek rutinleri benzer verileri aynı diziler üzerinde toplanmıştır. GPU bellekte oluşturulan veri dizisinin her elemanı için başlangıç değeri olarak "*0*" değeri atanır.

Algoritma 14. MEX Fonksiyonu

14.10: çekirdek fonksiyonun koşturma parametrelerini belirle

if *(max_dimension<MaxThreadsPerBlock)*

{threads = max_dimension;}

else

{threads = MaxThreadsPerBlock; }

end if

14.11: *blocks = (max_dimension + threads – 1) / threads;*

dim3 threadsPerBlock (threads, 1, 1);

dim3 blocksPerGrid(blocks,1,1);

14.12: step_array çekirdek fonksiyonun koşturma parametrelerini belirle ve paralel fonksiyon çağrısını gerçekleştir

step_array<<<blocksPerGrid, threadsPerBlock>>>(d_A, Dp_A_GPU);

14.13: *mesh*grid paralel fonksiyon çağrısını gerçekleştir

meshgrid<<<blocksPerGrid2,threadsPerBlock>>>(Dp_X_GPU,

Dp_Y_GPU, Dp_A_GPU);

14.14: vector_field paralel fonksiyon çağrısını gerçekleştir

Vector_field<<<blocksPerGrid3,threadsPerBlock3>>>(p_kX_GPU,

p_kY_GPU, p_fi_GPU, p_eta_GPU, Dp_X_GPU, Dp_Y_GPU,

Dp_DX_GPU, Dp_DY_GPU, Dp_SX_GPU, Dp_SY_GPU);

...

Gerçekleştirilen bir sonraki işlem takip eden kod bloğunda gösterildiği şekilde ev sahibi kod üzerinde çekirdek çağırma parametrelerinin tanımlanması ve de çekirdek çağırma işleminin gerçekleştirilmesidir. Potansiyel alanın hesaplanması için bir diğer

değişle gradient operasyonunun kısmi türev işlemleri ile uygulayarak GPU üzerinde gerçekleştirmek için, üç farklı çekirdek fonksiyonu tanımlanmış ve bir arada kullanılmıştır. İlk iki çekirdek fonksiyonu MATLAB standart kütüphanesinde fonksiyon olarak tanımlanan "meshgrid" fonksiyonunun görevini icra eder. Sonuncusu ise vektör alanın hesaplanması için kısmi türev işlemlerini yürütmektedir.

MATLAB programlama ortamında "*gpuDevice*" komutunun çıktısı örnek NVIDIA GeForce GTX 670MX cihazı için Çizelge 2.9 ile gösterildiği şekildedir. Bu parametreler donanıma bağlı değişkenler olup, iplik, blok ve grid değerlerinin belirlenmesinde faydalanılmaktadır.

Algoritma 14. MEX Fonksiyonu

14.15: GPU belleğinden sistem belleğine veri transferini gerçekleştir

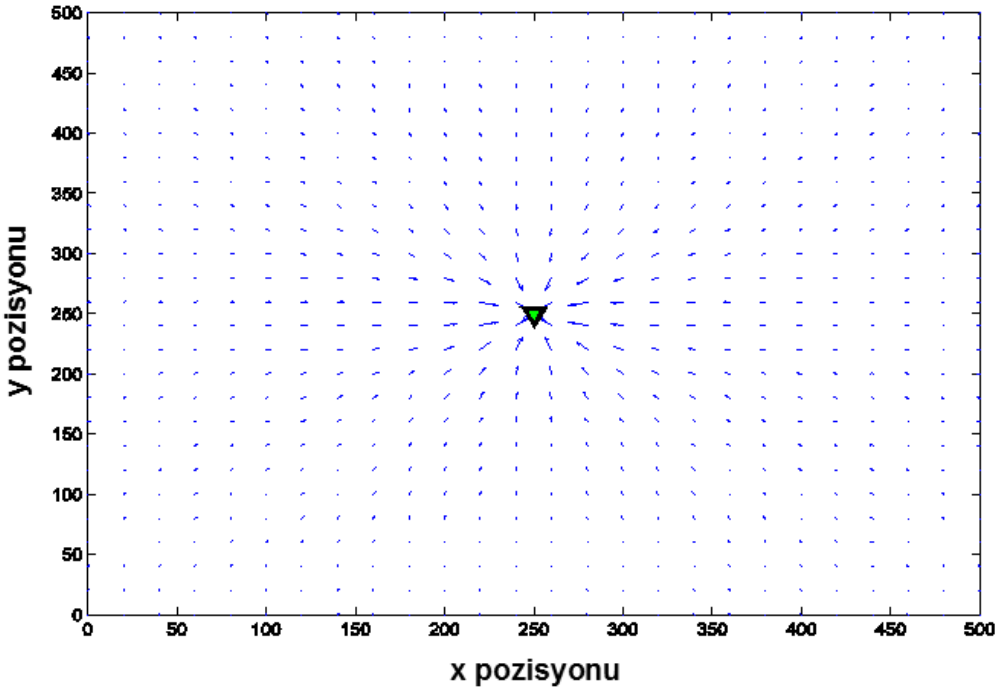
```
plhs[0] = mxGPUCreateMxArrayOnCPU (A_GPU);
plhs[1] = mxGPUCreateMxArrayOnCPU (X_GPU);
plhs[2] = mxGPUCreateMxArrayOnCPU (Y_GPU);
…
```

14.16: GPU belleğini boşalt

```
mxGPUDestroyGPUArray(Dp_alpha_GPU);
mxGPUDestroyGPUArray(Dp_ptype_GPU);
…
end
```

Çekirdek kodunun çağrılmasının ardından, geri dönüş değerleri cihaz belleğinden sistem belleğine kopyalanmalıdır. Veri transfer işleminin tamamlanmasının ardından GPU cihaz bellek alanının ayrılmasına daha fazla ihtiyaç yoktur, bu nedenle bu bellek alanı serbest bırakılır.

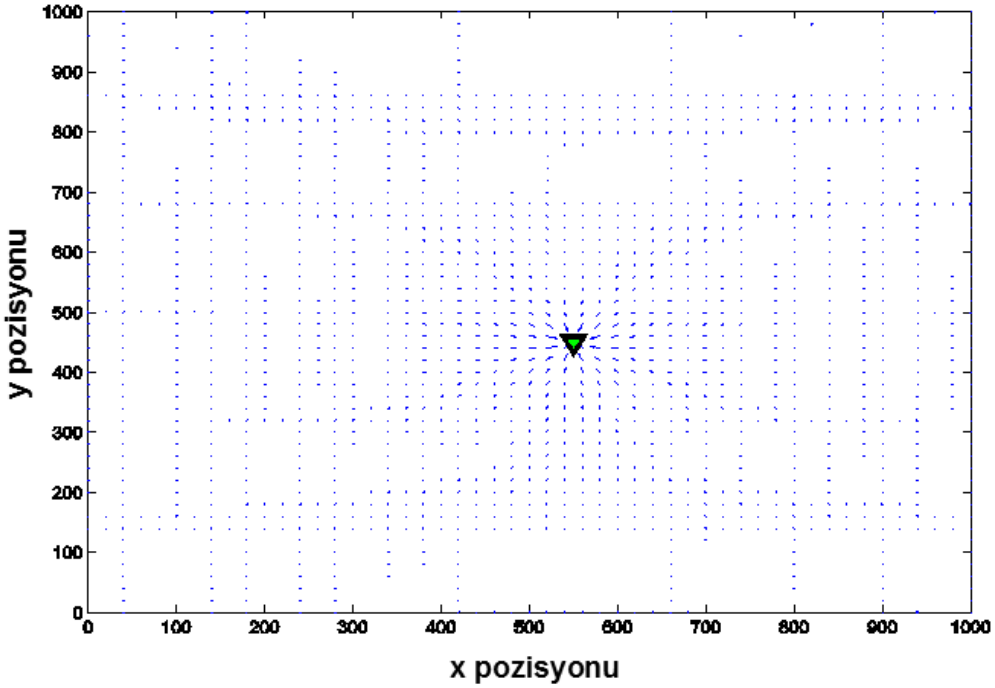
CUDA Çekirdek Kodları: Şekil 2.18 ile gösterildiği biçimde yazılım geliştirme ortamının en iç katmanını çekirdek kodları oluşturmaktadır. Çekirdek kodları farklı veriler üzerinde koşturulan SIMD paralel programlama yapısına uygun fonksiyon bloklarıdır. Amaç aynı kod bloğunu GPU mimarisi üzerinde yer alan çok sayıda işlemci üzerinde paralel olarak koşturarak hesaplama performansını arttırmaktır. Çekirdek kodu vasıtasıyla oluşturulan her bir ipliğe ait eşsiz bir iplik numarası ve blok numarası yaratılır. Her bir iplik belleğin farklı alanında yer alan verilere ulaşır ve de geri dönüş değerlerini farklı bellek alanlarına yazar. Tüm bu süreç GPU belleği üzerinde yürütülmektedir.



Şekil 8.3: Paralel olarak hesaplanan çeken noktasal kaynak vektör alanı gösterimi.

Potansiyel alanın hesaplanması amacıyla faydalanılan fonksiyonlar (*step_array*, *meshgrid* ve *vector_field*) önceki bölüm kapsamında yer alan *Algoritma 5 – Algoritma 6* ve *Algoritma 7* ile açıklanmıştır. Potansiyel alanında her bir (x, y) noktası GPU stream işlemci üzerinde çalışan bir iplik tarafından hesaplanır. Her bir iplik, iplik ve blok id parametrelerini kullanarak giriş akımı olarak tanımlanan bir bellek konumuna erişir. *Vector_field* algoritmasının içeriğinde tanımlanan kısmi türev işlemleri Çizelge 3.1 ile belirtilen denklemleri yerine getirmektedir. Bu yapı SIMD tipinde GPGPU tabanlı bir süreçtir. Vektör alanının toplam değeri, her bir (x, y) noktası için tüm toplam değerleri hesaplandıktan sonra hesaplanabilir. Her bir (x, y) noktası için tüm toplam değerleri hesaplandıktan sonra senkronize işlemi gerçekleştirilir ve platform komutları (baş ve hız) Bölüm 3.5 kapsamında açıklandığı şekilde komut tabloları oluşturmak için paralel olarak hesaplanabilir.

GPU üzerinde GPGPU teknikleri ile MATLAB CUDA desteğinden faydalanılarak paralel biçimde hesaplanmış olan tek nokta için çeken potansiyel alanın oluşturmuş olduğu vektör alan Şekil 8.3'de gösterildiği biçimde elde edilmiştir. Ölçek değeri 500 px. olan ve çözünürlük değeri 25 px. olarak belirlenen alan için, GPU üzerinde GPGPU teknikleri ile MATLAB CUDA desteğinden faydalanılarak paralel biçimde hesaplanmış olan tek nokta için çeken potansiyel alanın oluşturmuş olduğu vektör alan Şekil 8.3'de gösterildiği biçimde elde edilmiştir.



Şekil 8.4: Paralel olarak hesaplanan çeken noktasal kaynak vektör alanı gösterimi.

Ölçek değeri 1000 px. olan ve çözünürlük değeri 20 px. olarak belirlenen alan için, GPU üzerinde GPGPU teknikleri ile MATLAB CUDA desteğinden faydalanılarak paralel biçimde hesaplanmış olan tek nokta için çeken potansiyel alanın oluşturmuş olduğu vektör alan Şekil 8.4'de gösterildiği biçimde elde edilmiştir.

8.4 Farklı Grafik işlemci Mimarilerinin Hesaplama Performansı

CUDA tabanlı programlama performansının donanım özelliklerine bağlı olduğu, paralel koşturulan programın Çizelge 8.9 ile gösterilen değişik yeteneklere göre hesaplama performansında faklılıklar gösterdiği daha önceki bölümlerde açıklanmıştır.

Bu bölümde çalışma kapsamında ortaya konulan YPA tabanlı paralel programlama ile gerçekleştirilmiş yol planlama yaklaşımının farklı grafik işlemciler üzerindeki hesaplama performansı ele alınacak ve Çizelge 8.9 ile ortaya konulan donanımsal farklılıkların hesaplama performansını nasıl etkilediğine değinilecektir. Yukarıdaki Çizelge 8.9 kapsamında taralı olarak gösterilen hücre değerleri o satırdaki en iyi ve en kötü sonuçları göstermektedir.

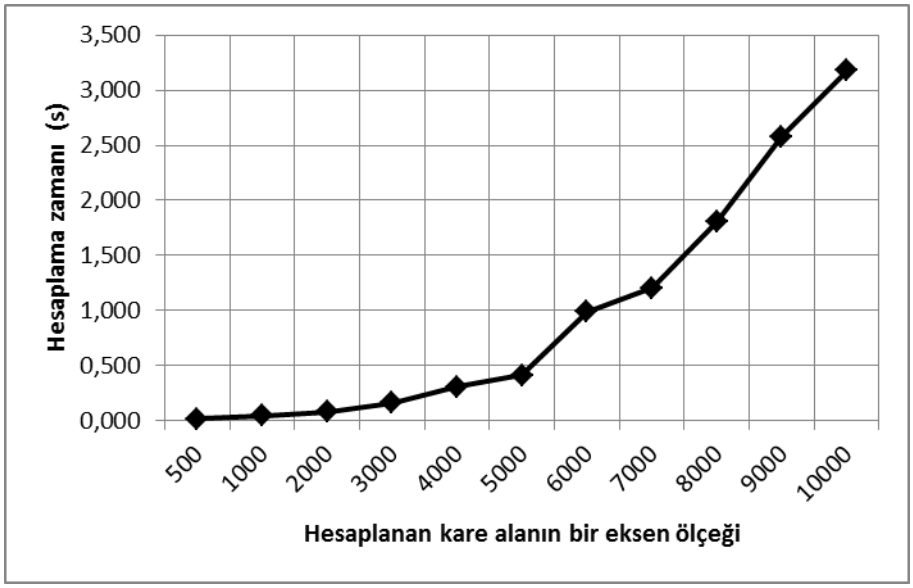
Çizelge 8.9: NVIDIA grafik işlemcilerin karşılaştırılması [58][59][60][61].

	GTX 480	**GTX670MX**	**Quadro 1000M**	**K20C**
Mikroişlemci Mimarisi	FERMI	KEPLER	FERMI	KEPLER
Hesaplama Kabiliyeti	2	3	2,1	3,5
Sürücü Versiyonu	6,5	6,5	5	6,5
Maks. Blok başına iplik sayısı	1024	1024	1024	1024
Blok başına paylaşımlı bellek miktarı	49152	49152	49152	49152
Maks. İplik organizasyonu	[1024 1024 64]	[1024 1024 64]	[1024 1024 64]	[1024 1024 64]
Maks. Grid organizasyonu	[65535 65535 65535]	[65535 65535 65535]	[65535 65535 65535]	[65535 65535 65535]
Toplam Cihaz Belleği (MByte)	1536	2048	2048	5120
Multiprocessor Adeti	15	7	2	13
Saat Hızı (MHz)	1400	600	700	1500
Güç İhtiyacı	225 W	75 W	45 W	225 W
Bellek Erişim Hızı	177 GB/s 384 Bit	67.2 GB/s 192 Bit	28.8 GB/s 128 Bit	208 GB/s 320 bit
Toplam Çekirdek Sayısı	480 Cores	960 Cores	96 Cores	448 Cores

Grafik işlemcileri özellikle harcadıkları güç miktarlarına bağlı olarak, kullanım yerleri açısından iki gruba ayırmak mümkündür. Bunlardan ilki mobil grafik işlemciler olup, bu tip işlemciler platform üzerinde yer alabilecek ölçü, güç tüketimi ve ağırlık gibi fiziksel özelliklere sahiptirler. İkinci grup grafik işlemciler ise daha çok masaüstü veya iş istasyonlarında kullanılan nispeten daha büyük ölçekli, daha fazla güç tüketen ve ağırlıkları olan işlemci grubudur. Bu tür işlemciler ise platformun üzerinden çok kontrol merkezinde yer alarak işlemlerin sonucunda oluşturulan yol planlamasının platforma aktarılması ile çalışabileceği değerlendirilen işlemcilerdir. Şimdi bu işlemciler üzerinde YPA tabanlı paralel yol planlaması yaklaşımının hesaplama performansını değerlendirebiliriz.

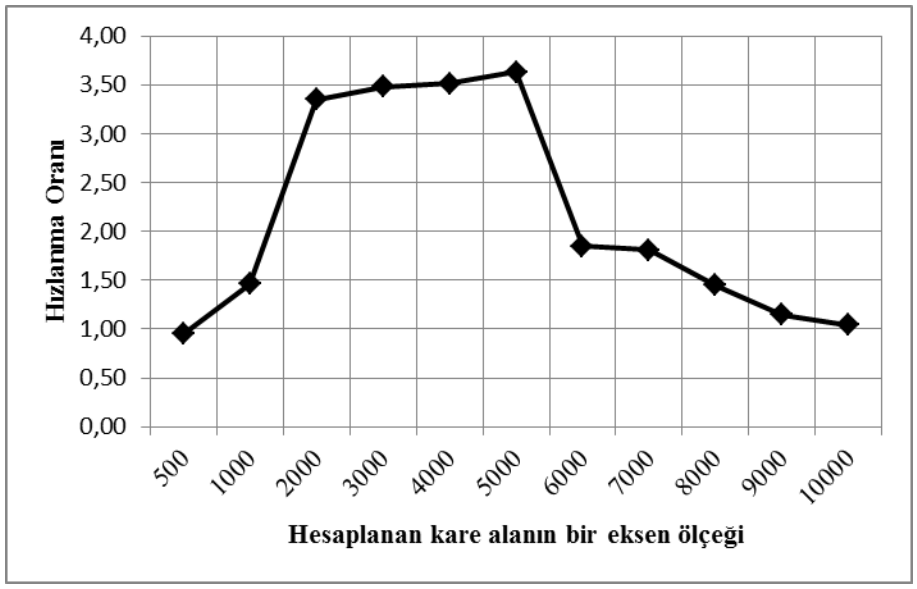
8.4.1 Mobil GPU mimarileri hesaplama performansı

Hesaplama performasının değerlendirileceği ilk donanım NVIDIA Quadro 1000M grafik işlemcisidir. Grafik işlemcinin temel özellikleri Çizelge 2.8 ile gösterilmiştir. Bu donanım üzerinde önceki bölümler kapsamında geliştirilen algoritmalar derlenerek çalıştırıldığında aşağıdaki Şekil 8.5 ile gösterilen performans sonuçları elde edilmiştir.



Şekil 8.5: NVIDIA Quadro 1000M grafik donanımı üzerinde algoritma performansı gösterimi.

NVIDIA Quadro 1000M grafik donanımı Çizelge 8.9 kapsamında ortaya konulan karşılaştırma tablosunda da görüleceği üzere en az güç tüketen grafik işlemcisi olması yanında en yavaş bellek erişimine sahip, en az multiprocessor ve çekirdek sayısına sahip işlemcidir. Çizelge 8.5 ile ortaya konulduğu şekilde sistem belleğini kullanmadan tek transfer ile gerçekleştirebileceği en büyük alan ölçeği çözünürlük değeri bir kabul edildiğinde 5000 x 5000 ölçeğidir. Bu nedenle Şekil 8.5 ile gösterildiği üzere hesaplama süreleri bu seviyedeki alan büyüklüğünün üstüne çıktığında hızla yükselmektedir. Çünkü hesaplamanın gerçekleştirilebilmesi için grafik işlemci belleği ile sistem belleği arasında veri transferi gerçekleştirmeye başlamış ve bu işlemlerim maliyetleri hesaplama performansına olumusz etki etmiştir.

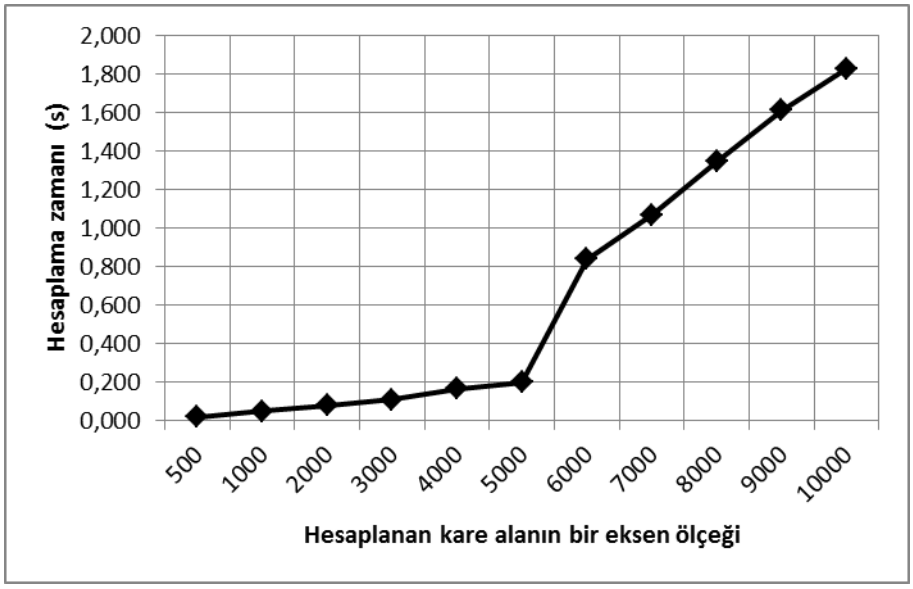


Şekil 8.6: NVIDIA Quadro 1000M grafik donanımı ve Intel i7 üzerinde karşılaştırmalı algoritma performansı gösterimi.

Şekil 8.6 ile gösterildiği üzere NVIDIA Quadro 1000M grafik işlemcisinin performansı 2000 x 2000 ölçeğinde ve birim çözünürlük değerinde Intel i7 merkezi işlem birimi ile sıralı hesaplama ile karşılaştırıldığında yaklaşık 3,3 kat, 5000 x 5000 ölçeğinde ve birim çözünürlük değerinde Intel i7 merkezi işlem birimi ile sıralı hesaplama ile karşılaştırıldığında yaklaşık 3,6 kat iyi performans üretmiştir.

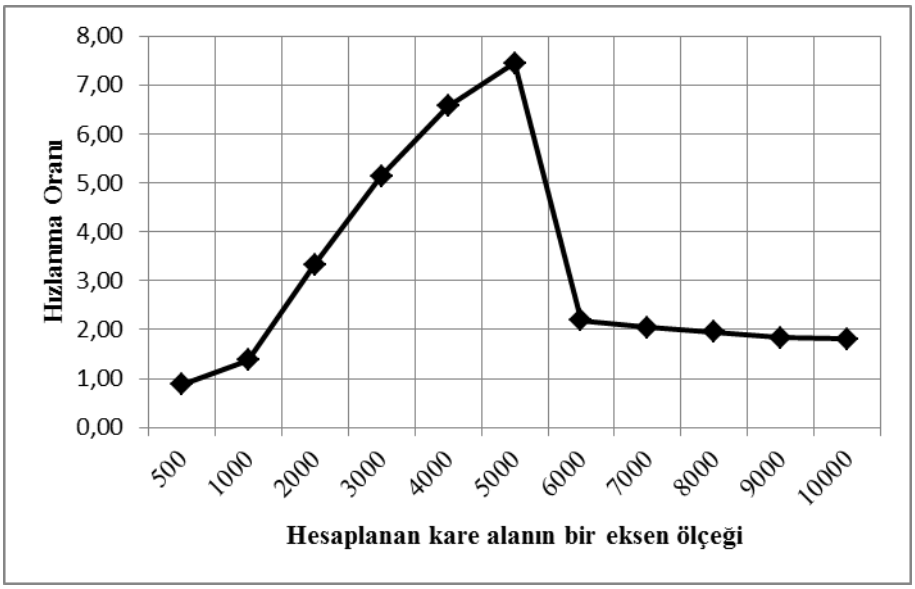
Seri hesaplama performansı neticesinde ortaya konulan ve S_2 olarak adlandırılan senaryonun Aerosonde platformları tarafından icra edilen bir senaryo olduğu kabulüne geri dönersek MATLAB ortamından çağrılan CUDA ve C++ programlama dilleri vasıtasıyla ortaya konulan NVIDIA Quadro 1000M üzerindeki özgün hesaplama yaklaşık 4,5 sn. içinde tamamlanmıştır. Bu süre zarfında platformların hareketine devam ettiği gerçeğinden hareket edilir ise hesaplama için geçen sürede her bir platform yaklaşık 180 m. yer değiştirecektir. Bu durumda Çizelge 5.1 ile ifade edilen formasyon şemalarından her hangi birini platformlar arasındaki mesafe 180 m. den az olacak şekilde uygulamak mümkün değildir. Ayrıca 180 m. mesafeden yakın engellerden sakınmak amacıyla gereksinim duyulan yol planlamasının istenen zaman aralığında üretilmesi, Çizelge 2.4 ile gösterilen sistem üzerinde CUDA ve C++ programlama dillerinden faydalanılarak geliştirilen özgün hesaplama yaklaşımı vasıtasıyla ve NVIDIA Quadro 1000M grafik işlemcisinin paralel programlama aracı olarak kullanıldığı durumlarda gerçekleştirilememektedir.

Hesaplama performasının değerlendirileceği bir diğer mobil donanım NVIDIA GTX 670MX Mobil grafik işlemcisidir. Grafik işlemcinin temel özellikleri Çizelge 2.9 ile gösterilmiştir.



Şekil 8.7: NVIDIA GTX 670MX Mobil grafik işlemcisi üzerinde algoritma hesaplama performansı gösterimi.

Şekil 8.7 ile gösterildiği üzere 5000 x 5000 ölçeğine kadar bir alanın paralel olarak modellenerek vektör alan üretilmesi performansı grafik işlemcinin belleğinin yeterli olmasından dolayı diğer seviyelerdeki alanlara istinaden daha iyi performans göstermektedir.



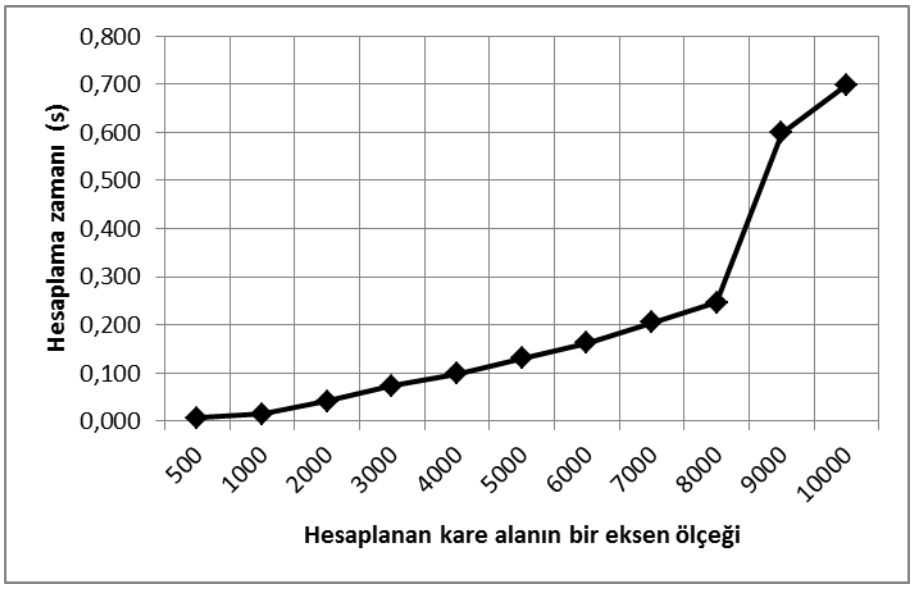
Şekil 8.8: GTX 670MX mobil grafik işlemcisi ve Intel i7 işlemcisi üzerinde karşılaştırmalı algoritma hesaplama performansı gösterimi.

Şekil 8.8 ile gösterildiği üzere NVIDIA GTX 670MX grafik işlemcisinin performansı birim çözünürlük değerinde ve 5000 x 5000 ölçeğinde Intel i7 merkezi işlem birimi ile sıralı hesaplama ile karşılaştırıldığında yaklaşık 7,4 kat iyi performans üretmiştir.

Seri hesaplam performansı neticesinde ortaya konulan ve S_2 olarak adlandırılan senaryonun Aerosonde platformları tarafından icra edilen bir senaryo olduğu kabulüne geri dönersek MATLAB ortamından çağrılan CUDA ve C++ programlama dilleri vasıtasıyla ortaya konulan NVIDIA GTX 670MX üzerindeki özgün hesaplama yaklaşık 2,2 sn. içinde tamamlanmıştır. Bu süre zarfında platformların hareketine devam ettiği gerçeğinden hareket edilir ise hesaplama için geçen sürede her bir platform yaklaşık 88 m. yer değiştirecektir. Bu durumda Çizelge 5.1 ile ifade edilen formasyon şemalarından her hangi birini platform aralığı 88 m. den az olacak şekilde uygulamak veya 88 m. mesafeden yakın engellerden sakınmak amacıyla gereksinim duyulan yol planlamasının istenen zaman aralığında üretilmesi seri hesaplama yaklaşımı ile Çizelge 2.4 ile gösterilen sistem üzerinde CUDA ve C++ programlama dillerinden faydalanılarak geliştirilen özgün hesaplama yaklaşımı vasıtasıyla ve NVIDIA GTX 670MX grafik işlemcisinin paralel programalama aracı olarak kullanıldığı durumlarda mümkün değildir.

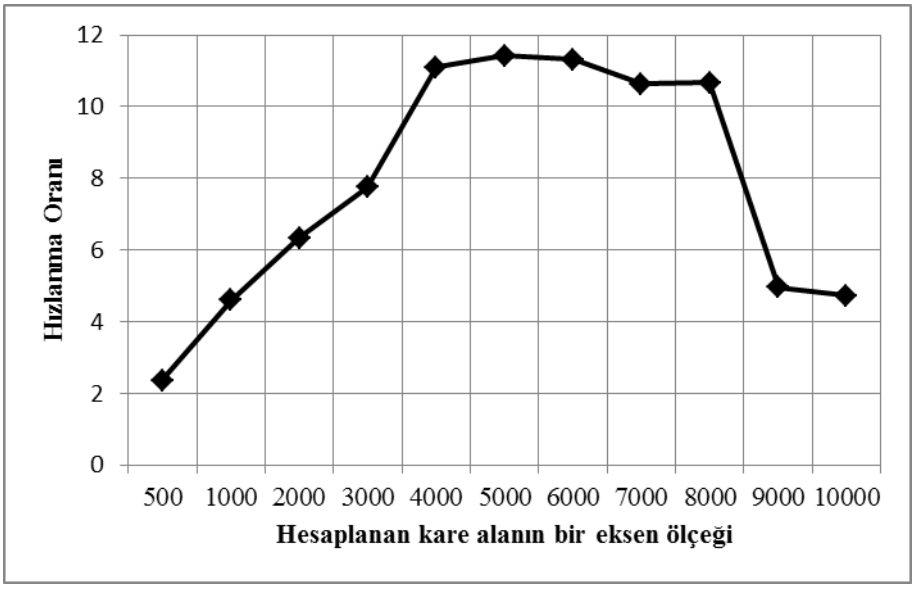
8.4.2 Masaüstü/iş istasyonu GPU mimarileri hesaplama performansı

Mobil grafik işlemcilerin yanısıra daha fazla güç tüketen, fiziksel olarak daha büyük ebatlarda ve daha çok sayıda multiprocessor ile çekirdek sayısına sahip genellikle masaüstü bilgisayar ile iş istasyonlarında kullanılan grafik işlemciler de mevcuttur. Bu tip grafik işlemcileri orta sınıf İHA platformlarının üzerinde kullanmak mümkün değildir. Daha çok yer istasyonlarında yer alabilirler. Ancak stratejik seviye ve üst sınıf İHA platformlarında araçların üzerinde yer alabilirler. Bölüm 8.3 kapsamında gerçekleştirilen çalışmalar NVIDIA Tesla K20c grafik işlemcisi üzerinde gerçekleştirilen sonuçlara istinaden üretilmiştir. Dolayısıyla bir önceki bölüm kapsamında mobil grafik işlemcileri ile elde edilen sonuçlardan daha iyi hesaplama performansları sergilemiştir. Ancak bir önceki bölüm kapsamında üretilen hesaplama performansına dair grafik sonuçların daha kolay karşılaştırılarak anlaşılabilmesi için görsel çıktılar tekrar bu bölüm kapsamında K20c donanımı için tekrar gösterilmiştir.



Şekil 8.9: NVIDIA TESLA K20C donanımı üzerinde algoritma performansı gösterimi.

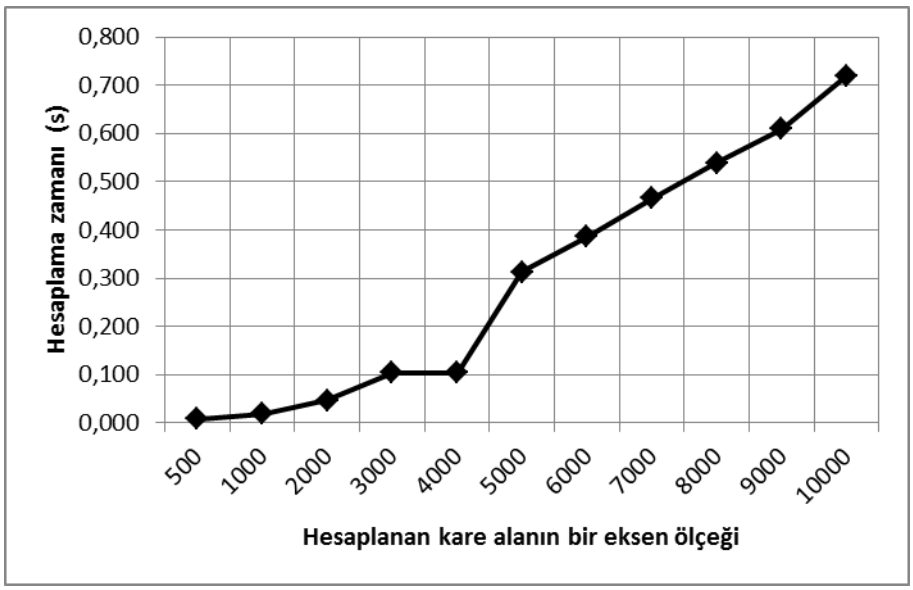
Şekil 8.9 ile tek bir çeken yönlü vektör alanın hesaplanmasına yönelik paralel algoritmaların NVIDIA Tesla K20c grafik işlemcisi üzerindeki performansı gösterilmiştir.



Şekil 8.10: NVIDIA TESLA K20C donanımı ve Intel i7 işlemcisi üzerinde karşılaştırmalı algoritma hesaplama performansı gösterimi.

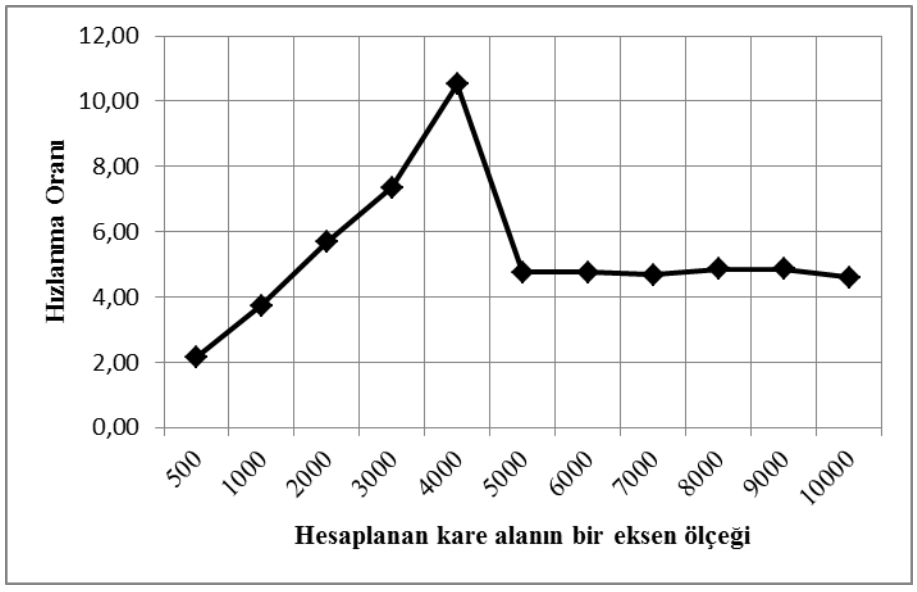
Şekil 8.10 ile NVIDIA Tesla K20c grafik kartı üzerinde koşturan paralel algoritmaların Intel i7 merkezi işlem birimi üzerinde sıralı hesaplama performansı ile karşılaştırması ortaya konulmuştur. Grafikten de görüldüğü üzere 5000 x 5000

ölçeğinde bir alanın hesaplanması esnasında yaklaşık 11,4 kat daha iyi performans değerleri elde edilmiştir.



Şekil 8.11: NVIDIA GTX 480 grafik donanımı üzerinde algoritma performansı gösterimi.

Şekil 8.11 ile tek bir çeken yönlü vektör alanın hesaplanmasına yönelik paralel algoritmaların NVIDIA GeForce GTX 480 grafik işlemcisi üzerindeki performansı gösterilmiştir.



Şekil 8.12: NVIDIA GeForce GTX 480 donanımı ve Intel i7 işlemcisi üzerinde karşılaştırmalı algoritma hesaplama performansı gösterimi.

Şekil 8.12 ile NVIDIA GeForce GTX 480 grafik kartı üzerinde koşturan paralel algoritmaların Intel i7 merkezi işlem birimi üzerinde sıralı hesaplama performansı ile

karşılaştırması ortaya konulmuştur. Grafikten de görüldüğü üzere 4000 x 4000 ölçeğinde bir alanın hesaplanması esnasında yaklaşık 10,5 kat daha iyi performans değerleri elde edilmiştir.

8.5 Çok Sayıda GPU İle Hesaplama Performansı

Çizelge 8.5 ile gösterilen biçimde, grafik kartlarının bellekleri sınırlıdır. Şayet vektör alanı hesaplanmak istenen potansiyel alanın gereksinim duyduğu bellek alanı grafik belleğin sahip olduğu bellek alanının üzerine çıkar ise grafik işlemci çok sayıda sistem belleği ile grafik belleği arasında veri transferi gerçekleştirmek durumunda kalacaktır. Ayrıca yüksek ölçekli ve içerisinde çok sayıda hücre barındıran potansiyel alanının hesaplanması esnasında Bölüm 6 kapsamında açıklanan paralel hesaplama algoritmalarına istinaden çekirdek başına düşen hesaplanması gereken hücre sayısı veya alt temel potansiyel alan sayısı çok büyüyebilir. Bu durum seri işlemcilere göre daha düşük hesaplama performansına sahip grafik işlemci çekirdeklerinde hesaplama sürecinin çok uzamasına ve hesaplama genel performansının olumsuz etkilenmesine neden olabilir. Bu durumun önüne geçmek için birden fazla grafik işlemcinin bir arada kullanılması yöntemine gidilebilir.

CUDA ile paralel programlamanın yapılabilmesi amacıyla birden fazla grafik işlemciden faydalanmanın birkaç yöntemi vardır. Bu yöntemlerin paralel hesaplama performansları genellikle grafik işlemcilerin ve grafik kartlarının birbirleri ile nasıl haberleştiklerine dolayısıyla sisteme nasıl bağlanarak dahil edildiklerine göre değişiklik göstermektedir. Birden fazla işlemcinin bir arada kullanılarak gerçekleştirilen GPGPU tabanlı paralel programlamalarda hesaplama performansını etkileyen en önemli dar boğaz grafik işlemcilerin birbirleri ile haberleşme, dolayısıyla birbirleri ve istemci arasında veri aktarım maliyetidir. Bu nedenle çalışmanın bundan sonraki kısımlarında birden fazla grafik işlemcinin bir arada kullanılmasıyla elde edilen hesaplama performansları grafik işlemcilerin sisteme olan fiziksel bağlantılarına göre gruplanarak irdelenmiştir.

8.5.1 Aynı grafik kartı üzerinde çok sayıda GPU ile hesaplama

Bir grafik kartı üzerinde birden fazla grafik işlemci ünitesi bir arada bulunabilir. Bu durum 2.5.4.3.1 başlığı altında ortak PCI veri yoluna bağlı işlemciler olarak açıklanmıştır. Örneğin NVIDIA GeFroce GTX TITAN Z grafik kartı üzerinde iki

adet grafik işlemci yer almaktadır. Bu iki grafik işlemciden aynı anda paralel programlama amacıyla faydalanaılabileceği gibi tek tek de faydalanmak mümkündür. Bu noktada karşılaşılan dar boğaz her iki işlemcininde ortak grafik belleğini kullanmasıdır. Ayrıca ortak veri yolu üzerinden sistem ile haberleşmek durumunda kalacaklardır. Dikkat edildiğinde YPA tabanlı yol planlamasının paralel olarak grafik işlemciler üzerinde gerçekleştirilmesinde karşılaşılan en büyük problemin grafik bellek miktarı ve de ana sistem ile haberleşme maliyetleri olduğu aşikardır. Bu durumda bu tip bir mimaride çok sayıda grafik işlemciden faydalanmanın önemli bir performans artışı sağlamayacağı değerlendirilmektedir.

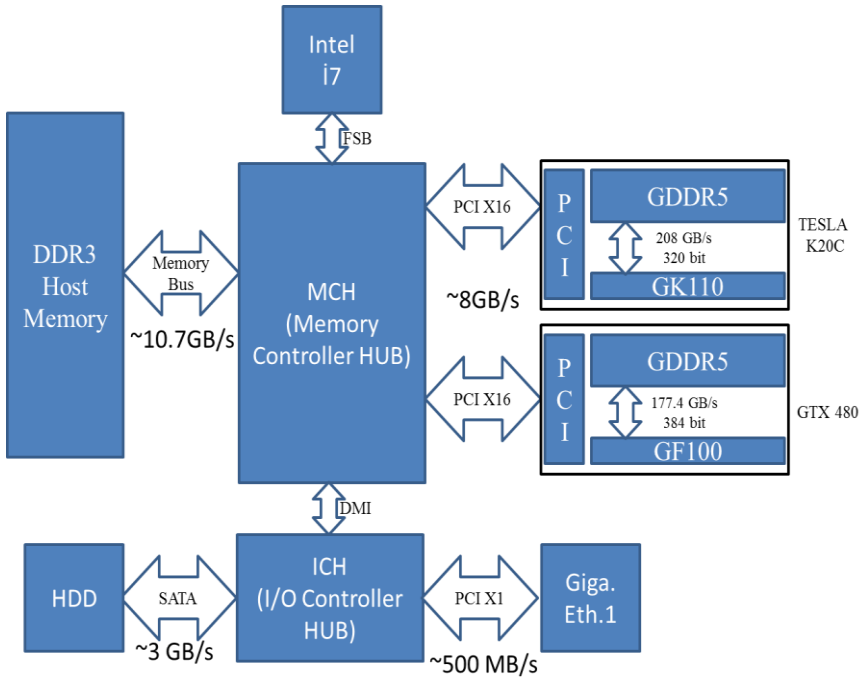
Bu tip bir paralel hesaplama maliyetinin ortaya konulabilmesi ve soonuca deneysel oalrak varılabilmesi için bu mimaride bir grafik işlemciye ihtiyaç duyulmaktadır. Maalesef ki bu tip bir grafik işlemci halihazırda elmizde bulunmadığı için bu deney gerçekleştirilememiştir.

8.5.2 Aynı ana makine üzerinde çok sayıda GPU ile hesaplama

Bir diğer çok sayıda grafik işlemciden aynı anda paralel programlama amacıyla faydalanılmasına imkan sağlayan yöntem 2.5.4.3.2 başlığı altında açıklandığı üzere aynı ana makine üzerinde yer alan farklı PCI veri yollarına bağlı işlemcilerden bir arada faydalılmasıdır.

Şekil 8.13: Aynı ana makine üzerinde yer alan farklı PCI veri yollarına takılmış üç NVIDIA GeForce GTX 480 donanımı gösterimi.

Bu yapı Şekil 8.13 ile gösterilmiştir. Şekilden de görüldüğü üzere üç adet GTX 480 grafik işlemci her birisi farklı PCI veri yollarına takılmak üzere aynı ana makine üzerinde yer almaktadır.



Şekil 8.14: Birden fazla grafik işlemcinin aynı ana makine üzerinde bağlantısının gösterimi.

Şekil 8.14 ile GPGPU tabanlı paralel programlama esnasında veri transferinin gerçekleştirildiği arayüzler ve bunların teorik haberleşme hızları gösterilmiştir. Bir önceki Şekil 8.13 ile gösterilen yapıdan farklı olarak bu defa farklı grafik işlemcilerin bir arada farklı PCI veri yollarına bağlanmadığı görülmektedir. Bu durumda her bir grafik işlemci kendisine tesis edilmiş olan PCI veri yolu üzerinden, diğer grafik işlemcilerden bağımsız olarak ana işlemci ile haberleşebilmektedir. Ayrıca her bir grafik işlemciye ait bağımsız grafik bellek alanı erişimi mümkün olmuştur. Böylece çok daha fazla sayıda akış işlemci ve çekirdek ile çok daha fazla bellek ihtiyacı duyan problemlerin hesaplanması imkanı oluşmuştur.

Her ne kadar her bir grafik işlemci farklı PCI veri yolu üzerinden birbirleri ile veya ana makine ile haberleşseler dahi Şekil 8.14 ile gösterildiği üzere yapı üzerindeki en yavaş teorik veri aktarım imkanı bu noktada gerçekleşmektedir. PCI veri yolu üzerinden ölçülen pratikteki haberleşme hızları EK-A'da yer alan tablolar ile gösterilmiştir.

Programlama açısından birden fazla grafik işlemciden faydalanmak birkaç farklı organizasyon ve de algoritmada değişikliğe gereksinim duyacaktır. Öncelikle yapılması gereken işlem YPA tabanlı yol planlaması açısından durum ele alındığında, vektör değerleri hesaplanmak istenen potansiyel alanının belirli

ölçülerde alt alanlara ayrılması ve hesaplama amaçlı kullanılan grafik işlemciler üzerine bu alanların dağıtılmasının organize edilmesidir.

