ÇOKLU KAMERA ORTAMINDA NESNE TAKİBİ

Emrah ÜSTÜN
19 14 02 202

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Programı
Danışman: Prof. Dr. Emin Murat ESİN

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Şubat, 2022

ÇOKLU KAMERA ORTAMINDA NESNE TAKİBİ

Emrah ÜSTÜN
19 14 02 202
Orcid: 0000-0001-6580-1296

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Programı
Danışman: Prof. Dr. Emin Murat ESİN

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Şubat, 2022

JÜRİ VE ENSTİTÜ ONAYI

Bu belge, Yükseköğretim Kurulu tarafından 19.01.2021 tarihli *"Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"* ile bildirilen 6689 Sayılı Kişisel Verilerin Korunması Kanunu kapsamında gizlenmiştir.

ETİK İLKE VE KURALLARA UYUM BEYANI

Bu belge, Yükseköğretim Kurulu tarafından 19.01.2021 tarihli *"Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"* ile bildirilen 6689 Sayılı Kişisel Verilerin Korunması Kanunu kapsamında gizlenmiştir.

TEŞEKKÜR

Çalışmalarım süresince hiçbir desteğini esirgemeden yanımda olan aileme, fikirlerinden, bilim insanı kişiliğinden ve insaniyetinden çok şey öğrendiğim değerli hocam Prof. Dr. Emin Murat ESİN'e sonsuz teşekkürlerimi sunarım.

Emrah ÜSTÜN

Şubat, 2022

ÖZ

ÇOKLU KAMERA ORTAMINDA NESNE TAKİBİ

Emrah ÜSTÜN
Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Programı
Danışman: Prof. Dr. Emin Murat ESİN
Maltepe Üniversitesi Lisansüstü Eğitim Enstitüsü, 2022

Nesne tespiti, nesne takibi ve kameralar arası eşleştirme, video gözetimi ve endüstri 4.0 uygulamaları için bir gerekliliktir. Çoklu kamera ortamlarında her bir kameranın görüş açısı farklıdır. Bazı durumlarda kameraların görüş açıları kesişmeyebilir, alanda kör noktalar kalabilir. Bu tür durumlar, takip edilen nesnelerin kaybolmasına ve tekrar tespit edildiğinde farklı bir nesne olarak algılanmasına neden olacaktır.

Bu çalışmada; bir nesnenin herhangi bir kameranın görüş alanında belirlendiği andan itibaren hem kamera görüntüsü içinde, hem farklı kameralar arasında takip edilmesini sağlayan bir metot sunulmuştur. Bu metot, nesne bir kamera görüşünden çıkarken hangi kamera görüşüne doğru ilerlediğini açısal olarak kestirerek, nesnenin kameralar arası takip edilebilmesini sağlamaktadır.

Yapay sinir ağları gibi yüksek maliyetli tespit ve görsel eşleştirme algoritmalarının aksine, düşük maliyetli optimize işlemler ile gerçeklenebilen bir takip algoritmasıdır.

Anahtar Sözcükler: Kamera, Nesne Takibi, Video İzleme, Bilgisayarlı Görü.

ABSTRACT

OBJECT TRACKING IN MULTI CAMERA ENVIRONMENT

Emrah ÜSTÜN
Master Thesis
Graduate School
Computer Engineering Programme
Thesis Advisor: Prof. Dr. Emin Murat ESİN
Maltepe University Graduate School, 2022

Object detection, object tracking and multi-camera mapping is a requirement for video surveillance and industry 4.0 applications. In multi-camera environments, each camera has a different field of view. In some cases, the viewing angles of the cameras may not intersect, and blind spots may remain in the area. Such situations will cause the tracked objects to disappear and be tracked as a different object when detected again.

In this study; a method is presented that allows an object to be followed both within the camera image and between different cameras from the moment it is detected in the field of view of any camera. This method ensures that the object can be tracked between cameras by angularly estimating which camera view it is moving towards as it exits a camera view.

Unlike high-cost detection and visual matching algorithms such as artificial neural networks, it is a tracking algorithm that can be implemented with low-cost optimized processes.

Keywords: Camera, Object Tracking, Video Surveillance, Computer Vision

İÇİNDEKİLER

JÜRİ VE ENSTİTÜ ONAYI ii
ETİK İLKE VE KURALLARA UYUM BEYANI iii
TEŞEKKÜR iv
ÖZ v
ABSTRACT vi
İÇİNDEKİLER vii
TABLOLAR LİSTESİ ix
ŞEKİLLER LİSTESİ x
KISALTMALAR LİSTESİ xi
ÖZGEÇMİŞ xii
BÖLÜM 1. GİRİŞ 1
1.1. Problem 2
1.2. Amaç 4
1.3. Önem 5
1.4. Mevcut Yaklaşımlar 6
BÖLÜM 2. YÖNTEM 7
2.1. Ortak Düzlemin Oluşturulması 8
2.1.1. Kamera Düzlem Geometrisi 8
2.1.1.1. Lambert Yüzeyi 9
2.1.1.2. Uzay Açı Dönüşümü 10
2.1.1.3. Görüntü Koordinat Sistemi 10
2.1.1.4. Çoklu Kamera Ortamında Koordinat Düzlemi 13
2.1.2. Perspektif Dönüşümü 14
2.1.3. Global Harita 17
2.1.4. İzdüşüm Hesaplama 19
2.2. Tespit 20
2.3. Takip 22
2.3.1. Nesne Özniteliklerinin Eşleştirilmesi 23
2.3.2. Arama Bölgesi 29
2.3.3. Kameralar Arası Geçiş 32
2.3.4. Nesnenin Kaybolması 36

BÖLÜM 3. TEST VE UYGULAMA 37
3.1. Uygulama 37
3.2. İşlem Yükü İyileştirmesi 39
3.3. Test Kurulumu 40
3.4. Elde Edilen Sonuçlar 41
3.4.1. Başarı testleri 41
BÖLÜM 4. SONUÇ 47
KAYNAKÇA 49

TABLOLAR LİSTESİ

Tablo 2.1. Şablon Arama Aşamasında Korelasyon Katsayısı Değerlerinin Görselleştirilmesi Örneği 1 25
Tablo 2.2. Şablon Arama Aşamasında Korelasyon Katsayısı Değerlerinin Görselleştirilmesi Örneği 2 26
Tablo 3.1. Uygulamada kullanılan yazılım bileşenleri 38
Tablo 3.2. Bir Nesne Takip Örneğinin Tüm Aşamaları 42

ŞEKİLLER LİSTESİ

Şekil 2.1. Gerçek dünya koordinat sistemi ve kamera koordinat sistemi ilişkisi 8
Şekil 2.2. Lambert Yüzeyi Yansıması 9
Şekil 2.3. İğne deliği (pinhole) kamerada P noktasının görüntü (sensör) düzlemine izdüşümü 11
Şekil 2.4. 3 Boyutlu Nesnenin Kamera Sensörü Üzerinde Oluşumu 12
Şekil 2.5. İğne deliği (pinhole) kamerada koordinat düzlemi piksel dönüşümü 13
Şekil 2.6. Kesişim Alanı Bulunan İki Kameranın Tek Bir Düzlemde Gösterimi..... 14
Şekil 2.7. Kamera Açısına Göre Perspektif Etkisi 15
Şekil 2.8. Yüksek Açı İle Kurulmuş Kamera Düzeneği Örneği 16
Şekil 2.9. Perspektif Etkisi..... 16
Şekil 2.10. Test Ortamının Üçüncü Bir Kameradan Geniş Açılı Alınmış Fotoğrafı..... 18
Şekil 2.11. Test Ortamından Alınan Kamera Görüntüleri 18
Şekil 2.12. Elde Edilen Kuş Bakışı Global Harita 19
Şekil 2.13. Canny Kenar Tespit Karşılaştırmaları (Farklı Eşik Değerleriyle)..... 21
Şekil 2.14. Watershed Segmentasyon..... 22
Şekil 2.15. Şablon Eşleştirme Geometrisi 23
Şekil 2.16. Seçilen Bir Nesne İçin 110 Kare Boyunca Elde Edilen MKKD 27
Şekil 2.17. LPF Uygulanmış MKKD Ve Güvenilirlik Limitleri 28
Şekil 2.18. MKKD Güvenilirliğine Göre Yeni Konum Hesabı..... 29
Şekil 2.19. Hızı Sıfır Olan [t-1]. Karedeki Nesnenin [t]. Karedeki Arama Bölgesi 31
Şekil 2.20. Nesne Hızına Göre Arama Penceresinin Konumlandırılması 32
Şekil 2.21. Arama Penceresinin Kameralar Arası Geçişi 33
Şekil 2.22. Kameralar Arası Geçiş Anı..... 34
Şekil 2.23. Kesişimi Olmayan Kamera Kenar Doğru Parçası 35
Şekil 3.1. İşlem Akış Diyagramı..... 37
Şekil 3.2. Çoklu Kamera Ortamında Nesne Takibi Yazılım Uygulama Mimarisi 39
Şekil 3.3. Test Düzeneği Örneği..... 40
Şekil 3.4. Test Düzeneğinden Elde Edilen Kamera Görüntüleri Ve Oluşturulan Kuş Bakışı Global Harita 41
Şekil 3.5. Test Düzeneğinde Elde Edilmiş MKKD Grafik Örneği..... 46

KISALTMALAR LİSTESİ

TP : True Positive (Doğru Pozitif)

FP : False Positive (Yanlış Pozitif)

Kare : Tek bir görüntü, fotoğraf (Frame)

LTSM : Uzun/Kısa Vadeli Bellek (Long/Short Term Memory)

MKKD : Maksimum Korelasyon Katsayısı Değeri

FPS : Saniyedeki Kare Sayısı (Frame Per Second)

GPU : Grafik İşlemci Ünitesi (Graphical Processing Unit)

TPU : Tensor İşlemci Ünitesi (Tensor Processing Unit)

CSV : Virgülle Ayrılan Değerler Dosyası (Comma Separated Values)

ÖZGEÇMİŞ

Emrah ÜSTÜN

Bilgisayar Mühendisliği Tezli Yüksek Lisans Öğrencisi

Eğitim

Özel Öğrenci	2012	Orta Doğu Teknik Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği
Lisans	2009	Kocaeli Üniversitesi Mühendislik Fakültesi Elektronik ve Haberleşme Mühendisliği
Lise	2002	Ankara Atatürk Anadolu Lisesi

İş/İstihdam

2019-Halen	CV/AI Birim Yöneticisi - Trio Mobil
2012-2019	Kurucu Ortak, CTO - a2 Teknoloji
2010-2012	Yazılım Takım Yöneticisi - VCount Teknoloji
2009-2010	Yazılım Mühendisi - Pars Ar-Ge

BÖLÜM 1. GİRİŞ

Video gözetim ve endüstriyel bilgisayarlı görü uygulamalarında, nesneler görüntü işleme veya makine öğrenmesi yöntemleri ile algılanır. Nesne sayma veya nesne sınıflandırma problemleri için tek bir kare üzerinde algılama ve işleme yeterli bir çözüm olabilir. Ancak video gözetimi ve endüstriyel bilgisayarlı görü uygulamalarında, ilgilenilen nesnenin mevcut konumunun belirlenmesi, zaman düzleminde hareket yörüngesinin izlenmesi gerekecektir.

Evrişimli Sinir Ağı (Convolutional Neural Networks - CNN) mimarisine sahip yapay sinir ağları veya SIFT, SURF, ORB gibi öznitelik çıkarımına dayalı yüksek doğrulukta nesne tespit ve eşleştirme çözümü sağlayabilen metotlar bulunmaktadır. Ancak bu metotlar oldukça yüksek işlem yüküne ihtiyaç duydukları için kullanım alanları dardır. Bu çalışmada yüksek maliyetli yöntemlerden ziyade, nesnenin olası konumunda arama yaparak kareler arası eşleştirme üzerine çalışılmıştır. Bu yönüyle, düşük maliyet odaklı mimariler üzerinde nesne tespit ve kameralar arası nesne takip çözümü olarak planlanmıştır.

Uzun vadede nesne hareketinin izlenmesi, hareket modellerinin hesaplanabilmesi için de gerekli olduğundan, nesne takibi bazı durumlarda hareket tahmini problemlerinde kullanılabilir. Genel kabul görmüş bir takip metodu olan Şablon Eşleştirme (template matching), düzenli aydınlatmaya sahip alanlarda nesneleri başarılı şekilde takip edebilir [1].

Sahnenin [t] numaralı karesinde tespit edilen nesne için zaman düzleminde takip, nesneyi [t+1] numaralı karede aramak suretiyle gerçekleştirilebilir. Bir nesne, bir karede algılanıp bir kimliğe sahip olduktan sonra, daha sonraki karelerde de algılanarak takip edilmesi gerekir. Yani nesnenin yörüngesi elde edilmelidir. Bu bilgiler ayrıca sayma, anormallik tespiti veya güvenlik amaçlı uygulamalarda da kullanılabilir.

Bir kamera görüntüsü içinde nesne takibi gürbüz yöntemlerle verimli bir şekilde çözülebilmiş bir sorundur. Ancak bir kameranın çok sınırlı bir görüş alanı vardır, bu nedenle bir kameranın kurulu olduğu video analiz uygulamaları çok nadirdir. Büyük tesislerde binlerce kamera olabilir. Bu yönden gerçek dünyadaki bir sorunun çözümünde kameraların kullanılabilmesi için kameralar arasındaki geçişlerin takip edilmesi esastır.

Kameralar arasında nesne takip edebilmek için görüntünün kamera sensörü üzerinde nasıl oluştuğunun, gerçek dünya koordinat düzlemi ile kamera koordinat düzlemi arasındaki geçişte ne gibi dönüşümlere ihtiyaç bulunduğunun incelenmesi gerekmektedir.