Algoritma 15. Aynı ana bilgisayar üzerinde birden fazla GPU ile paralel hesaplama

```
…
15.1 Programlama amacıyla kullanılacak grafik işlemci sayısı kadar döngü oluştur.
     for g = i: gpuDeviceCount
15.2 Sırasıyla hesaplamada kullanılacak grafik işlemcileri aktif et
          cudaSetDevice (g);
15.3 Aktif grafik işlemci üzerinde çekirdek çağrılarını gerçekleştir
          kernel <<<…>>>(…);
15.4 Çekirdek çağrıları sonucunda elde edilen verileri asenkron olarak transfer et
          cudaMemcpyAsync(…);
15.5 Döngüden çık
     End
…
```

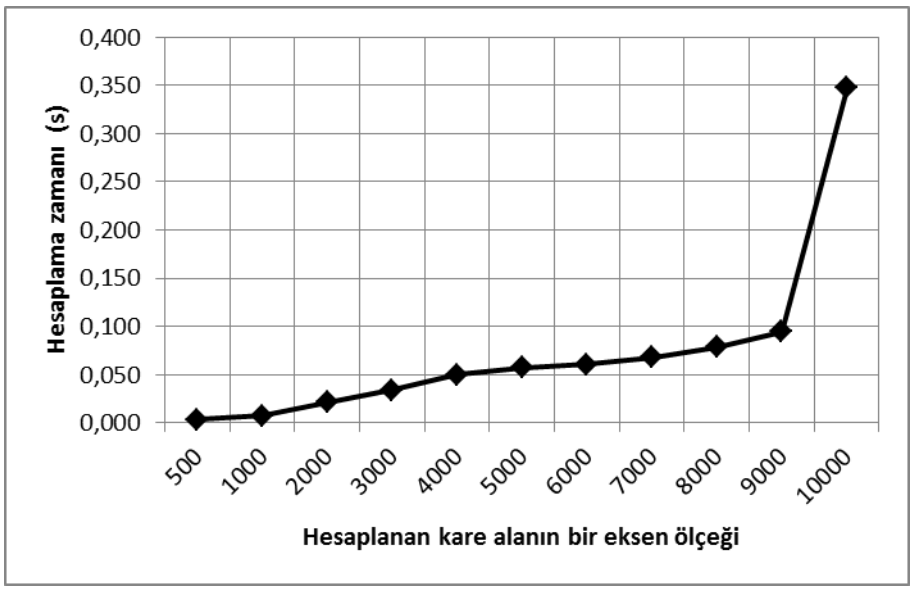
Bu işlemin akabinde paralel olarak koşturulacak olan iplikler asenkron olarak veri transferi imkanı sağlanarak işlemcilerin sırayla aktif edilerek çekirdek çağrılarının gerçekleştirilmesi ile sağlanabilir. Bu yapı *Algoritma 15* ile gösterilmiştir.

Çizelge 8.10: Aynı ana makine üzerindeki birden fazla grafik işlemci ile (K20c&GTX480) paralel hesaplama performansı.

Alan	Toplam Hesaplama Süresi (s.)	Bir Hücrenin Hesaplama Süresi (ns.)	Seri Hesaplama Karşında Hızlanma (x)
500	0,003	13	5,11
1000	0,007	7	9,48
2000	0,021	5	12,28
3000	0,034	4	16,67
4000	0,050	3	21,92
5000	0,057	2	26,03
6000	0,060	2	30,43
7000	0,068	1	32,17
8000	0,079	1	33,31
9000	0,094	1	31,48
10000	0,347	3	9,53

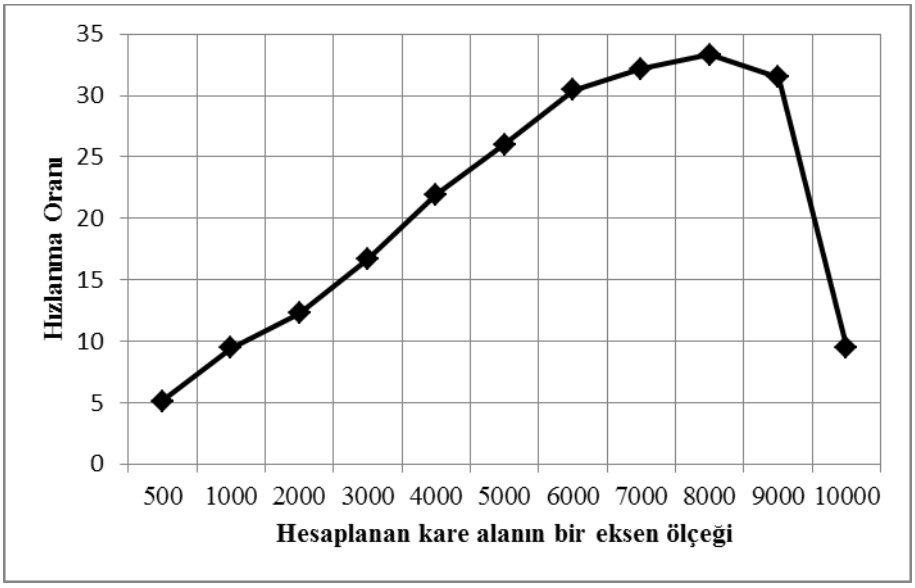
Çizelge 8.10 ile aynı ana makine üzerinde farklı PCI veri yollarına bağlanmış durumda olan NVIDIA K20c ve GTX 480 grafik işlemcilerinin bir arada kullanılması sonucu çeken yönlü potansiyel alanın hesaplanması soncunda elde edilen performans verileri gösterilmektedir. Çizelgeden de görüleceği üzere iki grafik

işlemcinin bir arada kullanılması sonucunda 9000 x 9000 ölçeğinde bir alanın da hesaplanması mümkün hale gelmiştir.



Şekil 8.15: Aynı ana makine üzerindeki birden fazla grafik işlemci (K20c ve GTX 480) ile paralel hesaplama performansı eğrisi gösterimi.

Çizelge 2.10 ile K20c grafik işlemcisinin özellikleri, Çizelge 2.11 ile GTX 480 işlemcisinin özellikleri gösterilmektedir. Çizelge 8.5 incelendiğinde 10000 x 10000 ölçeğinde bir alanın hesaplanabilmesi için gereksinim duyulan bellek ihtiyacı bu iki grafik işlemcinin toplam belleği ile karşılanamadığından PCI veri yolu üzerinden haberleşme ihtiyacı oluşmakta ve hesaplama performansında düşüş gözlemlenmektedir.



Şekil 8.16: Aynı ana makine üzerindeki birden fazla grafik işlemci (K20c ve GTX 480) ile paralel hesaplama hızlanma eğrisi gösterimi.

Bu durum Şekil 8.15 ile de ortaya konulmaktadır. Belirli bir formda alan ölçeğine bağlı olarak davranış gösteren hesaplama eğrisi 9000 ölçek değerini geçtiği anda hesaplama süresi olarak ani bir yükseliş göstermektedir.

Şekil 8.16'den de görüldüğü üzere seri hesaplama performansına kıyasla en iyi hesaplama performansının sağlandığı ölçek değeri 8000 x 8000 olarak tespit edilmiş olup bu noktadaki hesaplama performansı yaklaşık 33,3 kat olarak tespit edilmiştir.

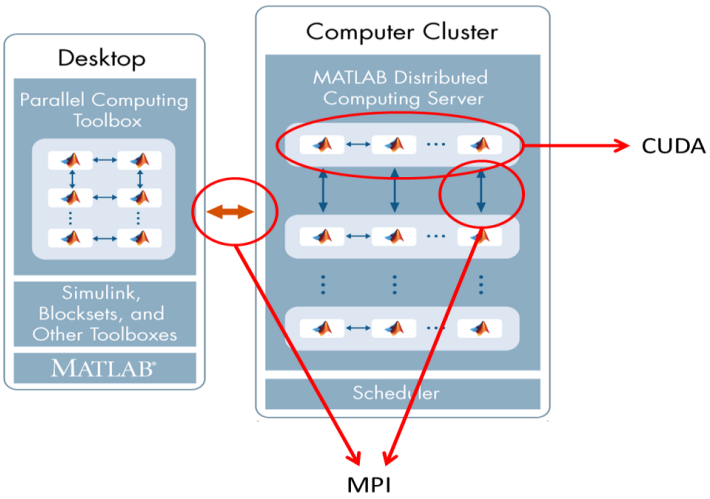
Daha önceki hesaplama performans değerlendirmeleri ile emsal oluşturması açısından Intel i7 işlemcisi üzerinde gerçekleştirilen seri hesaplama performansı ile karşılaştırıldığında elde edilen sonuçların 4000 x 4000 ölçeğindeki alan için yaklaşık 22 kat daha hızlı hesaplandığı görülmektedir.

Seri hesaplama performansı neticesinde ortaya konulan ve S_2 olarak adlandırılan senaryonun Aerosonde platformları tarafından icra edilen bir senaryo olduğu kabulüne geri dönerek, aynı ana makine üzerindeki birden fazla grafik işlemci (K20c ve GTX 480) ile paralel hesaplama elde edilen sonuçları tekrar değerlendirebiliriz. S_2 senaryosunun hesaplaması aynı ana makine üzerinde yer alan K20c ve GTX 480 işlemcilerinin bir arada kullanılması ile paralel olarak hesaplandığında seri hesaplamaya kıyasla elde edilen hızlanma 19,9 kat olarak elde edilmiştir. Bu durumda birim uzunluğun 20 metre dolayısıyla 4000 x 4000 ölçeğindeki bir alanın 80 km. x 80 km. olduğu kabul edildiğinde, alanın hesaplanması için gerekli süre yaklaşık 0,71 sn. olacaktır. Bu süre zarfında platformların hareketine devam ettiği gerçeğinden hareket edilir ise hesaplama için geçen sürede her bir platform yaklaşık 28 m. yer değiştirecektir. Bu durumda Çizelge 5.1 ile ifade edilen formasyon şemalarından her hangi birinin platformlar arasındaki mesafe 28 m. den az olacak şekilde uygulamak mümkün değildir. Ayrıca 28 m. mesafeden yakın engellerden sakınmak amacıyla gereksinim duyulan yol planlamasının istenen zaman aralığında üretilmesi, Çizelge 2.4 ile gösterilen sistem üzerinde özgün CUDA ile geliştirilmiş hesaplama uygulamaları vasıtasıyla ve NVIDIA Tesla K20c ile GeForce GTX 480 grafik işlemcilerinin paralel programlama aracı olarak bir arada kullanıldığı durumlarda gerçekleştirilememektedir.

8.5.3 Ağ ortamı üzerinde yer alan çok sayıda GPU ile hesaplama

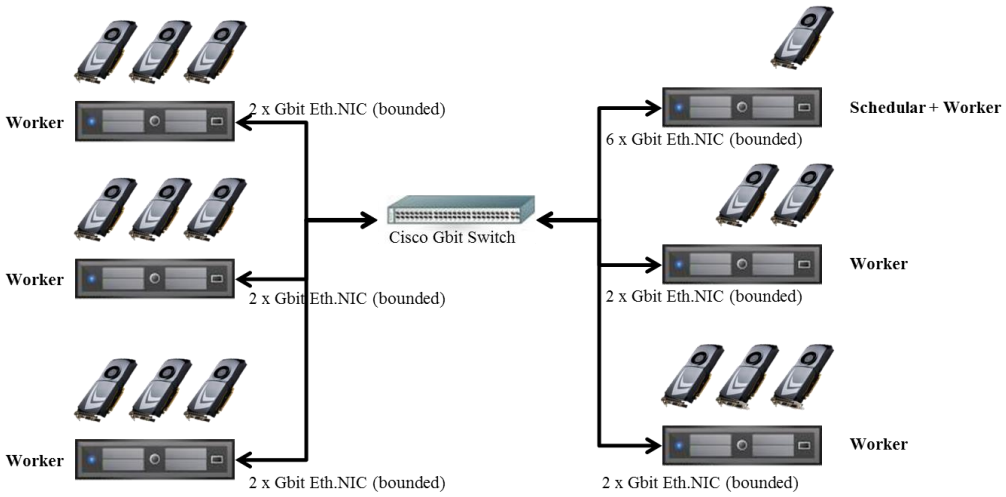
Çok sayıda grafik işlemciden faydalanarak GPGPU tabanlı paralel programlama yapma olanağı sağlamanın bir diğer yolu da bir ağ ortamı üzerinde yer alan çok

sayıdaki sunucu üzerinde yer alan grafik işlemcilerden faydalanmak olarak açıklanabilir. Bu yazılım geliştirme ortamını sağlamak için farklı yazılım altyapıları mevcuttur.



Şekil 8.17: MATLAB ortamında cluster teşkil edilmesinin ve CUDA proramlama icrasının yazılım desteği gösterimi [94].

Bölüm 8.3.3 kapsamında açıklanan gereksinimlere istinaden GPGPU tabanlı paralel programlama ortamının MATLAB ile entegrasyonunu Şekil 8.2 ile gösterilen şekilde sağlamak için çok sayıda grafik işlemcinin ağ üzerinde bir arada programlama amacıyla kullanılması için Şekil 8.17 ile gösterildiği biçimde MATLAB Distributed Server yazılım geliştirme altyapısından faydalanılmıştır [94]. MATLAB Distributed Server altyapısı CUDA tabanlı paralel programlama yapılması amacıyla bir bilgisayar grubu üzerinde paralel yazılım koşturmaya olanak sağlamaktadır.



Şekil 8.18: Ağ ortamında oluşturulan çok sayıda grafik işlemcinin bir arada kullanım konfigürasyonunun gösterimi.

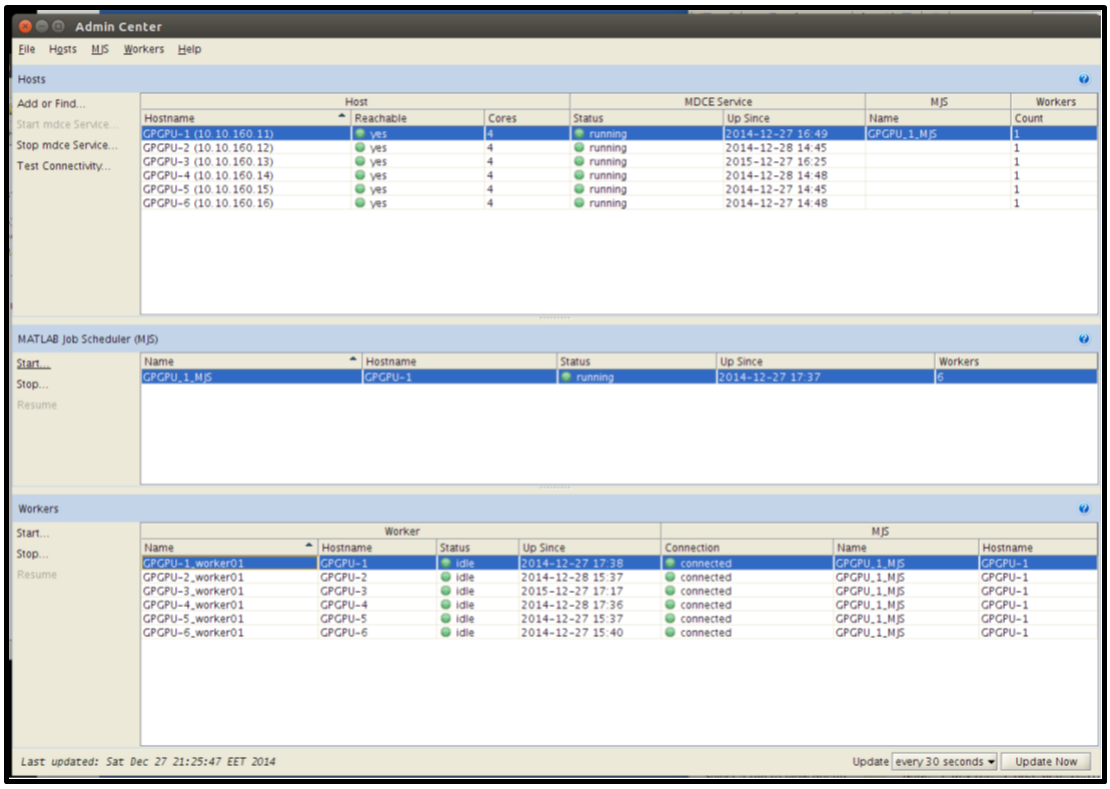
Şekil 8.18 ile tez çalışması kapsamında faydalanılacak olan ağ ortamı ve sunucular ile bu sunucular üzerinde yer alan NVIDIA grafik kartları gösterilmektedir. Sunuculardan bir adeti MATLAB Distributed Server mimarisi gereği planlayıcı (schedular) olarak planlanmış diğer sunucular ise işçi (worker) olarak konfigüre edilmiştir [94].

Çizelge 8.11: Ağ ortamında paralel hesaplama amaçlı oluşturulan grubun grafik işlemci özellikleri.

	NODE 1	**NODE 2**	**NODE 3**	**NODE 4**	**NODE 5**	**NODE 6**
GPU_1	GTX 480	GTX 480	GTX 480	GTX 480	GTX 480	GTX 480
GPU_2	-	GTX 480	Tesla C2070	GTX 480	GTX 480	GTX 480
GPU_3	-	-	GTX 480	GTX 480	Tesla C2050	GTX 480
GPU Belleği (GB)						
GPU_1	1,4996	1,4996	1,4996	1,4996	1,4996	1,4996
GPU_2	-	1,4996	5,9998	1,4996	1,4996	1,4996
GPU_3	-	-	1,4996	1,4996	1,4996	1,4996
GPU Akış İşlemci ve Çekirdek Sayıları						
GPU_1	15 x 32	15 x 32	15 x 32	15 x 32	15 x 32	15 x 32
GPU_2	-	15 x 32	14 x 32	15 x 32	15 x 32	15 x 32
GPU_3	-	-	15 x 32	15 x 32	15 x 32	15 x 32

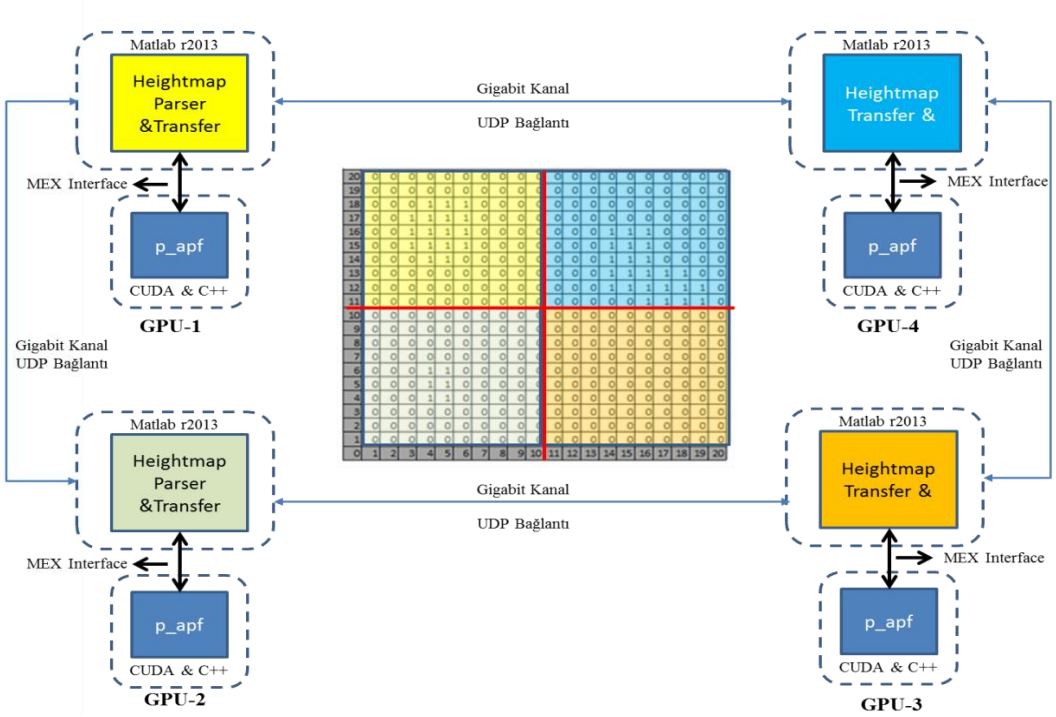
Çizelge 8.11 ile çalışma kapsamında faydalanılan ağ ortamı üzerinde yer alan sunucular ve bu sunucular üzerinde yer alan farklı PCI veri yollarına takılmış olan grafik işlemcilerin model, bellek miktarı ve akış işlemciler ile çekirdek sayıları gösterilmektedir.

MATLAB Distributed Server mimarisinde yer alan sunucular ve rolleri Şekil 8.19 ile gösterilmiştir. Sunucuların tamamı planlayıcı sunucu üzerinden yürütülmektedir. Çalışma kapsamında gerçekleştirilen testlerin icrası esnasında oluşturulan sunucu grubu üzerinde her hangi başka bir işlem veya program koşturulmamıştır.



Şekil 8.19: MATLAB dağıtılmış hesaplama sunucusunda grup dahilindeki sunucuların konfigürasyonunun gösterimi.

Oluşturulan sunucu grubu üzerinde Bölüm 6 ile açıklanan YPA tabanlı otonom yol planlamasına yönelik paralel algoritmalar koşturulmuştur.



Şekil 8.20: Ağ ortamı üzerinde yer alan birden fazla grafik işlemci ile hesaplama uygulamasının gösterimi.

Bir önceki bölüm kapsamında aynı ana makine üzerinde yer alan çok sayıda grafik işlemcisinin paralel programlama amacıyla bir arada kullanılmasına benzer biçimde alan verisi sunucu grubu üzerinde yer alan sunuculara ve bu sunucular üzerinde yer alan grafik işlemcilere belirli bir düzen dahilinde dağıtılmıştır.

Sunucu grubunda yer alan sunuculara yol planlaması icra edilecek olan alanın potansiyel değerlerinin hesaplanması ve vektör alanların üretilmesi amacıyla verinin dağıtımı sembolik olarak Şekil 8.20 ile gösterilmiştir. Engeller verilerini üzerinde tutan yükseklik haritası (heightmap) verileri de hesaplanan hücrelerin konumlarına göre parçalanmış ve sunucuların üzerinde yer alan grafik işlemci belleklerine aktarılmıştır.

Algoritma 16. Farklı sunucular üzerinde birden fazla GPU ile paralel hesaplama

```
…
16.1 MATLAB distributed server nesnesi oluştur
        MATLABpool open s
16.2 İşlem sürelerini tutacak değişken tanımlaması
        t1 = zeros(1, poolSize);
16.3 Dağıtılmış veri setlerini ayarla
         heightmap_d = codistribute (heightmap)
        heading_table_d = distributed (heading_table)
        speed_table_d = distributed (speed_table)
16.4 SIMD tipinde paralel hesaplama
        spmd (s)
16.5 Sunucudaki hesaplama süresini başlat
        tic;
16.6 Sunucu üzerindeki g sayısı kadar grafik işlemcide hesaplamayı gerçekleştir
        for g = i: gpuDeviceCount
                cudaSetDevice (g);
                kernel <<<…>>>(…);
                cudaMemcpyAsync(…);
        end
16.7 Süreyi durdur
        t1(n) = toc;
16.8 Hesaplama sonuçlarını geri transfer et
        heading_table = gather (heading_table_d)
        speed_table =  gather (speed_table_d)
16.9  end
16.10 MATLAB distributed server nesnesini sonlandır
        MATLABpool close
…
```

Verilerin sunucu grubu üzerine paylaştırılarak SIMD tipinde paralel bir programlama yapılmasına ve her bir sunucu üzerinde yer alan bir veya daha fazla sayıdaki grafik işlemciden faydalanarak hesaplamanın gerçekleştirilmesine yönelik olarak hazırlanan programa ait algoritma yapısı *Algoritma 16* ile gösterilmiştir. Algoritmadan da görüldüğü üzere ağ ortamı üzerinde yer alan sunucular dahilindeki grafik işlemcilerde gerçekleştirilen hesaplama sonucunda yol planlaması amacıyla faydalanılacak olan yön ve sürat komut tabloları planlayıcı sunucuya ağ ortamı üzerinden gönderilerek bir araya getirilmiştir.

Şekil 8.14 ile gösterilen biçimde GPGPU tabanlı paralel programlamanın icra edildiği sistem yapısı üzerindeki en büyük dar boğaz oluşturan nokta ağ ortamı üzerinden gerçekleştirilen haberleşmeden kaynaklanmaktadır. Doğal olarak hesaplanmak istenen alanın boyutu, alan çözünürlük değerinin sabit olduğu kabul edildiğinde, büyüdüğü sürece ağ ortamı üzerinden transfer edilen verinin de miktar olarak büyümesi anlamına gelecektir.

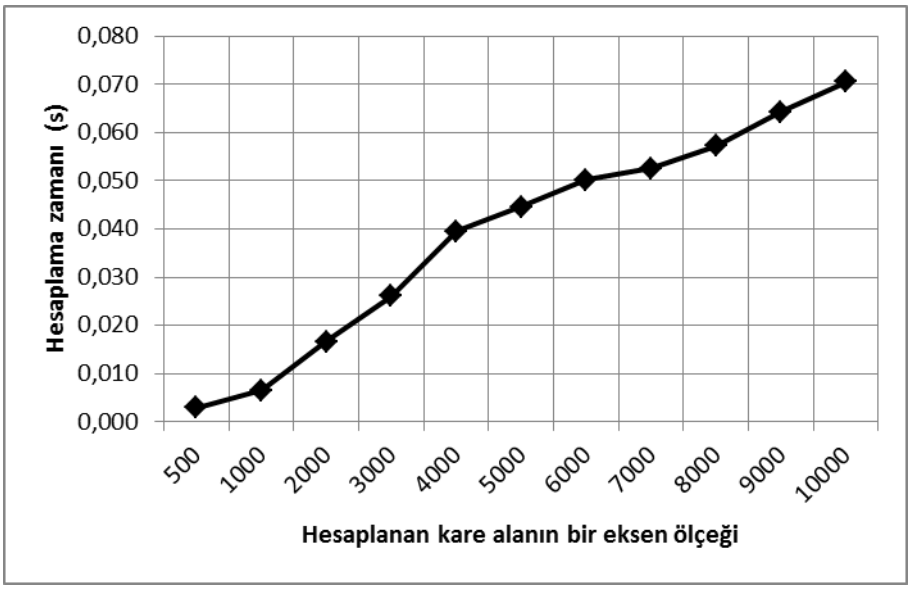
Ağ ortamı üzerinde yer alan sunucular üzerinde gerçekleştirilen YPA tabanlı yol planlaması sürecinin hesaplama performansının görülebilmesi için Bölüm 8.1 dahilinde açıklandığı üzere iki farklı test gerçekleştirilmiştir.

Çizelge 8.12: Ağ ortamındaki birden fazla grafik işlemci ile paralel hesaplama performansı.

Alan	Toplam Hesaplama Süresi (s.)	Bir Hücrenin Hesaplama Süresi (ns.)	Seri Hesaplama Karşında Hızlanma (x)
500	0,003	12	5,73
1000	0,007	7	10,33
2000	0,017	4	15,91
3000	0,026	3	21,58
4000	0,040	2	27,46
5000	0,045	2	33,46
6000	0,050	1	36,50
7000	0,053	1	41,44
8000	0,057	1	45,84
9000	0,064	1	46,15
10000	0,071	1	46,87

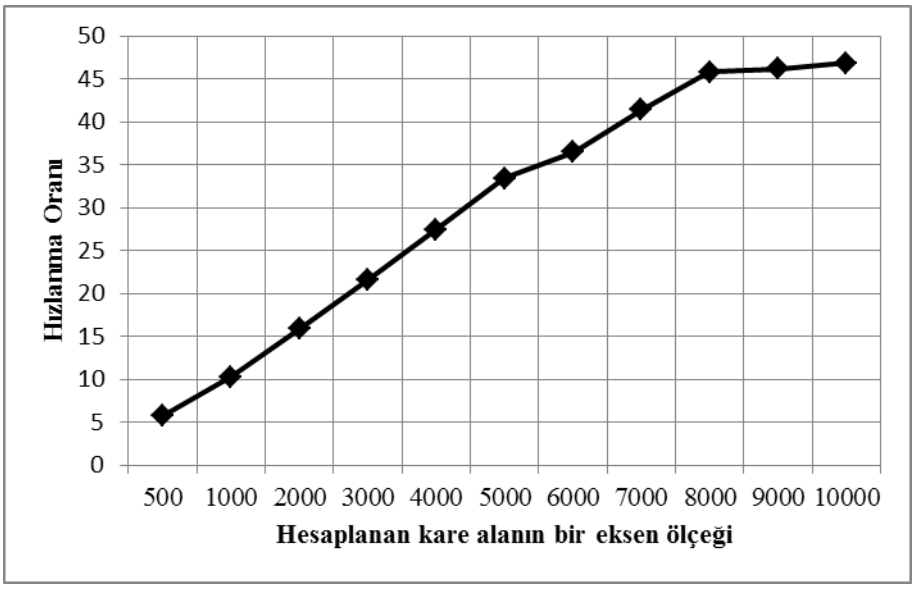
İlk performans değerlendirmesi kapsamında gerçekleştirilen çalışma kapsamında çeken yönlü noktasal potansiyel alan değişik ölçekteki çözünürlük değeri birim olan

alanlar içinde oluşturularak vektör alan hesaplaması gerçekleştirilmiş ve yön ile sürat tablolarının üretilmesi neticesinde bu işlemlerin aldığı zaman değerlendirilmiştir. Bu hesaplama performansları Çizelge 8.12 ile gösterilmiştir. Çizelge kapsamında elde edilen sürelere ağ ortamı üzerinden gerçekleştirilen haberleşme süreleri dahil edilmemiştir. Sadece her sunucu üzerinde gerçekleştirilen hesaplama ve veri transfer süreleri bu hesaplama performansları zamanlarına eklenmiştir. Şayet vektör alan değerlerinin ve oluşturulan komut tablolarının sunucu grubu üzerinde dağınık olarak tutulduğu kabul edilir ise en iyi hesaplama performansı Şekil 8.21'den görüldüğü üzere 10000 x 10000 ölçeğindeki alan için elde edilmiştir. Bu ölçeklerdeki potansiyel alan bundan önceki çalışmalar dahilinde grafik işlemcilerin bellek miktarlarının yetersiz olması nedeniyle hesaplanamamaktayken çok sayıda grafik işlemcinin bir arada kullanılması sonucunda yeterli grafik işlemci bellek miktarına ulaşılmıştır.



Şekil 8.21:Ağ ortamı üzerindeki birden fazla grafik işlemci ile paralel hesaplama performansı eğrisi gösterimi.

Daha önceki hesaplama performans değerlendirmeleri ile emsal oluşturması açısından Intel i7 işlemcisi üzerinde gerçekleştirilen seri hesaplama performansı ile karşılaştırıldığında elde edilen sonuçların 4000 x 4000 ölçeğindeki alan için yaklaşık 27,4 kat daha hızlı hesaplandığı Şekil 8.22'den görülmektedir.



Şekil 8.22: Ağ ortamı üzerindeki birden fazla grafik işlemci ile paralel hesaplama hızlanma eğrisi gösterimi.

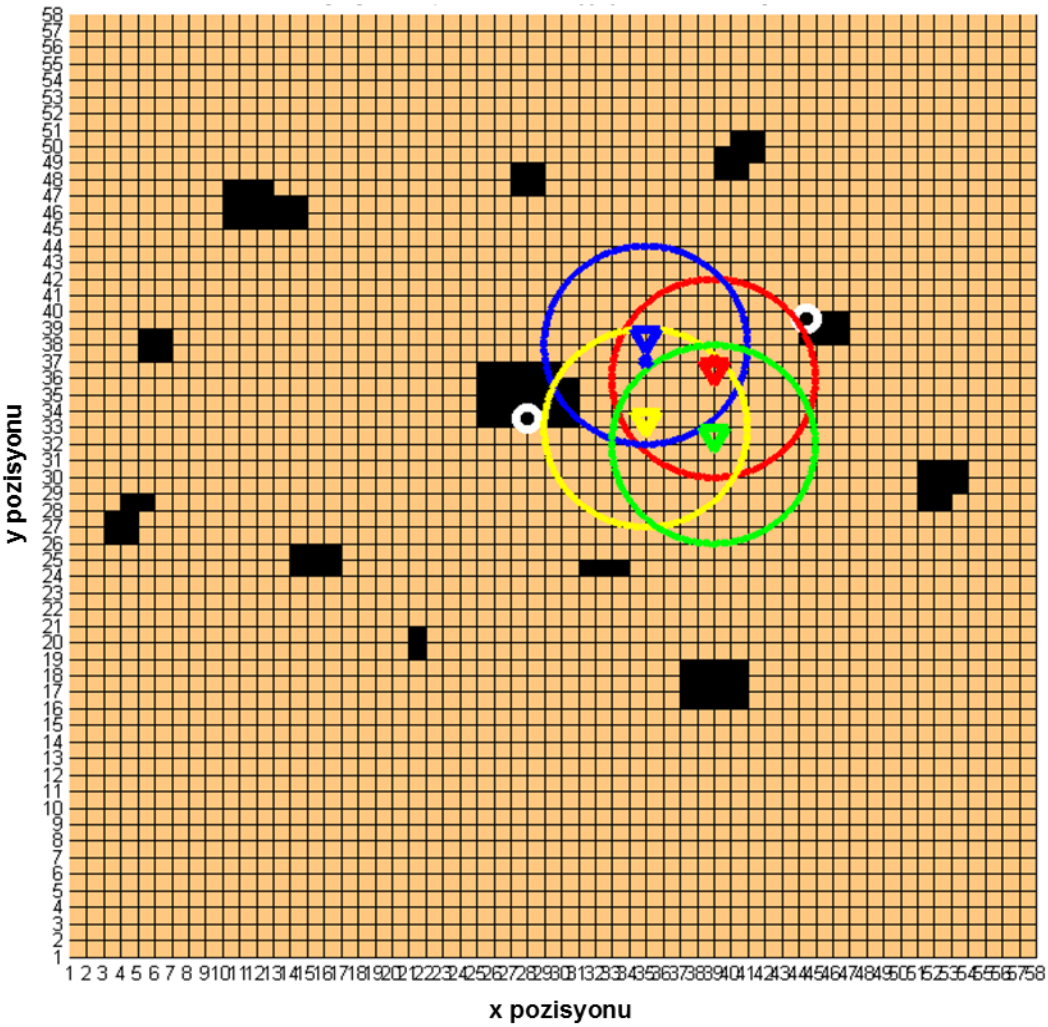
Seri hesaplama performansı neticesinde ortaya konulan ve S_2 olarak adlandırılan senaryonun Aerosonde platformları tarafından icra edilen bir senaryo olduğu kabulüne geri dönerek, ağ ortamı üzerindeki birden fazla grafik işlemci ile paralel hesaplama ile elde edilen sonuçları tekrar değerlendirebiliriz. S_2 senaryosunun hesaplaması ağ ortamı üzerinde yer alan grafik işlemcilerinin bir arada kullanılması ile paralel olarak hesaplandığında seri hesaplamaya kıyasla elde edilen hızlanma 26 kat olarak elde edilmiştir. Bu durumda birim uzunluğun 20 metre dolayısıyla 4000 x 4000 ölçeğindeki bir alanın 80 km. x 80 km. olduğu kabul edildiğinde, alanın hesaplanması için gerekli süre yaklaşık 0,54 sn. olacaktır. Bu süre zarfında platformların hareketine devam ettiği gerçeğinden hareket edilir ise hesaplama için geçen sürede her bir platform yaklaşık 22 m. yer değiştirecektir. Çizelge 5.1 ile ifade edilen formasyon şemalarından her hangi birinin platformlar arasındaki mesafe 22 m. den az olacak şekilde uygulanması mümkün değildir. Ayrıca 22 m. mesafeden yakın engellerden sakınmak amacıyla gereksinim duyulan yol planlamasının istenen zaman aralığında üretilmesi Çizelge 8.11 ile özellikleri gösterilen Şekil 8.18'daki sistem üzerinde özgün CUDA ile geliştirilmiş hesaplama uygulamaları vasıtasıyla hesaplandığı durumlarda gerçekleştirilememektedir.

8.5.4 Çok sayıda GPU ile hesaplamanın doğrulaması

Çalışmanın bu kısmına kadar farklı formasyon şemalarını destekleyen nitelikte çarpışmayı öneleyen ve engellerden sakınan bir yaklaşım dahilinde otonom

platformların yol planlaması YPA tabanlı GPGPU destekli paralel algoritmalar tasarlanarak farklı grafik işlemcileri üzerinde gerçekleştirilmiştir. Ortaya konulan farklı hesaplama denemeleri esnasında en iyi performans ağ ortamı üzerindeki haberleşme maliyetlerinin göz ardı edilmesi halinde ağ ortamı üzerinde yer alan çok sayıda grafik işlemcinin bir arada hesaplamaya dahil edilmesi ile elde edilmiştir.

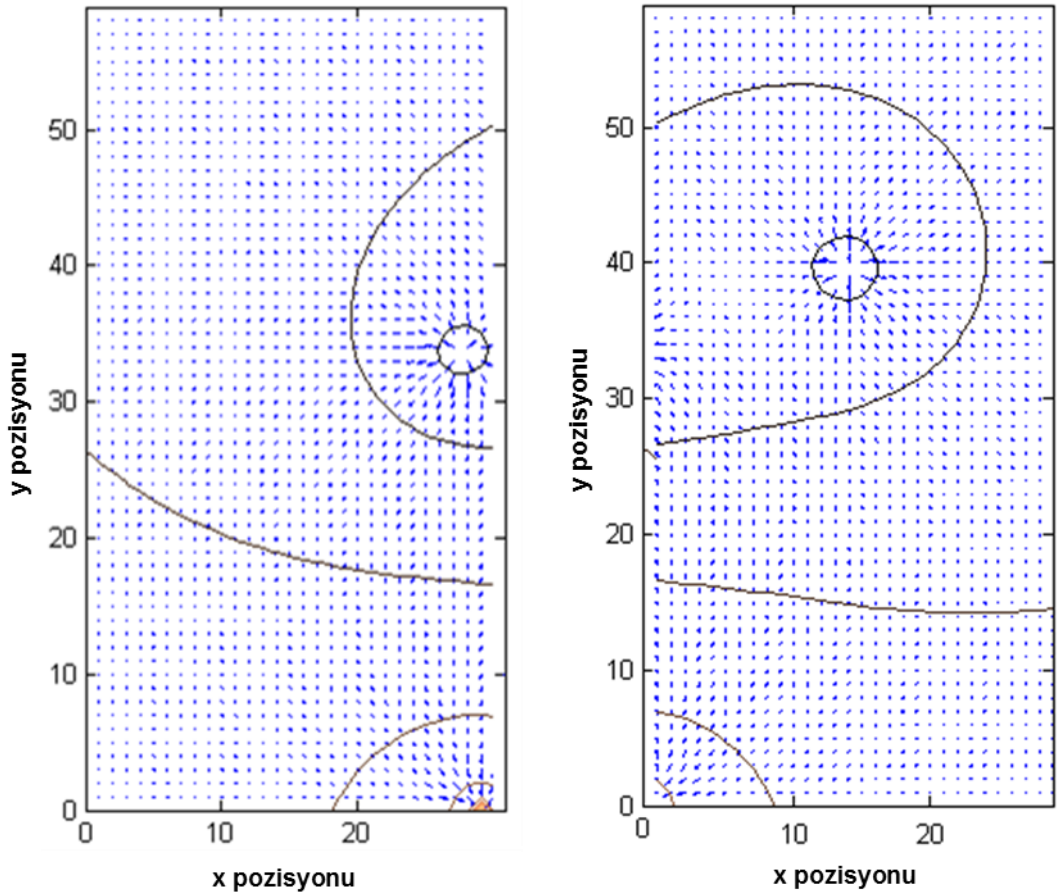
Çalışmanın bu bölümünde ağ ortamı üzerinde yer alan veya aynı ana sunucunun farklı PCI bağlantı noktalarında yer alan grafik işlemcilerin bir arada kullanılması sonucu elde edilen vektör alan çözümlerinin ne derece çözümü yansıtabildikleri kontrol edilecektir. Bu maksatla daha önceden seri hesaplama yaklaşımı ile ya da tek grafik işlemciden faydalanarak ortaya konulmuş olan vetör lanların birden fazla grafik işlemci yardımıyla hesaplanarak ortaya konulan çözümler ile karşılaştırması icra edilecektir.



Şekil 8.23: Çok sayıda grafik işlemci üzerinde potansiyel alan yaklaşımının paralel hesaplanma doğrulaması senaryo gösterimi.

Şekil 8.23 ile *t* anında üzerlerinde yer aldığı kabul edilen sanal algılayıcılar vasıtasıyla tespit edilen engellerden sakınarak formasyonu muhafaza eden ve seyrüseferi icra eden dört adet kare formasyonundaki platform görülmektedir. Formasyonun ve engellerin içinde bulunduğu alan değerlendirildiğinde; alan ölçeğinin 60 x 60 olduğu görülmektedir.

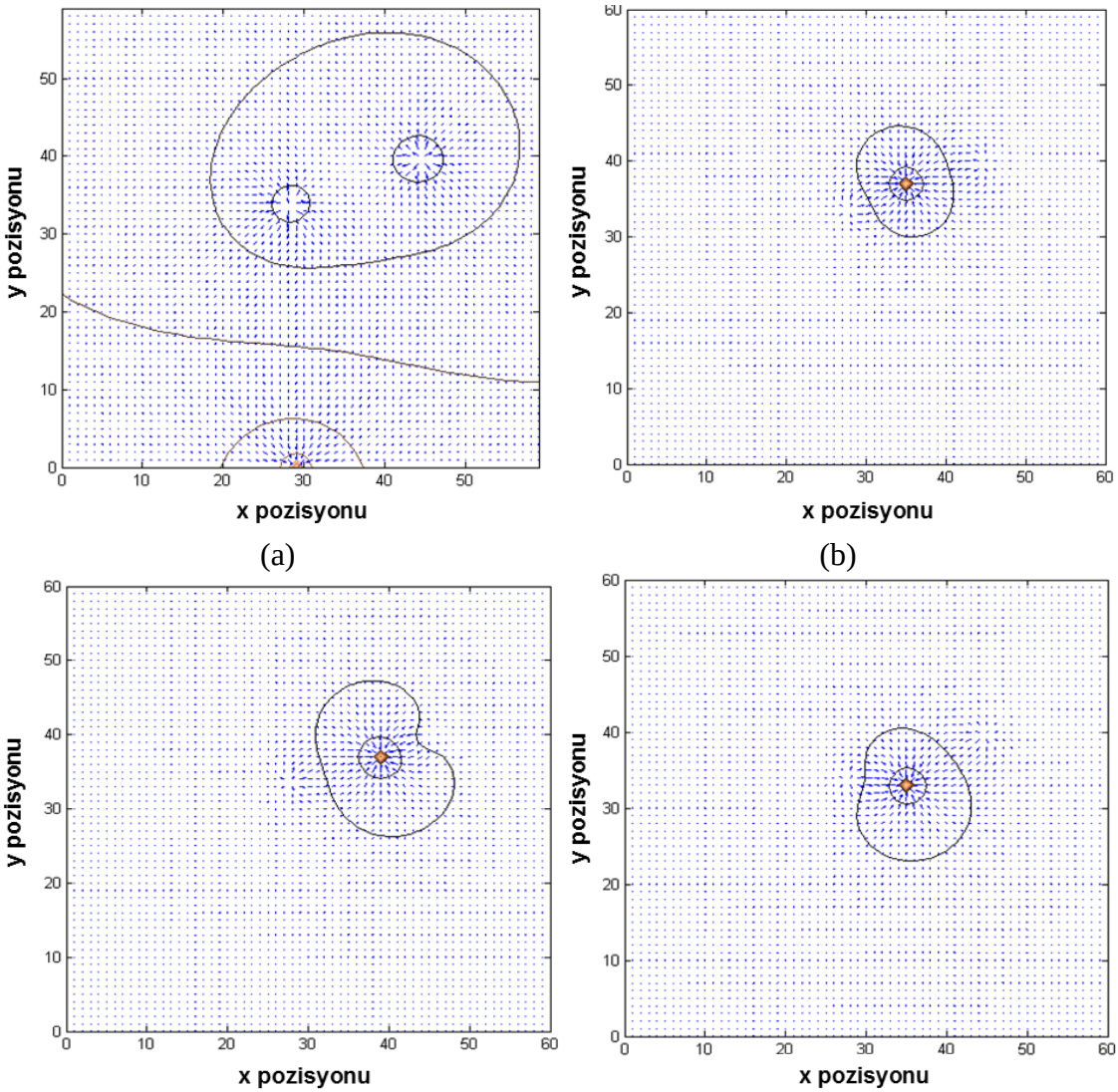
Bu alanın tek bir grafik işlemci ile hesaplanabilir olduğu ve ağ ortamı ya da PCI veri yolu üzerindeki haberleşme maliyetinin bu tip küçük ölçekli bir alan için çok grafik işlemci ile hesaplama yapmanın gereksiz olduğu konusu aşikârdır. Ancak yaklaşımın başarısını sınamak maksadıyla hesaplama performansını dikkate almayarak simülasyon çalışması iki grafik işlemci üzerinde gerçekleştirilmiştir.



(a) (b)

Şekil 8.24: Farklı grafik işlemciler ile hesaplanan tek potansiyel alan bileşenleri gösterimi.

Şekil 8.24 ile görüldüğü üzere alan, 30 x 60 ölçeğinde iki alt bölgeye ayrılmış ve formasyon dahilinde yer alan 1 numaralı platform için YPA ile üretilen iki farklı vektör alan hesaplanmıştır.



Şekil 8.25: Çok sayıda grafik işlemci ile hesaplanan ve birleştirilen vektör alanlar gösterimi.

Şekil 8.24 (a) ile gösterilen ve bir grafik işlemcisi üzerinde hesaplanan vektör alan, Şekil 8.24 (b) ile gösterilen ve farklı bir grafik işlemci ile hesaplanan vektör alan birleştirildiğinde Şekil 8.25 (a) ile gösterilen toplam vektör alan ortaya konulmuştur.