Kameralarda görüntü, ışığın lens aracılığıyla sensör üzerine doğru açıyla düşürülmesi sonucunda oluşur. Yani her bir piksel koordinatının gerçek dünyadaki karşılığı hesaplanabilir. Bu prensiple kameralar belli bir sapmayla algılama ve ölçme amacıyla kullanılabilmektedir.

Gerçek dünya bilgisi 2 boyutlu kamera sensörlerinde dijitalleştirildiğinde 1 boyut bilgisi kaybolmuş olur. Yani perspektif etkisi oluşmuş olur. Kameranın düz bir zemine baktığı varsayımı yapıldığında, kameranın zemine olan açısı ve ışığın kırılımındaki bozucu diğer parametreler de bilinirse kaybolan 3. boyut bilgisi geri elde edilebilir. Bu da perspektif etkisinin yaklaşık olarak ortadan kaldırılması anlamına gelmektedir. Bu sayede kameradaki bir pikselin gerçek dünyadaki konumu hesaplanabilir.

Piksellerin gerçek dünya konumlarının bilinmesi durumunda, farklı kameraların görüntüleri arasındaki piksel değerleriyle yapılabilecek dijital bir eşleştirme, gerçek dünya konum eşleştirmesi anlamına gelmiş olacaktır. Bu sayede görüntüde tespit edilen nesne veya insanları eşleştirebileceğimiz gibi, kamera konumları da hesaplanabilir, birden fazla kamera görüntüsünün aynı koordinat düzleminde izdüşümleri kestirilebilir.

1.1. Problem

Güvenlik, istatistik, endüstriyel bilgisayarlı görü uygulamaları için nesne ve insanların izlenmesi amacıyla kullanılan kameralar genelde yalnızca video kaydı almakta veya basit video analitik işlemler yapmaktadırlar. Herhangi bir durumda kayıtlar geriye yönelik incelenmektedir. Canlı izleme ise personel tarafından gözle yapılmakta, kamera sayısı arttıkça ve personel yoruldukça yapılan incelemenin doğruluk payı azalmaktadır.

Son yıllarda geliştirilen nesne takip algoritmaları ve bunları çalıştırabilecek güçlü donanımlar sayesinde bilgisayarlı görü yöntemleri bu problemin çözümü için katkı sağlamıştır. Ancak görüntü işleme veya yapay sinir ağları metotlarıyla geliştirilen bu algoritmaların makul işlem yüküne sahip olanları genelde nesne tespitini tek bir kare üzerinden anlık olarak yapmaktadır. Bu tespit algoritmaları çok faydalı görünse de, tek kare üzerinden nesne tespiti yapılabilmesi sebebiyle bu algoritmalar gerçek uygulamalar

için yeterli değildir. Nesne hareket ettikçe izlediği yolun belirlenmesi ve bir kamera alanından çıktığında başka bir işleme gerek olmadan geçiş yapılan alandaki diğer kameralardan takibe devam edilmesi gerekmektedir.

Bu amaçla kullanılacak olan sistemde;

- kamera görüş alanlarından her birinde nesne takibi,
- nesne alan dışına çıktığında hangi kamera alanına girmiş olabileceğinin kestirimi,
- bu yolla alandan alana geçişlerde nesne izlemenin sürekli yapılabilmesi,
- zamana bağlı olarak bulunduğu noktaların ve dolayısıyla izlediği yolun belirlenmesi

görevlerini yerine getirecek modüllerin bulunması gerekmektedir.

Çok kameralı ortamlarda kameralar arasında nesneleri takip edebilmek için bir nesnenin başka bir kameranın görüş alanına nereden geçtiğini bilmek gerekir. Ancak kameraların birbirlerine göre konumsal/açısal yerleşimleri ve izdüşümleri arasındaki ilişki bilinmediği için kameralar arasında piksel tabanlı bir haritalama yapılamayacaktır. Kameralar yan yana olsa bile, farklı görüş açılarına sahip oldukları için oluşturdukları görüntüler bitişik değildir. Kamera görüntüleri arasında bir kesişme bölgesi olmayabilir, yani kör nokta oluşabilir. Tek bir kameranın kendi görüntüsü içinde de kör noktalar oluşabilir.

Bir nesne kamera görüntüsünden çıkıp bir süre kör noktada kaldığında ve ardından herhangi bir noktadan görüntüye yeniden girdiğinde, yeni tespit edilen nesnenin daha önce takip edilen nesne olup olmadığına karar vermek için bir eşleştirme yöntemine ihtiyaç vardır.

Video analitiği yöntemleriyle nesne tespit ve eşleştirme, kayan pencerelerle nesne sınıflandırma döngüleri sonucunda üretilen bilgilerin derlenmesiyle yapılabilmektedir [2]. Sınıflandırma (ve benzerlik bulma) için yüksek sayıda döngülerin tekrar tekrar uygulanması, işlem yükü çok yüksek olan başlı başına zor ve maliyetli bir süreçtir. Nesnenin konumunun bulunduktan sonra takip işlemi için, kareler arasında görsel eşleştirme yaparak zaman düzleminde nesne takibi sağlayabilen yöntemler de bulunmaktadır [3]. Ancak, tek bir karede nesne tespiti dahi oldukça fazla işlem gerektiren bir süreçken, takip için kareler arası eşleştirme amacıyla görsel arama metotlarını

uygulayan bu algoritmaların kullanılması işlem yükünü daha da artıracaktır. Çok yüksek işlem gücü gerektiren uygulamaların maliyetleri arttıkça kullanılabilirlikleri azalır.

Nesne algılama yöntemlerinin yüksek maliyetleri sebebiyle optimizasyon ihtiyaçları bulunmaktadır. Yapılan optimizasyonlar nedeniyle performans düşebilir. Optimizasyonlar veya diğer çevresel faktörler nedeniyle bir nesne her karede tespit edilemeyebilir. Yani nesne sürekli görüntü içinde olsa dahi algılanmasında kesintiler olabilir.

Nesne algılama (veya eşleştirme) kesintileri, tek bir kameradaki kör noktalar ve kameralar arasındaki kör noktalar göz önüne alındığında, takip algoritmasının zaman düzlemindeki eşleştirme aşamasında, mesafe değişiklikleri veya anlık kaybolmalar için belirli bir toleransı bulunması gerekir.

Çok kameralı nesne takibindeki bir diğer önemli sorun, farklı kameraların koordinat düzlem dönüşümlerinin (izdüşüm hesaplarının) farklı olması sebebiyle doğrudan eşleştirme yapılamamasıdır. Yani gerçek dünyada aralarında a birim mesafe bulunan iki nesnenin, birinci kamerada aralarında a_1 birim, ikinci kamerada a_2 birim mesafe bulunacaktır. a, $a_1$ve a_2 arasındaki ilişki bilinmediği için doğrudan bir eşleştirme ve takip yapılamayacaktır. Bu çalışmada bu ilişkinin kurulabilmesi için bir yöntem üzerine çalışılmıştır.

1.2. Amaç

Bu çalışmada nesnelerin kamera(lar) üzerinden konum-zaman ilişkisi içinde takip edilmesi amaçlanmaktadır. Karelerden yapılacak tespit ve konum eşleştirmeleri suretiyle konum-zaman bilgisi elde edilebilecektir. Bu bilgi 2 boyutlu bir mekan düzleminde hareket izi olarak görüntülenebilecektir.

Tespit edilen nesneyi takip eden algoritma; nesnenin anlık olarak görünmeme, çok hızlı hareket etme, anlık tespit edilememe, kör noktada bulunma veya başka kameranın görüş açısına geçme durumlarına karşı toleranslı olarak planlanmıştır. Takip algoritması, nesnenin kameraya göre konumunu gerçek dünya izdüşümü ile eşleştirerek takip edecektir. Takip edilen nesne, farklı kameranın görüş alanına geçiş yapsa bile nesne takibi devamı mümkün olacaktır.

Takip algoritması, rota tahmini de yaparak anlık tespit hatalarının olumsuz etkilerini azaltacak ve kör noktada bulunan nesnenin de takip edilmesini sağlayacaktır.

Bu çalışma kapsamında düşük hızlı 1 adet nesnenin takibi ele alınacaktır. Bazı endüstriyel uygulamalarda (özellikle üretim hattı takibi), yüksek hızda hareket eden çok sayıda nesneyi takip etmek gerekebilir. Bu durumda kameralardan alınan karelerin zaman senkronizasyonu da ayrı bir problem olarak karşımıza çıkacaktır. Bir karedeki her piksel değerinin donanım üzerinden senkron olarak okunması gibi bir sorun varken, iki kameradan senkron kare alma problemi çözülmesi daha zor bir sorundur.

1.3. Önem

Günümüzde yapay zeka, bilgisayarlı görü veya sensör tabanlı tespit sistemlerinin pratik hayatta tamamen otomatize olarak kullanımında eksikler bulunmaktadır. Karelerden elde edilen konum bilgisi, kameranın görüntü düzlemi üzerinde, kendi koordinat sistemindedir. Buna karşılık kameranın, hareketin gerçekleştiği gerçek dünya düzlemindeki konumları ifade edebilmesi için görüntünün uygun bir dönüşüm işleminden geçirilmesi gerekir.

Bilgisayarlı görü sistemlerinin ürettiği veriler genellikle anlık veya kısa vadeli tespitlere dayanmaktadır. Yapay sinir ağı metotları kullanarak Uzun / Kısa Süreli Bellek (Long / Short Term Memory - LSTM) zaman bazlı çıktılar üretilmesi, gerçek dünya verilerine daha yakın sonuçlar alınmasını sağlayabilir. Ancak bu gibi yöntemler, yüksek hafıza ve yüksek işlem gücü gerektirir. Önerilen yöntem ile nesnenin sadece özniteliklerinin hafızada tutulması yeterli olacağından, uzun vadede işlenen ve kayıtta tutulan veri miktarı da azaltılmış olacaktır.

Bu çalışmada, takip edilecek olan nesnenin kamera(lar)daki öznitelikleri incelenerek, bilgisayarlı görü metotlarının tahmin çıktıları birleştirilecek ve sonuç olarak "nesne hareketi" çıktısı oluşturulacaktır. Nesne tespit ve eşleştirme algoritmalarının ürettiği anlık veri, "nesne hareketi" verisine dönüştürülerek, analiz ve istatistik sistemlerinde kullanılabilecek faydalı veri üretilmiş olacaktır.

1.4. Mevcut Yaklaşımlar

Literatürde birçok nesne takip metodu bulunmaktadır. Bunların içinde yüksek başarılı ancak yüksek maliyetli yapay sinir ağı kullanan karmaşık algoritmalar mevcuttur. Aynı zamanda piksel operasyonu seviyesinde görsel eşleştirme yapan, başarısı ve

maliyeti orta seviye sayılabilecek metotlar bulunmaktadır. Bazı düşük başarılı metotlar ise tespit doğruluğuna güvenerek, takip karmaşıklığını ve dolayısıyla maliyeti azaltır.

Kamera görüntüsünde kör nokta olmaması veya dar olması durumunda; tek bir nesne takip ediliyorsa, takip edilen nesne en yakın nesne ile konum bazında doğrudan merkez takibi ile eşleştirerek takip edilebilir [4].

Büyük bir kör nokta ve/veya aralıklı algılama durumunda, görüntüde çok fazla nesne yoksa, Kalman Filtresi ile izleme, açıyı tahmin ederek eşleştirmeyi kolaylaştıran, kabul görmüş gürbüz bir yöntemdir [5].

Tek kamera içi takip (Single Camera Tracking - SCT) ve kameralar arası takibi (Inter-Camera Tracking – ICT) ayrı ayrı değerlendirerek renk spektrumu analiziyle öznitelik eşleştirmesi yapan PMCSHR isimli bir metot önerilmiştir [6]. Kameraların ortak koordinat sistemine dönüşüm parametrelerini otomatik olarak hesaplamayı hedefleyen çalışmalar da bulunmaktadır [7].

BÖLÜM 2. YÖNTEM

Bu çalışmada nesne sınıflandırma veya tüm görüntüde nesne arama üzerine bir konuda çalışılmamıştır. Kullanıcı (veya harici bir sistem) tarafından belirlenen bir noktada bulunan nesnenin tespiti yapılarak, bu nesnenin hareketinin kameralar arası takibi yapılmıştır.

Herhangi bir kamera üzerinden, sisteme giriş bilgisi olarak bir piksel koordinat bilgisi alındıktan sonra;

- bu koordinatın etrafı taranarak seçilen nesnenin sınırları belirlenmiş,
- nesnenin kamera koordinat düzlemine göre konumu belirlenmiş,
- bu konum, bir referans noktaya göre gerçek dünya koordinatlarına dönüştürülmüş,
- zaman düzleminde nesnenin bulunduğu konumlar takip edilmiş,
- nesnenin görüş alanında bulunduğu kameranın görüntüsünden çıkması durumunda, hangi kameraya geçiş yaptığı kestirilmiş,
- yeni kamera alanında aynı işlemlere devam edilmiştir.

Bu aşamaları tamamlayabilmek için gerekli yazılım modülleri geliştirilmiş ve USB (Universal Serial Bus) haberleşmesi yapan kameralar ile düzenek hazırlanarak test edilmiştir.

Hazırlanan modüller entegre edilerek birden fazla kameranın varlığında bu kameraların gördüğü alanlarda hareket eden nesnenin izlediği yolun takip edilebildiği görülmüştür.

Bu çalışmada izlenen yöntemde, kamera görüş alanı sınırlarının gerçek dünyada hangi konuma denk geldiğinin ve kameraların birbirine göre konumlarının mesafe değerleriyle biliniyor olması gerekmektedir.

İhtiyaç duyulan temel işlem perspektif etkisinin giderilmesi olacaktır. Perspektif etkisinin giderilmesiyle kamera görüntüleri kuş bakışı yerleşime dönüştürülebilir. Bundan sonra dönüşüm matrisleri kullanılarak her bir pikselin gerçek dünyadaki izdüşümü, referans noktaya göre kuş bakışı harita üzerinde hesaplanabiliyor olacaktır.

Böylece tüm kameraların tek bir koordinat düzleminde birleştirildiği, kuş bakışı global harita oluşturulmuş olur.

2.1. Ortak Düzlemin Oluşturulması

Kamera görüntüleri 2 boyutlu matrislerden oluşur. Her kameranın koordinat düzlemi kendine özeldir ve bağımsızdır. Kameralar arası nesne takibi yapılabilmesi için, sisteme dahil olan kameraların görüş açıları arasındaki ilişkinin bilinmesi gerekecektir. Bunun için kamera koordinat düzlemleri, geometrik dönüşümler yapılarak ortak yapıda bir araya getirilmiştir.

Bu çalışmanın ilk kısmı, kamera koordinat düzlemlerinin birbiriyle ilişkisinin bulunarak "Global Harita" ismi verilen ortak koordinat düzlemine indirgenmesini barındırmaktadır.

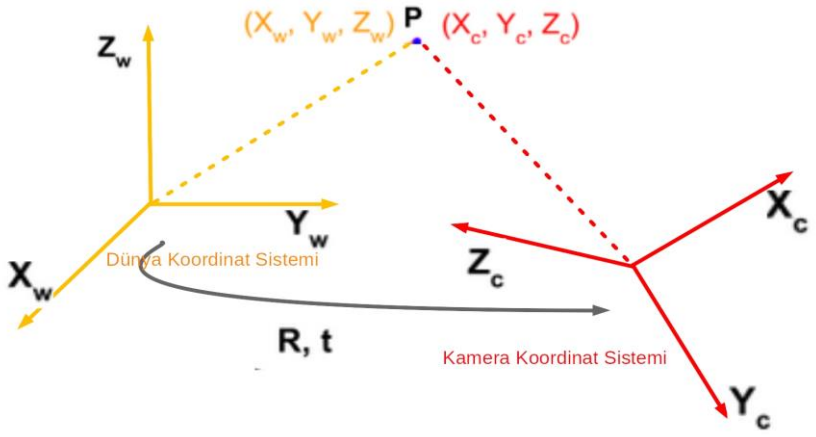
Global haritanın oluşturulabilmesi için;

- kamera açısından kaynaklı olarak oluşan perspektif etkisinin giderilmesi için konum kalibrasyonu,
- kuş bakışı kamera görüntülerinin ortak bir referans noktasına (x=0, y=0) göre dönüşümünün yapıldığı yeni bir ortak koordinat düzlemi oluşturulması

aşamaları uygulanmıştır.

2.1.1. Kamera Düzlem Geometrisi

Şekil 2.1'de sarı renkle gösterilen düzlem gerçek dünya koordinat sistemini, kırmızı renkle gösterilen düzlem kamera koordinat düzlemini göstermektedir. Gerçek dünyadaki bir P noktasının konumu (X_w, Y_w, Z_w) ise, kamera düzleminde (X_c, Y_c, Z_c) olarak tanımlanabilir.



Şekil 2.1. *Gerçek dünya koordinat sistemi ve kamera koordinat sistemi ilişkisi*

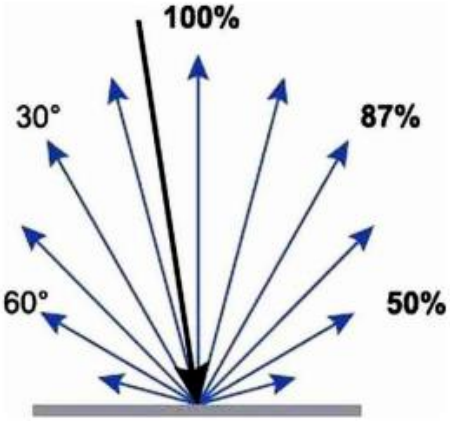
İki koordinat sistemi arasındaki dönüşüm parametrelerini hesaplayabilmek için öncelikle ortak bir referans noktası gerekmektedir. Dünya ve kamera koordinat düzlemlerinde (0,0,0) olarak tanımlayabileceğimiz ortak referans nokta herhangi bir konum seçilebilir.

Dönüşümü yapılmak istenen noktanın referans noktasına olan uzaklığını bularak diğer koordinat düzlemindeki yerini tespit edebiliriz.

P noktasının referans noktaya olan uzaklığı, gerçek dünya üzerinde seçilen referans nokta ve koordinat düzlemi yerleşimine göre ölçülerek (X_w, Y_w, Z_w) elde edilebilir. (Bu çalışmada kalibrasyon aşamasında referans noktasına göre uzaklıklar milimetre cinsinden ölçülmüştür.) Eğer kamerayı (0,0,0) referans noktası üzerinde gerçek dünya koordinat düzlem eksenleriyle aynı eksen üzerine yerleştirebilseydik, dünya koordinat düzlemi ve kamera koordinat düzlemi eşitlenebilirdi. Ancak bu pratikte mümkün değildir. Kamera herhangi bir yerde ve herhangi bir açıda konumlanmış olabilir. Bu durumda dünya ve kamera koordinat düzlemleri arasındaki ilişkiyi bulmak gerekmektedir. Eğer kamera referans (0,0,0) noktasına göre (t_X, t_Y, t_Z) noktasında bulunuyorsa bu değerleri öteleme (translation) değerleri olarak kabul edebiliriz.

2.1.1.1. Lambert Yüzeyi

Görsel bilgi ışık vektörleriyle taşınır. Işık kaynağı olmayan yüzeyler ve cisimler yansıtıcıdır, başka bir ışık kaynağından gelen ışığı yansıtarak etrafa dağıtırlar. Herhangi bir açıdan gelen ışığı, her yöne eşit dağıtan yüzeyler Lambert Yüzeyi olarak adlandırılır [8]. Lambert yüzeyi yansıması Şekil 2.2'de gösterilmiştir.



Şekil 2.2. *Lambert Yüzeyi Yansıması [8]*

Kameralarda ışık bilgisini dijital bilgiye dönüştüren sensörler, ışık enerjisiyle şarj olarak ışığın dalga boyuna göre elektriksel sinyal üretirler. Lambert yüzeylerinden yansıyan ışıklar kamera lensinde kırılarak sensöre düşer. Bu kırılma sayesinde gerçek dünyadaki bir alandan renk örneklemesi, bir pikselin renk bilgisi olarak dijital hale gelmiş olur. Böylece her bir pikselin gerçek dünyadaki yaklaşık konumu hesaplanabilir.

2.1.1.2. Uzay Açı Dönüşümü

Bir referans sıfır noktasına eşit uzaklıktaki iki noktanın arasındaki yay uzunluğunun açısal karşılığı uzay açıdır [9].

Kameranın referans sıfır noktası olarak kabul edildiği bir uzayda, görüntüdeki iki nokta arasındaki uzay açının izdüşümü, iki boyutlu görüntü üzerinde piksel birimiyle ölçülebilen mesafeye denk gelecektir.

Kamera koordinat düzleminin gerçek dünya koordinat düzlemiyle eşleştirilmesi için, yani uzay açı dönüşümünün uygulanabilmesi için üç parametre kullanılır: Sapma (yaw), eğim (pitch) ve dönme (rool) parametreleri. Bu dönüşümün uygulanabilmesi için genellikle dönüşüm/dönme matrisi (rotation matris) kullanılır ve sapma (translation) için bir vektör ile öteleme yapılır.

Şekil 2.1'deki çizime göre dönüşümü yapılması istenen P noktası, referans (0,0,0) noktasına göre gerçek dünyada X_w, Y_w, Z_w konumundaysa ve kamera düzlemine göre de X_c, Y_c, Z_c noktasındaysa, kamera düzlemindeki konumu dönüşüm matrisiyle çarpıp translation vektörü ile ötelersek bu eşitliği sağlamış oluruz.

Dönüşüm işlemi; R dönüşüm (rotation), t sapma (translation) olmak üzere,

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + t \tag{2.1}$$

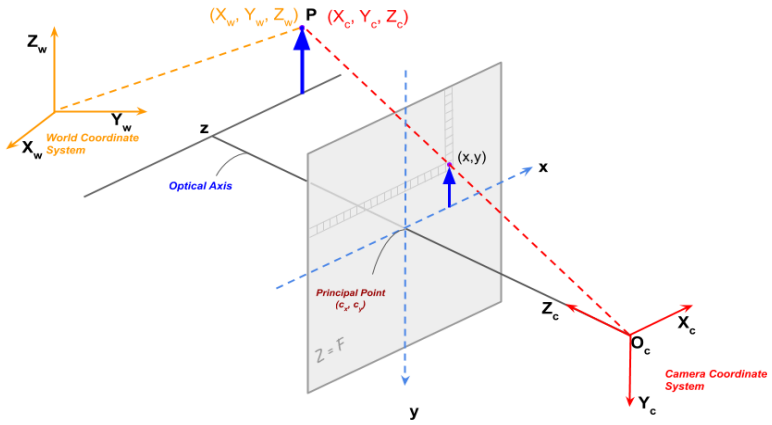
olarak tanımlanmaktadır.

Bu da kamera konumuyla bağıntılı olan dışsal (extrinsic) dönüşüm matrisini oluşturur.

2.1.1.3. Görüntü Koordinat Sistemi

Görüntü koordinat sistemi Şekil 2.3'teki gibi sembolize edilen bir kamera düzleminde, gerçek dünya koordinat düzlemi ile kamera koordinat düzlemi arasında kayma (shift) ve açı (rotation) bulunmaktadır. Görüntü üzerinde elde ettiğimiz veri,

kırmızı renkte sembolize edilen kamera koordinat düzlemindedir. Kamera koordinat düzleminden gerçek dünya koordinat düzlemine dönüşüm yapmak için sarı renkte sembolize edilen düzlemdeki izdüşümünün hesaplanması gerekecektir.



Şekil 2.3. *İğne deliği (pinhole) kamerada P noktasının görüntü (sensör) düzlemine izdüşümü [10]*

Kamera koordinat düzlemi, ışığın sensöre düşme açısı bakımından 3 boyutlu olsa da, bir 3 boyutlu kamera ile (TOF kamera vb.) çalışılmıyorsa görüntü 2 boyutlu oluşmaktadır. Yani veri, 2 boyutlu bir matriste X ve Y düzlemindeki piksellerden oluşan sayısal değerler olarak elde edilmekte, Z düzleminin bilgisi kaybedilmektedir. Oluşan veri, P noktasının sensör düzlemindeki izdüşümüne denk gelen x,y koordinatındadır. f parametresi odak uzaklığı olmak üzere, üçgenlerin benzerliğini kullanarak görüntüde oluşan x,y koordinatı Denklem 2.2 ve Denklem 2.3'te gösterildiği şekilde hesaplanabilir.

$$x = f\frac{X_c}{Z_c} \tag{2.2}$$

$$y = f\frac{Y_c}{Y_c} \tag{2.3}$$

Matris formunda yazılırsa;

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{2.4}$$

şekline gelir ve Denklem 2.5'te gösterilen kameranın iç parametrelerine bağımlı olan içsel (intrinsic) dönüşüm matrisinin elde edilmesini sağlar.

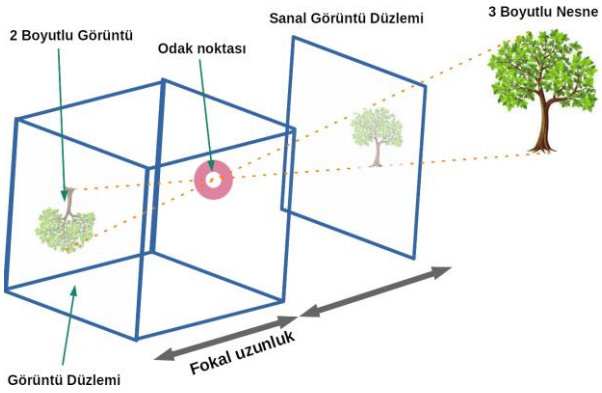
$$K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

Ancak pratikte düşük maliyetli kameralarda hata payları oluşmaktadır. Bu hata paylarından dolayı kaymaların da hesaplanabilmesi için başka parametreler kullanılmaktadır.

Kamera sensörünün kare olmaması durumunda ise odak uzaklığında kayma yaşanacaktır. x-y eksenlerindeki kaymayı f_x ve f_y ile sembolize edebiliriz. Sensör yerleşimindeki çok küçük sapmalardan dolayı optik merkez ile görüntü koordinat merkezi arasında da bir kayma olabilir. Bu sebeple c_x, c_y parametreleriyle bu kayma sembolize edilir. Sensör üretim hatasından dolayı x-y eksenleri arasında bir eğiklik (skew) olabilir, bu eğiklik de γ ile ifade edilir. Bu durumda intrinsic dönüşüm matrisi K Denklem 2.6'daki gibi ifade edilebilir.