Formasyonda yer alan diğer platformları yönlendiren ve parçalı olarak farklı grafik işlemciler üzerinde hesaplandıktan sonra bir araya getirilerek üretilen vektör alanlar Şekil 8.25 (c) ve Şekil 8.25 (ç) ile gösterilmiştir.

Formasyon dahilinde yer alan her bir platform için üretilen toplam vektör alanlar önceki çalışma döneminde gerçekleştirilen simülasyon sonuçları ile karşılaştırılarak doğrulanmış ve çok sayıda grafik işlemcisi kullanılarak paralel biçimde hesaplanan vektör alanların doğruluğu ortaya konulmuştur.

8.6 GPGPU Tabanlı Paralel Vektör Alan Hesaplanması Optimizasyonları

GPU üzerinde koşturulacak uygulama, paralel mimari gereği tüm iplikler için her durum için kararlı sonucu üretebilecek ve mümkün olduğunca optimize edilmiş aynı komut kümesini koşturacak şekilde ayarlanmalıdır.

Geliştirilen paralel yaklaşım, yol planlamasının yapılacağı alanın boyut ve çözünürlüğünden bağımsız olarak çalışabilmelidir (dynamic problem size).

Paralel hesaplama esnasında en iyi hesaplama performansının elde edilmesi için;

a. Bellek erişim maliyetlerinin,
b. Haberleşme maliyetlerinin (senkronizasyon ve ortak veri paylaşımı),
c. Hesaplama maliyetlerinin minimuma indirgenmesi gerekir.

GPGPU tabanlı paralel programlama esnasında hesaplama performansını etkileyen en önemli unsurlardan birisi haberleşme maliyetleridir. Şekil 6.1 ile gösterildiği şekilde çekirdek kod çağrısı öncesinde sistem belleğinden grafik işlemci belleğine veri transferi, çekirdek kodunun koşturulmasının ardından ise grafik işlemci belleğinden sistem belleğine doğru veri transferi gerçekleştirilmektedir. Bu iki yönlü veri transferi özellikle bu uygulamada olduğu gibi fazla miktarda veri transferine ihtiyaç duyduğunda en az hesaplama maliyeti kadar performansa etki edecektir. Tüm bu iki yönlü haberleşme PCI veri yolu üzerinden gerçekleştirilmektedir.

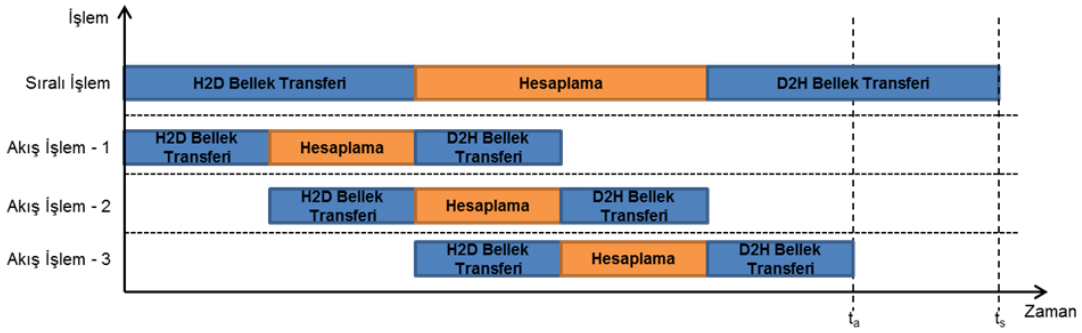
Sistem belleğinden grafik işlemci belleğine gerçekleştirilen veri miktarı, çalışma kapsamında ortaya konulan yaklaşım gereği hesaplama performansına etkisi önemsenmeyecek kadar küçültülmüştür. Yol planlamasının gerçekleştirileceği ortam verisine bağlı olarak belirlenen potansiyel alan parametreleri haricinde çekirdek çağrısı öncesinde grafik işlemci belleğine bir veri aktarılmamıştır.

Asıl haberleşme maliyetini oluşturan kısım ise çekirdek çağrısının gerçekleştirilmesi ve koşturulması neticesinde grafik işlemci belleğinde üretilen verinin, YPA tabanlı yol planlaması açısından bu veri iki adet yönlendirme tablosu olarak kabul edildiğinde, aktarılması maliyetidir.

Grafik işlemci belleğinin üzerinde bellek ihtiyacı duyan uygulamalarda çekirdek çağrısı öncesinde ve sonrasındaki veri transferlerinin haricinde ilave veri transferine ihtiyaç duyulmaktadır. Ancak grafik işlemci belleğine erişim sistem belleğine

erişime kıyasla yaklaşık 22 kat daha hızlı gerçekleşmektedir. Bu nedenle gerçekleştirilen çalışmalar kapsamında grafik işlemci bellek boyutunu aşan uygulamalarda sistem belleğine erişmek yerine çoklu grafik işlemcilerin bir arada kullanılması tercih edilmiştir.

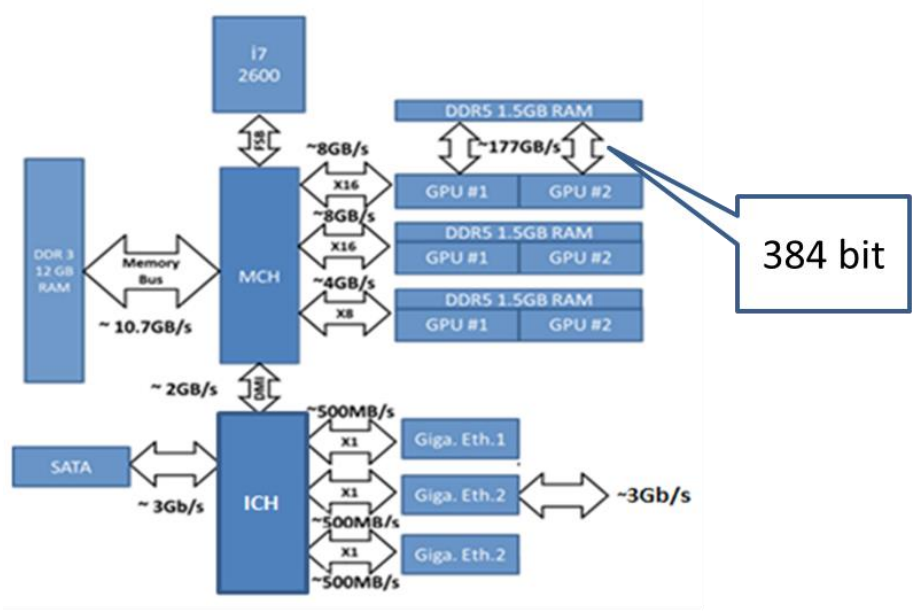
Çoklu grafik işlemcilerden faydalanılması esnasında hesaplama performansına etki eden bellek yönetimi açısından en önemli fayda sağlayan etken asekron veri tansferi olmuştur. Asenkron veri transferi *Algoritma 15* ile gösterildiği üzere bir grafik işlemci üzerindeki hesaplama tamamlanmadan diğer bir grafik işlemci üzerinde hesaplama yapılabilmesine imkan sağlamıştır.



Şekil 8.26: Asenkron veri transferi gösterimi.

Asenkron veri transferi imkanı sayesinde Şekil 8.26 ile gösterildiği şekilde hesaplama maliyetlerinin veri transferi maliyetleri üzerine dağıtılmasına imkan sağlanmıştır. Böylece özellikle birden fazla işlemcinin bir arada kullanıldığı hesaplamalar esnasında bir grafik işlemci üzerinde hesaplama sürerken diğer bir grafik işlemcinin ürettiği verilerin aktarılmasına imkan sağlanmıştır.

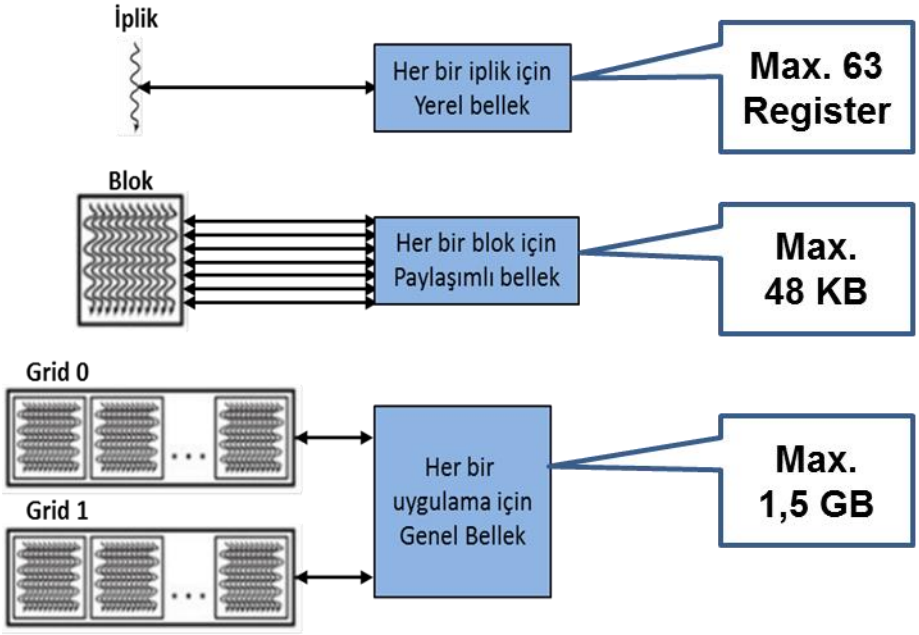
GPGPU tabanlı paralel algoritmaların tasarlanması esnasında performansın arttırılmasına yönelik olarak icra edilen bellek yönetimine ilişkin yaklaşımlardan bir diğeride ardışıl bellek gözlerinin veri yolu bant genişliğinin müsaade ettiği ölçüde bir arada okumak ve yazmaktır. Örneğin GTX 480 ve Tesla K20c grafik işlemcileri için veri yolu bant genişliği Şekil 8.27 ile gösterildiği biçimde 384 bit olarak belirlenmiştir. Bu nedenle aynı anda iki yönlü iletişim 192 bit uzunluğunda veri okuma yazma imkanı doğurmaktadır.



Şekil 8.27: GTX 480 ve K20c grafik işlemcilerin bellek erişim bant genişliği.

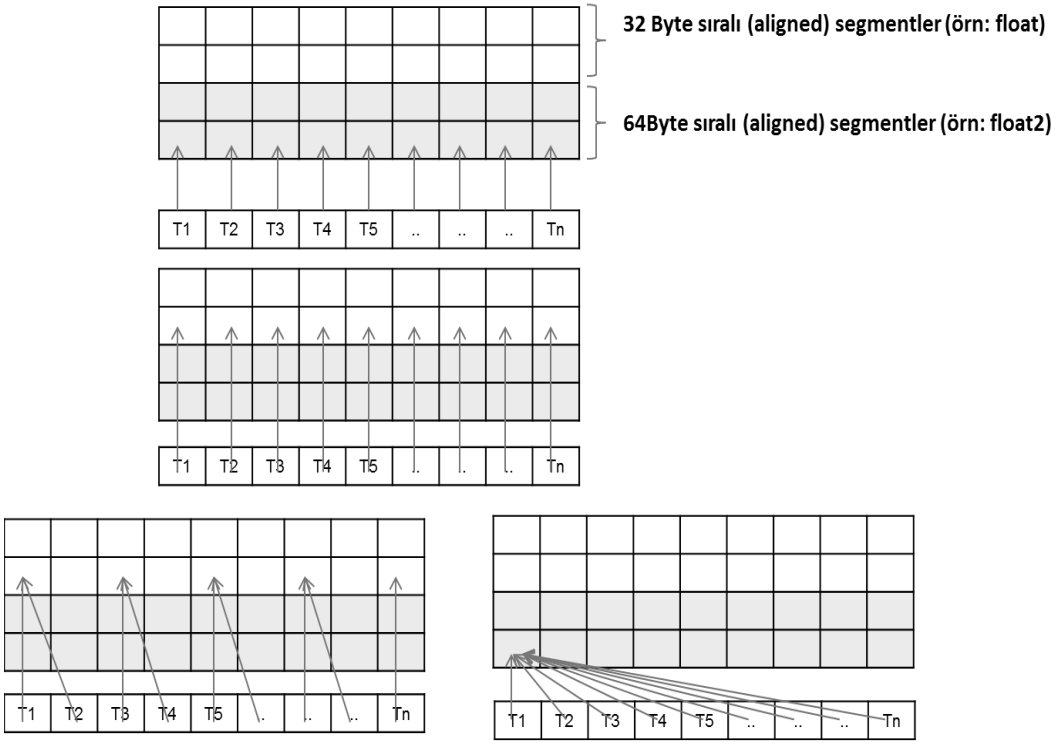
Çalışma kapsamında veri tipi olarak gerek potansiyel alan parametrelerinin temsilinde gerekse de komut tablolarında kullanılan verilerin temsilinde float (single precision) veri tipi kullanılmıştır. Bir float veri bellekte 32 bit ile temsil edildiğinden ardışıl olmak kaydıyla grafik işlemci belleğinden aynı anda 12 adet verinin okunup yazılması sağlanabilir. Bu duruma istinaden algoritmaların tasarımında özellikle belleğe yazma işleminde bu durumdan istifade edecek şekilde bir algoritma tasarımı gerçekleştirilmiştir.

GPU belleği double ve single precision veri temsiline izin verir. Ancak seçilen veri çözünürlüğü temsil edilebilen veri miktarını sınırlar. GPU belleğinde daha fazla veri depolayabilmek amacıyla single hassasiyetinde veri saklanmıştır. Single hassasiyetteki bir veri ile temsil edilen sürat ve yön tabloları yol planlaması esnasında otonom platformun davranışının belirlenmesinde yeterli hassasiyete sahiptir. Bu nedenle veri tipi olarak single hassasiyetinde float veri tipi belirlenmiştir.



Şekil 8.28: GPGPU tabanlı paralel programlama esnasında faydalanılan bellek mimarilerinin gösterimi.

Şekil 8.28 ile GPGPU tabanlı paralel programlama esnasında faydalanılabilecek hiyerarişik bellek alanları görülmektedir. İplik tarafından en hızlı erişilen bellek alanı register space olmak ile birlikte en az veri saklama imkanı da bu bellek mimarisindedir.



Şekil 8.29: GPGPU tabanlı programlama esnasında oluşabilecek memory coalesing durumu gösterimi.

Paylaşımlı belleğe erişimi esnasında mümkün olduğunca ipliklerin birbirleri ile çakışmasını, bir başka deyim ile memory coalesing durumunu engelleyecek şekilde tasarım yapılmıştır. İpliklerin eriştikleri veriler dizi yapısına uygun bir temsil ile bellekte saklanarak erişime sunulmuş ve bellek erişim paternleri ipliklerin id numaralarına uygun olarak düzenlenmiştir. Yukarıda yer alan Şekil 8.29 ile GPGPU tabanlı programlama esnasında memory colaesing durumunun oluşumu gösterilmiştir. İlk iki erişim doğru yaklaşımı modellerken, diğer iki erişim performansı olumsuz etkileyecek erişimleri göstermektedir.

GPU üzerindeki paralel uygulamaların performanslarının arttırılması için dikkat edilmesi gereken bir diğer konu işlemci üzerinde yer alan her bir SM (stream-multi processor)'in mümkün olduğunca en yoğun şekilde (occupancy) kullanılmasını sağlamaktır. Aksi taktirde sistem kaynakları boş yere meşgul edilir ve performans kaybı oluşur. Bu nedenle her bir çekirdek üzerinde koşacak olan kodun, ayrılan fiziksel sistem kaynaklarını en verimli şekilde kullanmasını sağlayacak (hardware utilization) bir algoritma geliştirilmesi gerekir. Çekirdek kodun çözümlediği problem boyutuna bağlı olarak Blok ve Grid değerleri optimize edilmelidir.

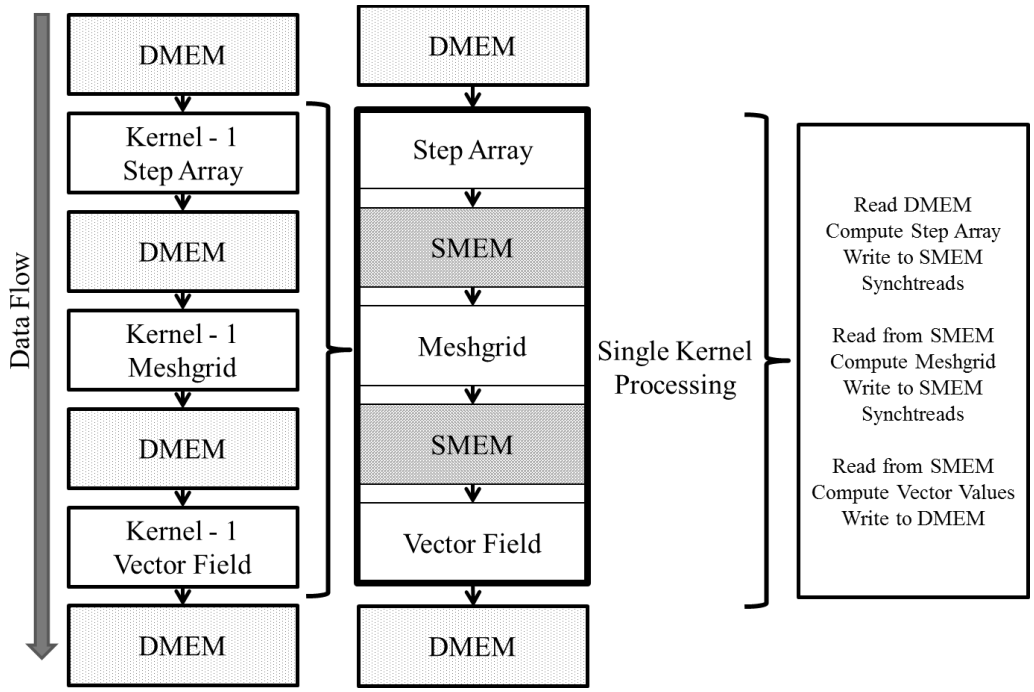
Çizelge 8.13: CUDA yoğunluk hesaplama örneklerinin gösterimi.

Blok Miktarı	Aktif İplik Sayısı	Yoğunluk
32	256	0.1666
64	512	0.3333
128	1024	0.6666
192	1536	1
256	2048	1

CUDA yoğunluk hesaplamasından örnekler Çizelge 8.13 ile gösterilmiştir. YPA tabanlı paralel yol planlamasının hesaplama algoritmasının tasarımı esnasında esnek bir blok ve iplik sayısı yaklaşımı uygulanmıştır. Şayet grafik işlemcinin üretebileceği iplik sayısı çalışmanın ilerleyen bölümlerinde açıklanan tek iplikte birden fazla ızgara hücresinin hesaplanması optimizasyonunda elde edilen dört hücrenin katı olarak yeterli oluyor ise yoğunluk olarak genellikle tam değere (%100) ulaşılmıştır. Ancak bu durum sağlanamdağında en düşük yoğunluk seviyesi %66 olarak gerçekleşmektedir.

GPGPU tabanlı paralel algoritmaların geliştirilmesi esnasında dikkat edilmesi gereken bir diğer nokta çok sayıda GPU kullanılarak gerçekleştirilen hesaplamalar esnasında PCI veri yolu ve ağ ortamından kaynaklanan haberleşme hızı dar boğazlarıdır. Bu nedenle bu kanallar üzerinden gerçekleştirilen haberleşmenin minimize edilmesi hesaplama performansını arttıracaktır. Çalışma kapsamında gerçekleştirilen uygulamalar esnasında özellikle ağ ortamındaki haberleşmeden kaynaklanan maliyetler çalışmanın ilerleyen bölümlerinde detaylı olarak açıklanmıştır. PCI veri yolu üzerinden gerçekleştirilen haberleşme maliyetleri hesaplama performans sonuçlarına dahil edilmiştir. PCI veri yolu üzerinden minimum veri aktarımını sağlayacak şekilde algoritma tasarımı sağlanmıştır. Bu durum daha önceki kısımlar dahilinde açıklanmıştır. Ayrıca birden fazla GPU kullanımı esnasında asenkron veri transferinden açıklanan şekilde faydalnılarak performans artışı sağlanmıştır.

Bu aşamaya kadar tğm CUDA tabanlı GPGPU uygulamaları için geçerli olan performansı arttırıcı optimizasyonlara değinilmiştir. Bu bölümden itibaren ise çalışma kapsamında ele alının YPA tabanlı yol planlaması uygulamasının performansının nasıl arttırıldığına dair uygulamaya özel gerçekleştirilen optimizasyonlara değinilmiştir.



Şekil 8.30: Tek çekirdek fonksiyonu kullanılarak vektör alanın hesaplanmasının gösterimi.

Çalışma kapsamında paralel potansiyel alan hesaplanması amacıyla geliştirilen yapı üzerinde uygulanan ilk optimizasyon, geliştirilen üç çekirdek kodunun bir çekirdek kodu şeklinde Şekil 8.30 ile gösterilen yapı dahilinde birleştirilmesidir. Böylece çekirdek kodu çağrılarında kaybedilen zamandan ve her koddan sonra üretilen verinin DMEM (device memory) üzerine yazılarak okunması için harcanan bellek transferi süresi kayıplarından istifade etmek amaçlanmıştır.

Algoritma 17. Birleştirilmiş Vector_field Çekirdek Algoritması

Girdi : $x,y,res,A_{type}[\,]$, $A_{\alpha}[\,]$, $A_{x}[\,]$, $A_{y}[\,]$, $A_{\varphi}[\,]$, $A_{\gamma}[\,]$, *X[ ],Y[ ],* $A_{slope}[\,]$, $A_{\rho}[\,]$

Çıktı : *SX[ ], SY[ ], Heading[ ], Magnitude[ ]*

17.1 İplik numarası değerini hesaplayarak bul

*int const i = blockDim.x * blockIdx.x + threadIdx.x;*

17.2 İplik numarası ve çözünürlük değerine bağlı olarak tek boyutlu A dizisinin değerlerini hesaplayarak ata

*A[i] = i * res;*

17.3 *syncthreads*

17.5 *X,Y* iki boyutlu mesgrid dizilerini hesapla

X[round_lower((i-mod(i,x))/x)+1, mod(i,x)] = A [mod(i,x)];

Y[round_lower((i-mod(i,x))/x)+1, mod(i,x)] = A [round_lower((i-1)/x)+1];

17.6 *syncthreads*

17.7 *i* iplik değeri ile temsil edilen noktanın vektör alan bilşen değerlerini sıfırla

a=round_lower((i-mod(i,x))/x)+1;

b=mod(i,x);

SX(a,b) = 0; SY(a,b) = 0;

17.8 *syncthreads*

17.9 İplik tarafından hesaplanan (a,b) noktasındaki toplam sınırlandırılmış

17.10 vektör değeri hesapla

17.11 ***for*** *s = 1:length(*A_{type}*)*

if *(*$A_{type}(s)$*)* $= C_{rot}$

$DX_{C_{rot}(s)}(a,b)$ değerini Denklem (3.21) ile hesapla

$DY_{C_{rot}(s)}(a,b)$ değerini Denklem (3.22) ile hesapla

$S_C^+(a,b)$ değerini Denklem (3.37) ile hesapla

$SX(a,b) = SX(a,b) + DX_{C_{rot}(s)}(a,b).S_C^+(a,b)$

17.12 $SY(a,b) = SY(a,b) + DY_{C_{rot}(s)}(a,b).S_C^+(a,b)$

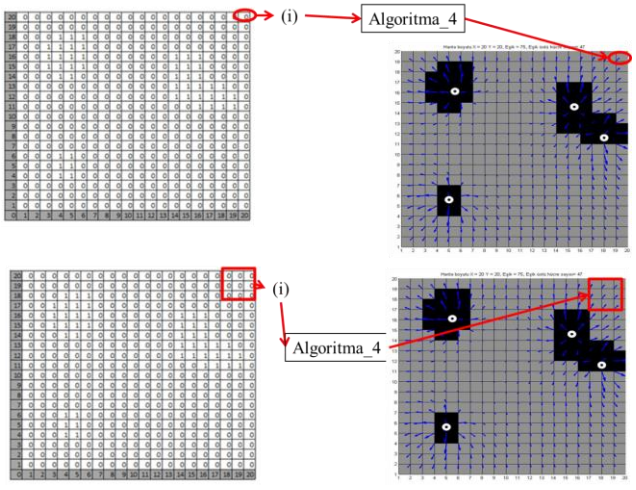
elseif *(*$A_{type}(s)$*)* $= I_{rot}$

$DX_{I_{rot}(k)}(a,b)$ değerini Denklem (3.22) ile hesapla

$DY_{I_{rot}(k)}(a,b)$ değerini Denklem (3.24) ile hesapla

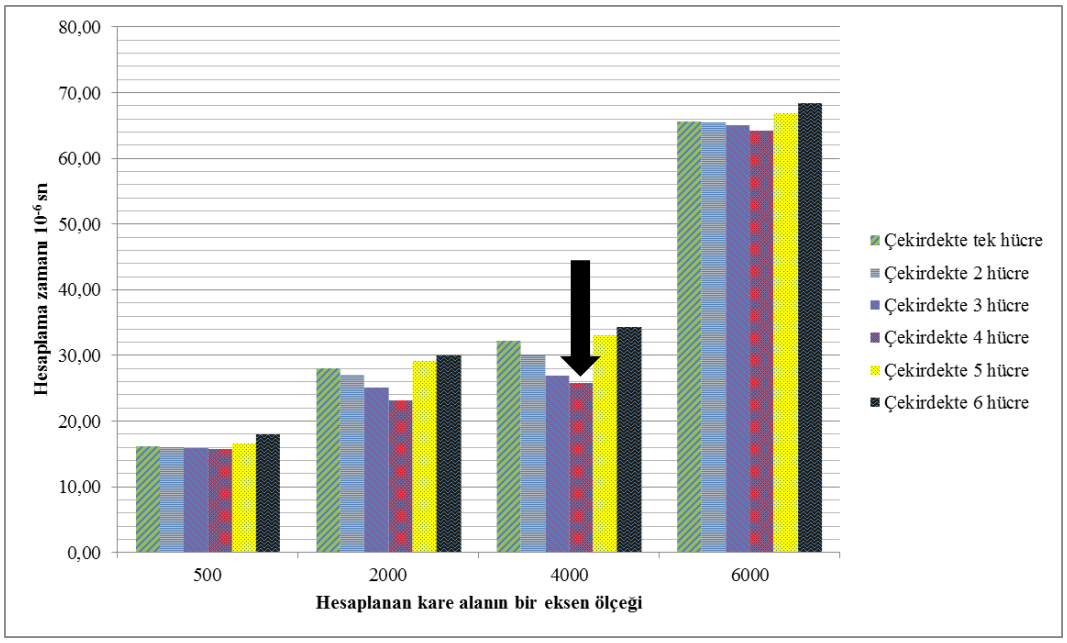
$S_I^-(a,b)$ değerini Denklem (3.37) ile hesapla
$SX(a,b) = SX(a,b) + DX_{I_{rot}(s)}(a,b).S_I^-(a,b)$
17.13 $SY(a,b) = SY(a,b) + DY_{I_{rot}(s)}(a,b).S_I^-(a,b)$
elseif $(A_{type}(s)) = R_p$
$DX_{R_p(s)}(a,b)$ değerini Denklem (3.29) ile hesapla
$DY_{R_p(s)}(a,b)$ değerini Denklem (3.31) ile hesapla
$S_R^-(a,b)$ değerini Denklem (3.37) ile hesapla
$SX(a,b) = SX(a,b) + DX_{R_p(s)}(a,b).S_R^-(a,b)$
17.14 $SY(a,b) = SY(a,b) + DY_{R_p(s)}(a,b).S_R^-(a,b)$
elseif $(A_{type}(s)) = R_n$
$DX_{R_n(s)}(a,b)$ değerini Denklem (3.30) ile hesapla
$DY_{R_n(s)}(a,b)$ değerini Denklem (3.32) ile hesapla
$S_R^-(a,b)$ değerini Denklem (3.37) ile hesapla
$SX(a,b) = SX(a,b) + DX_{R_n(s)}(a,b).S_R^-(a,b)$
$SY(a,b) = SY(a,b) + DY_{R_n(s)}(a,b).S_R^-(a,b)$
17.15 ***end if***
17.16 ***end for***
17.17
İplik tarafından hesaplanan (a,b) noktasındaki sürat ve yön tablo değerlerini hesapla
Heading (a,b) değerini Denklem (3.41) ile hesapla
Magnitude (a,b) değerini Denklem (3.42) ile hesapla

Şekil 8.30 ile gösterilen yapı *Algoritma 17* kapsamında ortaya konulmuştur. Algoritmadan da görüleceği üzere senkronizasyon noktaları oluşturularak, üç çekirdek kodu birleştirilerek tek çekirdek kodu şeklinde uygulanabilir. Böylece CPU'ya dallanma ve de tekrar çekirdek kodu çağrısı yapma maliyetlerinden tasarruf edilmiştir.



Şekil 8.31: Tek çekirdek içinde çok sayıda hücre değerinin hesaplanmasının gösterimi.

Paralel programlama yapısına uygun şekilde geçekleştirilen algoritma ve sistem yapısı dahilinde gerçekleştirilen bir diğer optimizasyon ise her bir iplik ile bir hücre değeri hesaplanması yerine her bir hücrenin birden hücreye ait vektör değerleri hesaplaması şeklinde gerçekleştirilen uygulamadır. İlk bakışta aynı çekirdek kod dahilinde hesaplanan vektör değerlerinin sıralı olarak hesaplanan işlemler olduğu şeklinde algılansa dahi gerçekleştirilen yaklaşım daha iyi performans sonuçları üretmiştir. Bunun nedeninin bir iplik için atanan kayıtçı, bellek denetleyicisi gibi kaynakların daha verimli kullanılması ve memory coalesing durumunun azaltılması olarak değerlendirilebilir. Gerçekleştirilen yaklaşım grafiksel olarak **Şekil 8.31** ile gösterilmiştir.



Şekil 8.32: Tek çekirdek içinde çok sayıda hücre değerinin hesaplanması performansı.

Şekil 8.32 ile farklı ölçeklerdeki alanlar için tek çekirdek kodu içinde birden fazla ızgara hücresinin hesaplanması sonucu elde edilen performans sonuçların görülmektedir. Bu sonuçlara istinaden farklı ölçeklerdeki alanlar için hesaplama yapılması esnasında bu durum göz önünde bulundurulmuştur.

Hesaplama performansının arttırılmasına katkıda bulunmak amacıyla gerçekleştirilen iyileştirmelerden bir diğeri NVIDIA CUDA Thrust Kütüphanesi kullanılarak YPA tabanlı yol planlaması içinde kullanılan bazı algoritmaların hızlandırılması olmuştur. Thrust kütüphanesi C++ programlama dilinde CUDA tabanlı Standart Şablon Kütüphaneleri (Standard Template Library – STL) olarak geliştirilmiş programlama

yapılarıdır. Thrust, yüksek performanslı paralel hesaplama uygulamalarının minimum programlama performansı ortaya konularak geliştirilmesi maksadıyla, CUDA ortamında kullanılabilen, C++ tabanlı yüksek seviye arayüz şeklinde tanımlanabilir. Thrust kütüphanesinde yer alan hazır paralel fonksiyonlar ile arama (scan), sıralama (sort) ve azaltma (reduce) gibi paralel programlama uygulamalarının geliştirilmesinde sıklıkla kullanılan genel fonksiyonlar bir arada sunulmaktadır. Çalışma kapsamında geliştirilen algoritmalar dahilinde Thrust kütüphanesinden faydalanılarak yapılan değişiklikler şu şekildedir;

a. Öncelikle çalışma kapsamında geliştirilen algoritmalarda kullanılan C++ programlama dilindeki dizi veri yapıları yerine thrust::device_vector ve thrust::host_vector veri yapıları kullanılmıştır,

b. Önceki dönem kapsamında geliştirilerek açıklanan paralel engel gruplama ve etiketleme (Parallel Connected Component Labeling - P_CCL) algoritması dahilinde kullanılan MATLAB ortamındaki Parallel Computing Toolbox kütüphanesinde yer alan "find" fonksiyonu yerine thrust::find fonksiyonu kullanılmıştır.

Yukarıda açıklanan optimizasyonlara ilave olarak CUDA derleyicisi olan NVCC üzerinde bir takım performansın iyileştirilmesine yönelik çalışmalar gerçekleştirilmiştir:

a. Fermi mimarisinde yer alan grafik işlemcileri üzerinde hesaplama icra edilirken hesaplama kabiliyeti (compute capabilities) 2.0 değerine "arch=sm_20" parametresi ile, Kepler mimarisinde ise 3.5 değerine "arch=sm_35" parametresi ile ayarlanmıştır.

b. Derleyici "-use_fast_math" parametresi ile ayarlanmış ve böylece her ne kadar matematik fonksiyonların uygulanmasında bir miktar hassasiyet kaybı yaşansa dahi gerçekleştirilen kontrol neticesinde sonuçlarda farklılık gözlemlenmemiştir.

c. Fermi mimarisindeki grafik işlemciler için derleme esnasında "-maxrregcount" parametresi kullanılarak her bir iplik tarafından kullanılan kayıtçı sayısı 32 ile sınırlandırılmıştır. Fermi mimarisinde her hangi bir parametre verilmeyerek iplik başına kullanılan kayıtçı sayısı 64 olarak bırakılmıştır.

Derleyici dışında NVIDIA ayarlarından grafik belleklerin Hata Kotarma Kodu (Error Correction Code – ECC) opsiyonu devre dışı bırakılarak hesaplamalar

gerçekleştirilmiş ve böylece bir miktar hesaplama performans artışı sağlanmıştır. ECC özelliğinin açık hale getirilmesi ile bellekte verilerin saklanması esnasında verinin bozulduğunu farketmek ve belli bir oranda bozulan veriyi kurtarmak mümkündür. Ancak bunun yanında NVIDIA grafik işlemcilerde ECC özelliği aktif edildiğinde cihaz belleğinin toplam kapasitesinden yaklaşık %12,5 oranında kayıp oluşur. Örneğin 5120 Mbyte olan K20c grafik işlemcisinin cihaz bellek miktarı ECC etkinleştirildiğinde 4480 Mbyte değerine düşer. Bu durum hesaplanabilir alan ölçek değerini etkileyecektir. Bunun yanısıra her ne kadar hesaplama performansı etkilenmese dahi bellek erişim performansında erişilen bellek miktarına bağlı olarak düşüş gözlemlenecektir.

Çizelge 8.14: Grafik belleği ECC açık durumda K20c işlemcisi ile hesaplama performansı.

Alan	ECC Durumu Kapalı Hesaplama		ECC Durumu Açık Hesaplama		Hesaplama Performansı Düşüş Oranı (x)
	Toplam Hesaplama Süresi (s.)	Bir Hücrenin Hesaplama Süresi (ns.)	Toplam Hesaplama Süresi (s.)	Bir Hücrenin Hesaplama Süresi (ns.)	
500	0,007	29	0,008	35	0,214
1000	0,015	15	0,018	18	0,206
2000	0,042	10	0,051	12	0,218
3000	0,073	8	0,088	10	0,205
4000	0,098	6	0,118	7	0,199
5000	0,130	5	0,157	6	0,207
6000	0,162	5	0,200	6	0,234
7000	0,205	4	0,249	5	0,213
8000	0,246	4	0,302	5	0,228

Çizelge 8.14 ile gösterildiği şekilde NVIDIA K20c işlemcisi için ECC modu aktif edilerek hesaplama gerçekleştirildiğinde ECC modu kapalı hesaplama karşısındaki hesaplama performansı gösterilmektedir. ECC modu aktif edildiğinde yaklaşık ortalama %20 hesaplama performansında düşme gözlenmektedir. 4000x 4000 ölçeğinde bir alanın hesaplanması esnasında yaklaşık 11,1 kat oranından 9,2 oranına gerilemiştir.

Bu bölüm kapsamında hesaplama performansına etki edeceği değerlendirilen tüm optimizasyonlar bir arada ele alınmıştır. Bölüm 6 kapsamında açıklanan YPA tabanlı otonom yol planlaması algoritması üzerinde tüm bu optimizasyon çalışması sırasıyla

tek tek uygulanmıştır. Her bir optimizasyon neticesinde hesaplamada bir miktar performans artışı elde edilmiştir.

8.7 Hesaplama Performanslarının Karşılaştırılması

Bu bölüm kapsamında, çalışmanın bir önceki bölümünde ortaya konulan paralel programlama performansını arttırmaya yönelik gerçekleştirilen çalışmaların etkileri karşılaştırmalı olarak irdelenmiştir. Bu kapsamda farklı grafik işlemci donanımları üzerinde algoritmanın son hali sınanmıştır.

Çizelge 8.15: Farklı grafik işlemciler ile merkezi işlem biriminin paralel hesaplama optimizasyonları sonrası hesaplama performansları.

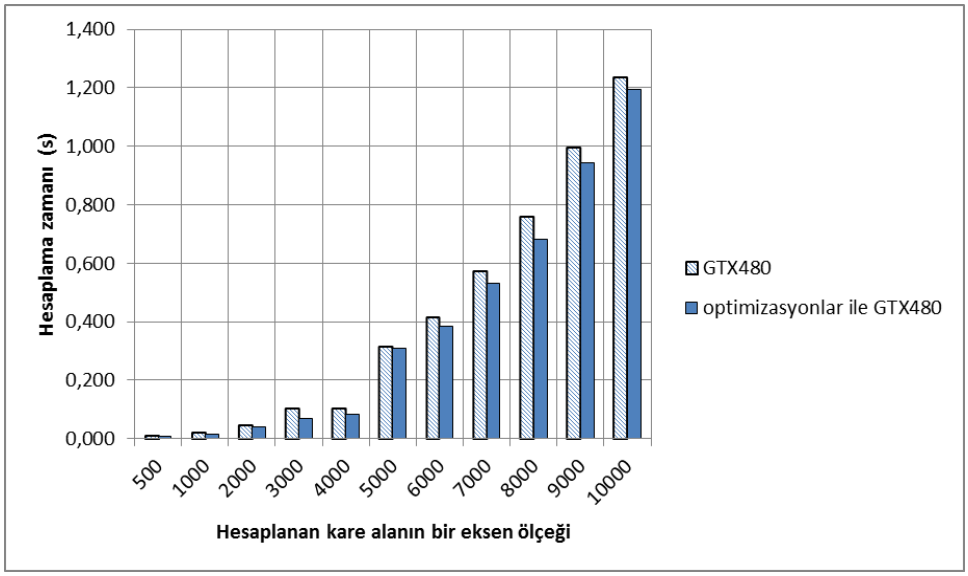
Alan	GTX480			K20c		
	Toplam Hesaplama Süresi (s.)	Bir Hücrenin Hesaplama Süresi (ns.)	Seri Hesaplama Karşında Hızlanma (x)	Toplam Hesaplama Süresi (s.)	Bir Hücrenin Hesaplama Süresi (ns.)	Seri Hesaplama Karşında Hızlanma (x)
500	0,007	28	2,41	0,007	27	2,51
1000	0,016	16	4,27	0,014	14	4,85
2000	0,041	10	6,40	0,038	9	7,02
3000	0,069	8	8,21	0,062	7	9,01
4000	0,082	5	13,23	0,071	4	15,40
5000	0,310	12	4,81	0,092	4	16,25
6000	0,384	11	4,78	0,103	3	17,75
7000	0,530	11	4,89	0,177	4	12,33
8000	0,680	11	4,92	0,210	3	12,45
9000	0,944	12	4,93	0,549	7	5,40
10000	1,194	12	4,66	0,652	7	5,07

Bir önceki bölüm kapsamında açıklanan hesaplamaya yönelik optimizasyonların algoritma üzerinde uygulanması sonucunda elde edilen farklı grafik işlemcileri üzerindeki performans sonuçları Çizelge 8.15 ile Çizelge 8.16 ve aşağıda yer alan grafikler ile ortaya konulmuştur.

Çizelge 8.16: Farklı grafik işlemciler ile merkezi işlem biriminin paralel hesaplama optimizasyonları sonrası hesaplama performansları.

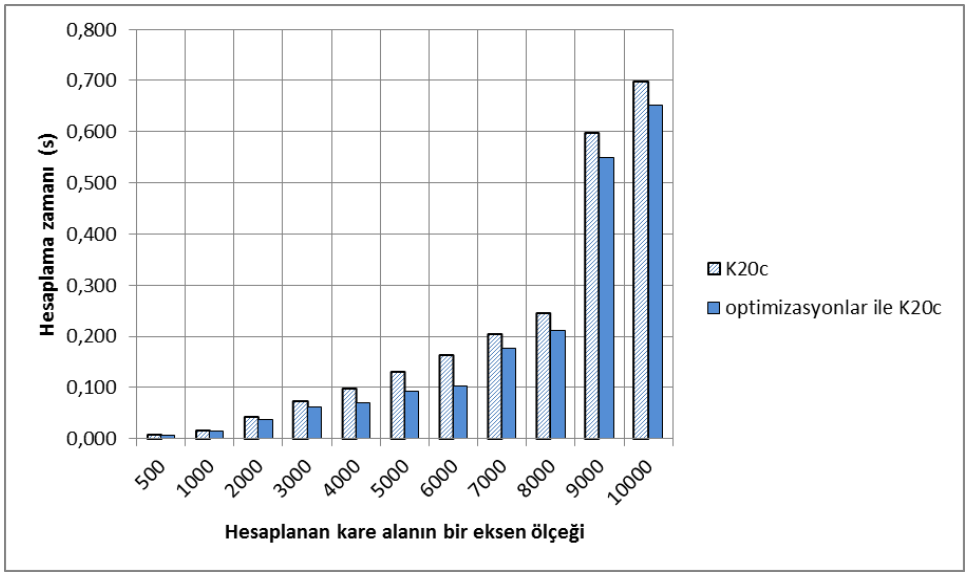
Alan	K20c&GTX 480			Cluster		
	Toplam Hesaplama Süresi (s.)	Bir Hücrenin Hesaplama Süresi (ns.)	Seri Hesaplama Karşında Hızlanma (x)	Toplam Hesaplama Süresi (ms.)	Bir Hücrenin Hesaplama Süresi (µs.)	Seri Hesaplama Karşında Hızlanma (x)
500	0,003	11	6,18	0,003	10	6,58
1000	0,006	6	10,95	0,006	6	11,56
2000	0,019	5	14,24	0,015	4	17,24
3000	0,029	3	19,49	0,024	3	23,04
4000	0,039	2	27,56	0,037	2	28,98
5000	0,050	2	29,94	0,042	2	35,54
6000	0,051	1	36,11	0,047	1	38,64
7000	0,055	1	39,68	0,050	1	43,15
8000	0,064	1	41,15	0,056	1	47,13
9000	0,080	1	37,19	0,062	1	48,11
10000	0,318	3	10,41	0,068	1	48,34

Çizelge 8.15 ile görüldüğü üzere hesaplamaya yönelik optimizasyonlar neticesinde ortaya konulan algoritmaların performansları karşılaştırma yapılabilmesi amacıyla farklı ölçeklerdeki alanlar için aynı grafik işlemciler ile hesaplanmıştır. Elde edilen yeni sonuçlar merkezi işlem birimi vasıtasıyla gerçekleştirilen sıralı hesaplama performansı ile karşılaştırılmış böylece optimizasyonların hesaplama performansına etkileri ortaya konulmuştur.



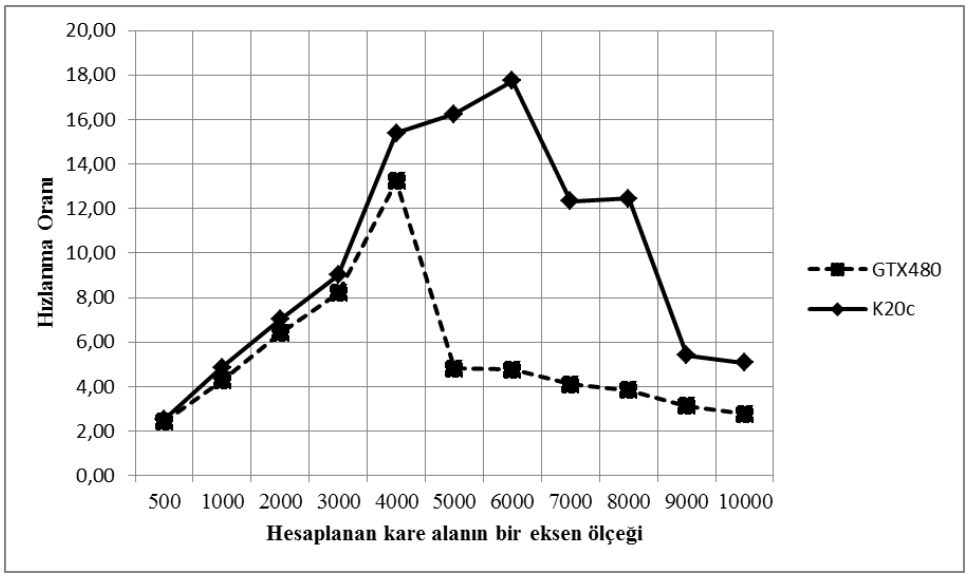
Şekil 8.33: Optimizasyonlar sonrasında NVIDIA GTX 480 grafik işlemcisinin hesaplama performansındaki değişim.

Bu durumda elde edilen sonuçlar incelendiğinde Şekil 8.33 ile gösterilen grafikten de görüleceği üzere sıralı hesaplamaya nazaran elde edilen en iyi performans oranı 13,23 kat ile 4000 x 4000 ölçeğinde üretilmiştir. Daha büyük ölçeklerdeki alanların hesaplanması istendiğinde grafik işlemci belleğinin yetersiz olmasından dolayı sistem belleği ile PCI veri yolu üzerinden haberleşme maliyetlerinin eklenmesi performansı olumsuz yönde etkilemiştir.



Şekil 8.34: Optimizasyonlar sonrasında NVIDIA K20c grafik işlemcisinin hesaplama performansındaki değişim.