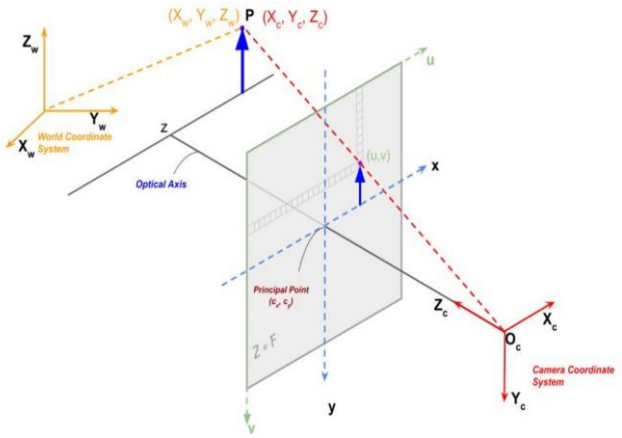
$$K = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Gerçek dünyadaki 3 boyutlu nesne üzerinden ışık yansıması ve bu yansımadan görüntü elde edilirken 3. boyutun kaybedilmesi Şekil 2.4'te gösterilmiştir.



Şekil 2.4. *3 Boyutlu Nesnenin Kamera Sensörü Üzerinde Oluşumu*

Şekil 2.5'te gerçek dünyadaki bir P noktasının konumu (X_w, Y_w, Z_w) olarak tanımlanmıştır. Kamerada oluşan görüntü 2 boyutlu olduğu için P noktası (u,v) konumunda görülmektedir.



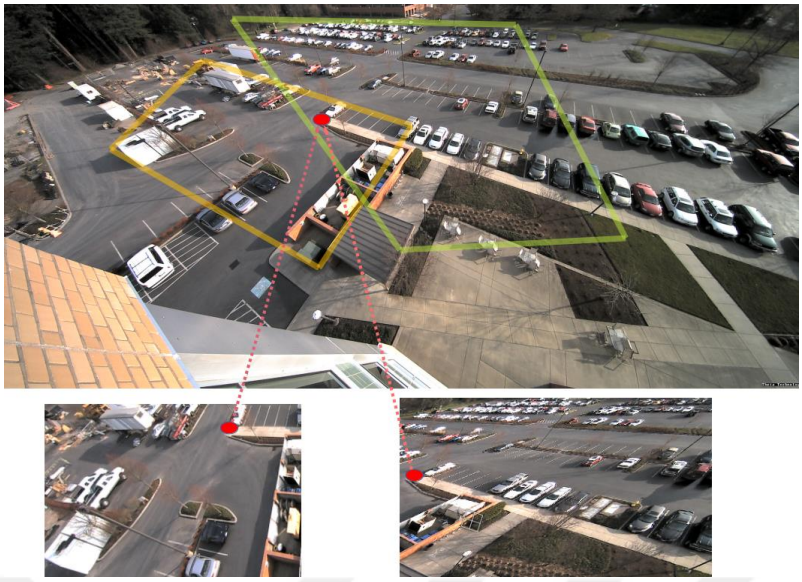
Şekil 2.5. *İğne deliği (pinhole) kamerada koordinat düzlemi piksel dönüşümü [10]*

$u = \frac{u'}{w'}$ ve $v = \frac{v'}{w'}$ olmak üzere, elde edilen intrinsic dönüşüm matrisinin piksel koordinatlarındaki karşılığı da Denklem 2.7'deki gibi ifade edilir.

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{2.7}$$

2.1.1.4. Çoklu Kamera Ortamında Koordinat Düzlemi

Donanımları ve parametreleri birebir aynı olsa dahi, aynı bölgeyi gören iki kameradan alınan görüntüler pratikte eş olamaz. Bu kameralar aynı sahneyi görüyor bile olsa, kameraların kendi konumlarından dolayı farklı uzay-açı hesabıyla ışıkları alacaklardır. Lambert yansıması prensibiyle, gerçek dünyada bir noktadan yansıyan ışık, farklı noktalara farklı açı ile ulaşacaktır. Gerçek dünya düzlemi ve tüm kameraların düzlemleri arasındaki farktan dolayı da oluşan görüntüler birbirinden farklı perspektif etkisine sahip olacaktır. Şekil 2.6'da aynı bölgeyi gören iki kamerada görüntülerin farklı perspektif etkileriyle oluşması sembolize edilmiştir.



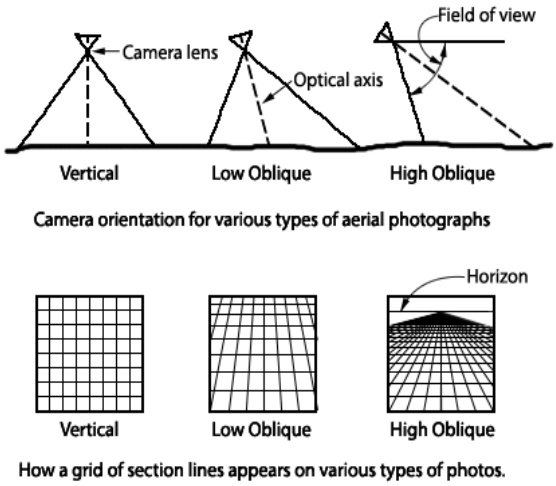
Şekil 2.6. *Kesişim Alanı Bulunan İki Kameranın Tek Bir Düzlemde Gösterimi*

Bu nedenlerle takip edilecek olan nesne; gerçek dünyada $\boldsymbol{\alpha}_0$ açıyla, saniyede d_0 birim mesafe hareket ediyorsa, Kamera-1'in görüntüsünde $\boldsymbol{\alpha}_1$ açıyla saniyede d_1 piksel mesafe hızla hareket ediyor, Kamera-2'nin görüntüsünde $\boldsymbol{\alpha}_2$ açıyla saniyede d_2 piksel mesafe hızla hareket ediyor olacaktır. Bundan dolayı, bir kamerada takibi başlanan nesnenin hareket karakteristiği diğer kamerada değişecektir.

Ortak bir koordinat düzleminin oluşturulması ve takibin bu düzlem üzerinden yapılması, ortak bir açı ve mesafe düzlemi oluşturulabilmesi anlamına gelmektedir. Hesaplanacak dönüşüm parametreleri, $\boldsymbol{\alpha}_1$ ve $\boldsymbol{\alpha}_2$'nin, farklı bir düzlemde Θ_0 açıya eşit olması sağlayacaktır. Bu sayede iki kamerada saniyede alınan d_1 ve d_2 piksel mesafeleri, ortak düzlemde D_0 piksel mesafe olarak eşitlenmiş olacaktır.

2.1.2. Perspektif Dönüşümü

Kamera görüntüsü, 3 boyutlu gerçek dünyadan Lambert etkisiyle yansıyan ışığın lens üzerinden geçip bir sensör üzerinden örneklenmesiyle oluşur. Yani 3 boyutlu bilgi, 2 boyuta indirilmiş olur. Bu da veri kaybı olarak yorumlanabilir. Bu veri kaybı ile görüntüde perspektif etkisi oluşmaktadır. Yüksek eğimli bakışta bu etki daha fazla oluşur.



Şekil 2.7. *Kamera Açısına Göre Perspektif Etkisi [11]*

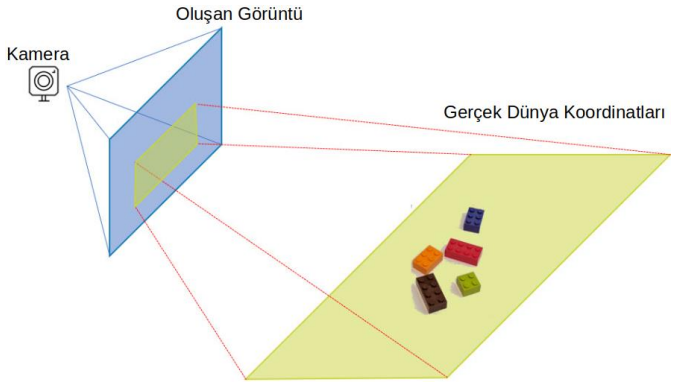
Şekil 2.7'de bir kameranın ızgara desenine bakış açısının değişiminin görüntüdeki etkisi sembolize edilmiştir. Kuş bakışı açısı 0^{0} kabul edilirse, açı arttıkça oluşan perspektif etkisi; görüntü üzerinde üst (uzak) tarafta yüksek, alt (yakın) tarafta düşük bir piksel konum değişimine sebep olmaktadır.

Kameranın düz bir zemine baktığı varsayımıyla, bir dikdörtgenin görüntüsü perspektif etkisiyle ters yamuk şeklinde oluşur. Affine dönüşümü ile bu görüntü kuş bakışı görünümüne dönüştürülebilir. Oluşan yamuğun 4 köşesinin koordinatları alınarak Affine dönüşümü ile dikdörtgenin tekrar elde edilmesi mümkündür.

3 boyutlu affine dönüşümleri bilgisayarlı görü ve tabanlı nesne tanıma alanında yaygın olarak kullanılmaktadır. İki farklı 3 boyutlu görüntü koordinat sistemi arasındaki ilişkiyi tanımlamak için 12 parametreli affine dönüşüm (öteleme, döndürme, her eksen boyunca farklı ölçek faktörü ve eğme) kullanılabilir. 3 boyutlu gerçek dünya düzlemini 2 boyutlu görüntü düzlemine dönüştüren bir modeli tanımlamak için ise 8 parametreli affine dönüşüm (iki öteleme, üç döndürme, iki ölçek faktörü ve görüntü uzayında çarpıklık bozulması) kullanılabilir [12].

Affine dönüşüm ile çapraz bakışlı bir kamera görüntüsü, referans noktalara göre kalibre edilerek kuş bakışı görüntü formatına dönüştürülebilir. Bu sayede, bir görüntü içinde ilgilenilen kısım referans nokta ile belirlenerek, bu referans noktanın (x=0,y=0) koordinatı olduğu varsayılarak işlem yapılabilir. Perspektif etkisinden kurtulmuş olan kuş bakışı görüntü üzerinde de mesafe-piksel değişimleri lineer olacağı için hesaplamalar daha doğru yapılabilecektir.

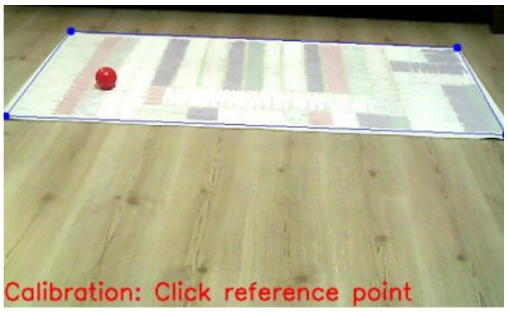
Bu işlemle gerçek dünyada referans (0,0,0) noktası olarak kabul edilen nokta için kuş bakışı görüntü elde edilebilmiş olacaktır [13].

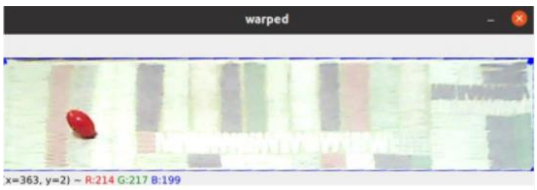


Şekil 2.8. *Yüksek Açı İle Kurulmuş Kamera Düzeneği Örneği*

Bu çalışma kapsamında Şekil 2.8'de görüldüğü gibi bir açı ile zemine bakılarak, alınan görüntü üzerindeki perspektif etkisi incelenmiştir. Bu görüntü geri çatılarak kuş bakışı görüntüye dönüştürülmüştür.

Bu dönüşüm OpenCV kütüphanesi [14] desteğiyle yapılmıştır. getPerspectiveTransform metoduyla dönüşüm matrisleri hesaplanmış, warpPerspective metoduyla kuş bakışı görüntü elde edilmiştir. Her kameranın kendi dönüşüm matrisi hafızada tutularak, nesne takibi aşamasında, gerektiğinde piksel koordinatlarının kamera düzlemi ve kuş bakışı düzlem arasındaki dönüşümü, perspektif dönüşüm metoduyla tekrar elde edilmiştir.





a) Perspektif etkisi yüksek görüntü b) Perspektif etkisi geri çatılmış kuş bakışı görüntü

Şekil 2.9. *Perspektif Etkisi*

Uygulama için Şekil 2.9'daki gibi bir düzenek kurulmuştur. Dört köşesi işaretlenen zemin, perspektif dönüşümü ile şekildeki gibi dikdörtgen halde elde edilebilmektedir.

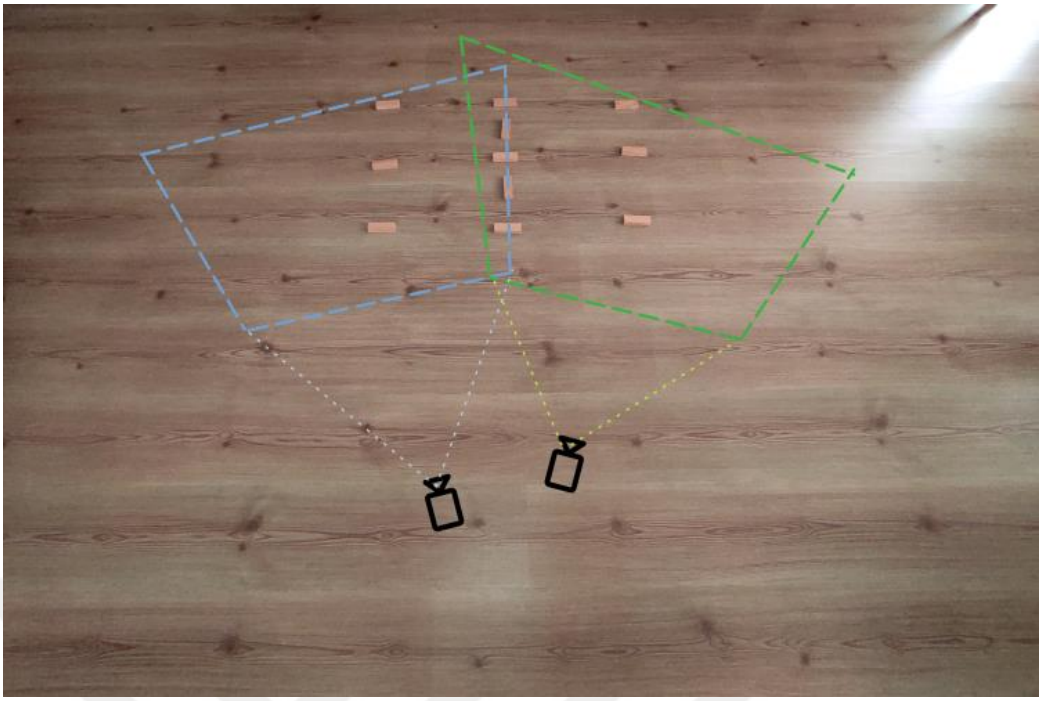
Kamera görüntülerinin kuş bakışı dönüşümleri yapılması sayesinde, kameraların açı (r: rotate) parametreleri aynı seviyeye çekilmiş olacaktır. Bu sayede görüntüler aynı düzlem üzerinde birleştirilmeye hazır hale gelmektedir. Bir sonraki aşama, öteleme (t: transition) parametrelerinin birbirine olan farkları kullanılarak kalibre edilmesi ve bu sayede tek bir harita üzerinde tüm kamera görüntülerinin gösterilebilmesidir.

2.1.3. Global Harita

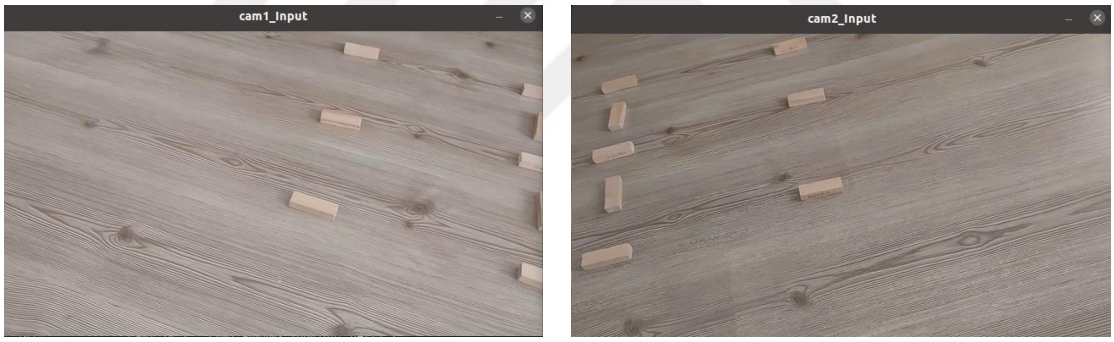
Kameralar arası nesne takibinin yapılabilmesi için tüm kamera düzlemlerinin birleşik olarak bulunduğu global bir harita mimarisi oluşturulmuştur. Tüm kameralar için gerçek dünyadan tek bir referans (0,0) noktası seçilmiştir. Bu referans (0,0) noktasının 2 boyutlu piksel koordinat düzlemindeki karşılığı da x=0, y=0 olarak hesaplanmıştır.

İdeal şartlarda lens bozunumu olmayan kameraların tepe bakışlı olarak 0° açı ile yan yana montajlanması durumunda, gerçek dünyada kabul edilen bir (0,0) koordinatındaki referans noktası, doğrudan görüntü üzerinde (0,0) koordinatına hizalı kabul edilebilirdi. Ancak perspektif etkisi ve kameraların farklı açı, farklı konumlarından dolayı, kamera görüntüleri global haritaya doğrudan yerleştirilemeyecektir.

Global haritanın işleyişini göstermek için düz bir zeminde test ortamı oluşturulmuştur. Ahşap parçaları zemine dağıtılmış ve Kamera-1 sol tarafa, Kamera-2 sağ tarafa farklı konum ve açılarda sabitlenmiştir. Şekil 2.10’daki fotoğraf sistemde kullanılmamış, yalnızca dokümantasyon amaçlı alınmıştır.



Şekil 2.10. *Test Ortamının Üçüncü Bir Kameradan Geniş Açılı Alınmış Fotoğrafı*

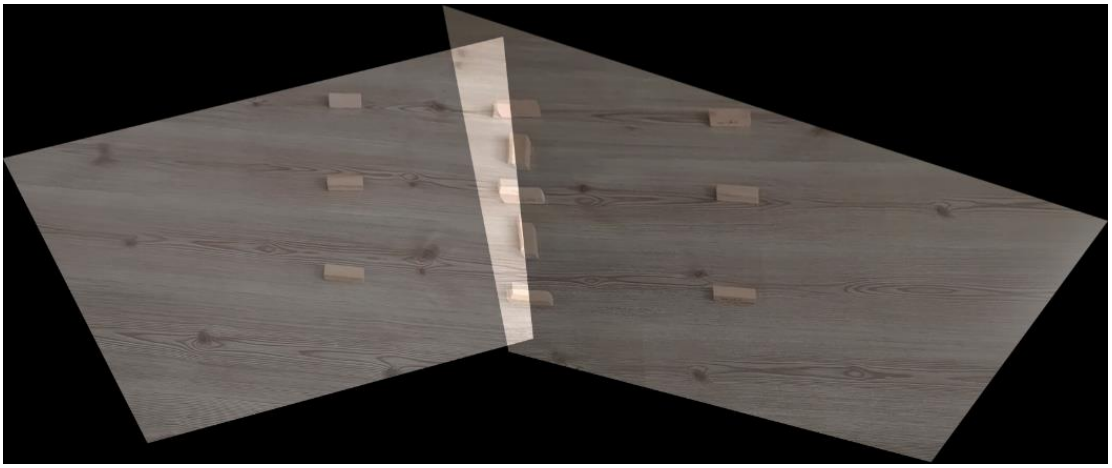


a) Kamera-1'in görüntüsü b) Kamera-2'nin görüntüsü

Şekil 2.11. *Test Ortamından Alınan Kamera Görüntüleri*

Herhangi bir dönüşüm yapmadan farklı açılarda duran kameralardan alınan görüntüler Şekil 2.11'de görülmektedir. Bu şekilde gerçek dünya koordinat düzlemi ile kamera koordinat düzlemi arasındaki hesaplama lineer olmayacaktır. Dönüşüm yapmadan global haritaya görüntülerin yerleştirilmesi de işlevsiz olacaktır.

Global haritayı oluşturabilmek için ilk aşama tüm kameraların kendi içlerinde Affine dönüşüme tabi tutulmuştur. Ardından seçilen ortak referans (0,0) noktasına göre mesafeler hesaplanarak Şekil 2.12'de gösterilen global harita üzerine yerleşim yapılmıştır.



Şekil 2.12. *Elde Edilen Kuş Bakışı Global Harita*

Bu aşamadan sonra piksel değerleri üzerinden yapılacak işlemler (görüntü işleme veya nesne takibiyle ilgili metotlar) kameraların kendi görüntüsü üzerinde bağımsız olarak yapılabilir. Ancak konumla ilgili operasyonlar ortak koordinat sistemi olan global harita üzerinden yapılabilecektir. Yani global haritada piksel değerleri önemsizdir, global harita yalnızca ortak konumlama sağlamaktadır. Global harita üzerinde bütün kameralardaki piksel mesafe ve gerçek dünya mesafe oranları eşit olacaktır.

Örneğin Kamera-1'in sol alt bölgesinde 100 piksellik bir yatay çizgi, gerçek dünyada 150 mm uzunluğa karşılık geliyorsa, Kamera-2'nin sağ üst bölgesindeki 100 piksellik bir dikey çizgi de gerçek dünyada 150 mm uzunluğa karşılık gelecektir.

2.1.4. İzdüşüm Hesaplama

Farklı kameraların görüntülerinin affine dönüşümü sonrasında global haritaya yerleştirilmesiyle tüm takip işlemlerinin global harita üzerinde yapılması mümkün olmuştur.

Ancak takibi yapılacak olan nesnenin tespiti her kamera için ayrı ayrı yapılıyor olmalıdır. Yani nesnenin özniteliklerini oluşturan modülün, her bir kamera için bağımsız çalışıyor olması varsayılmıştır. Bu yönden, her bir nesnenin konum ve boyut bilgisi, tespit yapılan kameranın kendi koordinat uzayındadır.

Takip işlemi, global harita üzerinde birleşik düzlemde yapılacaktır. Bu sebeple tespit sisteminden gelen nesne bilgisi, perspektif dönüşümüne tabi tutularak global haritadaki ortak koordinat düzlemindeki izdüşümü üzerinden işlem görecektir.

Bunun için, global harita oluşturulma aşamasında, her bir kameraya uygulanan affine dönüşümünde kullanılan dönüşüm parametreleri (dönüşüm matris değerleri) kayıtlı tutulmuştur. Bir kamera için nesne konum bilgisi oluşturulduğunda, o kamera için kayıtlı tutulan dönüşüm matrisleri ile nesnenin konum ve boyut bilgilerine dönüşüm uygulanarak, global harita (gerçek dünya) üzerindeki izdüşümü hesaplanmıştır. Bu sayede nesnenin global harita üzerinde yerleşimi yapılmıştır.

2.2. Tespit

Bu aşama, takip edilmesi istenen nesnenin belirlenmesi aşamasıdır. Kullanıcı (veya harici bir sistem) tarafından verilen koordinat bilgisi ile, takip edilmesi istenen nesnenin sınırlarının belirlenmesi için gerekli aşamalar uygulanmıştır. Alınan bilgi, tek bir piksel için x,y koordinat bilgisidir.

Kullanıcı herhangi bir geometrik şekle sahip, herhangi bir renkte, herhangi bir boyutta nesneyi takip etmek isteyebilir. Gerçeklenen uygulamada bilgisayar faresi tıklamasıyla süreç başlatılmıştır. Ancak bu verinin harici bir sistemden gönderilmesi de mümkün olabilir.

Takip aşamasındaki arama ve eşleştirme algoritmalarının ihtiyaç duyduğu, nesne özniteliklerinin çıkarılması için nesne sınırları belirlenmiştir. Bunun için nesnenin içinde dağınık renk değişimlerinin bulunmadığı ve arka plan ile renginin ayırt edilebilecek kadar farklı olması gerekliliği kabulü yapılmıştır. Kullanıcıdan alınan koordinat bilgisinin, nesnenin iç kısmında olması gerekliliğiyle, bu noktadan itibaren çevresinde ani renk geçişleri aranmıştır. Bu aslında kenar bulma olarak da yorumlanabilir.

Nesne tespiti aşamasının çıktısı nesnenin etrafını saran çerçeve konum bilgisidir. Bu çerçeve bilgisinin harici bir sistemden gönderilmesi durumunda da çalışmadaki diğer aşamalar bağımsız olarak uygulanabilecektir.

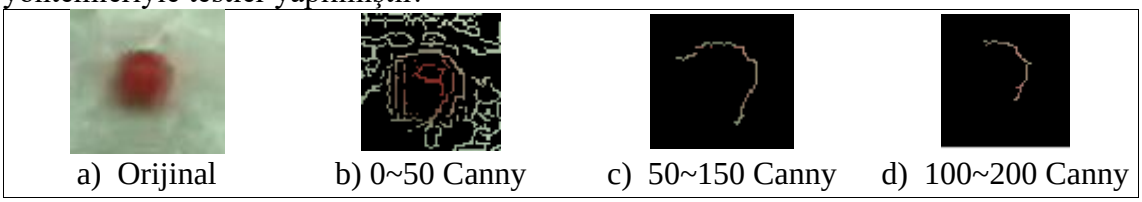
Sisteme giriş olarak verilen noktanın hangi kamera için gönderildiği tespit edilmiştir. Bunun için geri bildirim fonksiyonuna kamera ID'si gönderilmiştir. Seçilen (x,y) koordinatı ve kamera ID'si kaydedilmiştir.

Nesnenin bulunması için;

- renk aralıklı arama ve morfolojik işlemler
- Hough Detector,

- kenar tespit (Şekil 2.13)

yöntemleriyle testler yapılmıştır.



a) Orijinal b) 0~50 Canny c) 50~150 Canny d) 100~200 Canny

Şekil 2.13. *Canny Kenar Tespit Karşılaştırmaları (Farklı Eşik Değerleriyle)*

Kenar tespit algoritmalarıyla doğrudan nesnenin kenarları tespit etmek başarılı sonuç vermemiştir. Çünkü farklı kenar tespit algoritmaları farklı renk geçişlerine tepki verdikleri için nesnenin içinde ve dışında kılcal kenar bilgileri üretmektedir.

Daha sonra Watershed Segmentasyon metoduyla başarılı ve verimli bir tespit yöntemi elde edilmiştir. Watershed Segmentasyon algoritmasıyla nesnenin içindeki ve dışındaki pikseller farklı değerlerle etiketlenmektedir.

Watershed segmentasyon, görüntülere doğrudan uygulandığında aşırı segmentasyona sebep olan, ancak segmente edilecek bölümlerin önceden tanımlanması durumunda başarısını kanıtlamış bir yöntemdir. Waterfall algoritmasını kullanarak hiyerarşik ayırmayı temel alır ve homojen renk dağılımları bulunan alanları başarıyla ayırabilir [15].

Bu çalışma kapsamında segmente edilecek bölge (takibi yapılacak nesne) kullanıcı (veya harici sistem) tarafından belirtildiği için, Watershed segmentasyon bu çalışma kapsamı için uygun ve başarılı bir metot olmuştur. Watershed segmentasyon parametresiz olduğu için koşul ve konum bağımsız çalışabilmektedir.