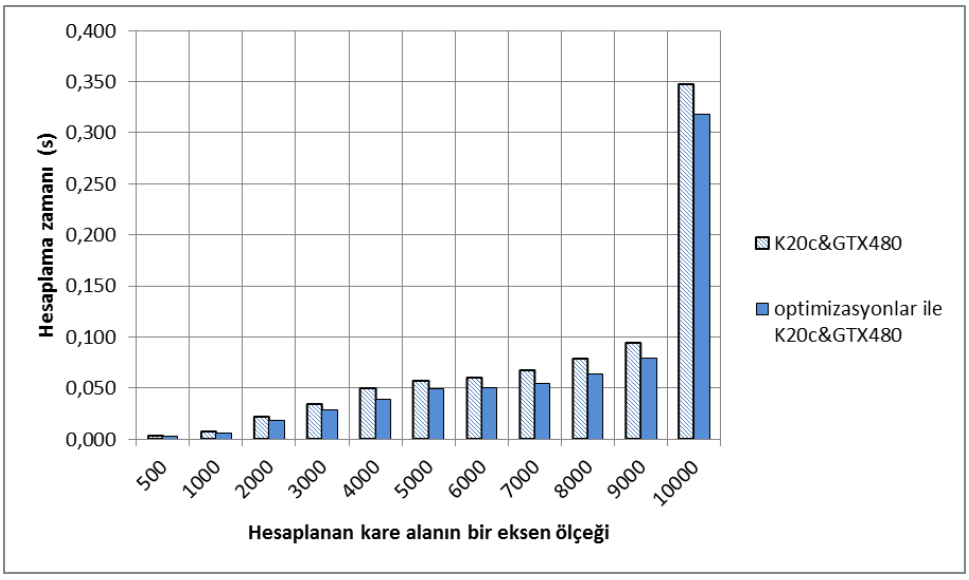
Hesaplamaya yönelik optimizasyonların uygulanarak, K20c donanımı için edilen sonuçlar incelendiğinde Şekil 8.34 ile gösterilen grafikten de görüleceği üzere sıralı hesaplamaya nazaran elde edilen en iyi performans oranı yaklaşık 17,75 kat ile 6000 x 6000 ölçeğinde üretilmiştir. Benzer şekilde bu ölçekten daha büyük alanların hesaplanması istendiğinde sistem belleği ile haberleşme maliyetleri perfomansa olumsuz etki etmiştir.



Şekil 8.35: Optimizasyonlar sonrasında K20c ve GTX 480 grafik işlemcilerin seri hesaplama karşısındaki hesaplama performansı.

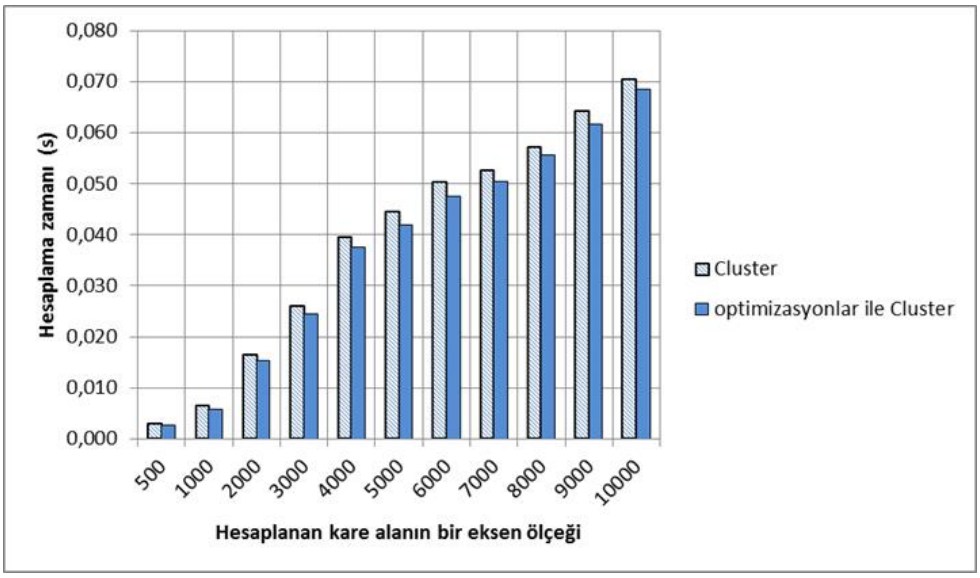
Şekil 8.35 ile gösterildiği biçimde tüm alanın hesaplanması için gerekli süreler karşılaştırılmıştır. Grafikten de görüldüğü üzere 4000 x 4000 ölçeğine kadar iki grafik kartının performansları arasında oluşan fark her ne kadar özellikleri Çizelge 2.11 ile gösterilen GTX 480 grafik işlemcisinin CUDA çekirdek sayısı özellikleri Çizelge 2.10 ile gösterilen K20c donanımınından sayısal olarak fazla olsa dahi iki mimari arasındaki bellek erişim hızlarından ve çekirdek çalışma frekans performanslarından kaynaklandığı değerlendirilmektedir. 4000 x 4000 ölçeğinden büyük alanların hesaplanması esnasında 9000 x 9000 ölçek değerine kadar iki donanım arasındaki performans farkı grafik donanımlarının farklı miktarda grafik belleğine sahip olmalarından kaynaklanmaktadır. Daha büyük ölçeklerde performansı belirleyen önemli kriter sistem belleği ile haberleşme maliyeti olduğundan aralarındaki fark azalmıştır.

NVIDIA GTX 480 kartı ile 4000 x 4000 ölçeğinde bir alanın hesaplanması için ihtiyaç duyulan 0,103 s.'lik süre optimizasyonların uygulanması ile 0,82 s. olarak, NVIDIA K20c donanım ile 6000 x 6000 ölçeğinde bir alanın hesaplanması için ihtiyaç duyulan 0,162 s.'lik süre ise optimizasyonların uygulanmasının ardından 0,103 s. olarak elde edilmiştir.



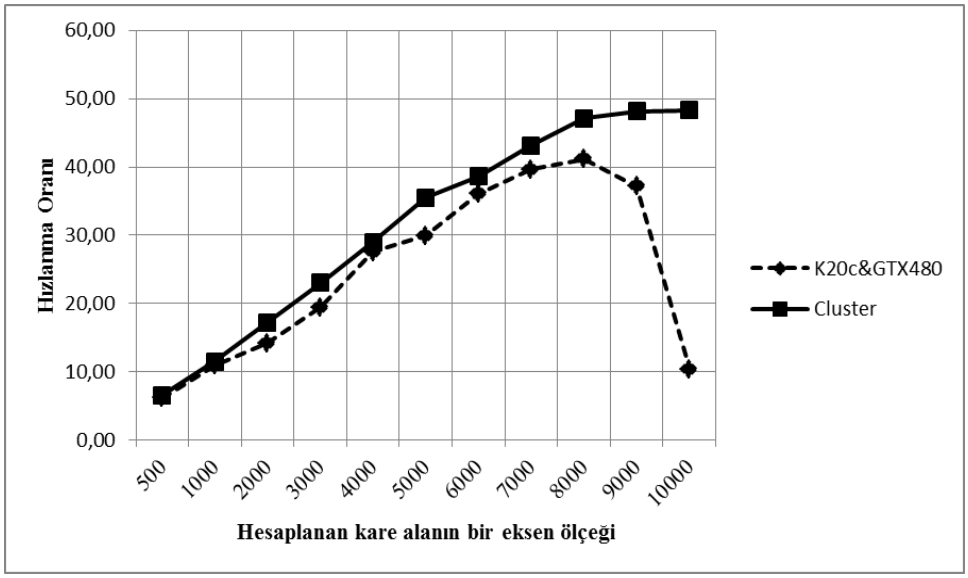
Şekil 8.36: Optimizasyonlar sonrasında K20c ve GTX 480 grafik işlemcilerin seri hesaplama karşısındaki hesaplama performansı.

Aynı şekilde hesaplama performansının arttırılmasına yönelik olarak bir önceki bölüm kapsamında açıklanan optimizasyonlar bu kez Şekil 8.36 ile sonuçları paylaşıldığı şekilde NVIDIA K20c ve GTX 480 grafik işlemcilerinin bir arada kullanıldığı hesaplama üzerinde uygulanmıştır. Bu durumda Çizelge 8.16'dan da görüldüğü üzere seri hesaplama karşındaki en iyi performans artışı 8000 x 8000 ölçeğindeki alanın hesaplanması esnasında 41,15 kat olarak gerçekleşmiştir. Grafikten de görüldüğü üzere 10.000 x 10.000 ölçeğindeki alanın hesaplanmasında karşılaşılan belirgin performans kaybının nedeni grafik işlemci belleklerinin hesaplam için yetersiz olmasından dolayı sistem belleğine erişim maliyetleridir.



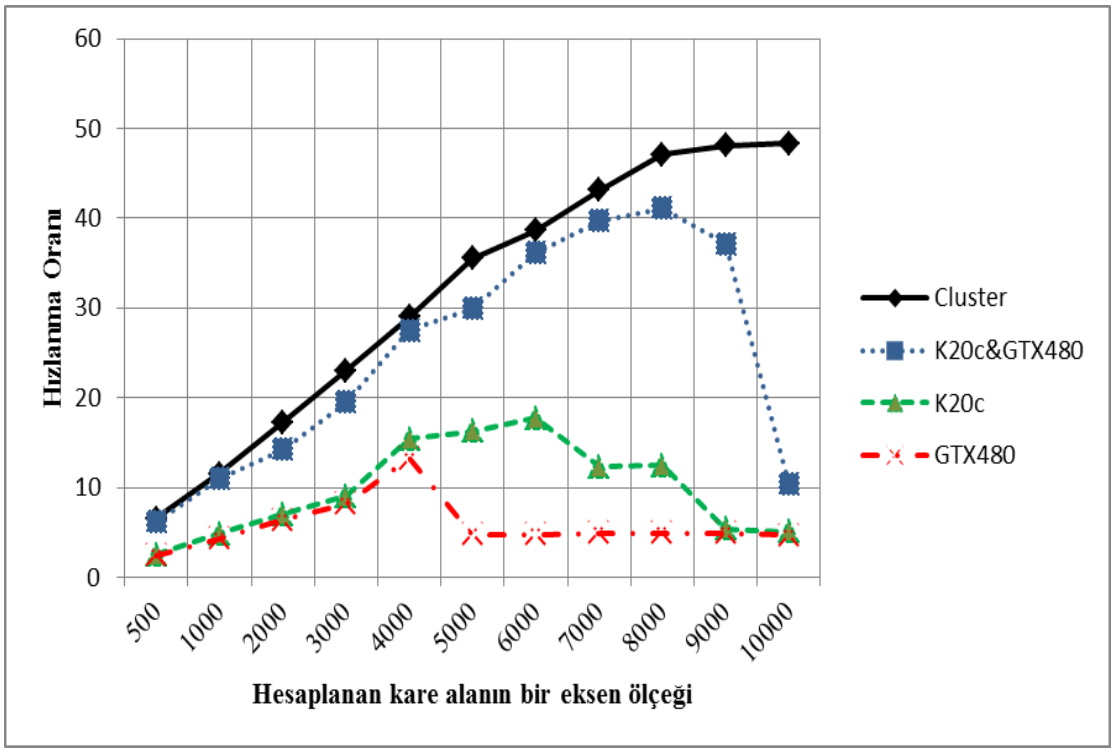
Şekil 8.37: Optimizasyonlar sonrasında ağ üzerindeki çok sayıda grafik işlemcinin birlikte gerçekleştirdikleri hesaplama performansındaki değişim.

Şekil 8.37 ile optimizasyonların uygulanması sonrasında ağ üzerinde yer alan ve Çizelge 4.3 ile açıklanan grafik işlemcilerin bir arada kullanılması neticesinde elde edilen hesaplama performans değişimi ortaya konmuştur. Çizelge 8.16 incelendiğinde seri hesaplama performansına kıyasla en iyi hesaplama performansının 10.000 x 10.000 ölçeğinde gerçekleştiği görülmektedir. Hızlanma oranı ise 48,34 kat olarak gerçekleşmiştir. Ancak unutulmaması gereken nokta hesaplama maliyetlerine ağ üzerinden gerçekleştirilen veri transfer sürelerinin eklenmemiş olduğudur.



Şekil 8.38: GTX 480 ve K20c ile alan hesaplama performansının cluster ile hesaplama performansı ile karşılaştırılması.

Çok sayıda grafik işlemcinin bir arada kullanılmasının iki yöntemi Şekil 8.38 ile karşılaştırmalı olarak ortaya konulmuştur.Grafikten de görüldüğü üzere 1000 x 1000 ölçeğinden itibaren fazla sayıda grafik işlemci kullanmanın avantajı ön plana çıkmaktadır. Özellikle K20c ve GTX480 grafik işlemcilerinin toplam bellek miktarının hesaplanan alan ihtiyacını kaşılamadığı 8000x8000 ölçeğinden itibaren aradaki fark büyümeye başlamaktadır. Ancak halen ağ ortamı üzerinden haberleşme maliyetleri hesaplama maliyetlerine eklenmemiştir.



Şekil 8.39: Farklı grafik işlemcilerin hesaplama performanslarının karşılaştırılması.

Şekil 8.39 ile optimizasyonların uygulanmasının ardından elde edilen hesaplama performanslarının karşılaştırması gösterilmiştir. En iyi hesaplama performans değerleri grafikten de görüldüğü üzere birden fazla grafik işlemcinin bir arada kullanılması ile gerçekleştirilen uygulamalarda elde edilmiştir.

GTX 480 ve K20c işlemcileri arasında oluşan performans farkı ise Şekil 2.16 ve Şekil 2.17 ile mimarileri gösterilen GK110 Kepler ve GF100 Fermi mimarilerinde yer alan ve bellek erişim performansını arttıran Load/Store Unit sayılarının farklı olmasından kaynaklanmaktadır (Kepler için 32 adet, Fermi için 16 adet). Bu durum bellek erişim bant genişliklerini etkilemekte ve belleğe erişim sürelerini dolayısıyla hesaplama performansını etkilemektedir. Benzer bir durum GTX 670MX ve Quadro 1000M grafik işlemcileri içinde geçerlidir.

Bu aşamaya kadar gerçekleştirilen değerlendirmeler kapsamında her ne kadar en iyi hesaplama performansının ağ ortamı üzerinde yer alan çok sayıda işlemci ile gerçekleştirildiği ön görülse dahi hesaplama performansına dahil edilmeyen ancak önemli ölçüde etkileyeceği değerlendirilen bir diğer husus ise ağ ortamı üzerinden veri aktarım maliyetlerinin etkisidir. Şekil 8.14 ile ortaya konulduğu üzere ağ ortamı üzerinde birden fazla grafik işlemcinin bir arada kullanılması istendiğinde oluşturulan sistemdeki en yavaş bağlantı noktası ağ iletişiminden kaynaklanmaktadır.

Çizelge 8.17: Şekil 8.18 ile gösterilen ağ üzerinde girdi verisi aktarım maliyetlerinin gösterimi.

Alan Ölçeği	Toplam Hücre Sayısı	Giriş Verisi Büyüklüğü (MB)	Toplam aktarım süresi (teorik) (s.)	Toplam aktarım süresi (gerçekleşen) (s.)
500	250.000	0,95	0,0071	0,0084
1000	1.000.000	3,81	0,0284	0,0311
2000	4.000.000	15,26	0,1137	0,1254
3000	9.000.000	34,33	0,2558	0,2847
4000	16.000.000	61,04	0,4548	0,4954
5000	25.000.000	95,37	0,7106	0,7521
6000	36.000.000	137,33	1,0232	1,1245
7000	49.000.000	186,92	1,3927	1,5852
8000	64.000.000	244,14	1,8190	1,9457
9000	81.000.000	308,99	2,3022	2,5642
10000	100.000.000	381,47	2,8422	2,9945

Şekil 8.18 ile gösterildiği gibi oluşturulan bir sistem üzerinde örnek alan ölçeklerinin oluşturduğu veri miktarı ve bu verinin iki yönlü olarak aktarılması istendiğinde ağ bağlantı hızına bağlı olarak gereksinim duyulacak olan teorik süre miktarları Çizelge 8.17 ile gösterilmiştir. Girdi verisi ile ifade edilen potansiyel alanın hesaplanabilmesi için gerekli olan yükselti alanı bilgisi, formasyon dahilinde yer alan diğer platformların konum bilgisi, hedefin konum bilgisi ve formasyon şeması bilgisidir. Bu bilgilerden sayısal yükselti bilgisi haricinde diğerleri haberleşme maliyeti açısından önemsenmeyebilir. Çıktı verisi ile kasıt ise formasyon dahilinde yer alan her bir platformun yönlendirilmesi kullanılacak ve genel yol planlamasını ortaya koyabilecek nitelikte her bir platform için iki adet olmak üzere hız ve yön komut tablolarıdır. Girdi ve çıktı verisinin bellekte kapladıkları alan hesaplanmasında veri tipi olarak single çönürlüklü float ve formasyon dahilinde dört platform olduğu kabul edilmiştir. Örnek alan ölçeği olarak 4000 x 4000 ölçeği seçildiğinde hesaplamanın yapılabilmesi için giriş verisi olarak 61,04 MB, çıktı verisi içinde 976,56 MB büyüklüğünde veri aktarılmasına ihtiyaç duyulacak olup, 2 Gbps band genişliğine sahip bir ağ ortamı üzerinden giriş verisinin aktarılması teoride 0,49 s., çıktı verisinin aktarılması ise teoride 3,81 s. gibi bir süre alacaktır.

En temel anlamda Şekil 8.20 ile gösterilen yapıya benzer bir sistem çözümü ile çıktı verisinin aktarım maliyetlerinden tamamen tasarruf edilebilir. Çünkü ortaya konulan yapı ile hesaplanmak istenen potansiyel alanın hangi bölümlerinin hangi grafik işlemci üzerinde hesaplandığını bilmek dolayısıyla sonuçların ağ ortamı üzerinde yer alan hangi sunucuda saklandığını bulmak mümkündür. Dolayısıyla şayet özel bir nedenden dolayı aracı yönlendiren tüm komut tablolarının tek bir sunucu üzerinde tutulmasına gerek duyulmuyor ise tablo verisi sistem üzerinde yer alan sunucu belleklerinde dağınık olarak saklanabilir.

Çizelge 8.18: Veri aktarım maliyetleri ile birlikte ağ ortamı üzerinde yer alan çok sayıda grafik işlemci ile gerçekleştirilen hesaplama performansları.

Alan Ölçeği	Toplam Hücre Sayısı	CPU Hesaplama Süresi (s.)	GPU Cluster Hesaplama Süresi (s.)	Veri aktarım süresi (gerçekleşen) (s.)	H2D + GPU Hesaplama (s.)	Hızlanma (X)
500	250.000	0,017	0,0026	0,0084	0,011	1,55
1000	1.000.000	0,068	0,0059	0,0311	0,037	1,84
2000	4.000.000	0,264	0,0153	0,1254	0,1407	1,88
3000	9.000.000	0,563	0,0244	0,2847	0,3091	1,82
4000	16.000.000	1,087	0,0375	0,4954	0,5329	2,04
5000	25.000.000	1,490	0,0419	0,7521	0,794	1,88
6000	36.000.000	1,833	0,0474	1,1245	1,1719	1,56
7000	49.000.000	2,176	0,0504	1,5852	1,6356	1,33
8000	64.000.000	2,620	0,0556	1,9457	2,0013	1,31
9000	81.000.000	2,963	0,0616	2,5642	2,6258	1,13
10000	100.000.000	3,307	0,0684	2,9945	3,0629	1,08

Bu durumda hesaplama maliyetini etkileyen tek haberleşme maliyeti girdi verilerinin aktarılmasından kaynaklanacaktır. Şekil 8.18 ile gösterildiği gibi oluşturulan bir ağ sistemi ve bant genişliklerinden faydalanılarak aktarılacak olan değişik ölçeklerdeki alanların hesaplanmasına yönelik haberleşme maliyetleri Çizelge 8.17 ile gösterilmiştir. Bu kapsamda 4000 x 4000 ölçeğindeki bir potansiyel alanın hesaplanması istendiğinde aktarılacak olan verinin haberleşme maliyeti 0,03 sn olarak gerçekleşmektedir.

Bu durumda Çizelge 8.18 ile de farklı ölçeklerde alanlar için gösterilen şekilde hesaplama performansında önemli kayıplar yaşanmaktadır. Örneğin 4000 x 4000 ölçeğindeki potansiyel alanın hesaplama performansı haberleşme maliyetleri

önemsenmeden seri hesaplama karşısında 28,98 kat hızlanmışken haberleşme maliyetlerinin eklenmesi ile bu oran 2,04 değerine gerilemiştir.

Seri hesaplama performansı neticesinde ortaya konulan ve S_2 olarak adlandırılan senaryonun Aerosonde platformları tarafından icra edilen bir senaryo olduğu kabulüne geri dönerek, GTX 480 işlemcisi ve K20c & GTX480 grafik işlemcilerinin birlikte kullanımı ile elde edilen hesaplama performansı sonuçları

Çizelge 8.19 ile gösterilmiştir. S_2 senaryosunun hesaplaması K20c & GTX480 grafik işlemcilerinin birlikte kullanımı ile paralel olarak hesaplandığında seri hesaplamaya kıyasla elde edilen hızlanma 19,85 kat olarak elde edilmiştir. Bu durumda birim uzunluğun 20 metre dolayısıyla 4000 x 4000 ölçeğindeki bir alanın 80 km. x 80 km. olduğu kabul edildiğinde, alanın hesaplanması için gerekli süre yaklaşık 0,7 sn. olacaktır. Bu süre zarfında platformların hareketine devam ettiği gerçeğinden hareket edilir ise hesaplama için geçen sürede her bir platform yaklaşık 28 m. yer değiştirecektir. Bu durumda Çizelge 5.1 ile ifade edilen formasyon şemalarından her hangi birini platform aralığı 28 m. den az olacak şekilde uygulamak veya 28 m. mesafeden yakın engellerden sakınmak amacıyla gereksinim duyulan yol planlamasının istenen zaman aralığında üretilmesi seri hesaplama yaklaşımı ile Çizelge 8.11 ile özellikleri gösterilen Şekil 8.18'daki sistem üzerinde özgün CUDA ile geliştirilmiş hesaplama uygulamaları vasıtasıyla hesaplandığı durumlarda mümkün değildir.

Çizelge 8.19: Farklı senaryolar için paralel hesaplama performansı.

Örnek Senaryo	Seri Hesaplama		GTX480			K20c & GTX480		
	Alanın Toplam Hesaplama Süresi (µs.)	Bir hücre Hesaplama Süresi (µs.)	Alanın Toplam Hesaplama Süresi (µs.)	Bir hücre Hesaplama Süresi (µs.)	Seri Hesaplamaya Göre Hızlanma (X)	Alanın Toplam Hesaplama Süresi (µs.)	Bir hücre Hesaplama Süresi (µs.)	Seri Hesaplamaya Göre Hızlanma (X)
S_1	781.600	0,78	185.213,27	0,19	4,22	87.919,01	0,088	8,89
S_2	14.015.741	0,80	1.214.535,62	0,08	11,54	706.082,67	0,044	19,85
S_3	13.865.910	0,87	1.081.584,24	0,07	12,82	655.288,75	0,041	21,16
S_4	777.260	0,78	181.179,49	0,18	4,29	90.274,10	0,090	8,61
S_5	92.015	0,77	14.652,07	0,12	6,28	8.245,07	0,069	11,16
S_5	781.840	0,78	187.491,61	0,19	4,17	89.251,14	0,089	8,76
S_6	770.140	0,77	190.158,02	0,19	4,05	91.465,56	0,091	8,42
S_7	29.926	0,75	7.108,31	0,18	4,21	2.613,62	0,065	11,45
S_8	155.100	0,78	31.782,79	0,16	4,88	13.393,78	0,067	11,58
S_9	18.865.910	1,18	1.969.301,67	0,12	9,58	984.650,84	0,062	19,16
S_{10}	77.965.946	0,78	17.481.153,81	0,17	4,46	8.941.048,85	0,089	8,72
S_{11}	1.041.525	0,83	210.835,02	0,17	4,94	47.063,94	0,038	22,13
S_{12}	269.616.305	2,70	63.588.751,18	0,64	4,24	33.244.920,47	0,332	8,11

Çizelge 8.20: 4000x4000 ölçeğinde birim çözünürlük değerine sahip çeken alanın hesaplama performansının farklı yöntem ve donanımlar ile hesaplama performansı karşılaştırması**.**

	Hesaplama Donanımı	Toplam Alan Hesaplama Süresi (s)	Bir Hücre Hesaplama Süresi (ns)	Seri Hesaplama Karşısında Hızlanma Oranı
Seri Hesaplama	**Intel i7**	1,0866	68	-
Matlab Paralel Hesaplama Araç Takımı	**NVIDIA K20c**	0,2264	14	4,8
CUDA ile programlanmış	**Quadro 1000M**	0,3094	19	3,51
	GTX 670MX	0,1654	10	6,57
	GTX 480	0,1033	6	10,52
	K20c	0,0979	6	11,1
	K20c+GTX480	0,0496	3	21,92
	Cluster	0,0396	2	27,46
CUDA ile programlanmış ve optimize edilmiş	**K20c**	0,0705	4	15,40
	GTX480	0,0821	5	13,23
	K20c+GTX480	0,0394	2	27,56
	Cluster	0,0375	2	28,98
İletişim maliyetleri dahil edilmiş ve optimize edilmiş	**Cluster**	0,5329	33	2,04

Çizelge 8.20 ile 4000 x 4000 ölçeğinde ve birim çözünürlük değerine sahip alan içinde yer alan tek bir çeken kaynağa ait vektör alanın hesaplama performansı farklı hesaplama programları ve farklı donanımlar üzerinde elde edilen sonuçlara istinaden karşılaştırmalı olarak ortaya konulmuştur. Çizelgeden de görüldüğü üzere seri hesaplama performansına nazaran elde edilen en iyi hesaplama zamanı CUDA programlama dilinde gerçekleştirilmiş ve önceki bölümler kapsamında açıklanan optimizasyonların uygulanması ile şayet ağ ortamı üzerinde gerçekleştirilen veri

transfer maliyetleri göz önünde bulundurulur ise aynı ana sunucu üzerinde yer alan birden fazla grafik işlemcinin bir arada kullanılması ile elde edilmiştir.

Çizelge 8.21: S_2 senaryosu kapsamında tanımlanan alanın hesaplama performansının farklı yöntem ve donanımlar ile hesaplama performansı karşılaştırması.

	Hesaplama Donanımı	Toplam Alan Hesaplama Süresi (s)	Bir Hücre Hesaplama Süresi (ns)	Seri Hesaplama Karşısında Hızlanma Oranı	Platformlar Arası Min. Mesafe (m)
Seri Hesaplama	**Intel i7**	14,00	875	-	560
Matlab Paralel Hesaplama Araç Takımı	**NVIDIA K20c**	3,59	224	3,9	144
CUDA ile programlanmış ve optimize edilmiş	**Quadro 1000M**	4,50	282	3,1	180
	GTX 670MX	2,20	137	6,4	88
	GTX 480	1,20	76	11,5	48
	K20c	1,00	63	13,9	40
	K20c+GTX480	0,71	44	19,9	28
	Cluster	0,54	34	26,0	22

Çizelge 8.21 ile S_2 olarak tanımlanan senaryonun hesaplama performansının farklı paralel programlama yöntemleri ve farklı grafik işlemciler üzerindeki başarısı karşılaştırmalı olarak ortaya konulmuştur. Haberleşme maliyetleri de göz önünde bulundurulduğunda en iyi hesaplama performansı aynı ana makine üzeride yer alan çok sayıda GPU ile elde edilmektedir. Bu durumda en yüksek hızı 40 m/s olan Aerosonde platformlarından oluşan dört platformun yol planlaması gerçekleştirildiği kabul edildiğinde platformların arasındaki mesafe 28 m. olacak şekilde gerçek zamanlı engellerden sakınan ve çarpışmayı önleyen bir genel yol planlaması gerçekleştirilebilinecektir.

9. SİMÜLASYON VE UYGULAMA

Çalışmanın bu bölümünde, bu aşamaya kadar ortaya konulan yaklaşımların farklı mobil araç modellemelerinden faydalanılarak örnek senaryolar kapsamında ihtiyaç duyulan yol planlaması ihtiyacına ne derece yanıt verebildiği incelenmiştir. Simülasyon ve uygulamalar ile yaklaşımın başarısı ortaya konulmaya çalışılmıştır.

Mobil araç olarak çalışma kapsamında belirlenen İHA platformlarının simülasyon ortamında davranışlarının temsil edilebilmesi için öncelikle YPA tabanlı yol planlaması problemlerinde yaygın olarak faydalanılan noktasal yüklü araç dinamiğine dayanan simülasyonlar gerçekleştirilmiştir. Ardından çalışma kapsamında uygulamalarda kullanılması planlanan AR Drone quadrotor platformunu temsil edebilecek döner kanatlı drone modellemesi yapılmış ve simülasyon çalışmalarında faydalanılmıştır.

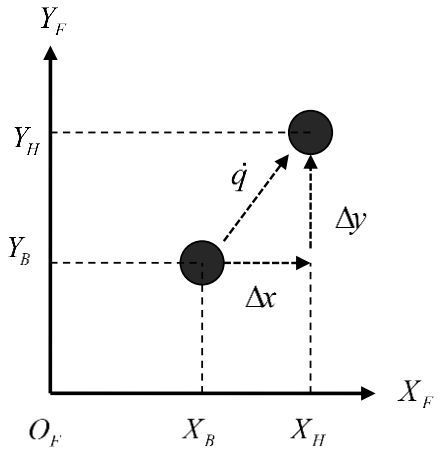
Son olarak yaklaşımın başarısının görülebilmesi açısından farklı uygulamalar gerçekleştirilmiş ve yaklaşımın başarısı ortaya uygulamalı olarak konulmuştur.

9.1 Noktasal Yüklü Araç Dinamiği

Noktasal yüklü araç dinamiği gerçek aracın en basit modelini temsil eder ve hareket planlama çalışmalarında potansiyel alan yaklaşımları üzerine icra edilen çalışmaların etkinliğinin incelenmesi sırasında en yaygın kullanılan modelidir [95]. Simülasyon ortamında Şekil 1.1 ile gösterildiği biçimde en temel mobil robotu temsil etmenin yöntemi olarak noktasal yüklü araç dinamiği kullanılabilir. Bu modelleme ile simülasyon ortamında Şekil 2.12 ile gösterilen açık hareket planlaması bileşenlerinden yol planlaması modülüne yanıt üretilebilir.

Her ne kadar İHA platformları üç boyutlu ortamda hareket yeteneğine sahip mobil araçlar olsalar dahi, çalışma kapsamında ortaya konulan ve hesaplama performansını arttırmayı amaçlayan çok katmanlı modelleme uygulamasında, her katman iki boyutlu bir hareket ortamı olarak kabul edilebileceğinden, bu katmanlar üzerinde planlanan yolun etkinliğinin ölçülmesi amacıyla iki boyutlu ortamda her aracın, *x-y*

ekseninde iki serbestlik derecesiyle noktasal yüklü bir aracın dinamiklerini gösterdiği varsayılmıştır [13][14].



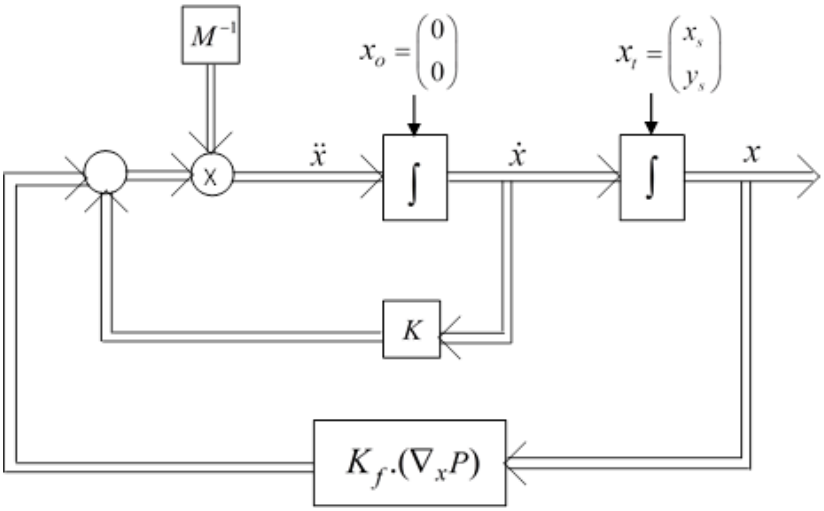
Şekil 9.1: Noktasal yüklü tek aracın hareketinin modellenmesi.

Bu kapsamda iki boyutlu uzayda aracın hareketinin modellenebilmesi amacıyla faydalanılan temel parametreler Şekil 9.1 ile gösterilmiştir. Orjin noktası O_F ile temsil edile iki boyutlu hareket uzayında noktasal yüklü aracın mevcut konumunun (X_B, Y_B) olduğu kabul edildiğinde aracın *x*-eksenindeki yer değişikliği Δx ile, *y*-eksenindeki yer değişikliği ise Δy parametresi ile temsil edilebilir. Bu durumda aracın toplam yer değişikliğini $\dot{q}$ parametresi, yeni konumunu ise (X_H, Y_H) temsil etmektedir.

Herhangi bir tip aracın hareketi ele alındığında, bu hareketi onun hız vektörüyle karakterize etmek mümkündür. Bu vektörün zamana göre integrali aracın uzay içindeki yolunu vermektedir. $\dot{x}$ ile gösterilen hız vektörü aracın uzay içindeki pozisyonudur. Noktasal yüklü bir aracın dinamik denklemini elde etmek için Denklem (9.1)' deki basit hareket denklemi kullanılarak, Denklem (9.2)'deki eşitlik üretilebilir:

$$F = M.a \tag{9.1}$$

$$M.\ddot{x} = -K_f(\nabla_x P) - K.\dot{x} \tag{9.2}$$



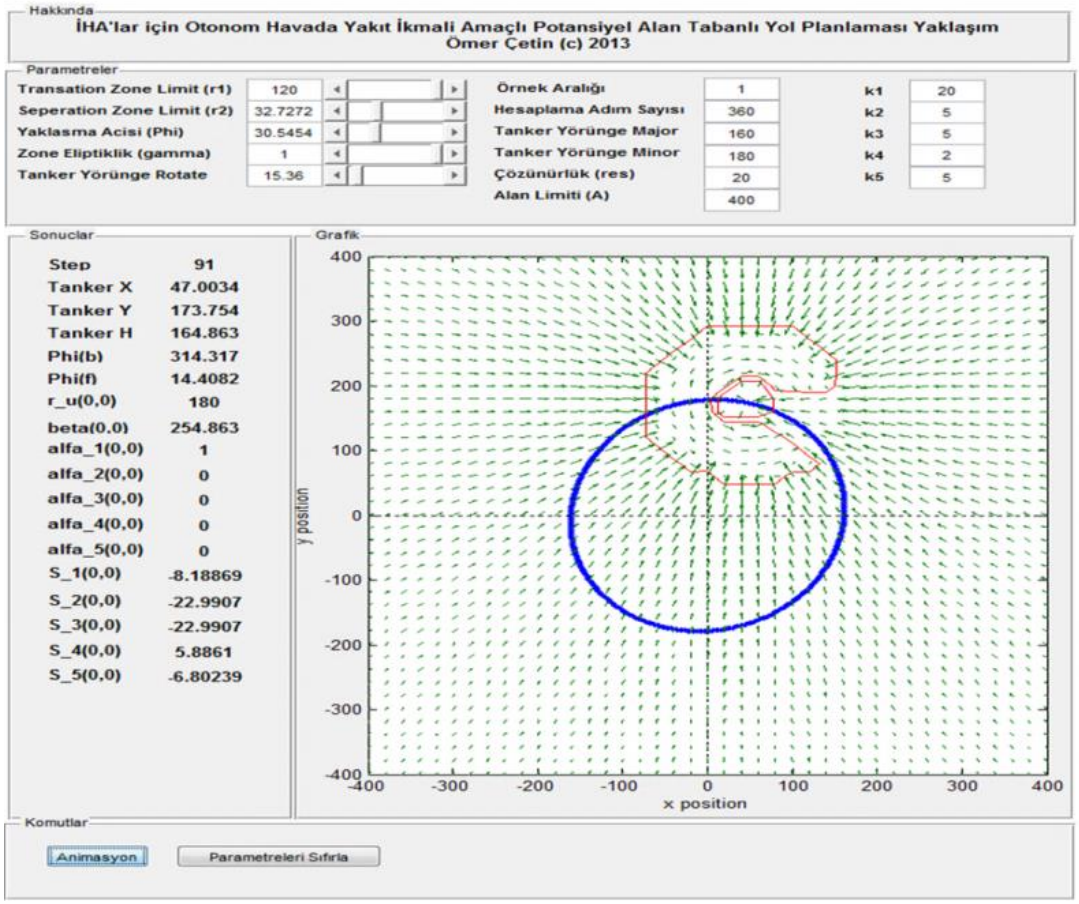
Şekil 9.2: Noktasal yüklü tek aracın sistem blok diyagramının gösterilmesi [13][14].

Noktasal yüklü aracın kütlesi $M = m.I_2$ şeklinde ifade edildiğinde, $-\mathrm{K_f}(\nabla_\mathrm{x}\mathrm{P})$ ifadesi sistemin giriş gücünü, $K = k.I_2$ ise sabit bir katsayıyı ifade etmektedir. Bu eşitlikler kullanılarak oluşturulan sistem blok diyagramı Şekil 9.2'de gösterilmiştir.

9.1.1 Noktasal yüklü araç dinamiği modeli ile havada yakıt ikmali simülasyonu

Çalışmanın bu bölümünde Çizelge 2.4 ile gösterilen konfigürasyon altında gerçekleştirilmiş simülasyon ortamında, ortaya konulan teorik yaklaşımın gerçekleştirilen sistem şeklinde başarısı sınanacaktır. Çizelge 2.11 ile özellikleri verilen NVIDIA GeForce GTX480 ekran kartından potansiyel alanın hesaplanmasında paralel donanım mimarisi olarak faydalanılmıştır. HYİ amaçlı tasarlanan vektör alanın paralel olarak hesaplanarak ortaya konulmasında NVIDIA CUDA paralel hesaplama yazılım desteğinden Bölüm 8.3.3 ile açıklanan şekilde faydalanılmıştır.

Çalışma esnasında parametrik değerlerdeki değişimin etkilerini görmek ve vektör alandaki değişimleri izleyebilmek amacıyla Şekil 9.3'de gösterilen biçimde bir grafik kullanıcı birimi geliştirilmiştir. Şekilden de görüldüğü üzere her bir parametre bağımsız olarak kolaylıkla değiştirilebilmektedir. Böylece yol planlaması yaklaşımı oldukça dinamik ve de esnek olarak sağlanmış, farklı tipte İHA platformlarının uygulayabileceği nitelikte bir yapı ortaya konulmuştur.



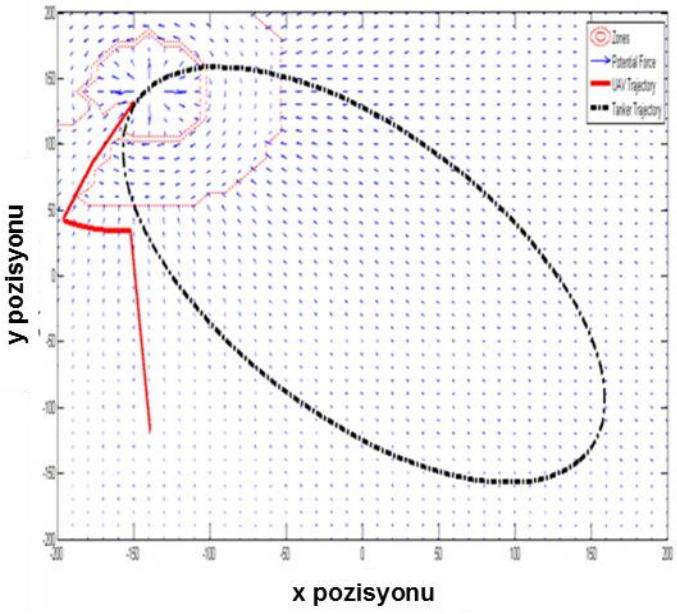
Şekil 9.3: Sigmoid sınır fonksiyonları ile desteklenmiş YPA tabanlı genel yol planlaması yaklaşımı.

HYİ görevi için geliştirilen YPA tabanlı otonom yol planlaması yaklaşımının başarısını gösterebilmek ve de gerçek zamanlı bir simülasyon ortamında sistemin hesaplama performansını ortaya koyabilmek amacıyla Çizelge 9.1'de yer alan parametrelere istinaden hesaplamalar gerçekleştirilmiştir.

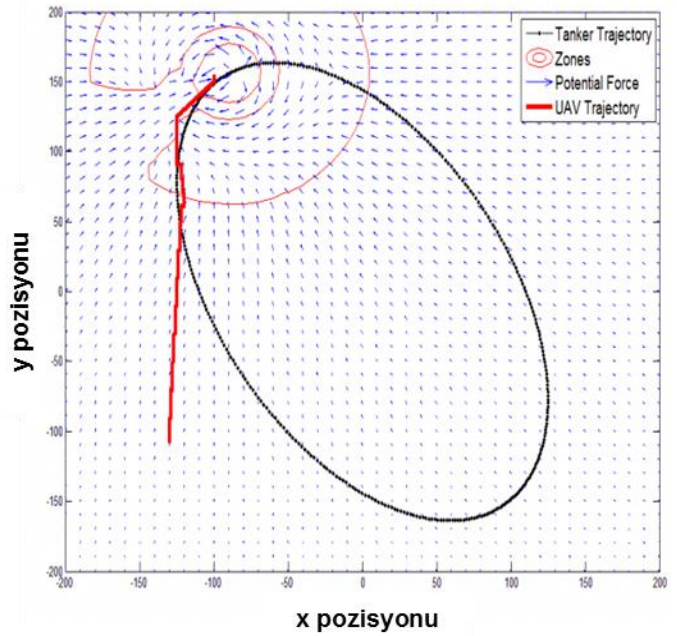
Çizelge 9.1: Simulasyon parametere değerleri.

Parametre	Değer
$[x_{s1}\ x_{s2}], [y_{s1}\ y_{s2}]$	[200 -200],[200 -200]
resolution	1
$q_u(0)$	-130, -110
$q_t(0)$	75,-150
δ	20
q_c	0,0
ax_{maj} , ax_{min}	100,50
$[r_1\ r_2]$	[90 40]
γ	1
$[k_1\ k_2\ k_3\ k_4\ k_5]$	[8 1 1 2 8]
ρ	0.091

Sigmoid ve ikili sınır fonksiyonları ayrı ayrı simülasyonlar dahilinde kullanılarak, ortaya çıkan yol planlamasının karşılaştırmalı değerlendirilmesine olanak sağlanmıştır. Dinamik araç temsili olarak simülasyon ortamında noktasal yüklü araç dinamiği modelinden faydalanılmış ve böylece platform tarafından ortaya konulan yol belirlenmiştir. Bu yol incelenerek yol planlamasının etkinliği tartışılmıştır.



(a) HYİ görevi için ikili sınır fonksiyonu tabanlı YPA modeli ile üretilen otonom yol.



(b) HYİ görevi için sigmoid sınır fonksiyonu tabanlı YPA modeli ile üretilen otonom yol.

Şekil 9.4: HYİ görevi için YPA tabanlı otonom genel yol planlaması.

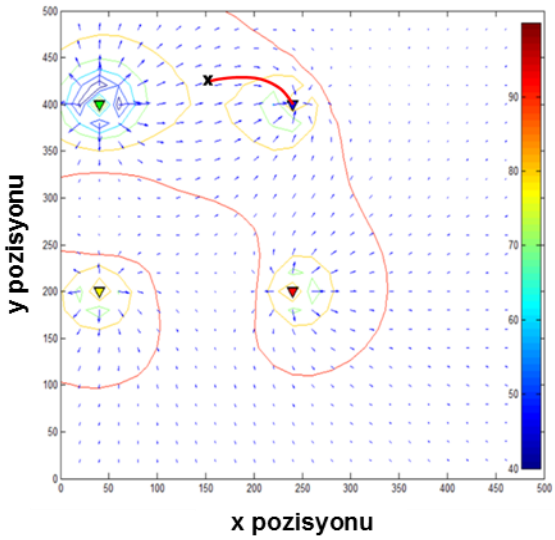
İHA platformunun hareketlerinin modellenmesi amacıyla noktasal yüklü paracığın ikili ve sigmoid sınır fonksiyonları ile modellenmiş vektörel alan içerisinde hareket ettirilmesi ile üretilen otonom gerçek zamanlı yol planlaması Şekil 9.6'de görüldüğü şekildedir. Elde edilen paternler her ne kadar benzerlikler gösterseler de iki patern arasındaki en büyük fark, sigmoid sınır fonksiyonları ile modellenen alan içerisindeki yol planlamasının nispeten daha yumuşak dönüşlere sebebiyet vermesi ve uygulamalarda kolaylık sağlamasıdır. Şekil 9.6 (a)'da yer alan keskin dönüş manevraları ihtiyacı Şekil 9.6 (b)'de görülmemektedir.

Çalışmanın bu bölümünde İHA sistemleri için HYİ görevi amacıyla otonom yol planlaması yaklaşımı sigmoid fonksiyonlar ile sınırlandırılmış birden fazla farklı karakterde YPA'nın bir arada kullanılması ile modellenmiştir. Tez çalışmasının önceki döneminde geliştirilen ikili (binary) üyelik fonksiyonları ile sınırlandırılmış yaklaşım yerine sigmoid fonksiyonlardan faydalanılmış ve böylelikle İHA platformlarının hareket yeteneklerine daha uygun bir patern üretilerek HYİ görevinin daha etkin olarak gerçekleştirilmesine imkan verecek nitelikte bir yaklaşım ortaya konulmuştur. Otonom yol planlama algoritmalarının dizayn edilmesi aşamasında ortaya çıkan bir önemli nokta da üretilen sonuçların platformlar tarafından uygulanabilir nitelikte olmasıdır. Özellikle İHA sistemleri gibi hareket kabiliyetleri insanlı sistemlere nazaran daha düşük olarak değerlendirilen platformlar için daha esnek ve yumuşak manevralara gereksinim duyan yaklaşımların geliştirilmesi gerekmektedir. Ortaya konulan yaklaşımın parametrik karakterine istinaden farklı tipte ve yetenekte platformlar için uyarlanabilmesi mümkündür.