Watershed Segmentasyon ile kenarları farklı renkte boyanan imaj üzerinden nesnenin çerçeve bilgileri çıkarılmıştır.

a) Kamera görüntüsü	b) Watershed çıktısı	c) Nesnenin çerçevesi	d) Genişletilmiş nesne öznitelik penceresi

Şekil 2.14 *Watershed Segmentasyon*

Bir sonraki aşama olan takip işleminde nesne çerçevesi içinden elde edilen öznitelikler kullanılacaktır. Şekil 2.14.(c)'de görüldüğü gibi nesne çerçevesi nesnenin kenarlarına yerleştiği için, nesnenin kenar öznitelikleri kaybedilmiştir. Bu sebeple sabit bir genişletme uygulanmıştır. Genişletme değeri olarak, yapılan testler sonucunda 20 piksel çerçeve ofseti kullanılmıştır. (w_1, h_1) boyutlarındaki nesne için 4 yönden 20 piksel genişletme yapmak suretiyle, $(w_1 + 40, h_1 + 40)$ boyutlarında genişletilmiş bir çerçeve alınmıştır.

Sonraki aşamada genişletilmiş çerçeve ile nesnenin bilgileri kayıt altına alınmıştır. Bu süreç tamamlandıktan sonra sistem "bekleme" modundan "takip" moduna geçecektir.

2.3. Takip

Sistemde "bekleme" ve "takip" olmak üzere iki mod bulunmaktadır. Nesne konumu belirlendikten sonra takip modu aktifleşir ve her kare analiz edilmeye başlanır.

Tespit modülü tarafından takip modülüne nesnenin koordinatları, boyutları ve piksel değerleri gönderilir. Bu aşamadan sonra arama işlemi başlatılacaktır. Arama işlemi, her yeni gelen karede nesnenin konumunun tekrar bulunmasıyla yapılacak ve bu konum bilgileri birleştirilecektir. İşlem yükünü azaltmak amacıyla [t+1]. karede yapılacak olan arama, nesnenin [t]. karedeki konumuna göre olası yeni konumunda yapılacaktır.

Nesnenin yeni konumunun bulunması için tespit modülünden gelen öznitelikler kullanılarak piksel tabanlı eşleştirme ve benzerlik çıkarımı yapılacaktır. En benzer konum nesnenin yeni konumu olarak tanımlanmış olacaktır.

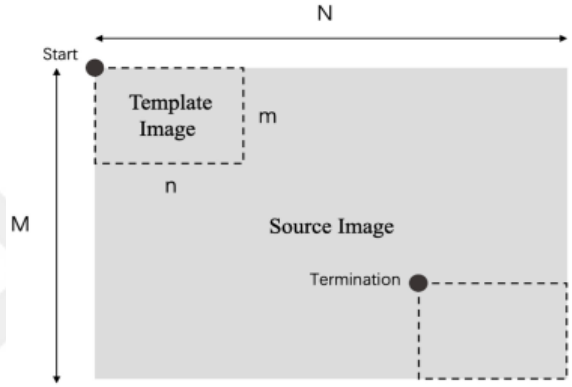
2.3.1. Nesne Özniteliklerinin Eşleştirilmesi

İlk karede sınırları belirlenen ve öznitelikleri çıkarılan nesnenin olası yeni konumunda arama yapılabilmesi için farklı eşleştirme ve benzerlik bulma yöntemleri incelenmiştir. Yapılan çalışma düşük işlem yükü odaklı olduğu için, tüm karede arama

yapılmamakta, yalnızca olası konumda küçük bir alanda arama yapılmaktadır. Bu sebeple düşük maliyetli arama yöntemleri üzerinde durulmuştur.

Bunun için kayan pencere kullanarak şablon eşleştirme (template matching) yapılmış ve benzerlik araması sağlanmıştır.

Şablon eşleştirme yöntemleri, bilgisayarlı görü problemlerinin çözümü için oldukça yaygın kabul görmüş yöntemlerdir. Şablon, arama yapılacak olan görüntü üzerinde kayan pencere yöntemiyle gezdirilerek her konumda benzerlik hesabına tabi tutulur [16]. Pencere kaydırma yöntemi Şekil 2.15'te gösterilmiştir.



Şekil 2.15. *Şablon Eşleştirme Geometrisi [16]*

Şablon eşleştirme mimarisinde temel fikir, (x, y) noktasını Y_{axb} içinde aramaktır. Böylece $X(1:m, 1:n)$ ve $Y(x:(x+m-1), y:(y+n)-1)$ arasındaki maksimum benzerlik noktası elde edilecektir [16].

Şablon eşleştirme için, şablonun her konumunda hesaplanması gereken bir benzerlik fonksiyonu bulunmaktadır. "Tam kare farkları (TM_SQDIFF)", "Korelasyon Katsayısı (TM_CCOEFF)", "Korelasyon (TM_CCORR)" fonksiyonlarıyla testler yapılmıştır. Yapılan testler sonucunda benzerlik fonksiyonu olarak korelasyon katsayısı hesaplama (TM_CCOEFF) fonksiyonu ile en yüksek başarı elde edildiği gözlemlenmiştir. TM_CCOEFF fonksiyonu Denklem 2.10'da gösterilmiştir.

$$T'(x', y') = T(x', y') - 1/(w.h) \,.\, \sum x'', y'' \, T(x'', y'') \tag{2.8}$$

ve

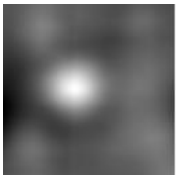
$$I'(x+x', y+y') = I(x+x', y+y') - 1/(w.h) \,.\, \sum x'', y''\, I(x+x'', y+y'') \tag{2.9}$$

olmak üzere,

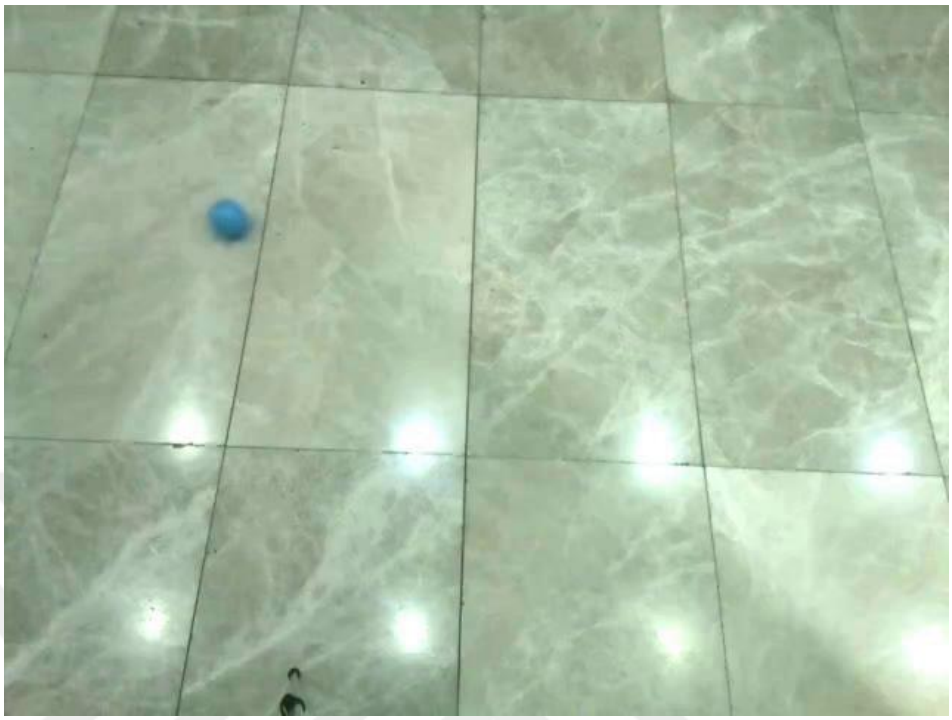
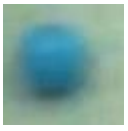
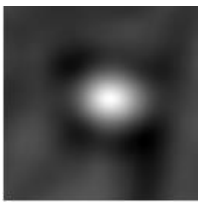
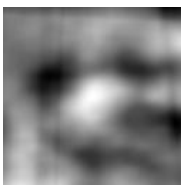
$$R(x,y) = \sum_{x',y'} \left(T'(x',y').I'(x+x', y+y')\right) \tag{2.10}$$

Şablon penceresinin her konumu için, tüm piksel verilerinde (RGB görüntüde her kanal için 3'er kez olmak üzere) korelasyon katsayısı hesaplanmıştır. Bu sayede şablon eşleştirmenin hem desen, hem renk duyarlı olması sağlanmıştır. Bu hesaplama için OpenCV kütüphanesinin *matchTemplate* metodu kullanılmıştır. Her bir konum için elde edilen korelasyon katsayısı değeri, benzerlik oranını vermekte ve arama penceresinden elde edilen Maksimum Korelasyon Katsayısı Değeri (MKKD)'nin elde edildiği konum, şablonun (yani takip edilen nesnenin) yeni konumu olarak kabul edilmektedir.

Tablo 2.1. Şablon Arama Aşamasında Korelasyon Katsayısı Değerlerinin Görselleştirilmesi Örneği 1

Tam Görüntü Karesi				
	Arama Yapılan Şablon	Arama Penceresi	[0-255] Değerlerine Normalize Edilmiş Korelasyon Katsayı Değerlerinin Görselleştirilmesi	MKKD
Örnek 1				23.219.123
Örnek 2				22.969.744
Örnek 3 Şablon görüntüden çıkmış				6.101.203

Tablo 2.2. Şablon Arama Aşamasında Korelasyon Katsayısı Değerlerinin Görselleştirilmesi Örneği 2

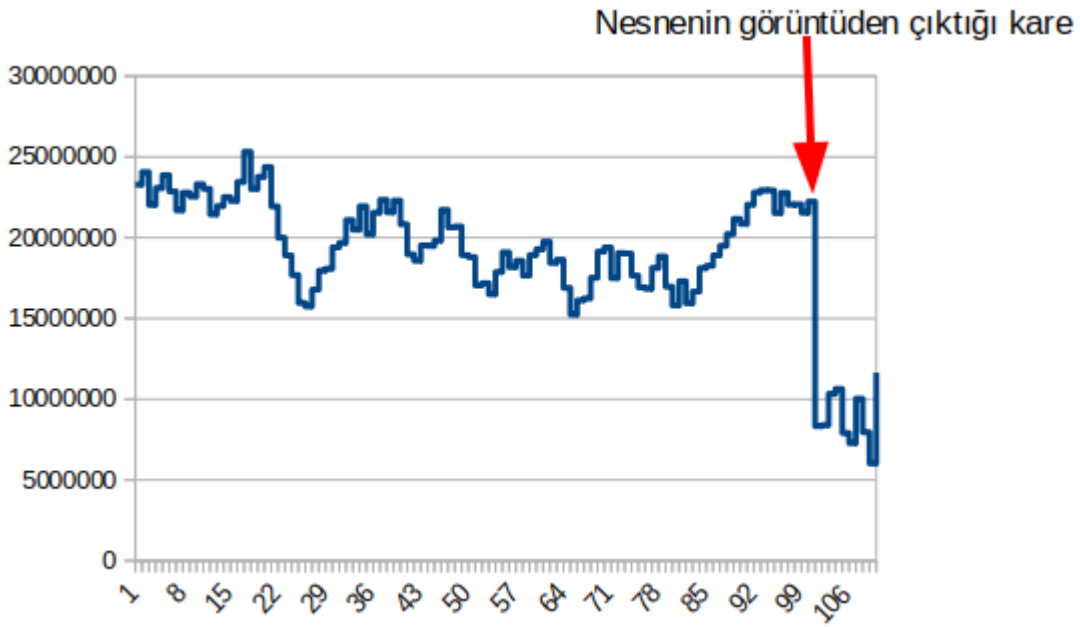
Tam Görüntü				
	Arama Yapılan Şablon	Arama Penceresi	[0-255] Değerlerine Normalize Edilmiş Korelasyon Katsayı Değerlerinin Görselleştirilmesi	MKKD
Örnek 4				12.367.449
Örnek 5				13.677.701
Örnek 6 Şablon görüntüden çıkmış				7.995.102

Tablo 2.1 ve Tablo 2.2'de görüldüğü üzere; Örnek 1, Örnek 2, Örnek 4 ve Örnek 5'te aranan nesne şablonu görüntüde bulunduğunda MKKD yüksek elde edilmiş, şablonun bulunmadığı Örnek 3 ve Örnek 6'da düşük elde edilmiştir.

Buradan görülüyor ki, MKKD her zaman nesnenin konumunu vermemektedir. Bu bir benzerlik değeridir. Şablonun (seçilen nesnenin) özniteliklerine ve arka plana göre, şablonun görüntüde bulunma/bulunmama durumlarında elde edilen MKKD büyük oranda değişiklik gösterebilmektedir. Tablo 2.1'de görüldüğü üzere kırmızı nesnenin görüntüde bulunma/bulunmama durumunda elde edilen değer çeyrek orana kadar düşerken, Tablo 2.2'de mavi top nesnesinin görüntüde bulunma/bulunmama durumunda elde edilen değer yalnızca yarıya kadar düşmüştür. Eşleştirmenin yapılabilmesi için elde edilen MKKD'nin güvenirliğine karar verecek bir limit hesaplanması gerekecektir.

Yapılan testler sonucunda, [t]. kare üzerinde şablon belirlendiğinde [t+1]. karede şablonun görüntüde bulunması durumunda elde edilen MKKD, önceki sonuçlara yakın elde edilmektedir. Yani her bir karede şablon tekrar belirlendiğinde, bir sonraki karede MKKD benzer çıkmaktadır. Bu şablonun rengine, şeklinde, arka plan ışıklılığına göre değişmektedir. Bu yüzden her bir nesne için farklı davranışlar elde edilecektir.