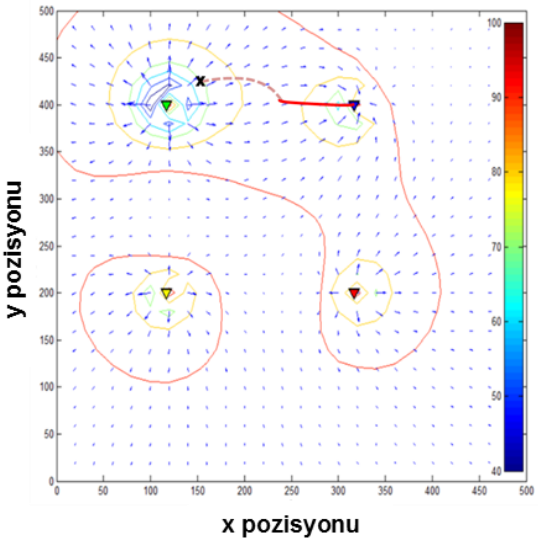
Çalışma kapsamında ele alınan konulardan bir diğeri ise grafik işlemcilerin paralel hesaplama yeteneklerinden faydalanılarak gerçek zamanlı hesaplama ihtiyacı duyan yaklaşımın performansının arttırılmasıdır. GPU mimarilerinin SIMD tipinde problemlerin çözümünde etkin yapılar olmasına istinaden, bu yaklaşıma uygun olan ızgara tabanlı potansiyel alan yaklaşımı MATLAB ortamında GPU tabanlı çok çekirdekli donanımlar ile paralel olarak hesaplanmıştır.

9.1.2 Noktasal yüklü araç dinamiği modeli ile formasyon halinde çarpışmayı önleme ve engellerden sakınma simülasyonu

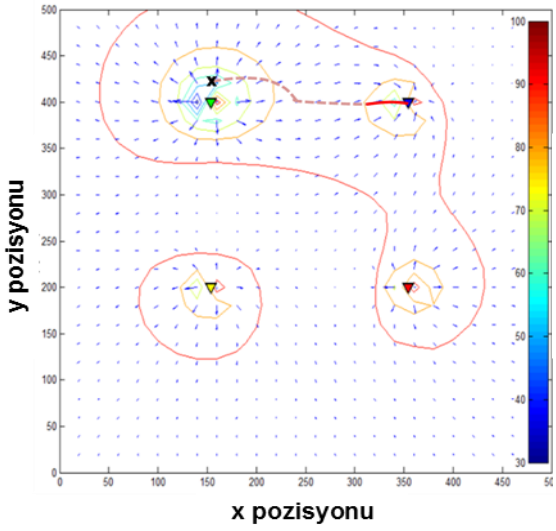
Çalışma kapsamında ortaya konulan formasyon kontrolü yaklaşımının etkinliği aşağıdaki örnekler ile noktasal yüklü araç dinamiği modelinden faydalanılarak ortaya konulmuştur. Zaman içinde yol planlaması kapsamında ele alınan platformun hareketlerinin vektörel alan yönlendirmesi ile nasıl bir değişim gösterdiği irdelenmiştir. Platformun zaman karşısında izlediği yol noktasal yüklü araç dinamiği modelinden faydalanılarak ortaya konulmuştur.



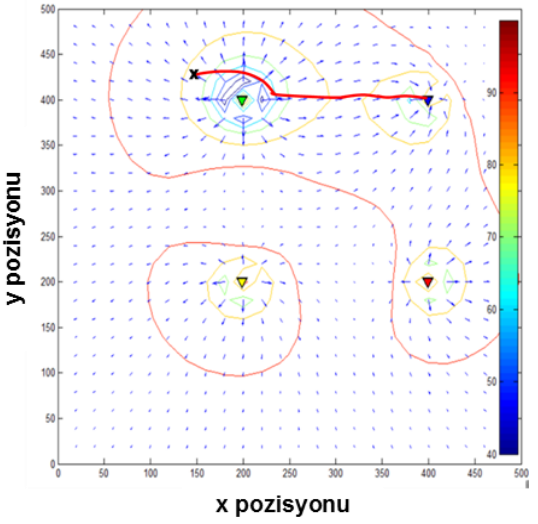
(a) t_0 ve t_{10} süreleri arasındaki yol gösterimi.



(b) t_{10} ve t_{20} süreleri arasındaki yol gösterimi.



(c) t_{20} ve t_{30} süreleri arasındaki yol gösterimi.



(ç) t_0 ve t_{40} süreleri arasındaki yol gösterimi.

Şekil 9.5: Vektör alan içerisinde konumun muhafazası için noktasal yüklü araç dinamiği ile modellenmiş aracın yolunun gösterilmesi.

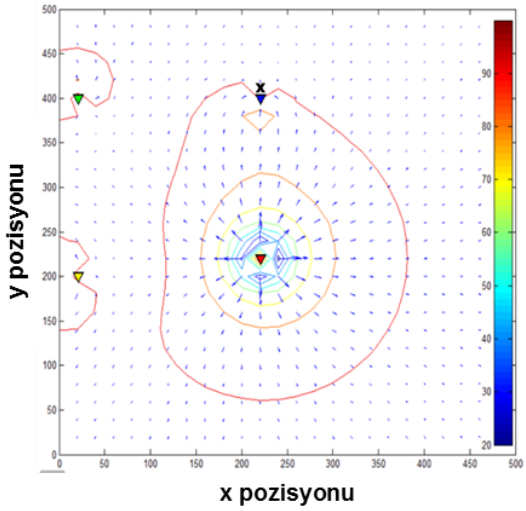
İlk simülasyon çalışması platformun formasyon içindeki konumu muhafaza etmesini göstermeyi, ikinci simülasyon çarpışmayı önlemek amacıyla yaklaşımın etkinkiliği, sonraki çalışmalar ise engellerden kaçınılması kapsamında yaklaşımın ne derece etkin yol planlamaları ürettiğini ortaya koymayı amaçlamaktadır.

Şekil 9.5 ile box formasyonunda hareket eden 4 adet platform görülmektedir. Kırmızı platform lider İHA'yı temsil etmekte, şekilde görülen vektör alan ise mavi ile temsil edilen 1 numaralı İHA'yı yönlendiren alan durumundadır. 2 ve 3 numaralı sarı ve yeşil ile gösterilen platformlar formasyon dahilindeki diğer platformlardır.

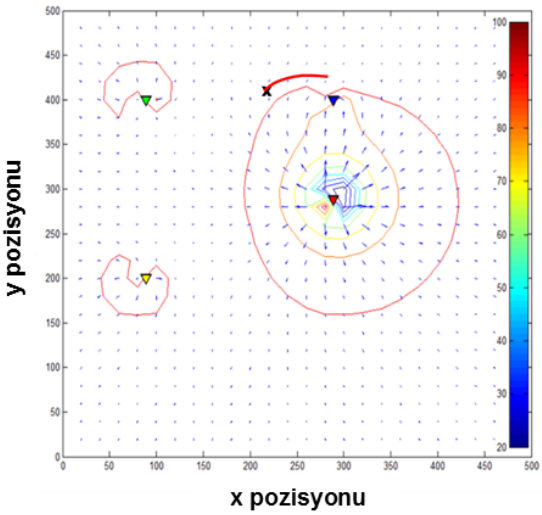
Şekil 9.5 ile gösterilen formasyon düzeninde amaçlanan mavi İHA'nın formasyon dahilinde sürekli doğru konuma yönlendirilmesini ve konumunu muhafaza etmesini amaçlar. Mavi platformun bulunması gerek noktanın etrafında oluşan vektörler sürekli olarak formasyon dahilindeki doğru konumu işaret etmektedir. Doğru pozisyon lider kırmızı İHA'nın konumuna göre hesaplanmaktadır. Şekil 9.5 (a) başlangıç pozisyonunu gösterirken, Şekil 9.5 (b,c ve ç) zaman içerisinde oluşan anlık durumlara ait potansiyel alanları temsil etmektedir.

Her bir *t* anında formasyon dahilinde yer alan platform sayısı kadar potansiyel alan hesaplaması yapılır. Bu durumda bu örnekte kırmızı platfom için üretilen potansiyel alan modelleri gösterilmiştir. Bu nedenle kırmızı dışında formasyon dahilinde yer alan platformlar iten kaynak noktaları olarak davranırlar ve kırmzı İHA'nın kendilerine bir çarpışma durumundan sakınmak amacıyla engel modeli gibi davranırlar.

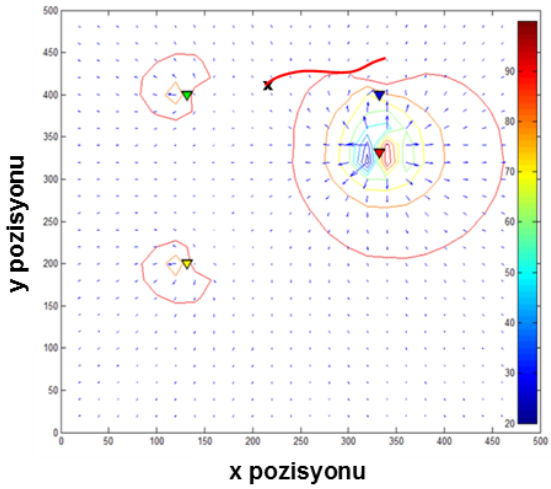
Şekil 9.6 ile gösterilen yapıda tekrar mavi platform için üretilen potansiyel alanlar gösterilmektedir. Kırmızı İHA ise bilinçli olarak formasyon dahilinde olması gerek noktadan saptırılmış ve süreç dahilinde kırmızı İHA'ya doğru hareket ettirilmiştir. Potansiyel alan içinde oluşan vektörler incelendiğinde mavi İHA'yı yönlendiren kuvvetlerin bu defa formasyon dahilinde platformun olması gereken noktanın dışında bir pozisyona platformu yönlendirdikleri görülebilir. Bunun nedeni çarpışmayı önlemek amacıyla olunması gereken noktadan ayrılmasına sebebiyet verilmesidir.



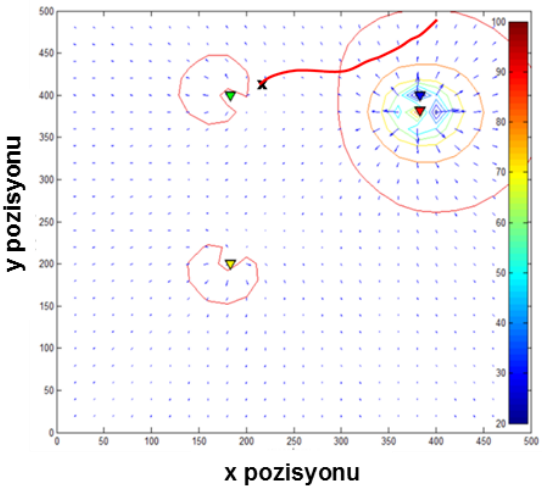
(a) t_0 ve t_{10} süreleri arasındaki yol gösterimi.



(b) t_{10} ve t_{20} süreleri arasındaki yol gösterimi.



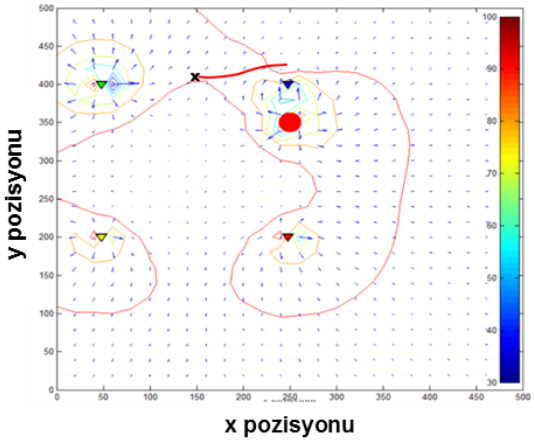
(c) t_{20} ve t_{30} süreleri arasındaki yol gösterimi.



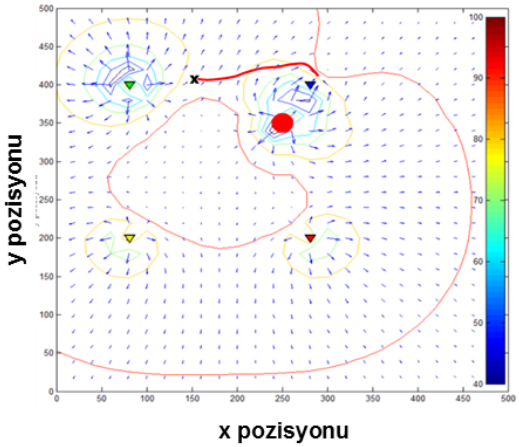
(ç) t_0 ve t_{40} süreleri arasındaki yol gösterimi.

Şekil 9.6: Vektör alan içerisinde çarpışmanın önlenmesi için noktasal yüklü araç dinamiği ile modellenmiş aracın yolunun gösterilmesi.

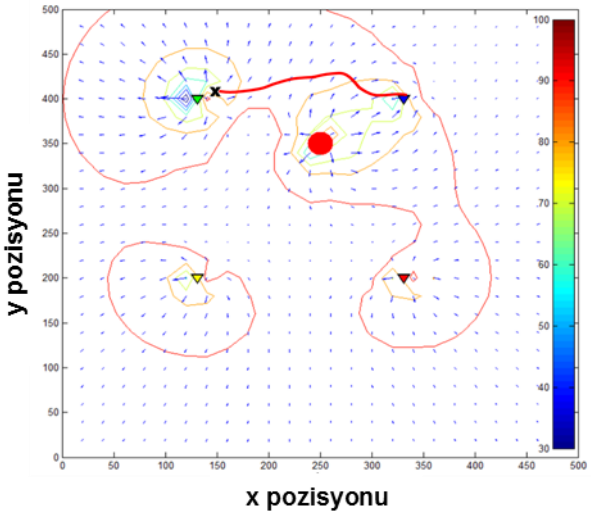
İten alan vektörlerinin kuvveti iki platform birbirlerine yaklaştıkça artacak ve çarpışmanın önüne geçilecektir. Bu durum zamana bağlı olarak modellenen ve Şekil 9.6 (b, c ve ç) ile gösterilen potansiyel alan modellerinden görülebilir.



(a) t_0 ve t_{10} süreleri arasındaki yol gösterimi.



(b) t_{10} ve t_{20} süreleri arasındaki yol gösterimi.

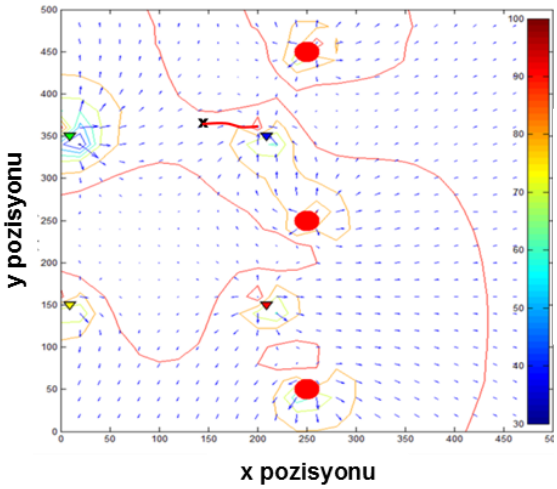


(c) t_{20} ve t_{30} süreleri arasındaki yol gösterimi.

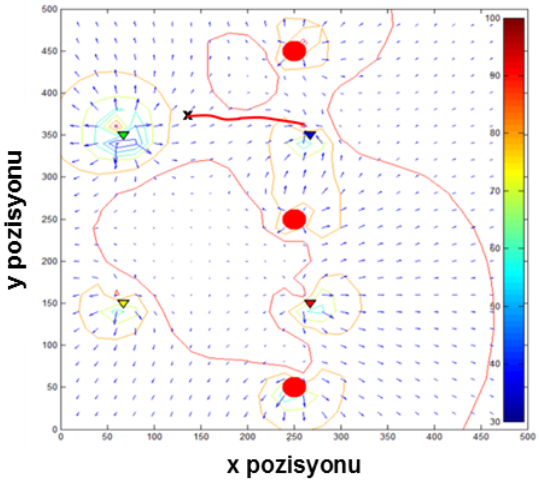
Şekil 9.7: Vektör alan içerisinde engelden kaçınma için noktasal yüklü araç dinamiği ile modellenmiş aracın yolunun gösterilmesi.

Şekil 9.7 kapsamında bu kez sabit bir engel tanımlaması yapılmıştır. Şekil 9.7 (a) ile görüldüğü üzere kırmızı İHA engelin yanında olduğu durumda formasyon düzeninde olması gereken konumun dışında bir noktaya yönlendirilmiştir. Böylece engel ile çırpışma ihtimali azaltılmaya çalışılmıştır. Kırmızı platformun formasyon düzeninde dışa doğru bir kaçınma hareketi yapmasının nedeni içe doğru oluşan vektörel kuvvetlerin şiddetinin engel ve lider platformun iten kuvvet alanlarının birleşimi ile nispeten daha kuvvetli olmasındandır. Beklenen davranış da çarpışmayı önlemek için bu şekilde olmalıdır. Şekil 9.7 (b ve c)'den de görüleceği üzere platform engelden uzaklaştıkça engelin iten potansiyel etkisi platformun formasyon dâhilinde olması gereken konumun çeken potansiyel etkisine nazaran azalacak ve engel bertaraf edildikten sonra platformun tekrar formasyon düzenindeki yerini alması sağlanacaktır.

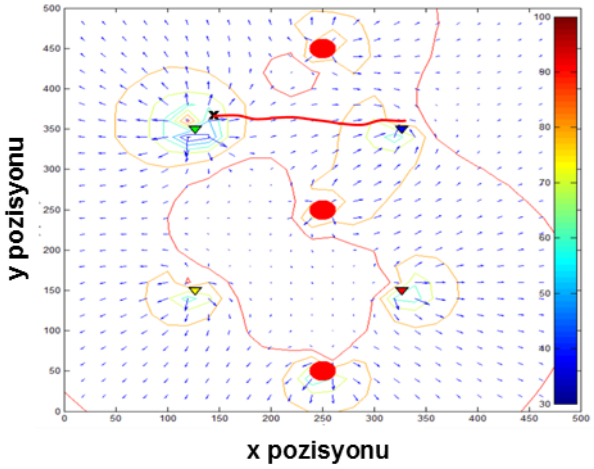
Şekil 9.8 ile alan içerisinde birden fazla engel olduğu durum gösterilmiştir. Şekil incelendiğinde bu sefer kırmızı platformun formasyonun dışına doğru yönlendirilmesi yerine neredeyse istikametinde devam etmesine yönelik vektör kuvvetlerin üretildiğinin görülmesidir. Bu kapsamda platformlar üzerinde oluşan kuvvetlerin platforma etkileri gösterilmiştir.



(a) t_0 ve t_{10} süreleri arasındaki yol gösterimi.



(b) t_{10} ve t_{20} süreleri arasındaki yol gösterimi.

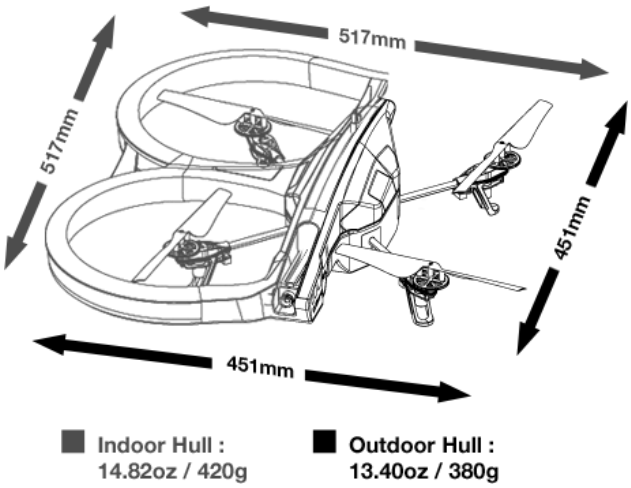


(c) t_{20} ve t_{30} süreleri arasındaki yol gösterimi.

Şekil 9.8: Vektör alan içerisinde engelden kaçınma için noktasal yüklü araç dinamiği ile modellenmiş aracın yolunun gösterilmesi.

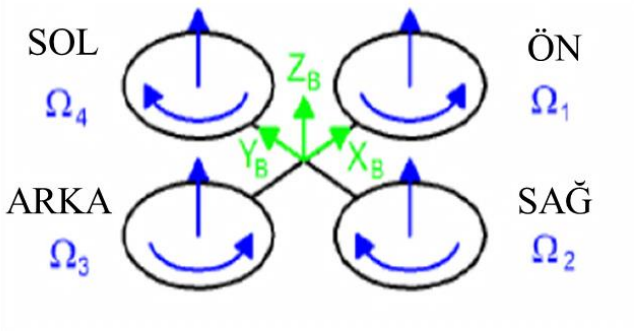
9.2 Döner Kanatlı İHA Modellemesi

Tez kapsamında gerçekleştirilen teorik çalışmaların Parrot AR Drone [96] platformu üzerinde uygulamalı olarak sınanmasına karar verilmiştir. Bu kapsamda bu bölüm dahilinde temel quad-rotor olarak adlandırılan ve mini bir İHA platformunun temel niteliklerini taşıyan Şekil 9.9'de gösterilen AR Drone platformunun otonom kontrolüne yönelik gerçekleştirilen çalışmalara değinilecektir.



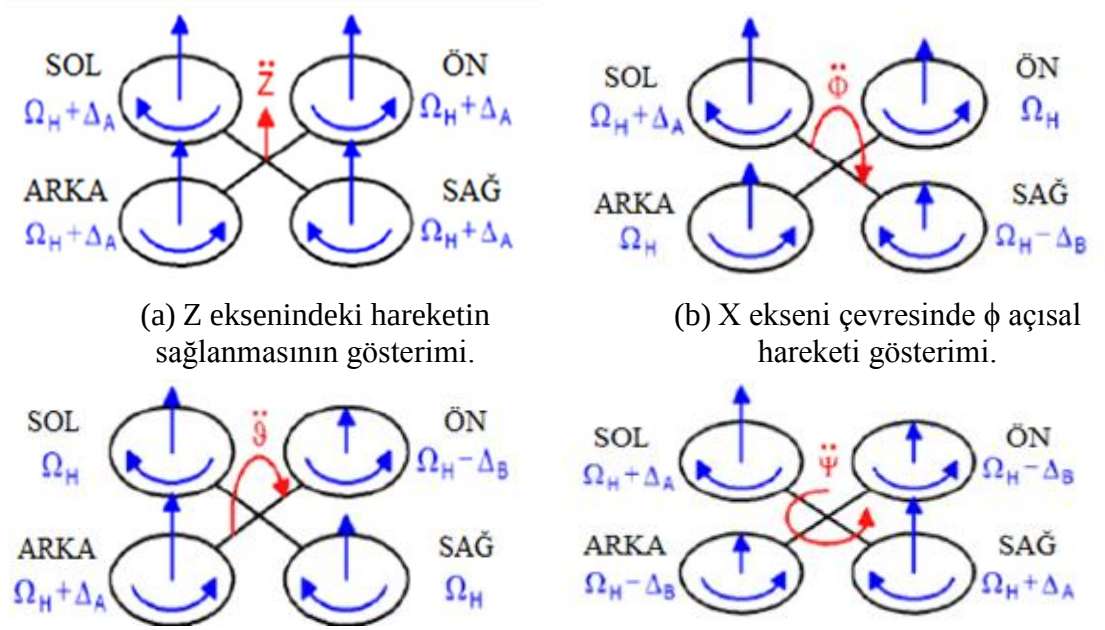
Şekil 9.9: Parrot AR Drone quadrotor platformu [96].

Simülasyon ortamında Şekil 1.1 ile gösterildiği biçimde noktasal yüklü araç dinamiğinin ardından platform davranışlarını modelleme amacıyla faydalanılan bir diğer yöntem hareketli robot modellemesidir. Çalışma kapsamında uygulama platformu olarak AR Drone quadrotor platformu belirlendiğinden hareketli robot modellemesi olarak döner kanatlı İHA modellemesi uygun olarak tespit edilmiştir. Bu modelleme ile simülasyon ortamında Şekil 2.12 ile gösterilen açık hareket planlaması bileşenlerinden yol planlaması modülüne aynı noktasal yüklü araç dinamiğinde olduğu gibi yanıt üretilebilir.



Şekil 9.10: Quadrotor platformun davranış kontrolünün gösterimi.

Quadrotor platformu adında anlaşılacağı üzere üzerinde birbirinden bağımsız dönüye sahip olan dört adet pervane (rotor) sisteminin ve kontrolünün yer aldığı bir platformdur. Şekil 9.10 ile gösterildiği üzere platform, üç boyutlu uzaydaki her hareketini bu bağımsız pervanelerin hareketinin bir koordinasyon şeklinde kontrolü ile sağlamaktadır.



(a) Z eksenindeki hareketin sağlanmasının gösterimi.

(b) X ekseni çevresinde ϕ açısal hareketi gösterimi.

(c) Y ekseni çevresinde θ açısal hareketinin gösterimi.

(ç) Z ekseni çevresinde ψ açısal hareketinin gösterimi.

Şekil 9.11: Quadrotor platformun farklı davranışları icra etmesinin gösterimi.

Platformun Z ekseni yönünde hareketi için, Ω_H olarak temsil edilen temel devir sayısına Δ_A kadar değişim her pervaneye eşit olacak biçimde Şekil 9.11 (a)'da gösterildiği şekilde uygulanır. Platformun *x*-ekseni çevresinde ϕ açısal hareketini uygulaması için Y ekseni üzerinde yer alan sol motoruna Δ_A ilave devri, sağ motoruna ise ters yönlü olarak Δ_B devri Şekil 9.11 (b)'de gösterildiği şekilde uygulanır. Platformun *y*-ekseni çevresinde θ açısal hareketini uygulaması için *x*-ekseni üzerinde yer alan arka motoruna Δ_A ilave devri, ön motoruna ise ters yönlü olarak Δ_B devri Şekil 9.11 (c)'de gösterildiği şekilde uygulanır. Son olarak platformun Z ekseni çevresinde ψ açısal hareketini uygulaması için *x*-ekseni üzerinde yer alan arka ve ön motoruna Δ_B ilave devri, *y*-ekseni üzerinde yer alan sol ve sağ motoruna ise ters yönlü olarak Δ_A devri Şekil 9.11 (d)'de gösterildiği şekilde uygulanır. Δ_A ve Δ_B değerleri motor devirlerindeki açısal hız değişimleri olarak kabul edilmelidir.

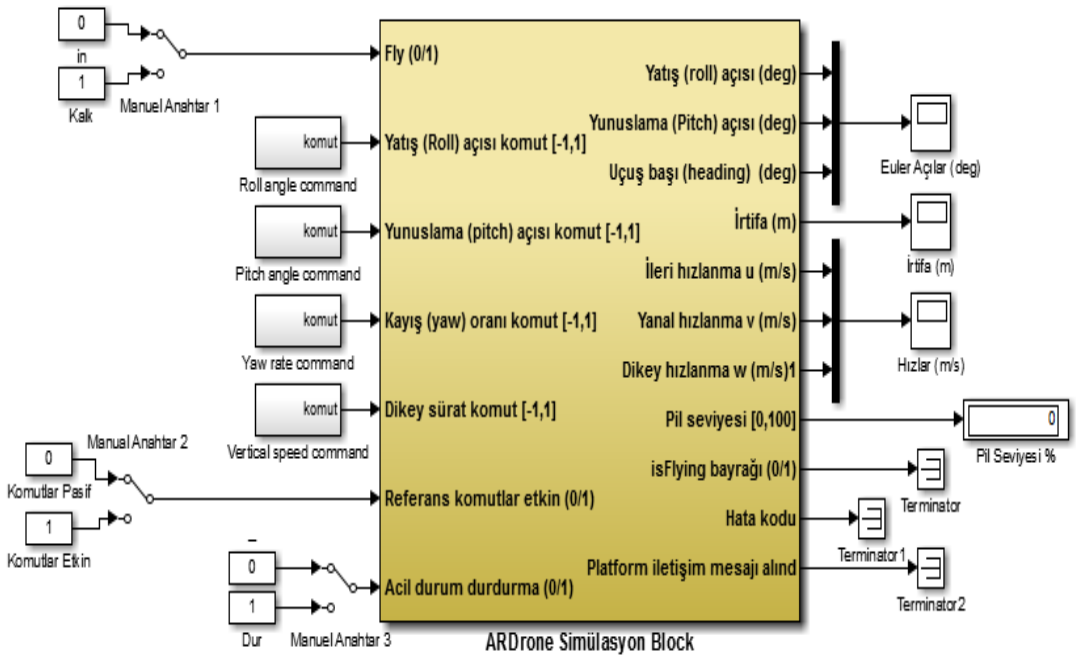
Tüm bu tanım quadrotor platformunun üç boyutlu uzaydaki hareketini nasıl şekillendirdiğinin anlaşılması için verilmiştir. Esasında devrin kontrol edilmesi için kapalı çevrim halinde koşturulan bir kontrol mekanizması mevcuttur. Bu mekanizma platformdan beklenen hareketi yapması için hangi rotora ne kadar devir değişikliği uygulancağını kapalı bir çevrim içinde, platform üzerinde yer alan algılayıcılardan aldığı geri dönüş verilerine göre hesaplayarak belirler. Bir diğer deyim ile platformun

yapması beklenen davranış her daim aynı devir değişikliği miktarları ile sağlanamaz bunu etkileyen çok sayıda parametre vardır. Bunlar aerodinamik etkiler, donanımsal etkiler, dış ortamdan kaynaklanan rüzgar gibi bozucular vb. olarak sıralanabilir. Bu kapsamda çalışma çerçevesinde simülasyon ortamında ve uygulamada quadrotor platformunun hareketinin modellenerek denetlenmesi amacıyla takip edilen başlılar altındaki yapılardan faydalanılmıştır.

9.2.1 İHA modelinin simülasyon ortamında geliştirilmesi

Çalışma kapsamında gerçekleştirilecek olan teorik çalışmalar belirtildiği üzere Şekil 9.9 ile gösterilen AR Drone Quadrotor [96] platformlarının bir arada kullanılmasıyla uygulamalı olarak sınanacaktır. Çalışmaların doğrudan gerçek platform üzerinde uygulanması öncesinde MATLAB Simulink ortamı üzerinde sınanması amacıyla

Şekil **9.12** ile gösterilen blok yapısında gösterilen simülasyon yapısı geliştirilmiştir. Bu blok sistem tanımlarından faydalanılarak aracın dinamiklerini benzetebilecek nitelikte geliştirilmiştir [97].

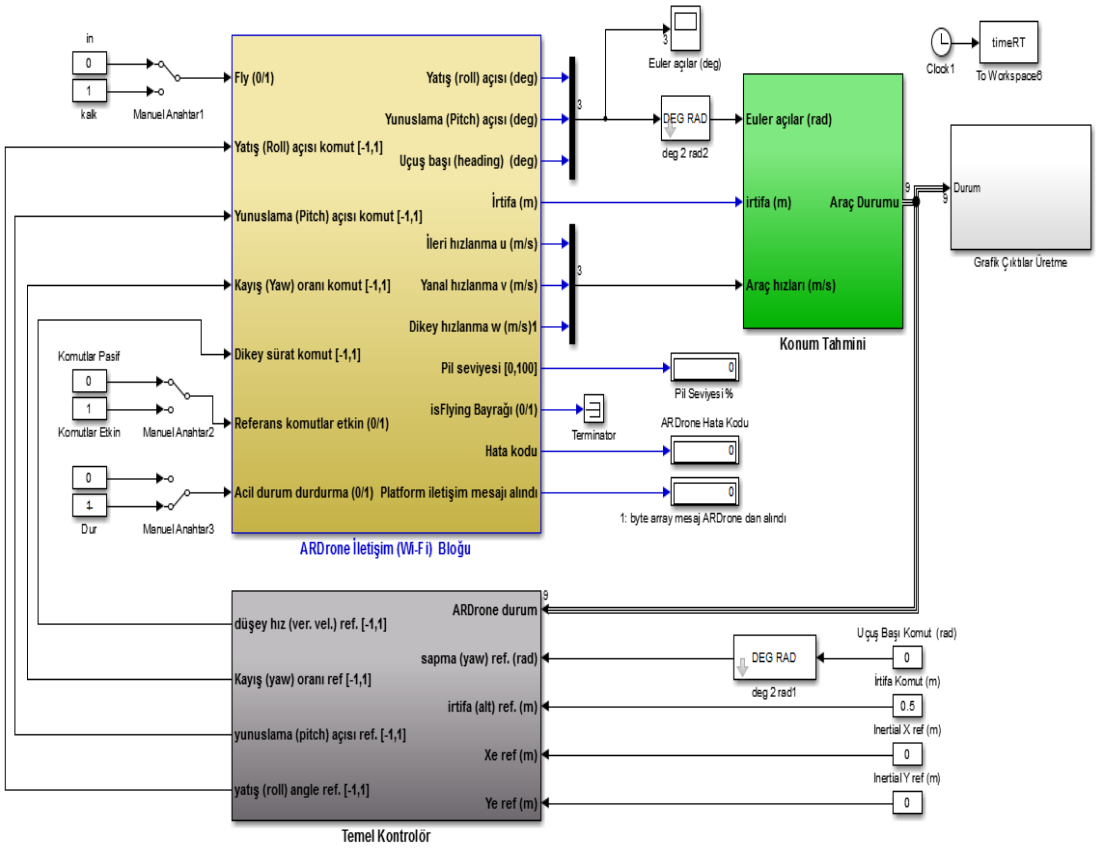


Şekil 9.12: MATLAB Simulink ortamında geliştirilen AR Drone modeli gösterimi.

AR Drone platformunun davranışlarının benzetilmesi amacıyla simülasyon bloğu içerisinde davranış modeli aşağıdaki Denklem seti (9.3) ile temsil edilen durum modelini baz alarak ortaya konulmuştur.

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du \qquad \textbf{(9.3)}$$
$$x|_{t=t_0} = x_0$$

x durum vektörü, u giriş vektörü, y çıktı vektörü ve x_0 ise başlangıç durumunu temsil eden durum vektörüdür. Tez çalışması kapsamında geliştirilen navigasyon, yol planlaması ve yönlendirme algoritmaları öncelikle bu simülasyon bloğu üzerinde sınanmaktadır. AR Drone platformunun gerçek zamanlı olarak kontrolünün sağlanması amacıyla, platform üzerinde yer alan kontrol yapısına kablosuz 802.11n bağlantı protokolü üzerinden AT komutları gönderilerek gerçek zamanlı kontrol sağlanabilir. Bu protokol platform üzerinde yer alan algılayıcıların elde ettiği verilerin paylaşılmasına da olanak vermektedir.



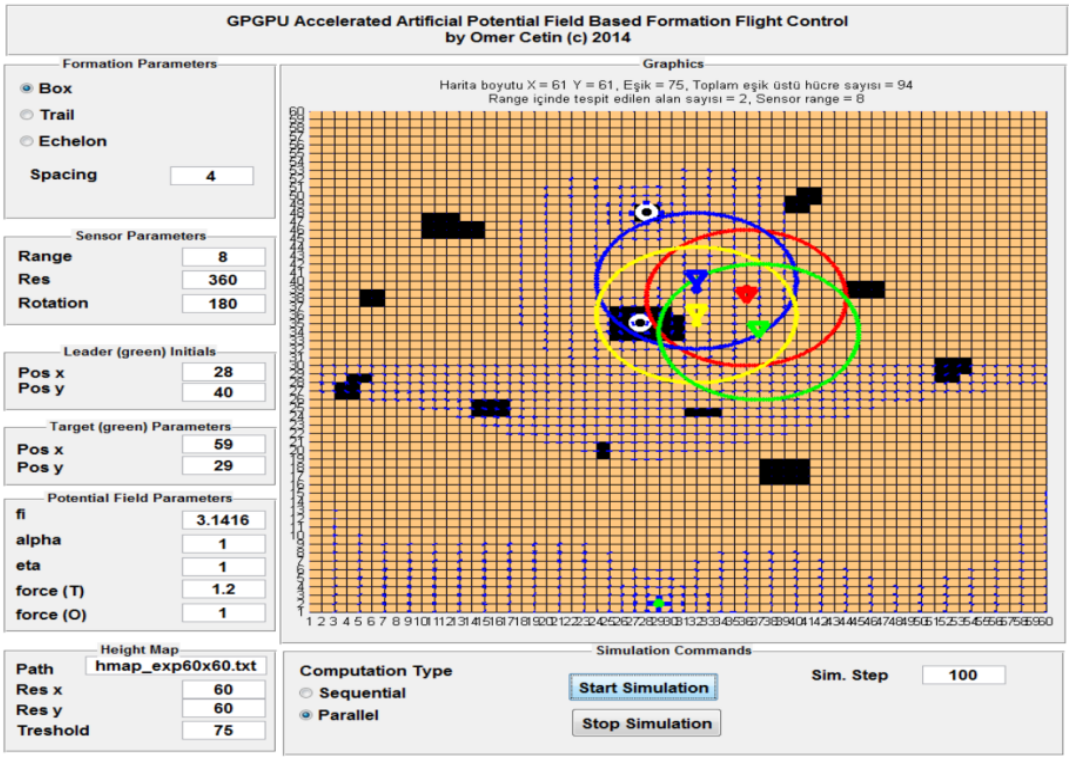
Şekil 9.13: MATLAB Simulink ortamında geliştirilen gerçek zamanlı AR Drone platform kontrolü.

AR Drone platformunun gerçek zamanlı kontrolünün sağlanması amacıyla Şekil 9.13 ile gösterilen model MATLAB Simulink ortamında geliştirilmiştir. Real-Time Windows Target versiyon R2013b modülünden faydalanılarak 802.11n kablosuz

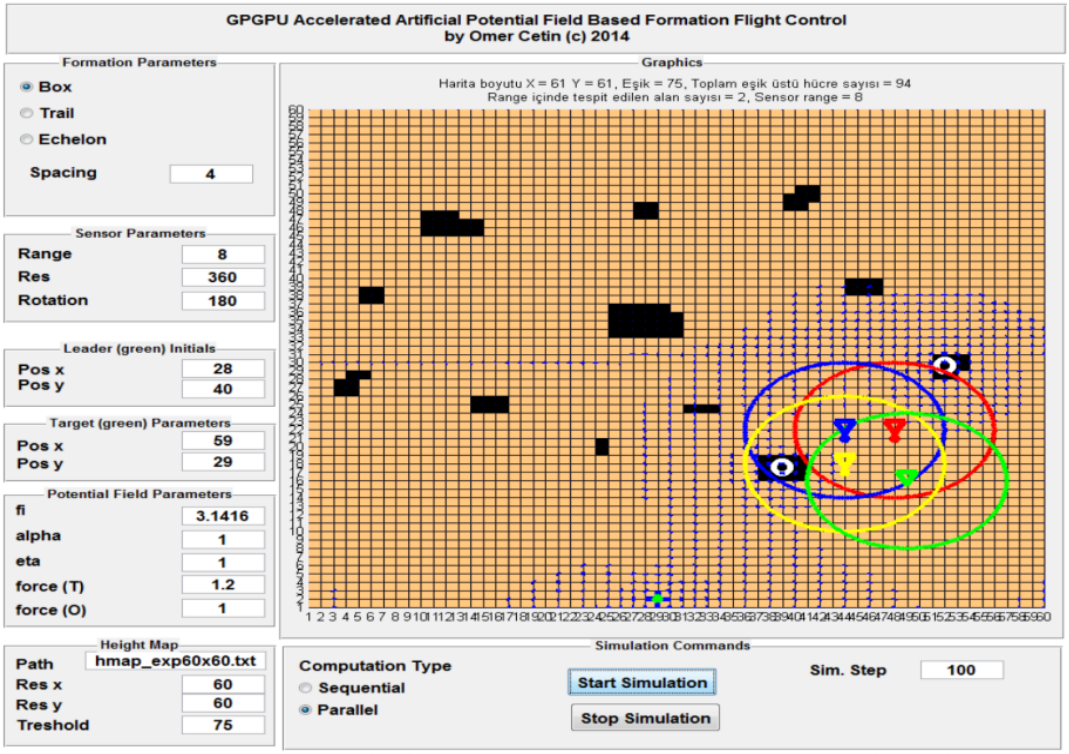
bağlantı üzerinden AR Drone platformunun geçek zamanlı kontrolü ve sensor verilerinin aktarılması sağlanmıştır. Bu yapı Parrot AR Drone platformu ile kullanıcı arasında bir arayüz görevi görür.

9.2.2 Döner kanat İHA modeli ile formasyon halinde çarpışmayı önleme ve engellerden sakınma simülasyonu

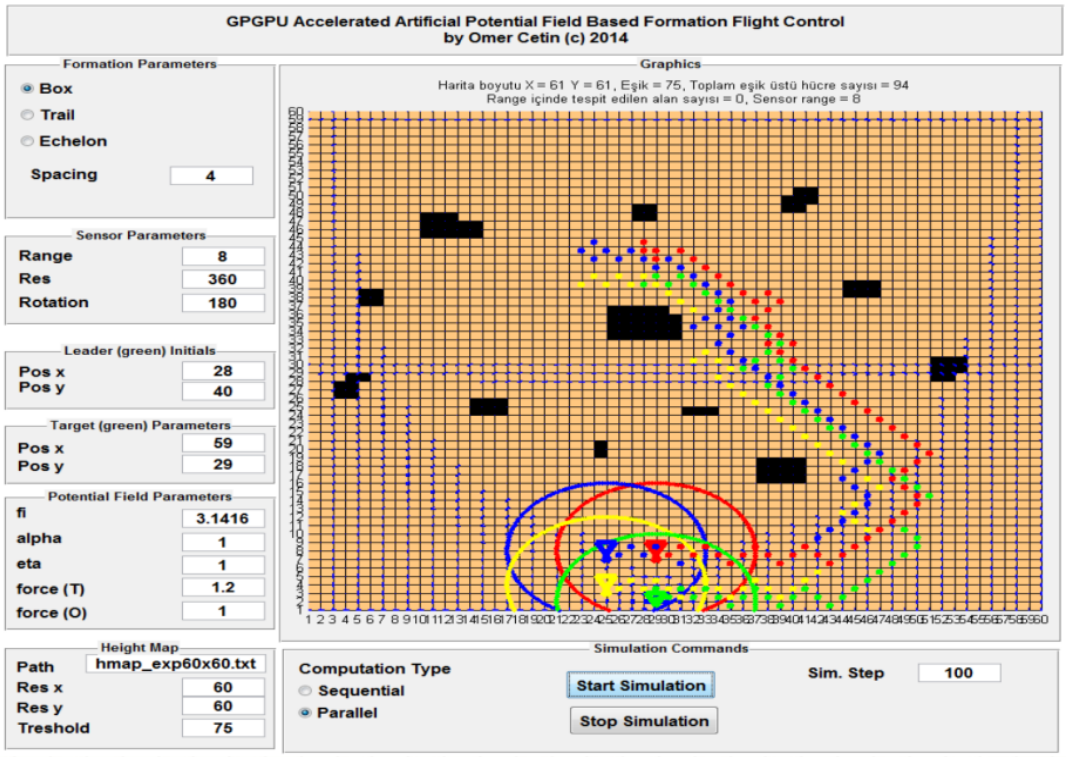
Çalışmanın bu bölümünde Şekil 9.12 ile gösterilen MATLAB Simulink ortamında geliştirilen AR Drone modelinden faydalanılarak, çalışma kapsamında ortaya konulan yaklaşımların başarısı sınanımmış ve sonuçları paylaşılmıştır.



(a) t_0 ve t_{10} süreleri arasındaki yol gösterimi.



(b) t_{10} ve t_{20} süreleri arasındaki yol gösterimi.

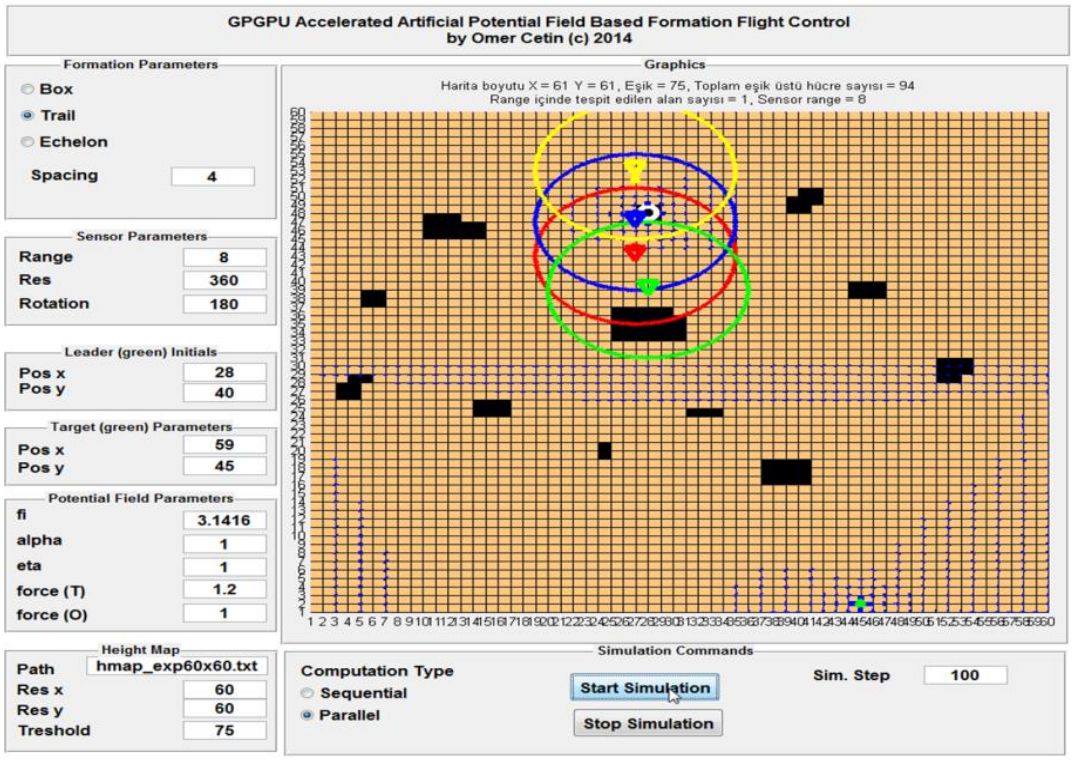


(c) t_{20} ve t_{30} süreleri arasındaki yol gösterimi.