Nesnenin tüm hareketi boyunca MKKD benzerliğini birbirine yakın tutabilmek için her karede nesnenin yeni konumu tespit edildikten sonra, şablon tekrar güncellenmiştir. Yani her bir şablon, yalnızca kendinden bir sonraki karede aranmıştır.

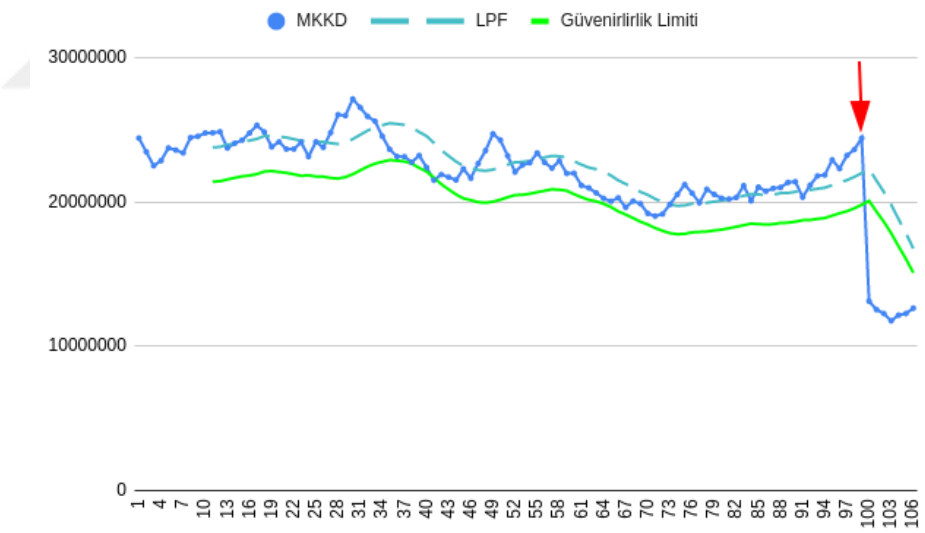


Şekil 2.16. *Seçilen Bir Nesne İçin 110 Kare Boyunca Elde Edilen MKKD*

Şekil 2.16'da nesne şablonunun kareler arasında hareketi boyunca elde edilen MKKD grafiği görülmektedir. [0-100]. kareler arasında şablon görüntüde, [100-110]. arasında görüntüde değildir. [100]. karede şablon görüntüden tamamen çıktığında MKKD'de ani düşüş görülmüştür. (Bkz: Tablo 2.1 ve Tablo 2.2)

Anlık gürültülerden dolayı MKKD'de yüksek frekanslı dalgalanmalar görülmektedir. Nesnenin yön değiştirmesi, konuma göre ışıklılığın değişmesi gibi sebeplerden dolayı nesne mesafe kat ettikçe düşük frekanslı değişimler gözlemlenmektedir. Bu yüzden ani dalgalanmaların yalıtılması için LPF (Düşük Frekans Geçiren Filtre) gerekmektedir. Bunun için belirli bir boyutta pencere ile ortalama değerler hesaplanmıştır. Pencere boyutu zamana göre belirlenmiş ve yarım saniye olarak (FPS/2) olarak tanımlanmıştır. Her kare için MKKD güvenilirlik limiti, önceki karelerin ortalama değerinin %90 üzeri olarak tanımlanmıştır.

$$GüvenilirMKKDEşiği_t = 0.9 * \left(Ortalama_{MKKD}([t-1]:[t-1-(FPS/2)])\right) \quad (2.11)$$



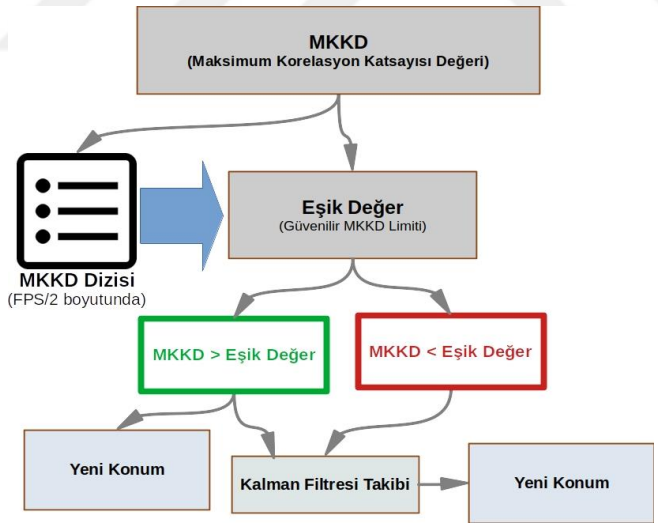
Şekil 2.17. *LPF Uygulanmış MKKD Ve Güvenilirlik Limitleri*

Her bir karede elde edilen MKKD, güvenilir MKKD eşiği üzerindeyse eşleştirme güvenilir olarak tanımlanır ve nesne konumu buna göre güncellenir. MKKD değerinin güvenilir olmadığı durumda ise, anlık bir gürültüden kaynaklı olarak konum sapmalı olarak hesaplanmış olabilir. Bu durumda veri güvenilir değildir. Güvenilir olmayan

MKKD hesaplandığı durumda takibin devam edebilmesi için Kalman Filtre Takibi yapılmıştır.

Kalman Filtresi, beyaz gürültü bozunumuna sahip lineer dinamik bir sistemin anlık durumunu tahmin etme yöntemidir. Günümüzde fabrikalarda üretim süreçleri, uçak-gemi seyrüsefer sistemleri, finans problemleri gibi günlük hayatımızda önemli konularda kullanılmaktadır. Dinamik bir sistemin kontrol ve takibi için kullanılır. Kalman filtresi, her zaman ölçümü yapılamayan (veya doğru ölçümü yapılamayan) sistemler için eksik bilgilerin, önceki veriler ışığında istatistiksel olarak tahmin edilmesi için bir çözüm sunar [17].

Hesaplanan MKKD'nin güvenilir olduğunda nesnenin konum bilgisi güncellenir, Kalman Takip'e bu veri beslenir. Güvenilir olmayan veri geldiğinde ise Kalman Tahmin bilgileri ile konum hareketine devam edilir. Ancak nesnenin açı değiştirmesi, farklı ışıklılıkta bir zemine geçmesi gibi değişimlerden dolayı güvenilir MKKD eşiğinin bu duruma adapte olabilmesi için, elde edilen veriler güvenilir olmasa bile kayıt altında tutulmaya devam edilir ve yeni güvenilir MKKD eşiği bu veriler üzerinden hesaplanır. (Bkz: Şekil 2.17) Yeni konum hesabı karar diyagramı Şekil 2.18'de paylaşılmıştır.



Şekil 2.18. *MKKD Güvenilirliğine Göre Yeni Konum Hesabı*

2.3.2. Arama Bölgesi

Takibi yapılacak olan nesne tüm görüntüde aranmamaktadır. Nesnenin bulunduğu kamera ve bulunduğu bölge değerlendirilerek, nesnenin bulunma olasılığı yüksek olan

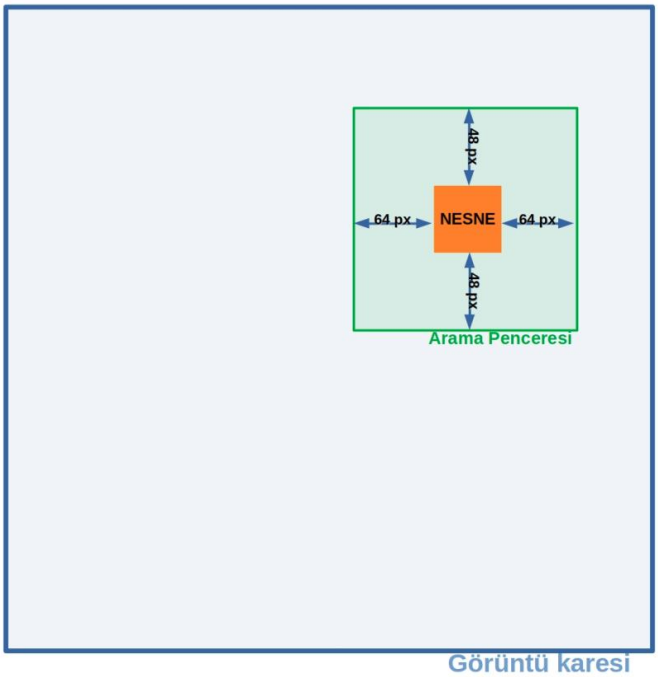
bölgede araması yapılmıştır. Arama pencere boyutunun hesaplanması için nesnenin boyutu, mevcut hareketinin yönü ve hareket hızı değerlendirilmiştir.

Testler için hazırlanan çalışma düzeneğiyle ilgili bilgiler şu şekildedir: Her bir kamera, yaklaşık 1 metre genişlikteki bir zemini görmektedir. 640*480 çözünürlükte çalışılmıştır. Yani 1 piksel, genişlikte yaklaşık 1,5 milimetrelik bir alanı görmektedir. Bu koşullarda, takip edilecek olan nesne maksimum 1 metre/saniye hızda hareket ettirilmiştir. Test düzeneği daha yüksek konumlandırılsaydı, örneğin bir kamerada 20 metrelik bir alan genişliği görülseydi, 20 metre/saniye = 72km/s hız için çalışılması mümkündür.

Bu düzenek için en yüksek hız olarak kabul edilen, 1 metre/saniye ile nesne yatay olarak saniyede 640 piksel mesafe almaktadır. 20 FPS kare hızında çalışmıştır. Her karede yatayda maksimum 32 piksellik hareket oluşacaktır, çözünürlük en/boy oranıyla, dikeyde ise 24 piksel hareket elde edilecektir. Anlık gürültülerden dolayı hata payını düşürmek amacıyla 2 kat toleranslı olarak, her karede 64*48 piksellik hareket limiti belirlenmiştir. Yani arama penceresi nesne boyutunun çevresinde sol ve sağ 64*2, üst ve alt 48*2 piksel olmak üzere genişletilmiştir.

Arama penceresi (AP) boyutlarının hesaplanması Denklem 2.12'de gösterilmiştir. Arama penceresinin görüntü karesi üzerindeki yerleşimi Şekil 2.19'da gösterilmiştir.

$$(AP_w, AP_h) = \left(Nesne_w + 2 * (Kare_w/FPS), Nesne_h + 2 * (Kare_h/FPS)\right) \quad (2.12)$$



Şekil 2.19. *Hızı Sıfır Olan [t-1]. Karedeki Nesnenin [t]. Karedeki Arama Bölgesi*

Bu arama penceresi, nesnenin [t-1]. karede tespit edilen konumuna göre [t]. kare üzerinde bulunmaktadır. [t-1]. karede hızı olmayan (veya bilinmeyen) nesne ile, bu nesnenin [t]. karedeki arama penceresinin orta noktaları kesişecektir. Yani nesne kendi etrafında aranır. Eğer nesnenin bilinen hızı ve yönü varsa [t]. karedeki arama penceresi bu hareket ve yöne göre tahmini olarak yönlendirilir.

Her ne kadar; arama penceresi, bu test düzeneğinin kabul edilen en yüksek hızı olan 1 metre/saniye hız için alınabilecek en fazla mesafeye göre boyutlandırılmış olsa da, MKKD hesaplama hatalarını tolere edebilmek için arama penceresi tahmini nesne hareketi yönünde ötelenmiştir.

İlk aşamada (nesnenin tespit edildiği ilk karede), nesne hareketli dahi olsa, nesnenin hareket ve yön bilgisi bilinmediği için sıfır kabul edilmiştir. Hız ve yön bilgisi elde edildiğinde arama penceresinin konumu da, bir sonraki hareketin olası konumuna göre yerleştirilecektir. Yön ve hareket hızının kestirilmesi için nesnenin önceki konumuna göre (x,y) koordinatlarında aldığı mesafe hesaplanacaktır. [t] karesindeki arama penceresinin konum hesaplamasında kullanılacak olan nesne hızı, [t-1] karesinde, [t-1] ile [t-2] kareleri arasındaki konum farkı ile hesaplanmış olacaktır. Sistemin sabit FPS'te çalıştığı varsayılarak, hız bilgisi her bir kare için vektör olarak kaydedilecektir.

s hız, N nesnenin orta noktası olmak üzere,

$$s_{[t]} = \left(N_{[t-1]} - N_{[t-2]}\right) \tag{2.13}$$

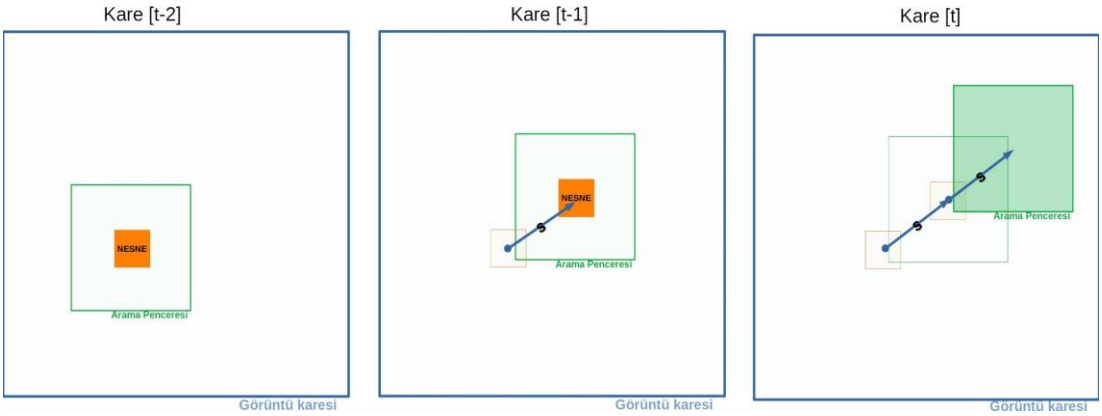
olarak hesaplanmıştır.