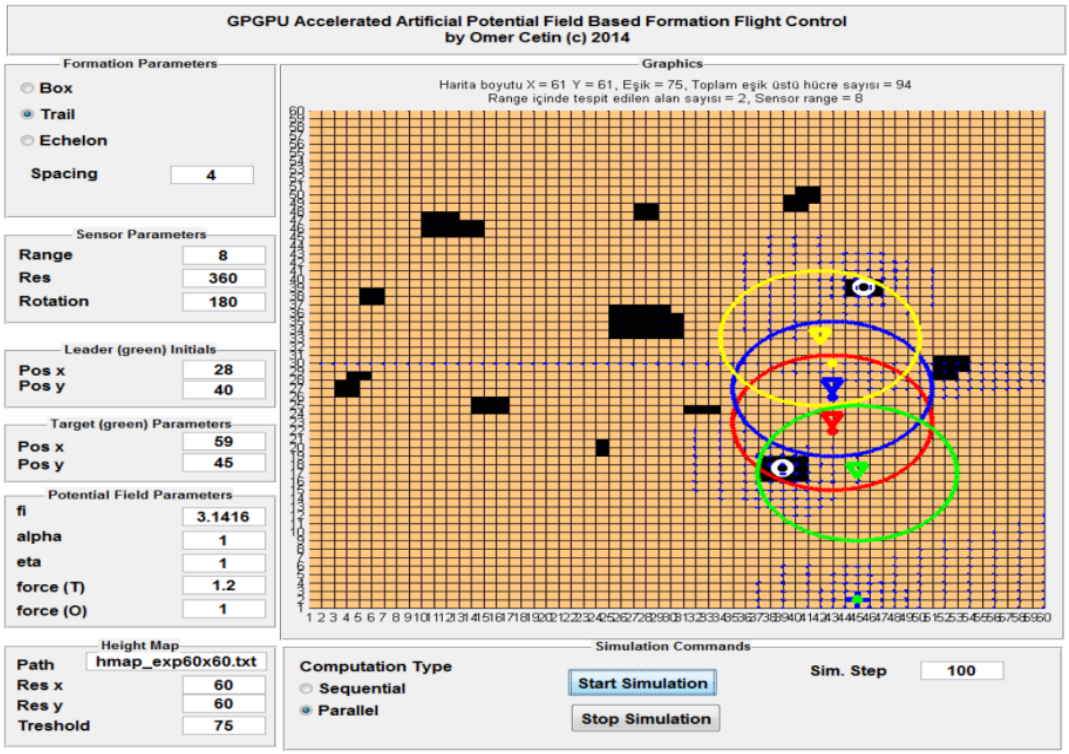
Şekil 9.14: Simülasyon ortamında box formasyonun uygulamasının gösterimi.

Şekil 9.14 ile çalışma kapsamında geliştirilen ve ortaya konulan formasyon yaklaşımının paremetrelerinin etkisinin görülmesi ve yaklaşımın sınanaması

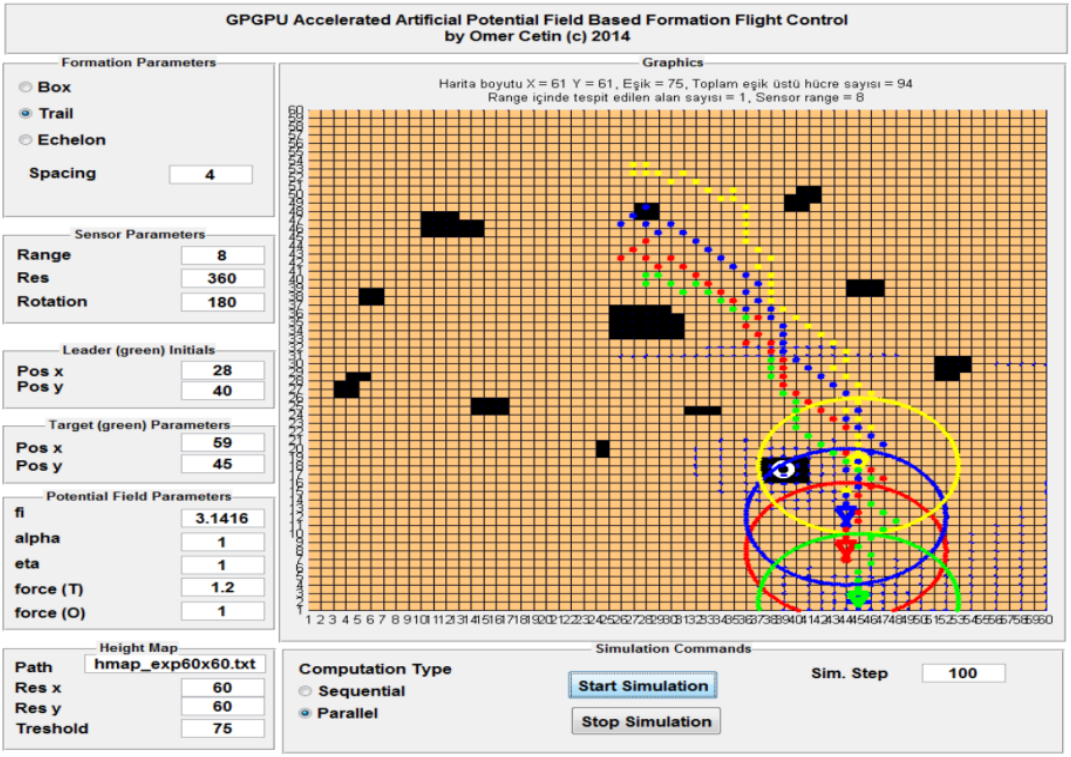
amacıyla geliştirilen simülasyon arayüzü gösterilmiştir. Simülasyon artamında ilk ortaya konulacak olan formasyon düzeni box formasyonudur. Şekil 9.14 (a) ile başlangıç konumları gösterilmiştir. Engellerin yer aldığı alan içerisinde platformlar üzerinde yer alan algılayıcıların tespit ettiği engellerin modellemesi yapılarak hareket yönlendirilmiştir. Şekil 9.14 (b) ile bir müddet sonra platformların yeni konumları gösterilmiştir. Şekil 9.14 (c) ile de platformların hareket süresinin sonunda simülasyon boyunca izledikleri rota ortaya konulmuştur.



(a) t_0 ve t_{10} süreleri arasındaki yol gösterimi.



(b) t_{10} ve t_{20} süreleri arasındaki yol gösterimi.



(c) t_{20} ve t_{30} süreleri arasındaki yol gösterimi.

Şekil 9.15: Simülasyon ortamında trail formasyonun uygulamasının gösterimi.

Şekil 9.15 ile aynı simülasyon çalışması bu kez bir diğer formasyon düzeni olarak, Trail için geçeklermiş ve sonuçları ortaya konulmuştur. Potansiyel alan tabanlı

otonom formasyon yaklaşımı doğası gereği aynı zamanda otonom olarak formasyon değişikliklerinin de yapılmasını ve emniyet ile gerçekleştirilmesini sağlamaktadır.

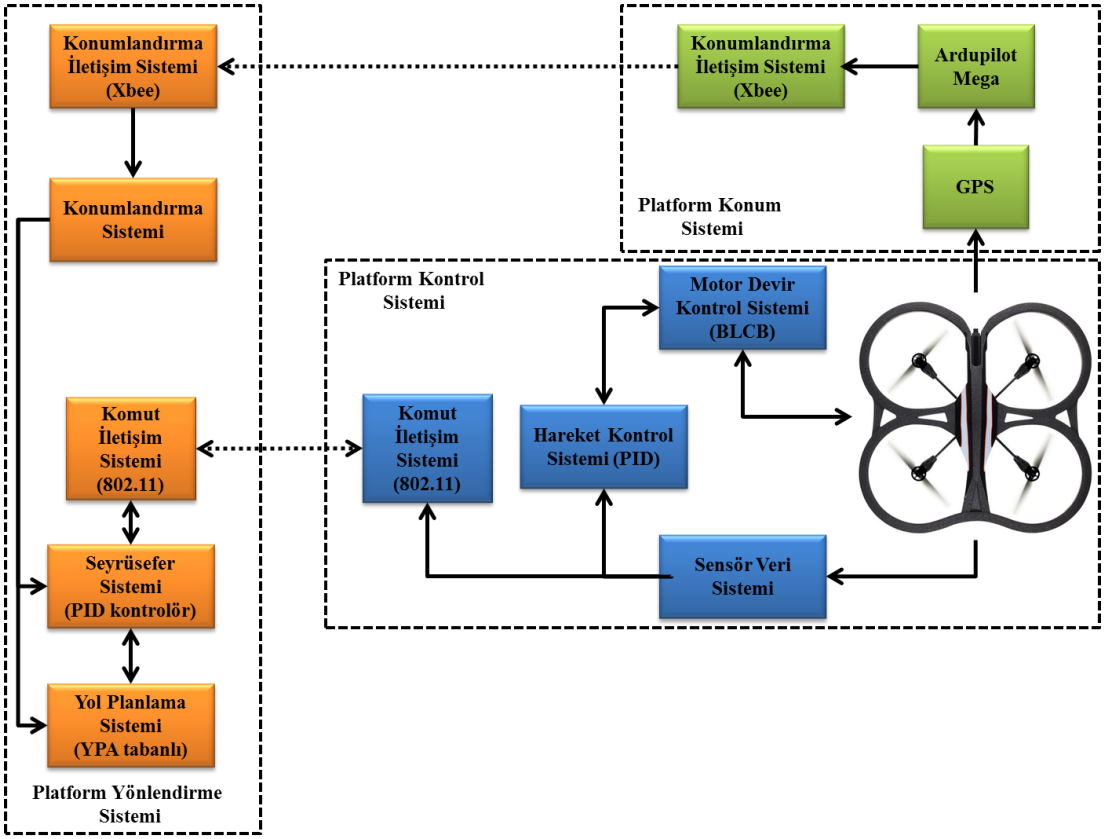
Yukarıdaki döner kanat İHA modelinden faydalanılarak gerçeklenen simülasyon çalışmaları göstemiştir ki, çalışma kapsamında ortaya konulan yaklaşım quadrotor platformlardan oluşan bir İHA formasyonunu başarı ile gerçekleştirmiş ve de engeller içeren bir alan içinde formasyonu muhafaza ederek seyrüsefer amaçlı yol planlaması üretilebilmesine imkan sağlamıştır.

9.3 Uygulamaya Yönelik Platform Sistemlerinin Geliştirilmesi

AR Drone klasik quadrotor dizaynına sahip bir platformdur [96]. Üzerinde yer alan 4 adet bağımsız sabit motor (brushless motor) platforma güç vermektedir. Bu motorlar Bölüm 9.2 kapsamında açıklandığı şekilde değişken oranlı güç kaynakları olarak kullanılmaktadırlar. Her motorun devir kontrolü fırçasız motor kontrol kartı (Brushless Motor Control Board - BLCB) adı verilen ayrı bir kontrol ünitesi ile sağlanmaktadır. Her bir BLCB kendi üzerinde ATMEGA8L 8bit mikro denetleyiciye ve acil durum güç kesici ünitesine sahiptir. Bu motorlar ve BLCB üniteleri karbon bir çapraz şase ile birbirlerine bağlanmıştır. Bu karbon çapraz gövde bağlantısının merkez noktasında motorlara elektrik gücü sağlayan Lityum Polimer Pil (Lithium-Polymer Battery – LiPo) yer almaktadır ve bu pil 3 hücreli 11,1 V ve 1000 mAh güç desteği sağlamaktadır. Ancak uçuş süresinin arttırılması için 2200 mAh, üç hücreli 25C LiPo pil ile beslenmiş ve uçuş süresi yaklaşık 20 dakikaya uzatılmıştır.

AR Drone quadrotor platformu üzerinde motor kontrol üniteleri (BLCB) haricinde iki adet kontrol kartı daha yer almaktadır. Bu kartlar üzerinde Parrot P6 olarak adlandırılan (32bits ARM9-core, 468 MHz) bir işlemci, Wi-Fi iletişim çipi, platform üzerinde yer alan dikey ve yatay kameraları kontrol eden üniteler ve algılayıcılar (sensor) yer almaktadır. Gerçek zamanlı bir Linux işletim sistemi (busybox) ile platformun tüm kontrolü sağlanmaktadır. Platform üzerinde 93 derecelik geniş bir lense sahip HD görüntü sağlayan ve ileri bakan bir kamera ile, platformun altına doğru yerleştirilmiş VGA çözünürlüğünde iki adet kamera yer almaktadır. Yatay kamera daha çok platformun seyrüseferinde algılayıcı olarak kullanılmak üzere yerleştirilmiştir. Ana kart üzerinde yer alan 40 MHz hızında çalışan 16 bit PIC mikro denetleyicisi diğer algılayıcılar ile haberleşerek platformun temel hareket kontrolünü sağlamaktadır. Bu algılayıcıları 3 eksenli ivmeölçer, iki eksenli cayroskop

(gyroscope), bir bağımsız dikey eksenli ilave gyroscope ile 2 adet ultrasonic irtifa algılayıcı ve baro metrik irtifa algılayıcısından oluşmaktadır. Tüm bu algılayıcıların birleşimi platformun temel hareketlerini ölçümleyebilen dahili bir ölçüm sistemi Ataletsel Ölçme Birimi (Inertial Measurement Unit – IMU) şeklinde davranmaktadır. Platform üzerinde yer alan gömülü yazılım şeklindeki işletim sistemi platformun algılayıcıları ve kameraları tarafından alınan verileri üzerinde yer alan Wi-Fi yongası ile 802.11n protokolüne uygun bir bağlantı üzerinden gerçek zamanlı olarak (200 Hz hızında) aktarabilmektedir [96].



Şekil 9.16: AR Drone quadrotor platformunun kontrol ve yönlendirme yaklaşımı.

Quadrotor sisteminin temek kontrol yaklaşımı Şekil 9.16 ile gösterilmiştir. Bu kapsamda kontrol sisteminin temel unsurları aşağıdaki başlıklar altında ele alınarak işlevleri ile birlikte açıklanmıştır.

Çalışmanın sınanması amacıyla belirlenen platform AR Drone quadrotor platformu olarak belirlenmiş hazır bir kittir. Bu platform üzerinde GPS konumunu algılamasına imkân verecek nitelikte bir sistem yoktur. Bu problemin çözümü için APM kiti ile desteklenmiş GPS anteni ve de bu bilgiyi paylaşmaya imkân sağlayacak Xbee iletişim modülleri temin edilmiştir. GPS konum hassasiyeti iki ile sekiz metre

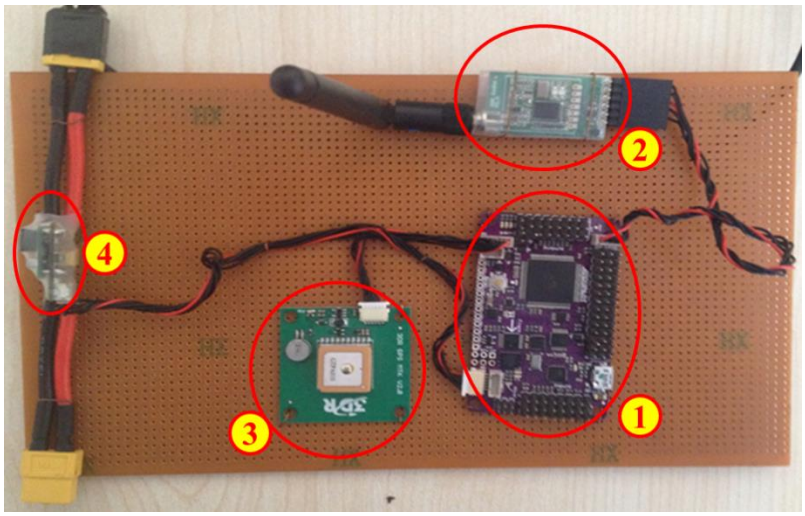
arasında değişmektedir. Platformun maksimum sürati yaklaşık 10 m/sn, tavan irtifası yaklaşık 100 metredir.

Platform üzerinde yer alan ve gömülü yazılım şeklinde çalışan uygulamalardır. AR Drone platformu üzerinde yer alan 4 adet fırçasız motorun devir kontrolünü sağlayana Motor Kontrol Birimi ve bu birimin komutlarını üretmek için faydalanılan PID tabanlı bir kontrol sistemi yer almaktadır. Platform üzerinde yer alan ve Algılayıcı Veri Sistemi tarafından denetlenen algılayıcılardan gelen platformun davranışına ilişkin bilgileri ve kullanıcı tarafından İletişim Sistemi üzerinden gönderilen komutları girdi olarak alan Kontrol Sistemi, motorların kontrolüne bu bilgiler ile karar vermektedir.

9.3.1 Konumlandırma sistemi

Yer kontrol istasyonu bünyesinde yer alan "Konumlandırma Sistemi" çalışmanın bu döneminde üzerinde durulan konulardan birisidir. AR Drone quadrotor platformu üzerinde GPS alıcısı bulunmamaktadır. Bu nedenle platformun GPS koordinat sistemi üzerindeki konumunu tespit edebilmek amacıyla ArduPilot Mega 2.0 (APM) kiti üzerine yerleştirilmiş 3DR GPS alıcısından faydalanılmıştır [98. APM kiti üzerinde yer alan GPS alıcısı elde ettiği konum bilgilerini APM kartı üzerinden haberleşme amacıyla kullanılan Xbee kablosuz modemine iletmektedir.

APM kiti üzerinde Atmega 2560 işlemcisi ve Genlik Modülasyonu (Pulse With Modulation – PWM) adım motor sürücü devreleri ve telemetri verilerine okuyabilecek nitelikte algılayıcıları bulunan ticari bir üründür [99].



Şekil 9.17: APM 2.0 ile GPS konum bilgilerinin tespit edilerek aktarılması.

GPS tabanlı harici ortam konumlandırma sisteminin AR Drone quadrotor platformuna entegre edilebilmesi amacıyla gerçekleştirilen çalışmalar kapsamında Şekil 9.17 ile gösterilen deney düzeneği kurulmuştur. Şekil üzerinde gösterilen (1) numaralı bileşen APM kartı olup, üzerindeki işlemci ile (3) numara ile gösterilen GPS alıcısı üzerinden edinilen konum bilgilerini işleyerek (2) numara ile gösterilen Xbee kablosuz iletişim modülü vasıtasıyla yer istasyonuna aktarmak için kullanılmaktadır. Sistemin güç beslemesi (4) numara ile gösterilen kontrol donanımı üzerinden AR Drone platformunun güç kaynağı üzerinden sağlanması amaçlanmakta ve böylece ilave yük getirilmemesi planlanmaktadır.

Şekil 9.18: Xbee alıcısı ile donatılmış kontrol istasyonu gösterimi.

APM kartı üzerinden alınan telemetri ve GPS konum verileri (2) numara ile gösterilen Xbee kablosuz iletişim modülü vasıtasıyla yer kontrol istasyonu olarak kullanılan bilgisayarın USB noktasına bağlı olan ve (5) numara ile **Şekil 9.18**'de gösterilen Xbee alıcısına gönderilir.

Şekil 9.19: Xbee alıcısı ile donatılmış kontrol istasyonu gösterimi.

Şekil 9.17 ve Şekil 9.18 ile gösterilen düzeneğin kurulması ve gerekli konfigürasyon ayarlarının yapılması neticesinde ekran çıktısı Şekil 9.19 ile gösterilen yer kontrol

istasyonu üzerinden platforma ait üç boyutlu koordinat bilgileri harici ortamda alınmıştır.

9.3.2 Seyrüsefer sistemi

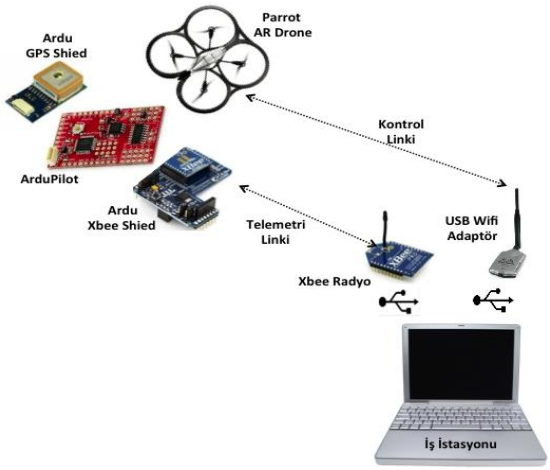
Seyrüsefer sistemi, yol planlama sisteminden aldığı yön (uçuş başı), sürat gibi komutlara istinaden platforma gönderilmek üzere belirlenecek olan yatış, kayış, hucüm açısı, dönüş ve sürat gibi komutları hesaplar.

9.3.3 Yol planlama sistemi

Tez kapsamında gerçekleştirilen ve geliştirilmesine devam eden örnek görevlerin planlanması ve icrası esnasında kullanılan yol planlama sistemi platformun kontrol sistemi içerisinde bu alt sistem içinde yer almaktadır. Geliştirilen yapay potansiyel alan tabanlı yol planlamasına yönelik algoritmalar bu blok dahilinde GPU donanımları üzerinde koşan parallel uygulamalar şeklinde gerçekleştirilmiştir.

9.3.4 İletişim sistemi

İletişim sistemi platformun davranışına karar veren yer kontrol sistemlerinin ürettiği komutların platforma aktarılmasını ve platformun davranışları hakkındaki bilgilerin yer istasyonuna geri gönderilmesi amacıyla faydalanılan sistemdir. Çalışma kapsamında gömülü yazılım olarak AR Drone quadrotor sistemleri üzerinde yer alan ve AT Commands adı verilen protokol vasıtasıyla Kullanıcı Veri Birimi Protokolü (User Datagram Protocol – UDP) tipindeki 802.11n bağlantı üzerinden bu işlemler gerçekleştirilecektir.



Şekil 9.20: İletişim alt yapısı gösterimi.

Şekil 9.20 ile gösterildiği biçimde yer kontrol istasyonu olarak kullanılan iş istasyonu ile ArDrone arasında iki iletişim protokolü teşkil edilmiştir. Bu iletişim protokolleri İletişim Sistemi ile kontrol edilmektedir. Quadrotor'un kontrol komutlarını aktaran "Kontrol Linki" AT Commands adı verilen protokolün iletişimini sağlarken,

“Telemetri Linki” adı verilen iletişim altyapısı da başta GPS bilgileri olmak üzere platform üzerinde yer alan APM sistemi tarafından üretilen bilgileri kontrol istasyonuna aktarmaktadır.

Ek B’de yer alan Şekil B.1 ile APM ve GPS alıcılarının AR Drone platformu üzerine entegre edilmiş şekilleri görülmektedir. Ayrıca standart AR Drone camera sisteminde ve gövde yapısında bir takım modifikasyonlar yapılarak Quadrotor’un APM ve Xbee modüllerini taşıyabilmesi sağlanmıştır. Ek B’de yer alan Şekil B.2 ile platformun üzerinden çıkarılan ağırlıklar ve ağırlık merkezinin değişmemesi için gerçekleştirilen modifikasyonlar ile veri alışverişinde kullanılan anten yapıları görülmektedir. Tez kapsamında gerçekleştirilen teorik çalışmaların Parrot AR Drone [96] platformu üzerinde uygulamalı olarak sınanmasına karar verilmiştir. Bu kapsamda bu bölüm dahilinde temel quad-rotor olarak adlandırılan ve mini bir İHA platformunun temel niteliklerini taşıyan Ek B’de yer alan Şekil B.3’de gösterilen AR Drone platformunun otonom kontrolüne yönelik gerçekleştirilen çalışmalara değinilecektir.

9.4 Uçuş Uygulamaları

Bu bölümde tez kapsamında uygulamaya yönelik çalışmalar başlığı altında gerçekleştirilen çalışmalara değinilecektir. Uçuş denemeleri ve hangi koşullar altında gerçekleştiği belirtilerek sonuçları paylaşılacaktır.

Şekil 9.21: Deney uçuşlarının gerçekleştirildiği alan gösterimi.

Uçuş denemeleri Şekil 9.21 ile gösterilen Erzurum Atatürk Üniversitesi kampüsü sınırları içinde yer alan model uçak pist bölgesinde gerçekleştirilecektir. Bu bölge yaklaşık 300 m. x 300 m. büyüklüğünde bir alandır. Ek C'de yer alan Şekil C.1 ile icra edilen deney uçuşlarından farklı anlara ait görüntüler verilmektedir.

9.4.1 Tek platform uçuş deneyleri

Şekil 9.22: Tek platform deney uçuşu için belirlenen engelsiz alandaki yol noktaları.

Uçuş bölgesi içinde bir görev haritası ve yol noktaları belirlenmiştir. Görev haritası Şekil 9.22 ile gösterildiği biçimdedir. *A1-A2-A3-A4* noktaları görevin icra edileceği 100 m. x 100 m. uzunluklarında kare uçuş görev bölgesinin sınırlarını kapsamaktadır. "Başlangıç" olarak gösterilen nokta görev uçuşuna başlamak üzere ilk kalkış pozisyonunu temsil etmektedir. "WP" ile isimlendirilen noktalar ise görev uçuşu süresince uğranması gereken noktaları temsil etmektedir. Bu görev belirli bir bölgenin belirli bir sabit yükseklikten görüntülenmesi görevi olarak tanımlanabilir.

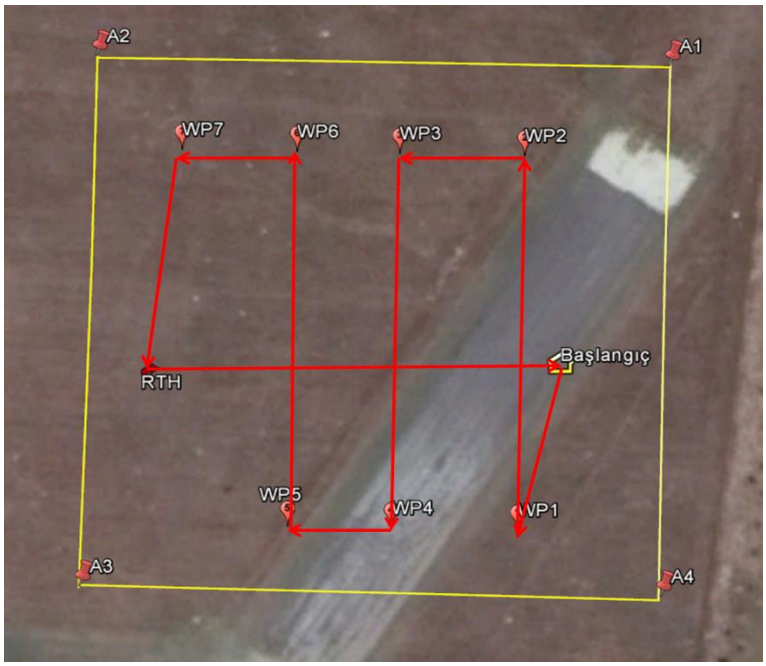
Çizelge 9.2 ile bölge üzerindeki tüm noktaların GPS konumlandırma sistemine göre enlem ve boylam bilgilerini göstermektedir. Hali hazırda bölge üzerinde bir coğrafi veya suni uçuşu engelleyecek nitelikte bir engel bulunmamaktadır. Dolayısıyla platformun belirlenen yol noktalarına düz rotalar oluşturarak otonom şekilde sırasıyla başlangıç noktasından başlayarak uğraması ve Eve Dönüş Noktası (Return

To Home Point – RTH) noktasına ulaşmasının ardından başlangıç noktasına geriye dönmesi beklenen harekettir.

Çizelge 9.2: Görev uçuşu tanımlaması noktaları ve GPS koordinatları.

Nokta Adı	Enlem	Boylam
A1	39.905455	41.238333
A2	39.905455	41.237163
A3	39.904583	41.237163
A4	39.904583	41.238333
Başlangıç	39.904939	41.238124
WP1	39.904685	41.238042
WP2	39.905301	41.238042
WP3	39.905301	41.237787
WP4	39.904685	41.237787
WP5	39.904685	41.237578
WP6	39.905301	41.237578
WP7	39.905301	41.237343
RTH	39.904939	41.237312

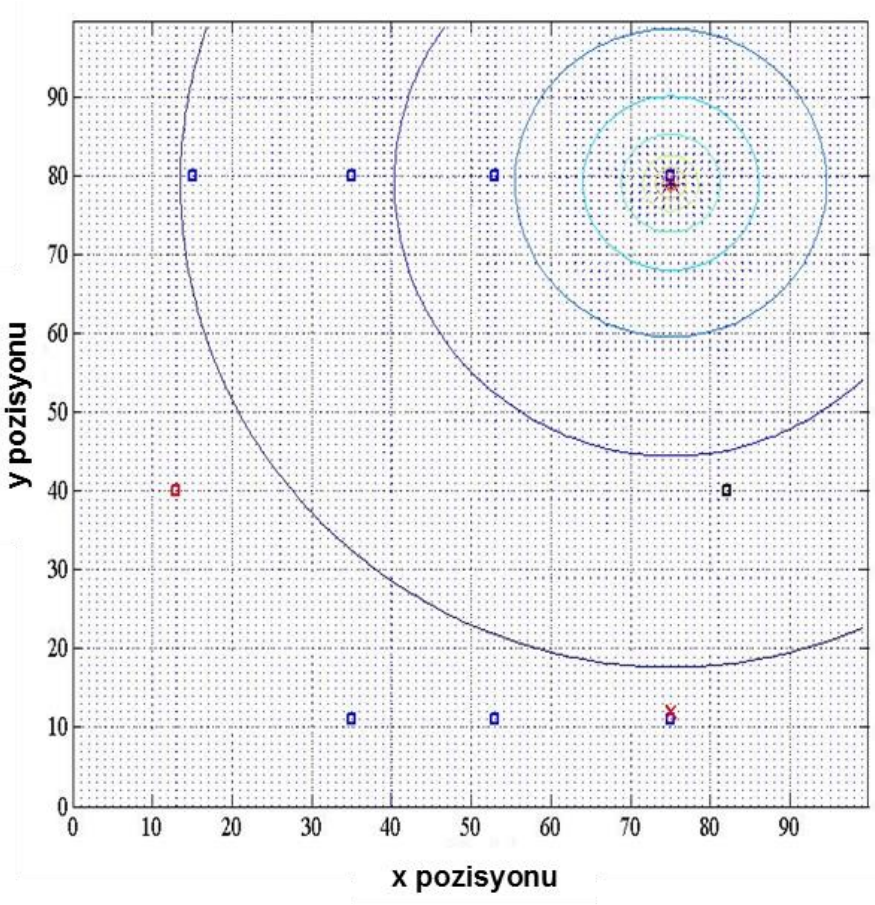
Çizelge 9.2 ile ifade edilen GPS koordinatlarında yer alan yol noktalarına (WP) sırasıyla ulaşması beklenen platform en son RTH noktasına ulaşarak başlangıç pozisyonuna geriye dönecektir.



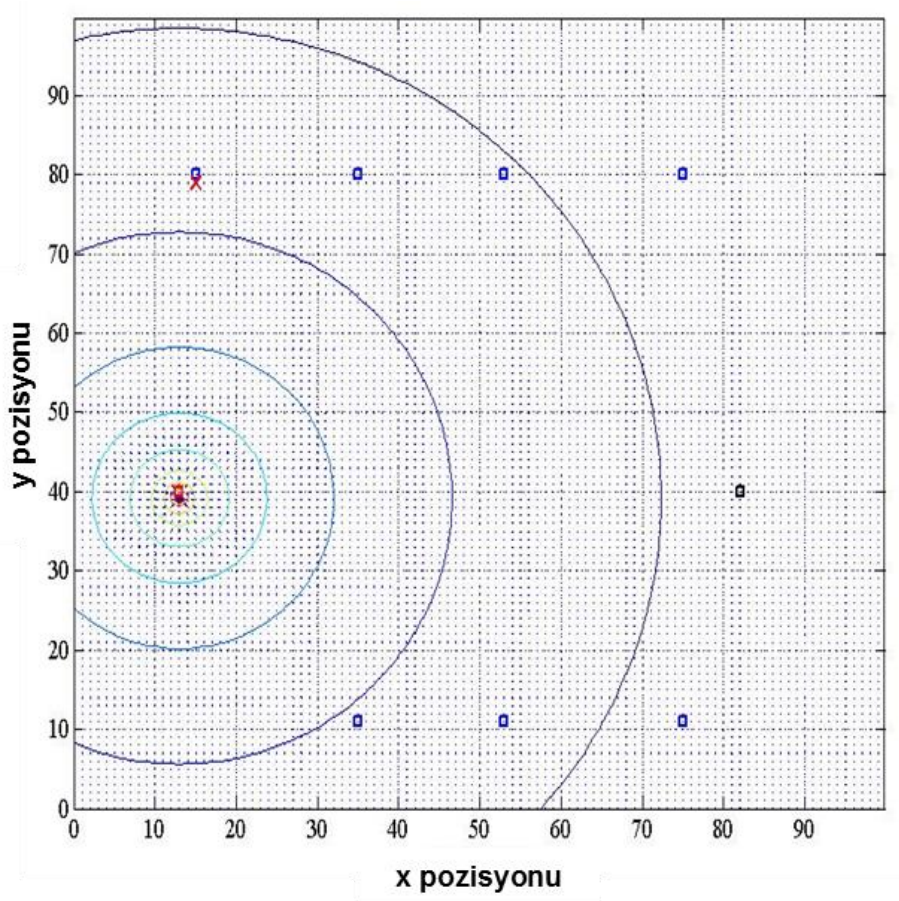
Şekil 9.23: Tek platform deney uçuşunun icra edilmesinde platformun izlemesi beklenen rota gösterimi.

Bu esnada platformdan beklenen uçuş rotası Şekil 9.23 ile gösterildiği gibi olacaktır. Ancak bu şekilde bir uçuş gerçekleşmesini beklemek maalesef gerçek koşullar altında mümkün değildir. Bunun nedeni rüzgâr, yer etkisi gibi platform üzerindeki

harici ortamdan kaynaklanan ve değişken etkilerin aerodinamik etkileri ve WP noktalarında platformun dönüş hareketinden kaynaklanan kontrol algoritmasındaki hata limitleri dahilindeki sapmalardır. Bu nedenle platform ilgili uçuş görevini uygularken teoride belirlenen rotayı birebir uygulayamayacak ancak mümkün olduğu kadar bu rotaya bağlı hareket etmeye çalışacaktır. YPA tabanlı kontrol ve yol planlaması yaklaşımı dahilinde rota incelendiğinde platformun her t anında bir ulaşması gereken ve çeken özellikte potansiyel alan etkisi altında olması sağlanacaktır. Farklı bir açıklama ile örneğin; platform başlangıç noktasında bulunduğu esnada WP1 noktası kaynaklı bir çeken alan etkisinde bu noktaya yönelmesi ve ulaşması sağlanacaktır. Platform WP1 noktasına ulaştığında bu noktanın çeken alan kaynağı olması ortadan kaldırılarak bir sonraki ulaşması istenen nokta olan WP2 noktası çeken potansiyel alan durumuna geçecektir. Alan içinde yer alan engel olmadığı için sadece tek bir kaynağa sahip ulaştıkça değişkenlik gösteren hedef noktalar olacaktır.



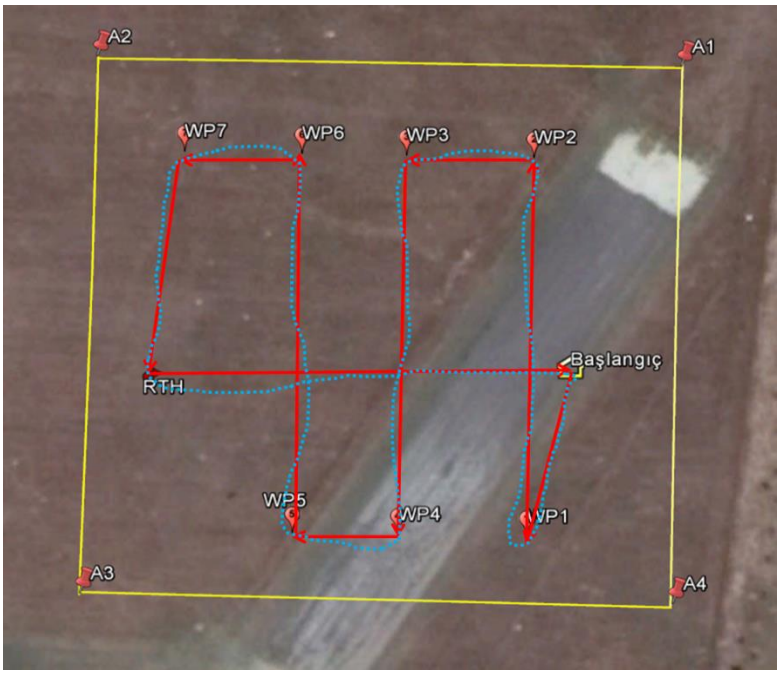
(a) t_n anında uygulamaya ait vektör alan gösterimi.



(b) t_{n+s} anında uygulamaya ait vektör alan gösterimi.

Şekil 9.24: Tek platfrom deney uçuşunun gerçekleştirilmesi esnasında hesaplanan vektör alan örnekleri.

YPA tabanlı kontrol ve yol planlaması amaçlı potansiyel alanın çözünürlük değeri gerçekte yaklaşık 100 m. x 100 m. olarak belirlenen alan için 10 cm olarak belirlenmiş ve 1000 x 1000 ölçeğinde bir potansiyel alan modeli oluşturulmuştur. Platformun WP1 noktasına ulaştıktan sonra WP2 noktasına ulaşması için hesaplanan vektör alan gösterimi Şekil 9.24 (a) ile gösterilmiştir. Platform WP7 noktasına ulaştıktan sonra RTH noktasına yönlendirilmesi için hesaplanan vektörel alan Şekil 9.24 (b) ile gösterilmiştir.



Şekil 9.25: Engelsiz alanda icra edilen tek platfrom deney uçuşu neticesinde platformun izlediği rota.

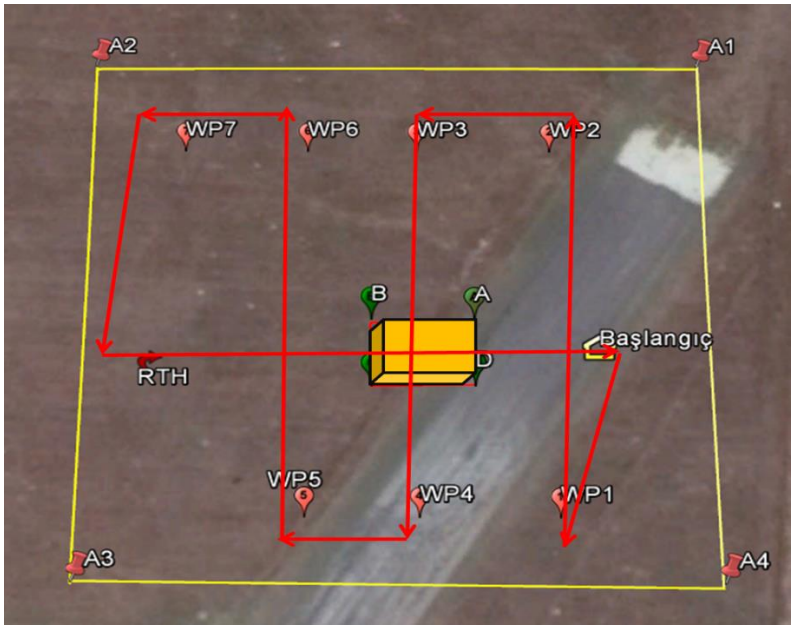
Şekil 9.25 ile platformun uçuş sonunda izlediği gerçek rota ile teoride belirlenen rota bir arada gösterilmiştir. Yukarıda sayılan nedenlerden dolayı rotadaki sapmalar tahmin edildiği gibi oluşmuş ancak platform başarı ile YPA kontrolü ve yol planlaması ile istenen görevi icra etmiştir.

Şekil 9.26: Tek platform deney uçuşunun icra edileceği engel içeren alan gösterimi.

Şekil 9.26 ile ikinci uçuş ortamı tanıtılmaktadır. Şekil 9.22 ile gösterilen ve Çizelge 9.2 ile ifade edilen koordinat noktalarına ilave olarak Şekil 9.26 ile gösterilen ve (A-B-C-D) noktaları ile tanımlanan engel uçuş alanına eklenmiştir. Engel köşe noktaları Çizelge 9.3 ile verilmiş ve küp şeklinde bir sütre olarak tanımlanabilir.

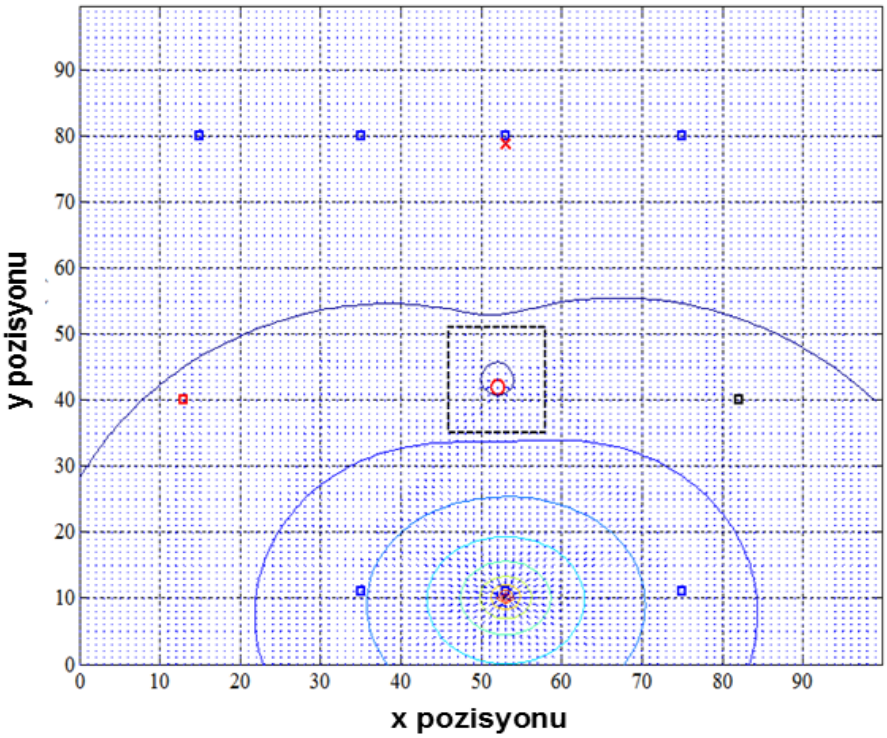
Çizelge 9.3: Engel tanımlama noktaları ve koordinatları.

Nokta Adı	Enlem	Boylam
A	39,905010	41,237893
B	39,905010	41,237700
C	39,904898	41,237700
D	39,904899	41,237893

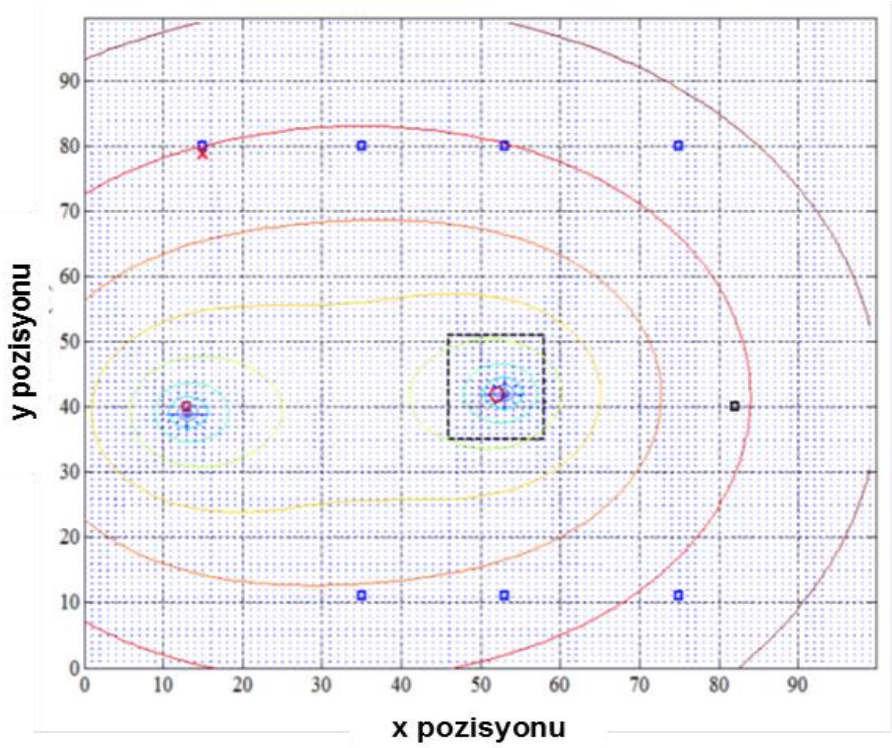


Şekil 9.27: Görevin icrası için beklenen rotanın engel ile kesişiminin gösterimi.

Şekil 9.27 ile içerisinde engel bulunan alan içerisinde önceki teorik uçuş rotası gösterilmiştir. Şekilden anlaşıldığı üzere platformun uçuş rotası ile engel kesişmektedir. Şayet engel YPA tanımlamasına uygun biçimde iten potansiyel alan ile modellenecek olursa, uçuş alanı içerisinde yer alan platformun üzerinde ilk uçuştan farklı olarak iki etki eden alan bulunacaktır; engel ve varılmak istenen yol noktası. Bileşke vektörlerin hesaplanmasıyla oluşturulacak olan vektör alan platformun engellerden sakınarak hedeflenen noktaya ulaşmasına imkân sağlayacak kontrol ve yol planlaması yaklaşımını bir arada sunacaktır.



(a) t_n anında uygulamaya ait vektör alan gösterimi.

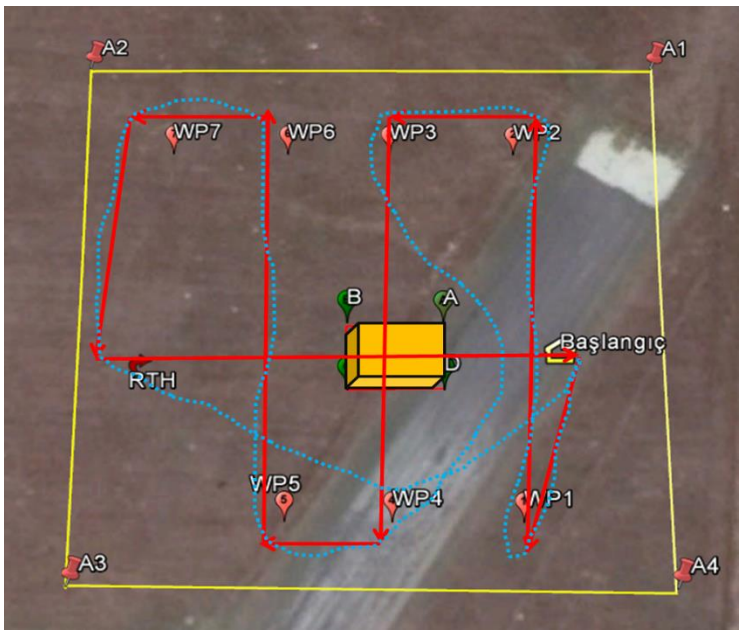


(b) t_{n+s} anında uygulamaya ait vektör alan gösterimi.

Şekil 9.28: Engel içeren alan içerisinde gerçekleştirilen görev uçuşu esnasında üretilen vektör alan örnekleri.

Bu tanıma istinaden WP3 noktasına ulaşmış ve WP4 noktasının çeken alan etkisi altındaki platform için oluşan vektör alan Şekil 9.28 (a) ile RTH noktasına ulaşmış

ve Başlangıç noktasına ulaşmaya çalışan platformu kontrol eden vektör alan ise Şekil 9.28 (b) ile gösterilmektedir.



Şekil 9.29: Engel içeren alan içerisinde tek platfrom ile icra edilen uçuş sonucunda platformun izlediği rotanın gösterimi.

İçerisinde engel yer alan uçuş alanı içinde otonom olarak tanımlanan yol noktalarını sırasıyla dolaşan platformun izlediği rota ve teorik olarak planlanan rota Şekil 9.29 ile gösterilmiştir.

Şekil 9.29'den de görüldüğü üzere platform otonom olarak engelden sakınmış ve güvenli bir rota izleyerek tüm kontrol noktalarına ve ardından başlangıç noktasına başarıyla ulaşmıştır. Sanal bir engel tanımlaması yapılarak bu engelden sakınmaya çalışılmasının nedeni quadrotor platformlarının güvenliği sağlamaktır. Bu yaklaşım ile gerçekten alan içinde bir engel olması ve bundan sakınılması arasında pratikte ve teoride hiçbir fark yoktur.

9.4.2 Çoklu platform uçuş deneyleri

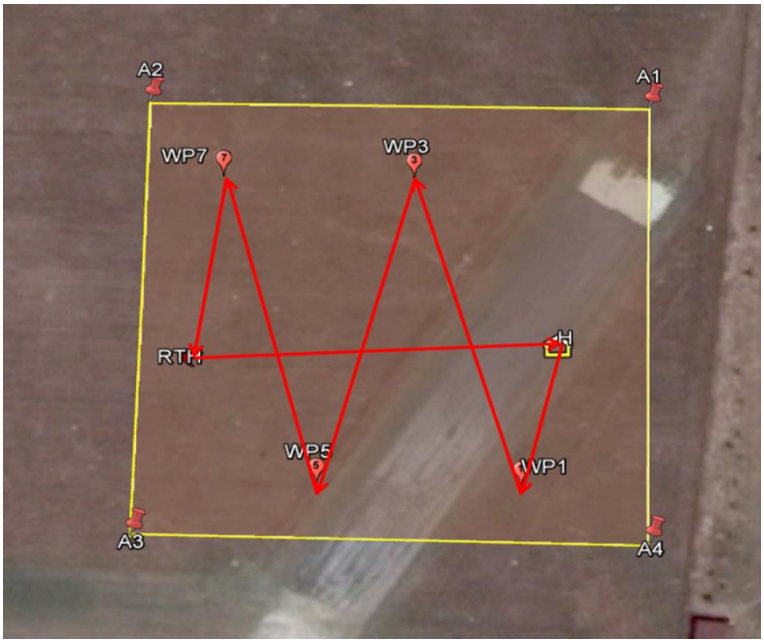
Şekil 9.30: Engel içermeyen alan içerisinde çoklu platform ile icra edilecek görev uçuşu tanımlasının gösterimi.

Şekil 9.30 ile gösterilen alan içinde Çizelge 9.2 ile gösterilen yol noktalarından dört tanesi (WP1-3-5-7) bırakılmış, diğerleri görev kapsamından çıkarılmıştır. Amaçlanan bu noktalara formasyonu muhafaza ederek sırasıyla lider platform tarafından uğranması ve son olarak RTH noktasından başlangıç konumuna dönülmesidir.

Alan içinde üç platform V formasyonunda görevi icra edecektir. Bu üç platformun her birinde farklı bir sensor (Kızılötesi - IR, television - TV, ısıl - thermal vb.) olduğu kabul edilmiş ve görevin icra edildiği bölgenin bu üç farklı sensor tarafından eş zamanlı görüntülenmesi amaçlanmıştır. Sensorların üçünü aynı anda taktik sınıfı bir İHA platformu taşıyabilir ancak bunun yerine üç mini İHA platformunun bu görevi bir arada uçarak yerine getirmesi öngörülmüştür.

Ek C'de yer alan Şekil C.2 ile formasyon uçuşunu gerçekleştircek olan AR Drone platformları görülmektedir. Platfromlar kalkışlarını takiben potansiyel alan tabanlı yol planlamasının belirlediği rotayı takip ederek uçuşlarını gerçekleştiriceklerdir. Akabinde her bir platformun uçuş rotası elde edilecek ve bu rotalar analiz edilerek yaklaşımın başarısı ortaya konulacaktır.

Bu kapsamda lider platform kendisine tanımlanan teorik rotayı YPA tabanlı kontrol ve yol planlaması kapsamında diğer platformlardan bağımsız olarak icra edecektir. Ancak diğer platformlar YPA tabanlı vektör alan içinde lider platform için çarpışmayı önler nitelikte sınırlandırılmış birer iten potansiyel kaynağı şeklinde davranacaklardır. Diğer platformlar çalışma kapsamında açıklandığı üzere lidere göre önceden belirlenmiş olan konumlarını uçuş süresince muhafaza etmeye çalışacaklardır. Bu esnada çevredeki sanal engeller ve formasyon dahilindeki diğer platformlar kendileri için sakınılması gereken noktaları oluşturacaktır.



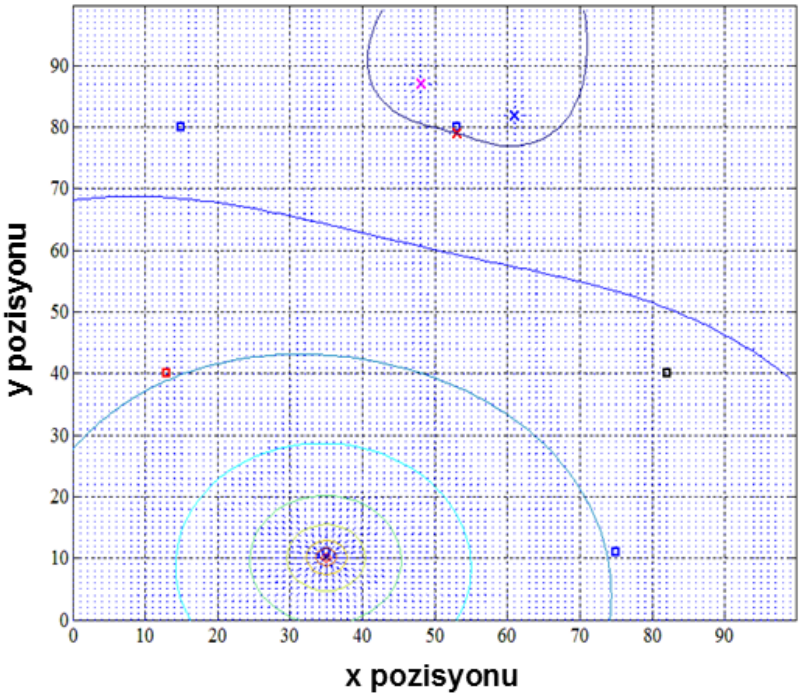
Şekil 9.31: Engel içermeyen alan içerisinde formasyon ile icra edilecek olan görevin beklenen rotası.

Bu durumda lider platformun teoride izlemesi gereken rota Şekil 9.31 ile gösterildiği şekilde gerçekleşmesi beklenmektedir. Ancak tıpkı tek platform tarafından icra edilen önceki uçuş uygulamalarında olduğu gibi platform teorideki planlanan rotadan farklı ancak mümkün olduğunca yakın bir yol izleyecektir.

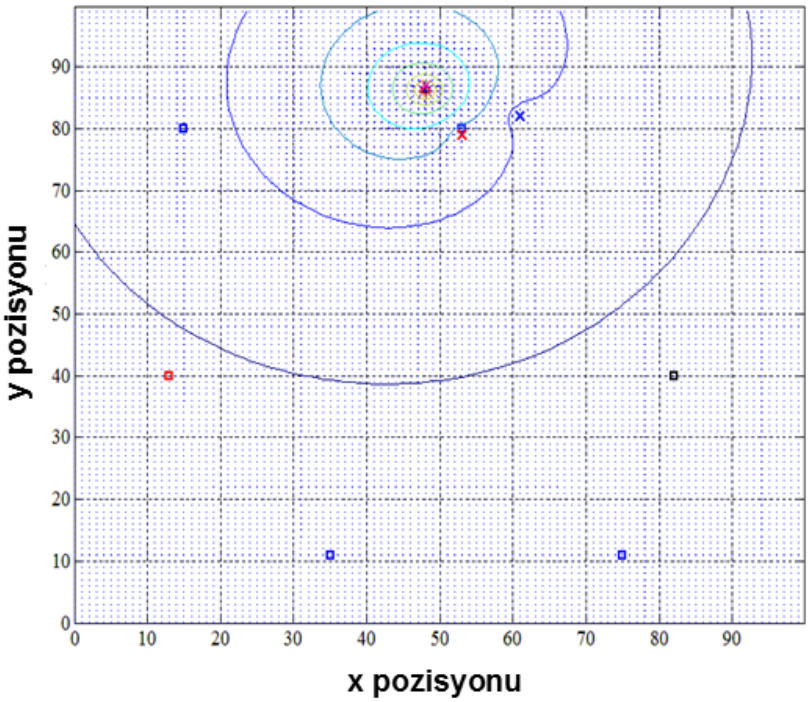
Kalkış komutunu takiben platformların havadaki görünüşleri Ek C'de yer alan Şekil C.3 ile gösterilmiştir. Platformun nasıl bir rota izleyeceğini kestirmek mümkün değildir. Çünkü çevre koşullarının ve koşulların aerodinamik etkilerini önceden hesaplayabilmek ya da bire bir modelleyebilmek mümkün değildir. Örneğin uçuş esnasında hamleli bir rüzgâr platformun rotasını doğrudan etkileyecektir. Platformun beklenen rotanın dışına çıkmasına neden olacaktır.

Formasyon halinde hareket eden platformların birbirleri ile çarpışmasını önlemek tez kapsamında ele alınan konulardan birisidir. Lider platform şayet teoride planlanan rotayı birebir uygulayacağı garanti edilebilseydi bu problemin çözümü oldukça kolay olacaktı. Ancak yukarıda açıklanan nedenlerden dolayı bilinmeyen bir davranış içinde bulunan platformun hareketlerine bağlı olarak birbirlerinin davranışı etkileyen bir kontrol yapısı tez kapsamında ortaya koyulmuştur.

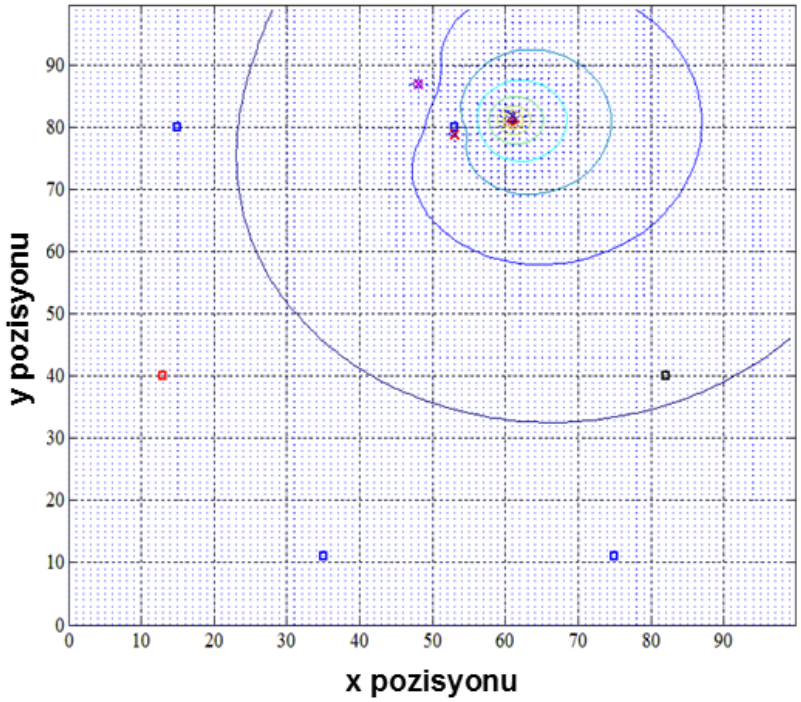
Platformların uçuşu esnasında formasyon dahilinde yer alan platformları görüntüledikleri anlardan birkaç kare Ek C'de yer alan Şekil C.4 ile gösterilmiştir. YPA tabanlı kontrol yaklaşımı dinamik olarak platformların davranışlarına göre güncellenmektedir. Gerçek zamanlı olarak gerçekleştirilen bu hesaplama sayesinde bu amaca ulaşılmıştır. Bu amaçla *t* anında her bir platform için kontrol amaçlı hesaplanan vektör alanlar aşağıda Şekil 9.32 ile örnek olarak gösterilmiştir.



(a) t_n anında uygulamaya ait vektör alan gösterimi.



(b) t_{n+s} anında uygulamaya ait vektör alan gösterimi.



(c) t_{n+r} anında uygulamaya ait vektör alan gösterimi.

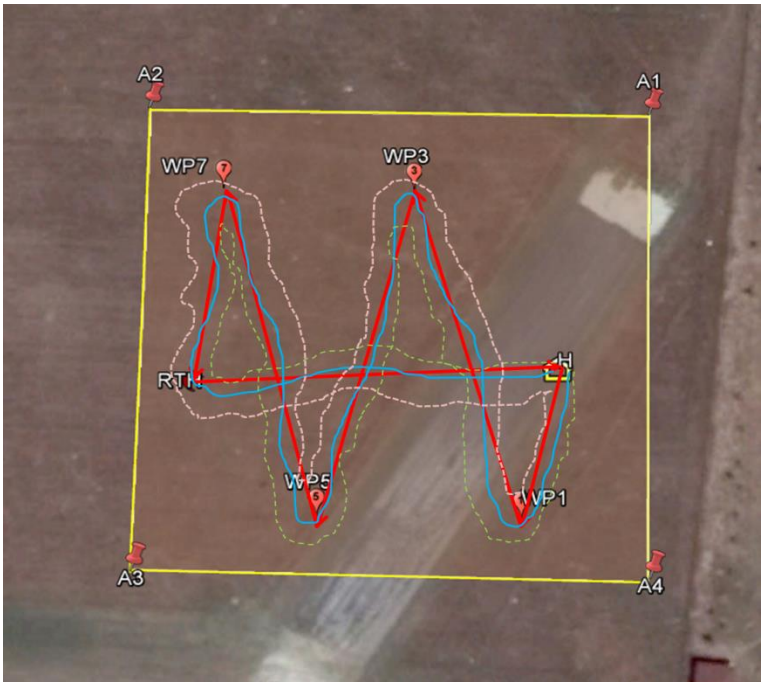
Şekil 9.32: Engel içermeyen alan içinde gerçekleştirilen formasyon uçuşu için hesaplanan vektör alanlarından örnekler.

Lider platforma etki eden kuvvetler incelendiğinde; hedef noktanın çeken, diğer platformların iten etkisi görülmektedir. Diğer platformlar için ise formasyon

şemasında olması gereken pozisyonları çeken, diğer platformların bulundukları nokta ise iten etkide davranırlar.

Şekil 9.33: Formasyon uçuşunun icra edilmesi sonucunda platformların izlediği rotaların gösterimi.

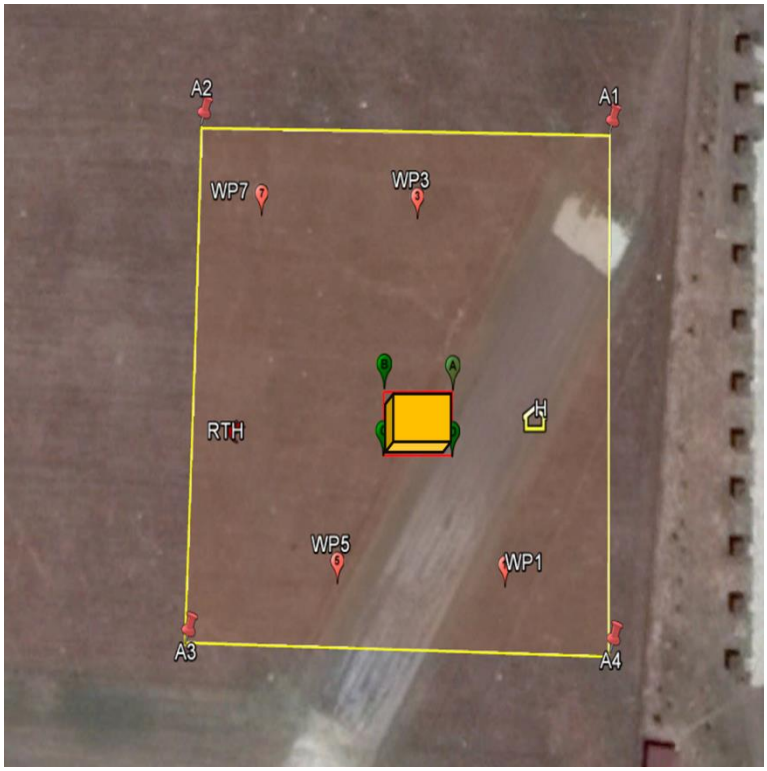
Şekil 9.33 değerlendirildiğinde platformların formasyonu muhafaza ederek yol noktalarını başarı ile ziyaret ettikleri görülmektedir.



Şekil 9.34: Formasyon uçuşunun gerçekleştirilmesi neticesinde ortaya çıkan rotalar ile beklenen rotanın karşılaştırılması.

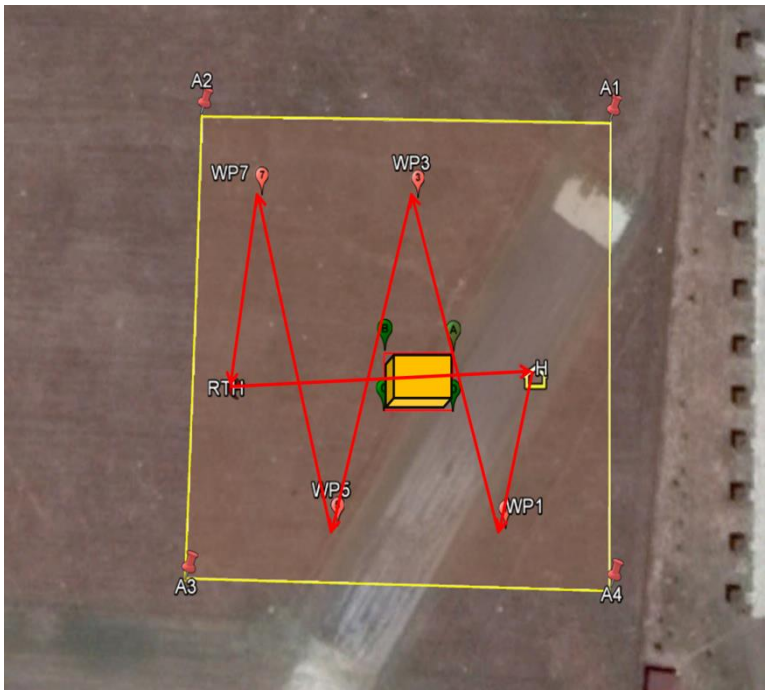
Tahmin edildiği gibi Şekil 9.34 ile de görüldüğü biçimde teoride planlanan rotadan farklı bir rota ortaya çıkmıştır. Ancak platformların rotaları incelendiğinde, formasyonu başarı ile muhafaza ettikleri ve yol noktalarına uğradıkları görülmektedir. Düz hat lider platformun rotasını temsil etmekte, kesikli çizgiler ise formasyondaki diğer platformların icra ettikleri uçuş rotasını göstermektedir.

Lider platform ile formasyonda yer alan platformlar arasındaki mesafe 5 m. olarak belirlenmiştir. Bu mesafe oluşan uçuş rotalarından ve Ek C'de yer alan Şekil C.5'den görüldüğü üzere azami şekilde muhafaza edilmeye çalışılmıştır.



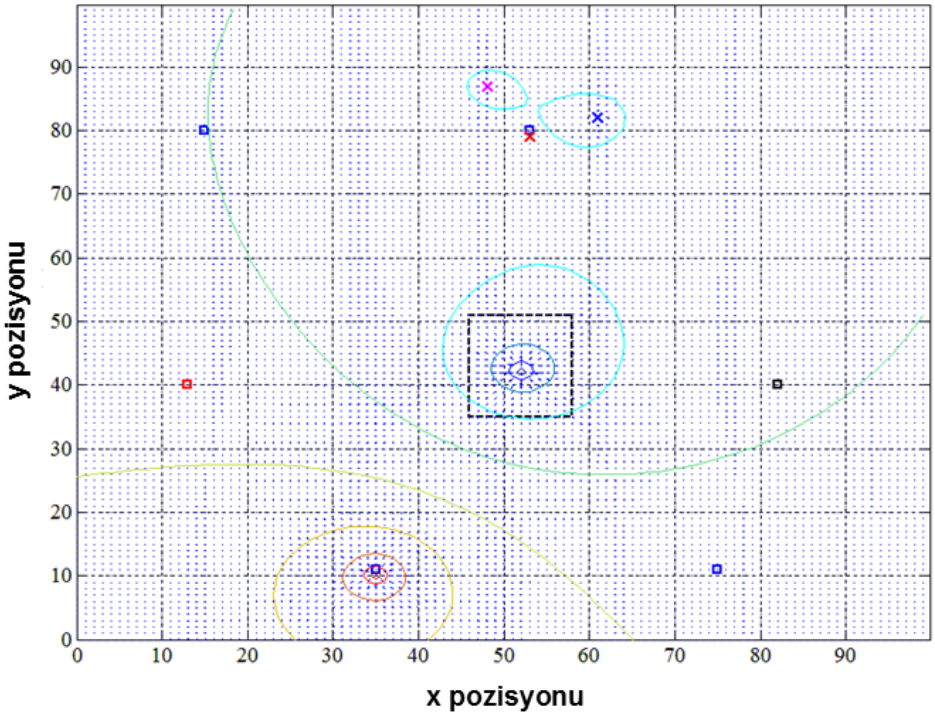
Şekil 9.35: Formasyon uçuşu için tanımlanan bölge içinde yer alan sanal engelin gösterimi.

Alan içerisinde doğal ya da suni bir engel bulunmadığından lider platform teoride planlanan rotaya yakın bir rota takip etmiştir. Şayet önceki senaryo dahilinde tanımladığımız aynı sanal engeli bu senaryo dahiline Şekil 9.35'da gösterilen biçimde tanımlarsak, teoride planlanan uçuş rotası Şekil 9.36 ile gösterildiği şekilde engel ile kesişecektir.

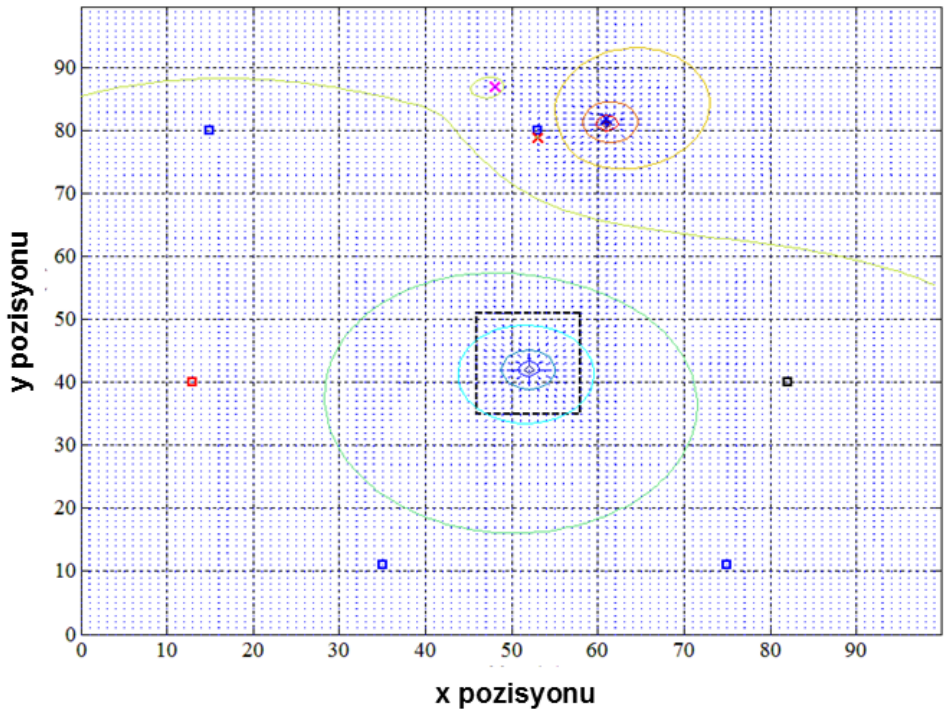


Şekil 9.36: Engel içeren alan dahilinde beklenen rota ile engelin kesişiminin gösterimi.

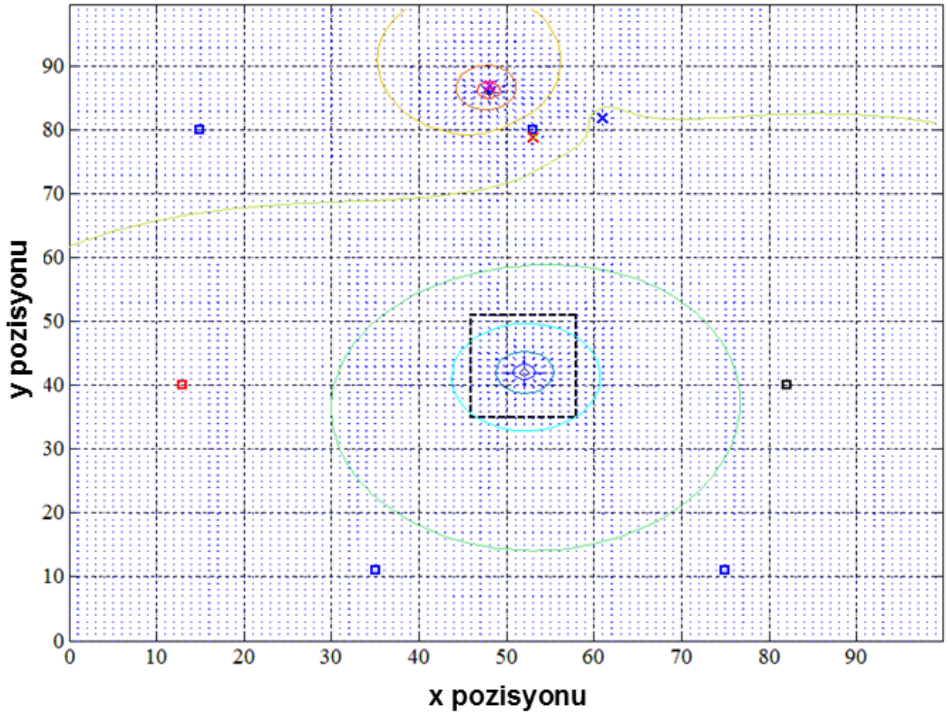
Tanımlanan sanal engelin koordinatları Çizelge 9.3 ile verilen konumdur. Bu durumda platformlar üzerindeki YPA tabanlı otonom yol planlaması ve kontrol amaçlı üretilen vektör alan incelendiğinde platformlar üzerinde etki eden faktörlere bir de iten karakterde engelin etkisi ilave edilecektir.



(a) t_n anında uygulamaya ait vektör alan gösterimi.



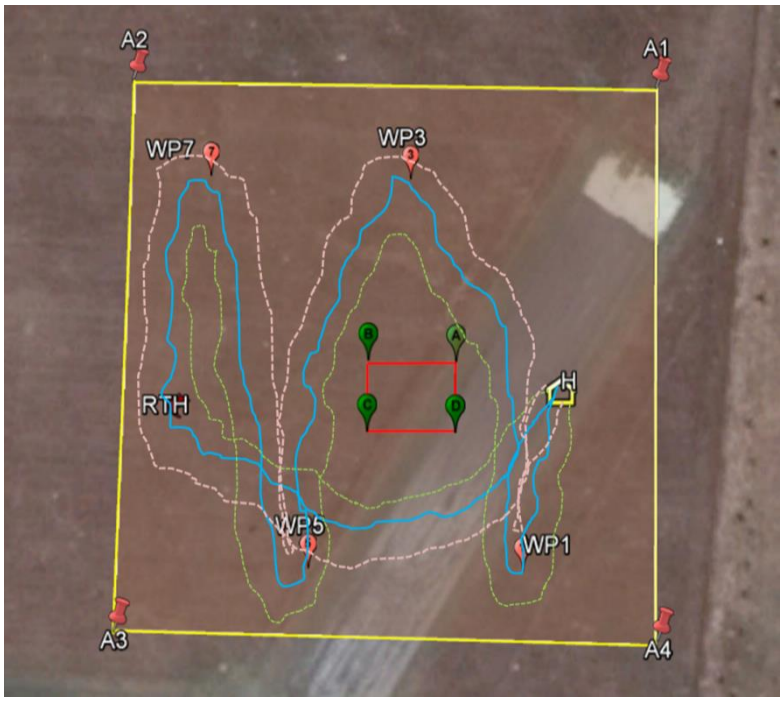
(b) t_{n+r} anında uygulamaya ait vektör alan gösterimi.



(c) t_{n+s} anında uygulamaya ait vektör alan gösterimi.

Şekil 9.37: Engel içeren alan içerisinde formasyon uçuşunu sağlamak için üretilen vektör alan örnekleri.

Bu durumda Şekil 9.37 ile *t* anında üç platformun hareketlerini belirleyen vektör alanlar örnek olarak gösterilmiştir.



Şekil 9.38: Engel içeren alan dahilinde icra edilen formasyon uçuşu sonucunda platformların izlediği rotaların gösterimi.

Sanal engelin tanımlanması sonucu gerçekleştirilen uçuş neticesinde oluşan rotalar Şekil 9.38 ile gösterilmiştir. Formasyon Ek C'de yer alan Şekil C.6 ile gösterildiği şekilde başarı ile muhafaza edilmiş ve görev tamamlanmıştır.

Şekil 9.39: Engel içeren alan içerisinde gerçekleştirilen formasyon uçuşunun beklenen rota ile karşılaştırılması.

Şekil 9.39 ile gösterildiği üzere formasyon dahilindeki platformlara ait uçuş sonundaki rotalar incelendiğinde teoride lider platform için planlanan rotaya mümkün olduğunca uygun bir uçuş gerçekleştirilmiş ve platformlar birbirlerinin arasında tanımlanan 5 m.lik formasyon mesafesini mümkün olduğunca muhafaza etmeyi başarmışlardır. Ayrıca tanımlanan sanal engelden başarı ile kaçınılmış ve görev dahilindeki yol noktalarına başarı ile ulaşılmıştır. Görevin tamamlanmasının ardından "*H*" noktasına dönen platformlar Ek C'de yer alan Şekil C.7 ile gösterilmektedir.

10. SONUÇ

Otonom sistemler arasında son yıllarda popüler duruma gelen İHA sistemlerinin teknolojilerindeki ilerlemeler neticesinde yeni otonom yeteneklerine olan ihtiyaç giderek artmıştır. Özellikle son yıllarda ortaya çıkan gelişmeler yakın gelecekte İHA platformlarının birçok insanlı hava aracı sisteminin yerini alacağının en önemli göstergesidir. Bu nedenle otonom İHA sistemleri bu tez çalışması kapsamında örnek uygulama alanı olarak belirlenmiştir. Otonom yeteneklerin arttırılmasına katkıda bulunulması amacıyla mümkün olduğunca hesaplama performansına ihtiyaç duyan formasyon halinde hareket edilmesi problemi örnek senaryo olarak belirlenmiştir. Hesaplama ihtiyacının arttırılması ve ortaya konulan yaklaşımın sınırlarının zorlanarak kabiliyetlerinin görülebilmesi amacıyla otonom İHA sistemlerinin bir formasyonu teşkil etmesi örnek uygulama olarak seçilmiştir. Ayrıca platformların bu formasyonu muhafaza ederek seyrüsefer icra etmesi, bu esnada birbirleri ve çevredeki engeller ile çarpışmalarını engelleyecek gerekli önlemleri alabilen bir otonom yol planlaması gerçekleştirmeleri bu tez çalışması kapsamında amaç olarak belirlenmiştir.

Çalışma kapsamında YPA tabanlı otonom yol planlamasının İHA gibi üç boyutlu ortamda hareket kabiliyetine sahip platformlar tarafından genel yol planlaması perspektifine uygun olarak icra edilebilmesine imkan sağlayacak şekilde çok katmanlı yaklaşım olarak isimlendirilen bir yapı ile gerçekleştirilmesi mümkün bir duruma getirilmiştir. Çok katmanlı yapı modellemesi ile halihazırda günümüzde kullanılan algılayıcı teknolojilerinden faydalanılarak otonom engel tespiti yapılabilmesine ve bunun yanı sıra tespit edilen engellere istinaden yol planlamasının gerçeğe yakın bir zaman zarfı içinde tekrar modellenerek hesaplanabilmesine imkan tanınmıştır.

Bu tez çalışması ile otonom platformların yol planlaması ihtiyacının gerçek zamana en yakın çözümler üretilerek karşılanması amaçlanmıştır. Literatürde yer alan otonom yol planlaması yönetmeleri incelenmiş ve bunlar içinde gerek benzetim tabanlı modeller olmalarından dolayı daha düşük hassasiyette algılayıcılar ile etkin sonuçlar üretebilmeleri gerekse de diğer yaklaşımlara nazaran daha hızlı sonuçlar üretebilmelerinden dolayı YPA tabanlı yol planlaması yöntemleri uygun yaklaşım olarak belirlenmiştir.

YPA tabanlı yaklaşımların literatürde tespit edilen yerel minimum, robot tuzağı, erişilemeyen hedefler gibi problemlerine çözüm oluşturması açısından genel yol planlaması metodolojisi kapsamında harmonik fonksiyonlardan faydalanılarak modellemeler gerçekleştirilmiştir. Birden fazla temel potansiyel alanın bir arada kullanılmasına ihtiyaç duyan HYİ ve formasyon uçuşu gibi görevlerin etkin olarak uygulanabilir yol planlaması sonuçları üretilebilmesi açısından sigmoid fonksiyonlardan faydalanılarak alt vektör alan sınırlandırmalarının gerçekleştirilmesi sağlanmıştır. Genel yol planlaması şeklinde harmonik fonksiyonlardan faydalanılarak modellenen ve sigmoid fonksiyonlar ile sınırlanan potansiyel alanlar her ne kadar literatürde karşılaşılan problemlerin çözülmesinde etkin olsa da hesaplama maliyetlerinin artmasına, dolayısıyla gerçek zamanlı çözümler üretilmesine olumsuz etki etmiştir.

Artan hesaplama performansının İHA platformları gibi yüksek süratli İHA sistemlerinde yaklaşımın etkin kullanımının sınırlandırılmasının önüne geçmek amacıyla paralel programlama yaklaşımları ile hesaplama performansının arttırılabileceği değerlendirilmiştir. YPA tabanlı yaklaşımların ızgara tabanlı yaklaşımlar olması, her bir ızgara hücresinin birbirinden bağımsız olarak ele alınarak hesaplanabilmesi yaklaşımın SIMD tipinde paralel algoritmalar geliştirilerek paralel şekilde hesaplanabilmesine olanak sağlamıştır.

Son dönemlerde paralel programlama maliyetlerini uygun hale getiren, araştırmacıların evlerinde dahi bir süper bilgisayar oluşturabilmesine imkan sağlayan GPGPU tabanlı yaklaşımlar, grafik işlemci mimarilerinin ucuz maliyetli, düşük güç tüketen ve de ölçek olarak uyarlanabilir yapılar olmalarından dolayı tercih edilen bir SIMD tipinde paralel uygulama geliştirme yaklaşımı olmalarına sebebiyet vermiştir. Ayrıca grafik işlemciler bu özelliklerinden dolayı mobil platformlar dahil olmak üzere farklı uygulamalar ile günümüzde yaygın olarak faydalanılan paralel programlama mimarileri olmuşlardır. Bu nedenlerden dolayı grafik işlemcilerin otonom platformlarda kullanılabilir yapılar olduğuna kanaat getirilmiş ve SIMD tipinde paralel uygulamaların grafik işlemciler üzerinde koşturan GPGPU tabanlı paralel algoritmalar geliştirilerek gerçekleştirilmesine karar verilmiştir.

YPA tabanlı otonom yol planlamasının paralel olarak hesaplanabilmesine imkan tanıyan iki yaklaşım çalışma kapsamında ortaya konulmuştur. Hücre bazlı yaklaşımın daha az kaynak ihtiyacı duyması ve nispeten zayıf olarak tanımlanan

grafik işlemci üzerindeki çekirdekler üzerine daha uygun şekilde paralel olarak yayılmasının daha etkin olacağı ortaya konulmuştur.

Çalışma kapsamında geliştirilen paralel algoritmanın başarısı, farklı mimarilerdeki mobil ve sabit grafik donanımları üzerinde sınanmıştır. Elde edilen performans sonuçları seri hesaplamalar ile karşılaştırmalı olarak örnek senaryolar dahilinde açıklanmıştır. Sonuç olarak tek bir grafik işlemciden faydalanılarak gerçekleştirilen hesaplamalar sonucunda seri hesaplama performansına nazaran 17,75 kat daha hızlı hesaplama sonuçları elde edilmiştir. Aynı ana sunucu üzerinde yer alan çok sayıda grafik işlemcinin bir arada kullanılması ile yaklaşık 41 kat seri hesaplamaya nazaran hesaplama performansında artış elde edilmiştir. Ağ ortamı üzerinde yer alan çok sayıda grafik kartından faydalanarak iletişim maliyetlerinin göz ardı edilmesi koşuluyla, bir başka değiş ile sonuçların bir sunucu üzerinde toplanması yerine dağıtılmış bir mimari üzerinde saklanması şartıyla yaklaşık 48 kat performans artışı elde edilmiştir.

Ortaya konulan simülasyon çalışmaları ve uygulamalar göstermiştir ki çalışma kapsamında ortaya konulan yaklaşım son dönemde geliştirilen ve hatta yakın gelecekte geliştirilecek olan İHA sistemlerinin otonom yol planlaması ihtiyacını karşılayabilecek niteliktedir. Çalışma kapsamında ortaya konulan GPGPU tabanlı paralel YPA algoritmasının çözünürlüğü 10 metre olarak belirlenmiş ve 40 km x 40 km. ölçülerinde bir alanın modellenmesi amacıyla koşturulması sonucunda her birisinden dört adet olmak üzere farklı formasyonlarda Aerosonde platformlarının birbirlerine en az 12 metre mesafede uçmalarına imkan sağlayacak hesaplama performansına ulaşılmıştır. Aynı şekilde Boeing X-45C platformlarının birbirlerine en az 72 metre, TAI ANKA platformlarının birbirlerine en az 18 metre ve Boeing tarafından geliştirilmesine devam edilen QF-16 Viper platformlarının birbirlerine en az 160 metre mesafede kol uçuşu gerçekleştirebilmeleri için gerekli hesaplama performansına ulaşılmıştır. Bu tanımlanan hesaplama süreleri içinde formasyon dahilinde yer alan her bir platform için ayrı ayrı yol planlaması gerçekleştirilebilmektedir.

Özellikle GPGPU tabanlı paralel hesaplama mimarilerinden yararlanarak sonuç üretebilen yaklaşım geniş ölçekli alanlar dahilinde hassas manevra yapılmasına imkan sağlayacak ve gerçek zamanlı tespit edilen sistem ihtiyaçlarına büyük oranla yanıt üretebilecek bir yaklaşım ortaya koymuştur. Elde edilen hesaplama performansı

sonuçları irdelendiğinde, grafik işlemciler üzerinde koşturulan algoritmanın alan ölçeğine ve çözünürlük değerine bağlı olan bellek ihtiyacının hesaplama performansına etki eden ana faktör olduğu görülmektedir. Tek bir grafik işlemci üzerinde gerçekleştirilen çalışmalarda özellikle PCI veri yolu üzerinden veri aktarım maliyeti ve blok başına düşen paylaşılan bellek alanının sınırlı olması hesaplama sürelerini olumsuz etkileyen ana unsurlar olmuştur. Ayrıca tek bir SM üzerinde koşturulabilen blok sayısı performansı sınırlayan bir diğer faktör olarak karşımıza çıkmıştır. Birden fazla grafik işlemcinin bir arada kullanıldığı uygulamalarda grafik işlemci toplam bellek alanından faydalanılarak daha geniş ölçekli alanların daha hızlı olarak hesaplanması mümkün hale gelmiştir. Ağ ortamı üzerinde yer alan grafik işlemcilerin bir arada kullanımı ile bellek kapasite kısıtlarının bir nebze önüne geçilmiş ancak bu sefer ağ ortamı üzerinden gerçekleştirilen haberleşme maliyetlerinin performansı olumsuz etkilediği görülmüştür.

Her ne kadar günümüzde İHA sistemlerinde kullanılan engel algılayıcı teknolojileri iki boyutlu algılama yeteneğine ve kısıtlı menzile sahip olsalar dahi bu konuda yakın gelecekte ilerleme kaydedileceği bir geçektir. Bu noktadan hareket ile YPA tabanlı yol planlaması amacıyla çok katmanlı olarak modellenen alanların üç boyutlu olarak ele alınmasının PCI x16 v.3.0 (15,75 GB/s) veya v.4.0 noktasına bağlı (31.5 GB/s) en az üç adet Compute Capability 3.0 ve üzerinde (Maxwell mimarisinde), CUDA doğrudan bellek erişimi (Direct Memory Access – DMA), uzaktan doğrudan bellek erişimi (Remote Direct Memory Access - RDMA), birleştirilmiş sanal adresleme (Unified Virtual Addressing – UVA) gibi teknolojileri destekleyen K80 gibi yüksek grafik işlemci bellek alanı sahibi (24GB) işlemcilerin bir arada kullanılması ile gerçek zamanlı sonuçlar üretebilecek şekilde ortaya konulabileceği değerlendirilmektedir.

Çarpışma ve engel sakınma tekniklerinin geliştirilmesi çalışmaları literatürde çok sayıda bulunmaktadır. Ancak bu tez ile öne sürülen özgün yaklaşım, yapay potansiyel alan ile ilgili işlemlerin paralel bir şekilde bir grafik işlemcisine yaptırılmış olmasıdır. Bu hesaplama yönteminin insansız hava araçlarına uygulanması özgün yaklaşımın ayrıca ne derece etkin olarak kullanılabilir olduğunun bir göstergesidir. Bu tez çalışmasında yol planlaması açısından konu alınan sistemler İHA platformları olsa dahi benzer modelleme teknikleri uygulanarak su altı araçlarının ve uzay platformlarının yol planlamasının da gerçekleştirilebileceği

değerlendirilmektedir. Tez önerisi kapsamında ortaya konulan hedeflere bu tez kapsamında gerçekleştirilen çalışmalar sonucunda başarı ile ulaşılmıştır. Tez çalışması kapsamında geliştirilen yaklaşımın hesaplama performansı başarısı ve uygulamadaki etkinliği hem simülasyon hem de uygulama çalışmaları sonuçları ile ortaya konulmuştur.

KAYNAKLAR

[1] **Goerzen, C., Kong, Z., & Mettler, B.** (2010). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. Journal of Intelligent and Robotic Systems, 57(1-4), 65-100.

[2] **Chen, H., Wang, X. M., & Li, Y.** (2009, November). A survey of autonomous control for UAV. In Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on (Vol. 2, pp. 267-271). IEEE.

[3] **d'Oleire-Oltmanns, S., Marzolff, I., Peter, K. D., & Ries, J. B.** (2012). Unmanned Aerial Vehicle (UAV) for monitoring soil erosion in Morocco. Remote Sensing, 4(11), 3390-3416.

[4] **Mancini, F., Dubbini, M., Gattelli, M., Stecchi, F., Fabbri, S., & Gabbianelli, G.** (2013). Using unmanned aerial vehicles (UAV) for high-resolution reconstruction of topography: The structure from motion approach on coastal environments. Remote Sensing, 5(12), 6880-6898.

[5] **Cetin, O., Kurnaz, S., & Kaynak, O.** (2011). Fuzzy logic based approach to design of autonomous landing system for unmanned aerial vehicles. Journal of Intelligent & Robotic Systems, 61(1-4), 239-250.

[6] **Beard, R. W., Kingston, D., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., & Goodrich, M.** (2005). Autonomous vehicle technologies for small fixed-wing UAVs. Journal of Aerospace Computing, Information, and Communication, 2(1), 92-108.

[7] **Bone, E., & Bolkcom, C.** (2003, April). Unmanned aerial vehicles: Background and issues for congress. LIBRARY OF CONGRESS WASHINGTON DC CONGRESSIONAL RESEARCH SERVICE.

[8] **Anderson, B. D., Fidan, B., Yu, C., & Walle, D.** (2008). UAV formation control: theory and application. In Recent advances in learning and control (pp. 15-33). Springer London.

[9] **Ismail, A. T., Sheta, A., & Al-Weshah, M.** (2008). A mobile robot path planning using genetic algorithm in static environment. Journal of Computer Science, 4(4), 341.

[10] **Rathbun, D., Kragelund, S., Pongpunwattana, A., & Capozzi, B.** (2002). An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments. In Digital Avionics Systems Conference, 2002. Proceedings. The 21st (Vol. 2, pp. 8D2-1). IEEE.

[11] **Hwang, Y. K., & Ahuja, N.** (1992). A potential field approach to path planning. Robotics and Automation, IEEE Transactions on, 8(1), 23-32.

[12] **Khatib, O.** (1986). Real-time obstacle avoidance for manipulators and mobile robots. The international journal of robotics research, 5(1), 90-98.

[13] **Keşoğlu, Y.** (Temmuz 2010). Potansiyel Fonksiyon Tabanli Yol Planlamada Harmonik Fonksiyonlar İle İki Değişkenli Üstel Fonksiyonlarin Karşilaştirilmasi. Yüksek Lisans Tezi, Hava Harp Okulu HUTEN.

[14] **Koç, T.** (Temmuz 2011). İnsansız Araçlarda Harmonik Potansiyel Fonksiyon Tabanlı Çoğul Panel Metodu İle Otonom Navigasyon, Yüksek Lisans Tezi, Hava Harp Okulu, HUTEN.

[15] **Lee, L. F.** (2004). Decentralized motion planning within an artificial potential framework (APF) for cooperative payload transport by multi-robot collectives (Doctoral dissertation, State University of New York at Buffalo).

[16] **Wallace, S.A.** (2010). Development Of A Small And Inexpensive Terrain Avoidance System For An Unmanned Aerial Vehicle Via Potential Function Guidance Algorithm. Master's Theses California Polytechnic State University.

[17] **International Civil Aviation Organization.** (2011). ICAO Cir 328, Unmanned Aircraft Systems (UAS), ISBN 978-92-9231-751-5,.

[18] **US Office of the Secretary of Defense** (2009). Unmanned Systems Integrated Roadmap 2009-2034.

[19] **The Unmanned Aerial Vehicle Systems Association**, UAS Components, https://www.uavs.org/index.php?page=uas_components [son erişim 30.03.2015]

[20] **US Office of the Secretary of Defense** (2007). Unmanned Systems Roadmap 2007-2032.

[21] **Sasiadek, J. Z., & Dulęba, I.** (2000). 3D local trajectory planner for UAV. Journal of Intelligent and Robotic Systems, 29(2), 191-210.

[22] **Gertler, J.** (2012, January). US unmanned aerial systems. Lıbrary Of Congress Washıngton Dc Congressıonal Research Servıce.

[23] **Colucci, F.** (2014). QF16: Unmanned Viper Takes Flight, Avionics Today Journal, http://www.aviationtoday.com/av/military/QF16-Unmanned-Viper-Takes-Flight_81908.html#.VRmm9OFQGVE [son erişim 30.03.2015]

[24] **Türk Hava Kuvvetleri Komutanlığı**, http://www.hvkk.tsk.tr/TR/Index.aspx, [son erişim 30.03.2015]

[25] **Stelth Movie**, http://www.imdb.com/title/tt0382992/, [son erişim 30.03.2015]

[26] **DARPA**. Making Connections At 45,000 Feet: Future UAVs May Fuel up in Flight. http://www.darpa.mil/NewsEvents/Releases/2012/10/05.aspx, [son erişim: 30.03.2015]

[27] **Reed J.** (January, 2012). Navy Getting Very Close to UAV Aerial Refueling. Defense.org http://defensetech.org/2012/01/26/navy-getting-very-close-to-uav-aerial-refueling/#ixzz2LdAefRd7, [son erişim: 30.03.2015].

[28] **Ackerman, E.** (2012). Quadrotors Turned Into Flying Wireless Battery Chargers, IEEE Spectrum, http://spectrum.ieee.org/automaton/robotics/robotics-hardware/quadrotors-turned-into-flying-wireless-battery-chargers, [son erişim: 30.03.2015]

[29] **Paul, T., Krogstad, T. R., & Gravdahl, J. T.** (2008). Modelling of UAV formation flight using 3D potential field. Simulation Modelling Practice and Theory, 16(9), 1453-1462.

[30] **Wang, X., Yadav, V., & Balakrishnan, S. N.** (2007). Cooperative UAV formation flying with obstacle/collision avoidance. Control Systems Technology, IEEE Transactions on, 15(4), 672-679.

[31] **Christopher I. Connolly and Roderic A. Grupen**. (1992). Applications of Harmonic Functions to Robotics. University of Massachusetts at Amherst.

[32] **Huang, H. M., Messina, E., & Albus, J.** (2003). Autonomy Level Specification For Intelligent Autonomous Vehicles. National Inst Of Standards And Technology Gaithersburg Md.

[33] **Buniyamin, N., Wan Ngah, W. A. J., Sariff, N., & Mohamad, Z.** (2011). A simple local path planning algorithm for autonomous mobile robots. International journal of systems applications, Engineering & development, 5(2), 151-159.

[34] **Masehian, E., & Sedighizadeh, D.** (2007). Classic and heuristic approaches in robot motion planning-a chronological review. World Academy of Science, Engineering and Technology, 23, 101-106.

[35] **Li, G., Yamashita, A., Asama, H., & Tamura, Y.** (2012, August). An efficient improved artificial potential field based regression search method for robot path planning. In Mechatronics and Automation (ICMA), 2012 International Conference on (pp. 1227-1232). IEEE..

[36] **Dudek, G., & Jenkin, M.** (2010). Computational principles of mobile robotics. Cambridge university press.

[37] **Rathbun, D., Kragelund, S., Pongpunwattana, A., & Capozzi, B.** (2002). An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments. In Digital Avionics Systems Conference, 2002. Proceedings. The 21st (Vol. 2, pp. 8D2-1). IEEE..

[38] **Alejo, D., Conde, R., Cobano, J. A., & Ollero, A.** (2009, April). Multi-UAV collision avoidance with separation assurance under uncertainties. In Mechatronics, 2009. ICM 2009. IEEE International Conference on (pp. 1-6). IEEE..

[39] **Meng, B. B., & Gao, X.** (2010, May). UAV path planning based on bidirectional sparse A* search algorithm. In Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on (Vol. 3, pp. 1106-1109). IEEE..

[40] **Xia, L., Jun, X., Manyi, C., Ming, X., & Zhike, W.** (2009, August). Path planning for UAV based on improved heuristic A∗ algorithm. In

Electronic Measurement & Instruments, 2009. ICEMI'09. 9th International Conference on (pp. 3-488). IEEE.

[41] **Redding, J., Amin, J. N., Boskovic, J. D., Kang, Y., Hedrick, K., Howlett, J., & Poll, S.** (August 2007). A real-time obstacle detection and reactive path planning system for autonomous small-scale helicopters. In AIAA Navigation, Guidance and Control Conference, Hilton Head, SC.

[42] **Jun, M., & D'Andrea, R.** (2003). Path planning for unmanned aerial vehicles in uncertain and adversarial environments. In Cooperative Control: Models, Applications and Algorithms (pp. 95-110). Springer US.

[43] **Vaščák, J.** (2007). Navigation of mobile robots using potential fields and computational intelligence means. Acta Polytechnica Hungarica, 4(1), 63-74.

[44] **Richards, A., & How, J. P.** (2002). Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In American Control Conference, 2002. Proceedings of the 2002 (Vol. 3, pp. 1936-1941). IEEE..

[45] **Schouwenaars, T., De Moor, B., Feron, E., & How, J.** (2001, September). Mixed integer programming for multi-vehicle path planning. In European control conference (Vol. 1, pp. 2603-2608)..

[46] **Han, S. C., Bang, H., & Yoo, C. S.** (2009). Proportional navigation-based collision avoidance for UAVs. International Journal of Control, Automation and Systems, 7(4), 553-565.

[47] **Han, S. C., & Bang, H.** (2004, December). Proportional navigation-based optimal collision avoidance for uavs. In 2nd International Conference on Autonomous Robots and Agents (pp. 13-15)..

[48] **Park, J. W., Oh, H. D., & Tahk, M. J.** (2008, August). Uav collision avoidance based on geometric approach. In SICE Annual Conference, 2008 (pp. 2122-2126). IEEE..

[49] **Han, T. D., & Abdelrahman, T. S.** (2011). hiCUDA: High-level GPGPU programming. Parallel and Distributed Systems, IEEE Transactions on, 22(1), 78-90.

[50] **NVIDIA**. (2010). CUDA Programming Guide Version 3.0.

[51] **NVIDIA**. (2012). CUDA C Programming Guide, Version 4.2.

[52] **Stone, J. E., Gohara, D., & Shi, G.** (2010). OpenCL: A parallel programming standard for heterogeneous computing systems. Computing in science & engineering, 12(1-3), 66-73.

[53] **Wang, B., Wu, T., Yan, F., Li, R., Xu, N., & Wang, Y.** (2009, December). RankBoost Acceleration on both NVIDIA CUDA and ATI Stream platforms. In Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on (pp. 284-291). IEEE.

[54] **Kirk, D.** (2007, October). NVIDIA CUDA software and GPU parallel computing architecture. In ISMM (Vol. 7, pp. 103-104).

[55] **Aila, T., Laine, S., & Karras, T.** (2012). Understanding the efficiency of ray traversal on GPUs–Kepler and Fermi addendum. NVIDIA Corporation, NVIDIA Technical Report NVR-2012-02.

[56] **Wittenbrink, C. M., Kilgariff, E., & Prabhu, A.** (2011). Fermi GF100 GPU architecture. IEEE Micro, 31(2), 50-59.

[57] **NVIDIA.** CARMA Development Kit, https://developer.nvidia.com/seco-development-kit, [son erişim: 30.03.2015]

[58] **NVIDIA** Quadro 1000M, http://www.nvidia.com/object/quadro-mobile-features-benefits.html, [son erişim: 30.03.2015]

[59] **NVIDIA** GeFroce 670MX, http://www.geforce.com/hardware/notebook-gpus/geforce-gtx-670mx, [son erişim: 30.03.2015]

[60] **NVIDIA** Tesla K20c, http://www.nvidia.com/object/tesla-workstations.html, [son erişim: 30.03.2015]

[61] **NVIDIA** GTX 480, http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-480, [son erişim: 30.03.2015]

[62] **Ge, S. S., & Cui, Y. J.** (2002). Dynamic motion planning for mobile robots using potential field method. Autonomous Robots, 13(3), 207-222.

[63] **Kim, J. O., & Khosla, P. K.** (1992). Real-time obstacle avoidance using harmonic potential functions. Robotics and Automation, IEEE Transactions on, 8(3), 338-349.

[64] **Connolly, C. I., & Grupen, R. A.** (1993). The applications of harmonic functions to robotics. Journal of robotic Systems, 10(7), 931-946.

[65] **Rimon, E., & Koditchek, D. E.** (1992). Exact robot navigation using artificial potential functions. Robotics and Automation, IEEE Transactions on, 8(5), 501-518.

[66] **Shi, Y., Eberhart, R., & Chen, Y.** (1999). Implementation of evolutionary fuzzy systems. Fuzzy Systems, IEEE Transactions on, 7(2), 109-119.

[67] **Yang, K., & Sukkarieh, S.** (2008, September). 3D smooth path planning for a UAV in cluttered natural environments. In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on (pp. 794-800). IEEE.

[68] **Koren, Y., & Borenstein, J.** (1991, April). Potential field methods and their inherent limitations for mobile robot navigation. In Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on (pp. 1398-1404). IEEE.

[69] **Ge, S. S., & Cui, Y. J.** (2000). New potential functions for mobile robot path planning. IEEE Transactions on robotics and automation, 16(5), 615-620.

[70] **Volpe, R., & Khosla, P.** (1990). Manipulator control with superquadric artificial potential functions: Theory and experiments. Systems, Man and Cybernetics, IEEE Transactions on, 20(6), 1423-1436..

[71] **Krogh, B. H.** (1984). A generalized potential field approach to obstacle avoidance control. RI/SME. in Proc. ASME Conf. of Robotic

Research: The Next Five Years and Beyond, Bethlehem, Pennsylvania.

[72] **Beard R.W. & McLain T.W.** (2003). Motion planning using potential field, tutorial. Provo, UTAH.

[73] **Koditschek, D. E., & Rimon, E.** (1990). Robot navigation functions on manifolds with boundary. Advances in Applied Mathematics, 11(4), 412-442..

[74] **Krogh, B. H., & Thorpe, C. E.** (1986, April). Integrated path planning and dynamic steering control for autonomous vehicles. In Robotics and Automation. Proceedings. 1986 IEEE International Conference on (Vol. 3, pp. 1664-1669). IEEE.

[75] **Brooks, R. A.** (1986). A robust layered control system for a mobile robot. Robotics and Automation, IEEE Journal of, 2(1), 14-23..

[76] **Arkin, R. C.** (1987). Towards cosmopolitan robots: Intelligent navigation in extended man-made environments..

[77] **Borenstein, J., & Koren, Y.** (1991). The vector field histogram-fast obstacle avoidance for mobile robots. Robotics and Automation, IEEE Transactions on, 7(3), 278-288.

[78] **Kazemi, M., & Mehrandezh, M.** (2004). Robotic navigation using harmonic function-based probabilistic roadmaps. In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on (Vol. 5, pp. 4765-4770). IEEE.

[79] **Hong, Z., Liu, Y., Zhongguo, G., & Yi, C.** (2011, April). The dynamic path planning research for mobile robot based on artificial potential field. In Consumer Electronics, Communications and Networks (CECNet), 2011 International Conference on (pp. 2736-2739). IEEE.

[80] **Qi, N., Ma, B., Liu, X. E., Zhang, Z., & Ren, D.** (2008, June). A modified artificial potential field algorithm for mobile robot path planning. In Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on (pp. 2603-2607). IEEE.

[81] **Shi, W. R., Huang, X. H., & Zhou, W.** (2010). Path planning of mobile robot based on improved artificial potential field. Jisuanji Yingyong/ Journal of Computer Applications, 30(8), 2021-2023.

[82] **Zhang, B., Chen, W., & Fei, M.** (2006, October). An optimized method for path planning based on artificial potential field. In Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on (Vol. 3, pp. 35-39). IEEE.

[83] **Shi, P., & Zhao, Y.** (2009, August). Global path planning for mobile robot based on improved artificial potential function. In Automation and Logistics, 2009. ICAL'09. IEEE International Conference on (pp. 1900-1904). IEEE.

[84] **Daily, R., & Bevly, D. M.** (2008, June). Harmonic potential field path planning for high speed vehicles. In American Control Conference, 2008 (pp. 4609-4614). IEEE.

[85] **Vehrs, C.** (1974). U.S. Patent No. 3,795,909. Washington, DC: U.S. Patent and Trademark Office.

[86] **Nevin, R. L.** (1988, April). Waveform trade-offs for medium PRF air-to-air radar. In Radar Conference, 1988., Proceedings of the 1988 IEEE National (pp. 140-145). IEEE.

[87] **Lockheed Martin**. AN/AAQ-33 Sniper, http://www.lockheedmartin.com/us/products/Sniper.html, [son erişim: 30.03.2015]

[88] **He, L., Chao, Y., Suzuki, K., & Wu, K.** (2009). Fast connected-component labeling. Pattern Recognition, 42(9), 1977-1987.

[89] **Duff, I. S., Grimes, R. G., & Lewis, J. G.** (1989). Sparse matrix test problems. ACM Transactions on Mathematical Software (TOMS), 15(1), 1-14.

[90] **Bolz, J., Farmer, I., Grinspun, E., & Schröoder, P.** (2003, July). Sparse matrix solvers on the GPU: conjugate gradients and multigrid. In ACM Transactions on Graphics (TOG) (Vol. 22, No. 3, pp. 917-924). ACM.

[91] **Chen, Y. H.** (1997). Determining parting direction based on minimum bounding box and fuzzy logics. International Journal of Machine Tools and Manufacture, 37(9), 1189-1199.

[92] **MATLAB** Parallel Computing Toolbox, http://www.mathworks.com/products/parallel-computing/?refresh=true, [son erişim: 30.03.2015]

[93] **MATLAB** MEX Library, http://www.mathworks.com/help/matlab/mex-library.html, [son erişim: 30.03.2015]

[94] **MATLAB** Distributed Server, http://www.mathworks.com/products/distriben/, [son erişim: 30.03.2015]

[95] **Kinoshita, T., & Imado, F.** (2006, October). The application of an uav flight simulator-the development of a new point mass model for an aircraft. In SICE-ICASE, 2006. International Joint Conference (pp. 4378-4383). IEEE.

[96] **Bristeau, P. J., Callou, F., Vissiere, D., & Petit, N.** (2011, August). The navigation and control technology inside the ar. drone micro uav. In 18th IFAC World Congress (Vol. 18, No. 1, pp. 1477-1484).

[97] **AR Drone Simulink Development Kit v1.1**, http://www.mathworks.com/matlabcentral/fileexchange/43719-ar-drone-simulink-development-kit-v1-1, [son erişim: 30.03.2015]

[98] **ArduPilot** Online Manual and Support Forum, http://code.google.com/p/ardupilot-mega/wiki/Introduction,[last access 10 September 2013]

[99] **ATMega328** and ATTiny Product Pages, Atmel Corporation, http://www.atmel.com/dyn/products/product_card.asp?part_id=4720,[last access 10 September 2013]

EKLER

EK A: Grafik işlemci belleğine veri aktarım performansları
EK B: Modifiye edilmiş AR Drone platformu
EK C: Uygulama görüntüleri

EK A Grafik işlemci belleğine veri aktarım performansları

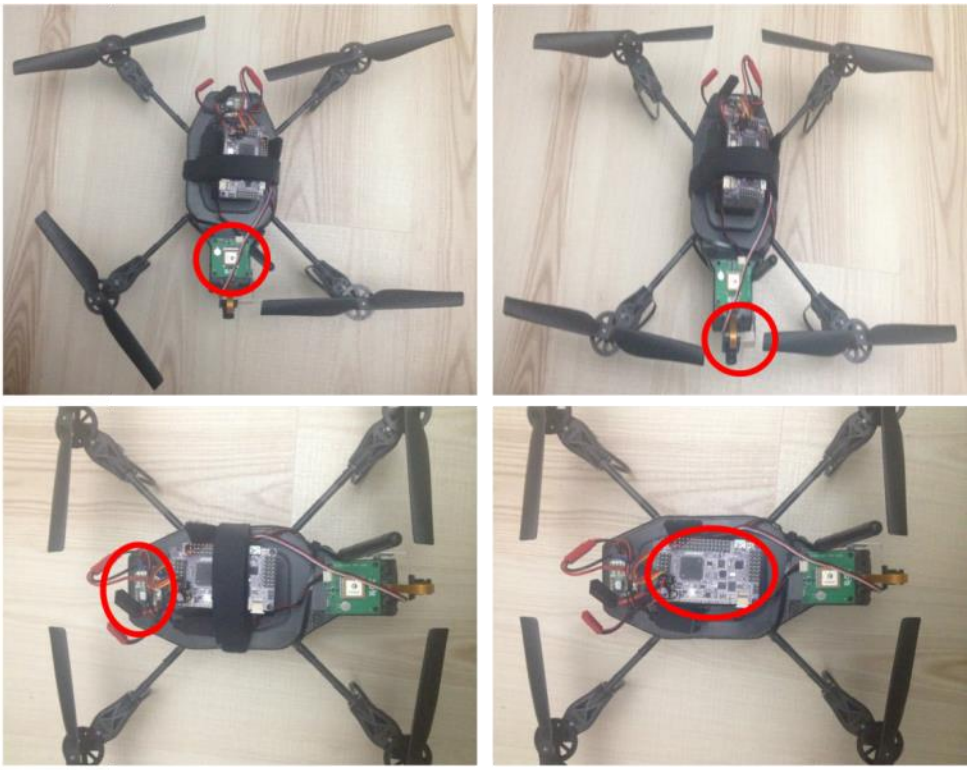
Çizelge A.1: Normal mod veri transferi performansı.

Transfer Edilen Veri Paketi Boyutu (byte)	H2D Transfer		D2H Transfer	
	Paket Transfer Süresi (milisaniye)	1 Byte Verinin Transferi için gerekli süre	Paket Transfer Süresi (milisaniye)	1 Byte Verinin Transferi için gerekli süre
1	0.120945	0.12094545	0.080556	0.08055563
2	0.084657	0.04232873	0.083177	0.04158836
4	0.075156	0.01878909	0.079043	0.01976073
8	0.075153	0.00939418	0.132308	0.01653854
16	0.121129	0.00757055	0.092695	0.00579346
32	0.079156	0.00247364	0.083177	0.00259927
64	0.087028	0.00135982	0.053094	0.00082959
128	0.074211	0.00057977	0.090208	0.00070475
256	0.075616	0.00029538	0.088346	0.00034510
512	0.078813	0.00015393	0.080611	0.00015744
1024	0.083738	0.00008178	0.085743	0.00008373
2048	0.080890	0.00003950	0.193367	0.00009442
4096	0.076649	0.00001871	0.089693	0.00002190
8192	0.069626	0.00000850	0.078647	0.00000960
16384	0.144404	0.00000881	0.100189	0.00000612
32768	0.094010	0.00000287	0.123590	0.00000377
65536	0.095148	0.00000145	0.097420	0.00000149
131072	0.129428	0.00000099	0.126653	0.00000097
262144	0.170025	0.00000065	0.171488	0.00000065
524288	0.238746	0.00000046	0.286868	0.00000055
1048576	0.512207	0.00000049	0.639680	0.00000061
2097152	0.803185	0.00000038	0.831811	0.00000040
4194304	1.448268	0.00000035	1.668041	0.00000040
8388608	3.042441	0.00000036	2.922982	0.00000035
16777216	5.727.555	0.00000034	5.822.589	0.00000035
33554432	11.114.330	0.00000033	11.711.289	0.00000035
67108864	21.125.643	0.00000031	23.188.993	0.00000035
134217728	43.515.846	0.00000032	46.126.621	0.00000034
268435456	85.132.103	0.00000032	91.881.462	0.00000034
536870912	167.821.899	0.00000031	183.025.040	0.00000034

Çizelge A.2: Asenkron mod (Page-Locked) veri transferi performansı.

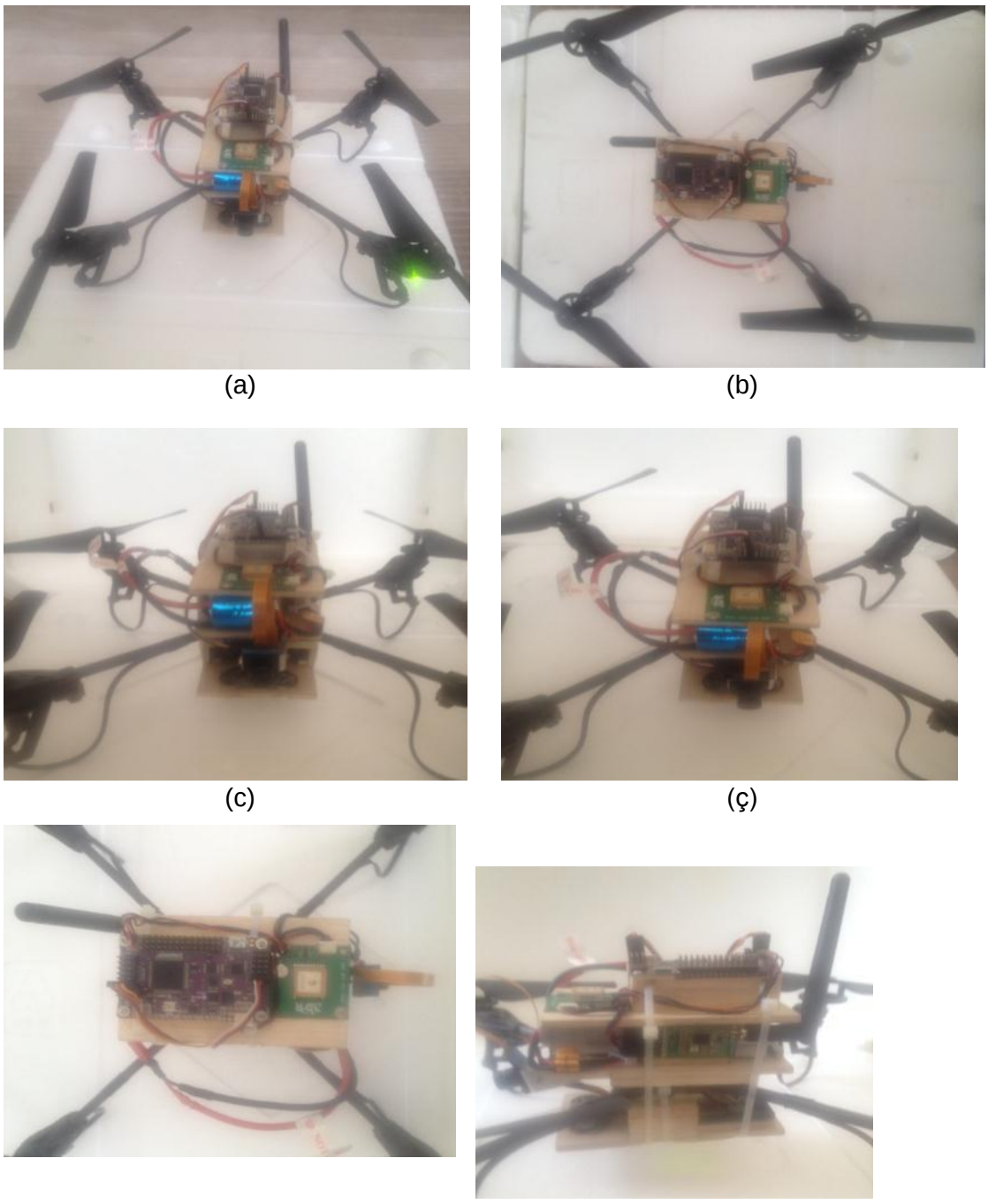
Transfer Edilen Veri Paketi Boyutu (byte)	H2D Transfer		D2H Transfer	
	Paket Transfer Süresi (milisaniye)	1 Byte Verinin Transferi için gerekli süre	Paket Transfer Süresi (milisaniye)	1 Byte Verinin Transferi için gerekli süre
1	0.008596	0.00859636	0.007270	0.00726982
2	0.008367	0.00418327	0.007596	0.00379782
4	0.008500	0.00212509	0.007284	0.00182109
8	0.008605	0.00107564	0.007305	0.00091309
16	0.008748	0.00054673	0.007599	0.00047491
32	0.008451	0.00026409	0.007284	0.00022764
64	0.008867	0.00013855	0.007369	0.00011514
128	0.008471	0.00006618	0.007168	0.00005600
256	0.008567	0.00003347	0.007247	0.00002831
512	0.008471	0.00001655	0.007252	0.00001416
1024	0.008503	0.00000830	0.007607	0.00000743
2048	0.008512	0.00000416	0.007581	0.00000370
4096	0.008788	0.00000215	0.007884	0.00000192
8192	0.009318	0.00000114	0.008535	0.00000104
16384	0.010892	0.00000066	0.010534	0.00000064
32768	0.013073	0.00000040	0.013489	0.00000041
65536	0.017873	0.00000027	0.019863	0.00000030
131072	0.027665	0.00000021	0.032564	0.00000025
262144	0.047945	0.00000018	0.058394	0.00000022
524288	0.087884	0.00000017	0.108503	0.00000021
1048576	0.168305	0.00000016	0.209466	0.00000020
2097152	0.328989	0.00000016	0.411165	0.00000020
4194304	0.650179	0.00000016	0.816003	0.00000019
8388608	1.293117	0.00000015	1.623180	0.00000019
16777216	2.579247	0.00000015	3.242362	0.00000019
33554432	5.150444	0.00000015	6.475569	0.00000019
67108864	10.291874	0.00000015	12.942956	0.00000019
134217728	20.583908	0.00000015	25.925356	0.00000019
268435456	41.148655	0.00000015	51.803799	0.00000019
536870912	82.316551	0.00000015	103.622200	0.00000019

EK B Modifiye edilmiş AR Drone platformu



Şekil B.1: AR drone platformu üzerine entegre edilen sistemlerin gösterimi.

Şekil B.2: AR drone platformu üzerinde gerçekleştirilen değişikliklerin gösterimi.



(a) (b)

(c) (ç)

(d) (e)

Şekil B.3: Hazırlanan Parrot AR Drone quadrotor İHA platformu.

EK C Uygulama görüntüleri

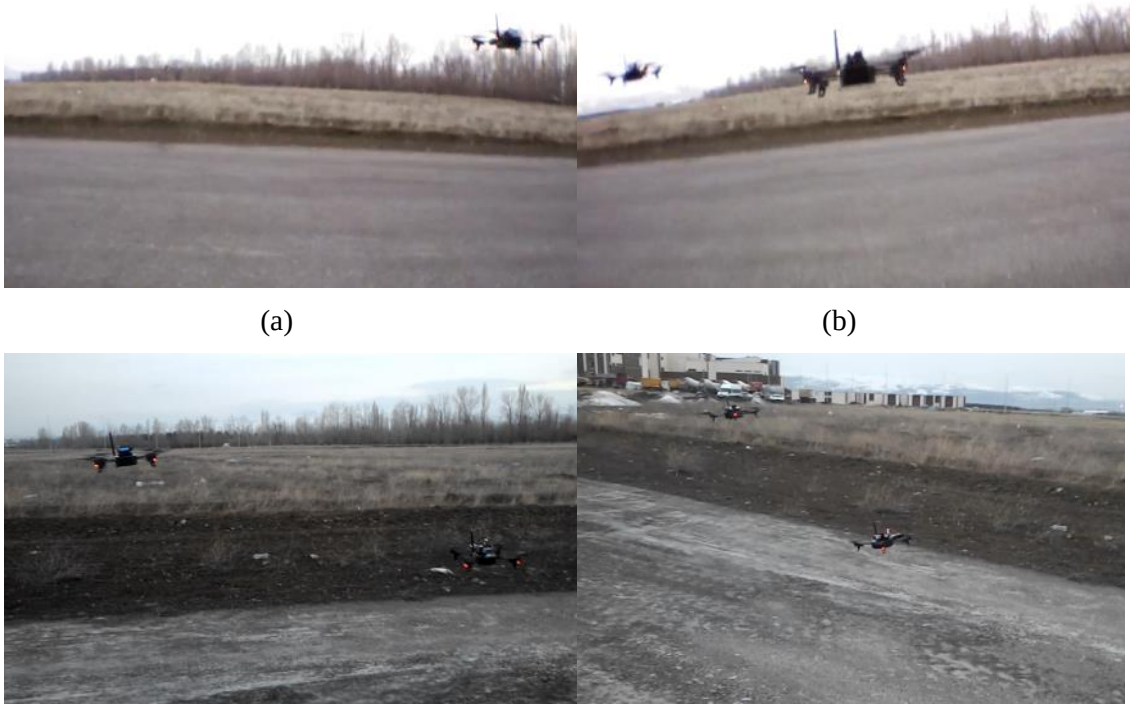
(a) (b)

(c) (ç)

Şekil C.4: Platformun uçuş esnasındaki görüntüleri.

Şekil C.5: Formasyon uçuşu gerçekleştirecek platformların başlangıç durumu.

Şekil C.6: Kalkışı takiben formasyon halindeki otonom platformların görünümü.



(a) (b)

(c) (ç)

Şekil C.7: Birbirlerinin kamerasından platformların uçuş anındaki görüntüleri.

(a)

(b)

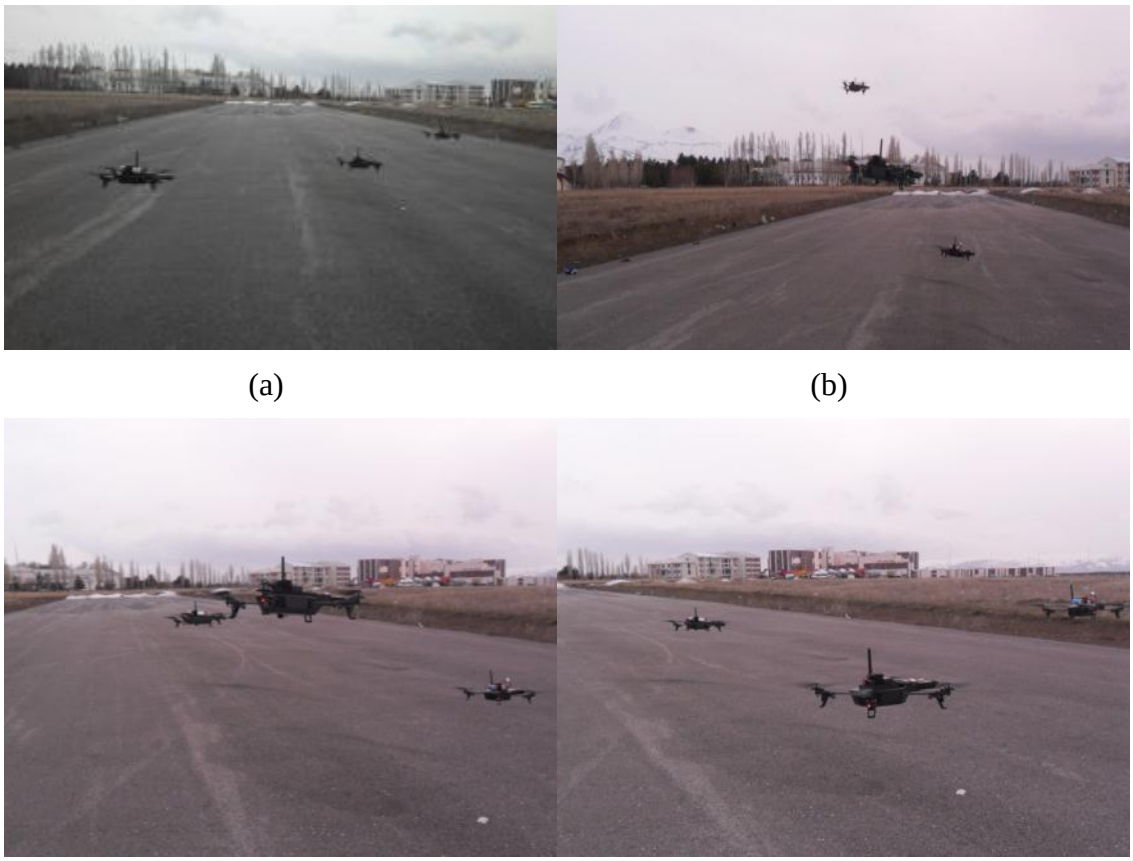
Şekil C.8: Platformların birbirleri arasındaki mesafeyi muhafaza etmelerinin uçuş anındaki görünümleri.

(a)

(b)

(c)

Şekil C.9: Uçuş esnasında platformlar tarafından çekilen görüntüler.



(a) (b)

(c) (ç)

Şekil C.10: Başlangıç noktasına geri dönen platformların görünüşü.

ÖZGEÇMİŞ

Ad Soyad: Ömer ÇETİN

Doğum Yeri ve Tarihi: 10 Nisan 1981

E-Posta: omer_cetin@outlook.com.tr

Lisans: Hava Harp Okulu
Bilgisayar Mühendisliği Bölümü (1999-2003)

Yüksek Lisans: Hava Harp Okulu HUTEN
Bilgisayar Mühendisliği Ana Bilim Dalı (2006-2008)

Mesleki Deneyim ve Ödüller:

Hava Muhabere Yüzbaşı Ömer Çetin 10 Nisan 1981 tarihinde İzmir'de dünyaya gelmiş olup, 1999 yılında Maltepe Askeri Lisesini bitirerek Hava Harp Okulu'na başlamıştır. 2003 yılında Hava Harp Okulu Bilgisayar Mühendisliği Bölümünden Bilgisayar Mühendisi Teğmen rütbesinde mezun olmuş ve 2005 yılında Muhabere Subay Temel Eğitimini başarı ile tamamlayarak Muhabere Subayı unvanı ile Hava Kuvvetleri Komutanlığındaki görev yaşantısına Diyarbakır 8'inci Ana Jet Üs Komutanlığı'nda İşletme Takım Komutanı olarak atanarak başlamıştır. 2006-2008 yılları arasında Havacılık ve Uzay Teknolojileri Enstitüsünde Bilgisayar Mühendisliği Ana Bilim Dalında yüksek lisans eğitimi tamamlamış ve Hava Harp Okulu Uzaktan Eğitim Sistemi Sistem Yöneticisi olarak 2010 yılına kadar görev yapmıştır. 2010 yılında Havacılık ve Uzay Teknolojileri Enstitüsünde Bilgisayar Mühendisliği Ana Bilim Dalında doktora eğitimine başlamıştır. 2011-2012 yılları arasında GPU tabanlı paralel programlama laboratuvar koordinatörlüğü ve 2011-2013 yılları arasında Bahçeşehir Üniversitesi, Hava Harp Okulu ve Japonya KDDI Araştırma Kuruluşu tarafından yürütülen projede araştırmacı olarak görev icra etmiştir. 2013 yılında 217 Numaralı Hava Radar Kıt'a Komutanlığına Kıt'a Komutanı olarak atanan Ömer ÇETİN, halen Erzurum Hava Radar Mevzi Komutanlığı Radar Kıtalar Tabur Komutanlığı Plan Koordinasyon Kısım Amirliği görevini icra etmektedir. Evli ve bir çocuk babası olan Ömer ÇETİN, 2008-2013 yılları arasında Hava Harp Okulu Bilgisayar Mühendisliği Bölümünde çeşitli lisans dersleri vermiştir.

Yayın ve Patent Listesi:

1. Kurnaz S., Cetin O., Ince F.; "Yazılım Mühendisliğinde Kalite ve UML, (UML and Quality Issues on Software Engineering)", Journal of Turkish Air Force Academy Aeronautics and Space Technologies Institute, Volume 1, Issue 2, Pages 1-12, 2003.

2. Cetin O., Kaplan M. C., Aydın A.; "Bulanık Mantık Tabanlı Yaklaşım ile Otonom İnsansız Hava Araçları için Yer Kontrol Sistemi Simülatör Dizaynı" USMOS 2009, 3. Ulusal Savunma Uygulamaları Modelleme ve Simülasyon Konferansı, 17-18 June, ODTÜ-Kültür Kongre Merkezi-ANKARA, 2009.

3. Kurnaz S., Cetin O., Kaynak O.; "Fuzzy Logic Based Approach to Design Flight Control and Navigation Tasks for Autonomous UAVs", Journal of Intelligent and Robotic Systems, Volume 54, Numbers 1-3, Pages 229-244, 2009.

4. Cetin O., Ozmen B., Kurnaz S., Temeltas H.; "Potential Field Based Navigation Task for Autonomous Flight Control of Unmanned Aerial Vehicles" UAV'09, 2nd International Symposium on Unmanned Aerial Vehicles, June 8-10, Nevada, USA, 2009.

5. Kurnaz S., Cetin O., Kaynak O.; "Fuzzy Logic Based Approach to Design of Flight Control and Navigation Tasks for Autonomous Unmanned Aerial Vehicles", Unmanned Aircraft Systems, Chapter 8, Pages 229-244, Springer ISBN 1402091362,9781402091360, 2009.

6. Kurnaz S., Cetin O.; "Autonomous Navigation and Landing Tasks for Fixed Wing Small Unmanned Aerial Vehicles"; Acta Polytechnica Hungrica, Volume 7, Number 1, Pages 87-102, 2010.

7. Kurnaz S., Cetin O., Kaynak O., "Adaptive Neuro-Fuzzy Inference System Based Autonomous Flight Control of Unmanned Air Vehicles", Expert Systems With Applications, Volume 37, Number 2, Pages 1224-1239, 2010.

8. Cetin O., Kurnaz S., Kaynak O.; "Fuzzy Logic Based Approach To Design Of Autonomous Landing System for Unmanned Aerial Vehicles", International Conference on Unmanned Aerial Vehicles, UAV2010, June 21-23, Dubai, UAE, 2010.

9. Cetin O., Kurnaz S., Kaynak O.; "Fuzzy Logic Based Approach to Design of Autonomous Landing System for Unmanned Aerial Vehicles", Journal of Intelligent & Robotic Systems, Volume 61, Numbers 1-4, Pages 239-250, 2011.

10. Cetin O., Kurnaz S., Kaynak O., Temeltas H.; "Potential Field-Based Navigation Task for Autonomous Flight Control Of Unmanned Aerial Vehicles"; International Journal of Automation and Control (IJAAC); Volume 5, Numbers 1, Pages 1-21, 2011.

11. Cetin O., Zagli I.; "Continuous Airborne Communication Relay Approach Using Unmanned Aerial Vehicles", International Conference on Unmanned Aerial Vehicles, UAV2011, June 12-15, Denver, USA, 2011.

12. Cetin O., Zagli I.; "Continuous Airborne Communication Relay Approach Using Unmanned Aerial Vehicles", Journal of Intelligent & Robotic Systems, Volume 65, Numbers 1-4, Pages 549-562, 2012.

13. Omer Cetin, Ibrahim Zagli and Guray Yilmaz; "Obstacle Free Airborne Communication Relay Approach by Using Autonomous Unmanned Aerial

Vehicles", International Conference on Unmanned Aerial Vehicles,UAV2012, June 12-15, Philadelphia, USA, 2012.

14. Bayraktar O., Özdemir F., Çetin Ö., Yılmaz G.; "İnsansız Hava Araçları İçin Otonom İniş Sistemi Simülatörü Tasarımı, (Autonomous Landing System Simulator Design for Unmanned Aerial Vehicles) ", Gazi Üniversitesi Bilişim Teknolojileri Dergisi, Volume 5, Issue 2, Pages 1-8, May 2012.

15. Cetin O., Zagli I.; "Continuous Airborne Communication Relay Approach Using Unmanned Aerial Vehicles", Recent Developments in Unmanned Aircraft Systems, Chapter 38, Springer, ISBN 978-94-007-3032-8, 2012.

16. Cetin O., Yilmaz G.; "GPGPU Accelerated Potential Field Based Autonomous Air Refueling Approach for UAVs", International Conference on Unmanned Aerial Vehicles, ICUAS13, May 28-31, Atlanta, USA, 2013.

17. Cetin O., Zagli I., Yilmaz G.; "Establishing Obstacle and Collision Free Communication Relay for UAVs with Artificial Potential Fields", Journal of Intelligent & Robotic Systems, Volume 69, Issue 1, Page 361-372, 2013.

18. Cetin O., Yilmaz G.; "Sigmoid Limiting Functions and Potential Field Based Autonomous Air Refueling Path Planning for UAVs"; Journal of Intelligent & Robotic Systems, Volume 73, Issue 1-4, Page 797-810, January 2014.

19. Cetin O., Yilmaz G.; "GPGPU Accelerated Real-Time Potential Field Based Formation Control for Unmanned Aerial Vehicles", International Conference on Unmanned Aerial Vehicles, ICUAS14, pp.103-114, May 27-30, Orlando-FL, USA, 2014.

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- Cetin O., Yilmaz G., 2013: GPGPU Accelerated Potential Field Based Autonomous Air Refueling Approach for UAVs. *International Conference on Unmanned Aerial Vehicles, ICUAS13*, Mayıs 28-31, 2013 Atlanta, ABD.

- Cetin O., Yilmaz G., 2014: Sigmoid Limiting Functions and Potential Field Based Autonomous Air Refueling Path Planning for UAVs. *Journal of Intelligent & Robotic Systems*, Volume 73, Sayı 1-4, Sayfa 797-810.

- Cetin O., Yilmaz G., 2014: GPGPU Accelerated Real-Time Potential Field Based Formation Control for Unmanned Aerial Vehicles. *International Conference on Unmanned Aerial Vehicles, ICUAS14*, Mayıs 27-30, 2014 Orlando-FL, ABD.

- Cetin O., Yilmaz G., 2015: Real-Time Autonomous UAV Formation Flight with Collision and Obstacle Avoidance in Unknown Environment, *Journal of Intelligent & Robotic Systems*, Aralık 2014 tarihinde gönderildi, kabul edildi, henüz basılmadı.

- Cetin O., Yilmaz G., 2015: Artificial Potential Field Based Autonomous Ring Formation Guidance for a Planar Constellation of Satellites, *Recent Advances in Space Technologies, RAST 2015*, kabül edildi, henüz sunulmadı, 16-19 Haziran 2015, Istanbul, Türkiye.

- TÜBİTAK 1001 - Bilimsel ve Teknolojik Araştırma Projelerini Destekleme Programı, proje no: 113E281, "Grup Halindeki Kooperatif İnsansız Hava Araçları için Grafik İşlemcilerin Genel Amaçlı Programlanması Tabanlı Çarpışma Önleme ve Fiziki Engellerden Sakınma Sistemi Tasarımı ve Geliştirilmesi", iki ara ve bir final rapor hazırlanarak gönderildi ve proje çalışması başarı ile tamamlandı, Nisan 2013 – Nisan 2015.