Nesnenin bir karede aldığı yolu bir sonraki karede de alması beklenerek, arama penceresinin merkez bölgesi bu olası konuma göre kaydırılmaktadır.

Arama penceresi orta noktası (AP);

$$AP_{[t]} = (N[t-1] - S[t]) \tag{2.14}$$

şeklinde hesaplanmaktadır. Hareket bilgisine göre AP konumunun değişimi Şekil 2.20'de gösterilmiştir.



Şekil 2.20. *Nesne Hızına Göre Arama Penceresinin Konumlandırılması*

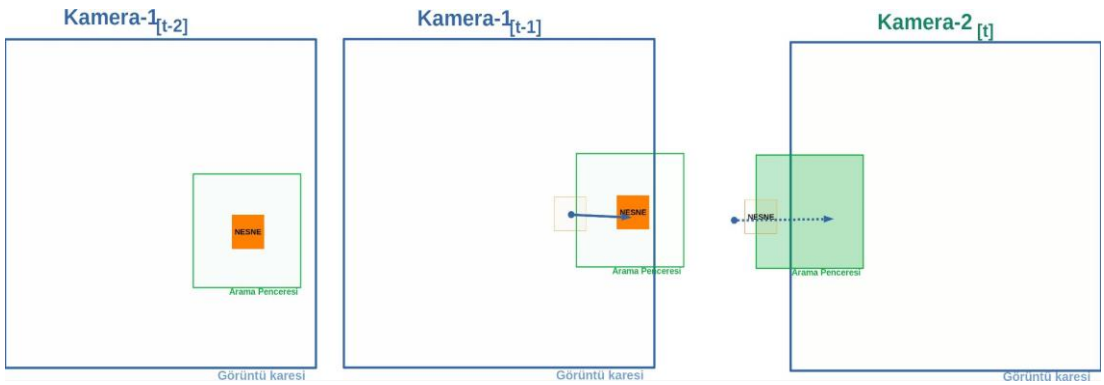
2.3.3. Kameralar Arası Geçiş

Nesnenin her yeni konum hesaplamasında, bulunduğu kameranın görüntüsünden çıkma ihtimali değerlendirilmektedir. Nesne görüntüden çıkarken kameradan alınan son karede nesnenin bir kısmı görünmüyor olabilir. Bu da bir sonraki karede hatalı takibe ve eşleştirmeye sebep olabilir. Bu sebeple nesnenin kameradan çıkmadan önce tam göründüğü son kare güvenilir olarak kabul edilmelidir.

Nesnenin bir kısmının görünmediği karede işlem yapılmaması için, kameralar arası geçiş işleminde nesnenin değil, arama penceresinin kamera dışına taşması kontrol edilmektedir. Arama penceresinde 1 piksel veya daha fazla alan görüntüden taşmışsa, kamera geçişi için gerekli işlemler başlatılır. Yani, nesne görüntüden çıkmadan en az 1 kare önce aramanın yapıldığı mevcut kamera değiştirilmektedir. Önceki kamerada takip bırakılıp, diğer kameraya geçiş yapılır, nesnenin yeni kamera görüntüsüne geçişi beklenir. Kameralar arası geçiş Şekil 2.21'de gösterilmiştir.

- Eğer kameralar arası kör nokta yoksa (görüntüde kesişim bölgesi varsa), nesne mevcut kameradan çıkmadan önce diğer kamera görüntüsünde belirmiş olacaktır. Bu durumda, arama diğer kameraya geçtiği anda, kayıp olmaksızın diğer kamerada eşleştirme yapılabilecektir.
- Eğer kameralar arası kör nokta varsa (görüntüde kesişim bölgesi yoksa), nesne mevcut kameradan çıktıktan sonra, diğer kamera görüntüsüne girene kadar bir süre kayıp olacaktır. Bu durumda nesnenin diğer kamera görüntüsüne girmesi beklenen noktada nesne bir süre aranacak, yani görüntüye girmesi beklenecektir. Nesnenin beklenme süresi ise "2.3.4 Nesnenin Kaybolması"

bölümünde anlatılan limit kadar tanımlanmıştır, ancak bu süre kamera geçişinde yüksek tutulabilir. Nesne, kameralar arasındaki kör noktada kaybolup, diğer kamera alanına tekrar girmezse takip sonlanacaktır.



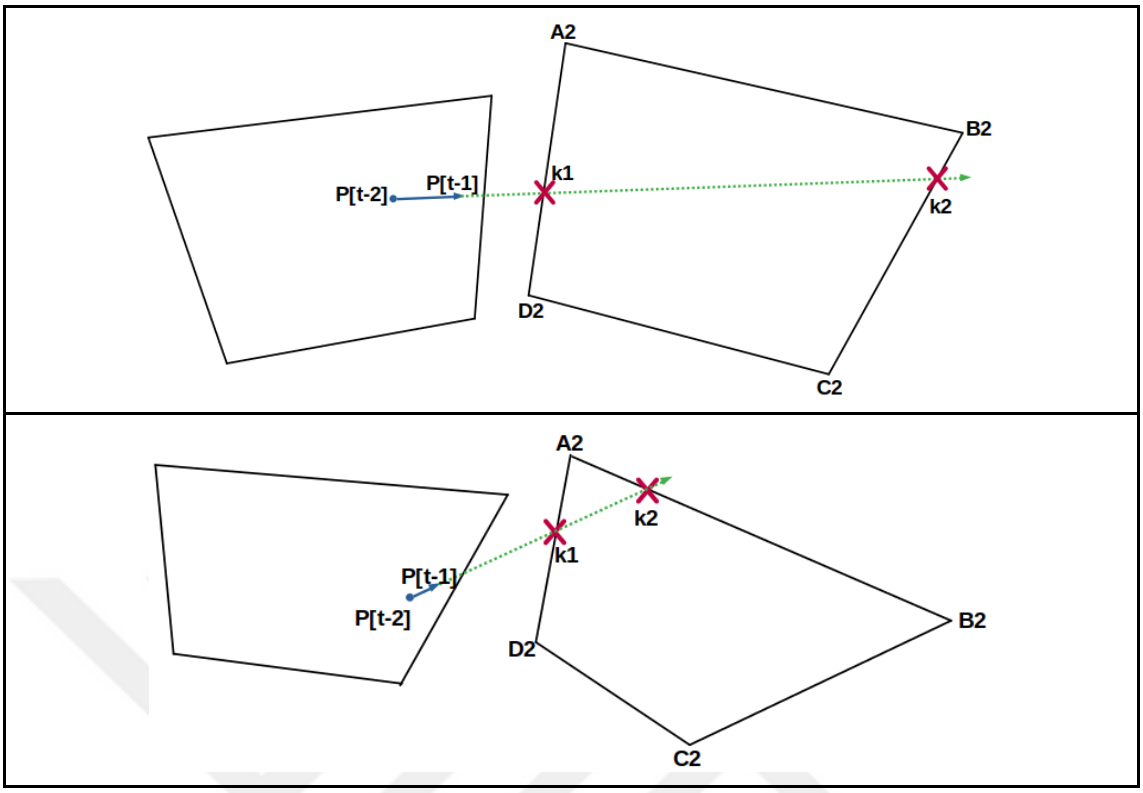
Şekil 2.21. *Arama Penceresinin Kameralar Arası Geçişi*

Diğer kameraya geçiş yapılması için görüntüden çıkma tahmini yapılırken, nesnenin (ve arama penceresinin) bulunduğu kamera kenarlarından hangisinden çıkacağı belirlenmektedir.

Nesnenin görüntüden çıkarken, son elde edilen [t-2]~[t-1] kareleri arasındaki hareket vektörü yönünde, geçiş yapacağı kameranın hangisi olduğu ve diğer kameradan yaklaşık hangi konumdan görüntüye gireceği tahmin edilmektedir.

P_G, bir kameradaki P noktasının global haritadaki izdüşümü olmak üzere, geçiş yapılan diğer kameradaki olası yeni konumun hesaplanması için;

- Mevcut kameradaki son 2 bilinen noktanın ($P_{[t-1]}$ve $P_{[t-2]}$), dönüşüm matrisleriyle global haritadaki konumları hesaplanmıştır. ($P_{G_{[t-1]}}$ve $P_{G_{[t-2]}}$)
- Geçiş yapılacak kameranın tespiti için, [$P_{G_{[t-1]}}$ve $P_{G_{[t-2]}}$] doğrusunu kesecek olan diğer kamera kenarları aranmıştır. Global haritadan tüm kameraların kenar bilgileri okunarak, iki doğrunun kesişim noktası bulunmuştur.
- $P_{G_{[t-1]}}$ noktasına en kısa pisagor uzaklığı bulunan kesişim noktası, geçiş yapılacak olan kameranın ilgili kenarını ve nesnenin diğer kameradan giriş yapacağı olası noktayı verecektir.
- Hesaplanan yeni nokta global harita koordinat düzlemindedir. Bu nokta hangi kameraya aitse, o kameranın dönüşüm matrisiyle geri dönüştürülerek, ilgili kameranın kendi düzlemindeki P_t tahmin koordinatı elde edilir.



Şekil 2.22. *Kameralar Arası Geçiş Anı*

Şekil 2.22'de paylaşılan örnekte çarpışma tahmin noktasının elde edilebilmesi için, nesnenin bilinen son hareket vektörü noktalarının, diğer kamera(ların) kenar vektörleriyle kesişim noktaları hesaplanmıştır. Bunun için, iki noktası bilinen doğru denklemi ve iki doğrunun kesişim noktası formülasyonları kullanılarak, k kesişim noktalarının konumları Denklem 2.16 ve Denklem 2.17'de gösterildiği gibi hesaplanır.

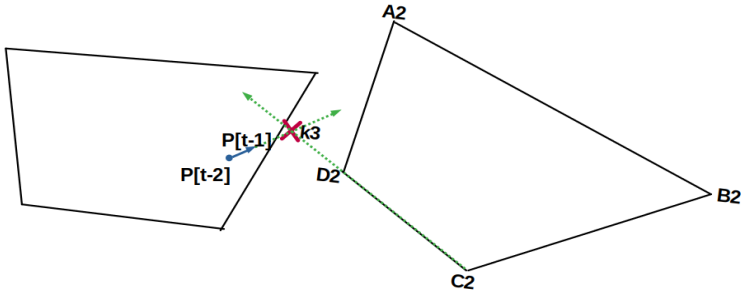
$$T = \frac{(D2_y - A2_y)(A2_x - P[t-2]_x) - (A2_y - P[t-2]_y)(D2_x - A2_x)}{(D2_y - A2_y)(P[t-1]_x - P[t-2]_x) - (P[t-1]_y - P[t-2]_y)(D2_x - A2_x)} \quad (2.15)$$

olmak üzere;

$$k_x = P[t-2]_x + (P[t-1]_x - P[t-2]_x)T \quad (2.16)$$

$$k_y = P[t-2]_y + (P[t-1]_y - P[t-2]_y)T \quad (2.17)$$

Her bir kameranın her bir kenar doğrusu için k kesişim noktaları hesaplanır. Ancak kesişimi olmayan kenarlar (doğru parçaları) için elde edilen k kesişim noktaları, bir P_t tahmin noktası adayı değildir.



Şekil 2.23. *Kesişimi Olmayan Kamera Kenar Doğru Parçası*

Şekil 2.23'te k_3 kesişim noktası, $[P_{[t-1]}, P_{[t-2]}]$ ve $[D_2, C_2]$ doğru parçalarının kesişiminde değildir. Bu yüzden Şekil 2.22'de görüldüğü gibi yalnızca k_1 ve k_2 , bir P_t tahmin noktası noktası adayıdır.

Tüm kameraların kesişim noktaları hesaplandıktan sonra;

$Kamera[1]_{k_1}$,$Kamera[1]_{k_2}$,$Kamera[1]_{k_3}$,$Kamera[1]_{k_4}$, … $Kamera[n]_{k_1}$,$Kamera[n]_{k_2}$,$Kamera[n]_{k_3}$,$Kamera[n]_{k_4}$ noktaları elde edilecektir.

Bu elde edilen tüm k noktaları için, ilgili kameranın 'doğru kenar parçası' (örneğin $[D_2, C_2]$) üzerinde olup olmadığı kontrolü;

$$min(D_{2_x}, C_{2_x}) \leq k_{3_x} \leq max(D_{2_x}, C_{2_x}) \tag{2.18}$$

ve

$$min\left(D_{2_y}, C_{2_y}\right) \leq k_{3_y} \leq max\left(D_{2_y}, C_{2_y}\right) \tag{2.19}$$

şeklinde yapılır.

Böylece, Şekil 2.22'deki eşleştirmede sadece k_1 ve k_2 kesişim noktaları değerlendirmeye tabi tutulmuştur.

Elde edilen tüm k noktaları içinden, $P_{[t-2]}$ noktasına en yakın mesafede olan k noktası, P_t tahmin noktası olarak kullanılır.

Bu aşamadan sonra arama tamamen diğer kamera geçiş yapar, nesne P_t noktasında kabul edilerek arama penceresinin yerleşimi yapılır. Kameralar arası geçişte kör nokta, süre, ışıklılık farkından dolayı nesnenin MKKD dizisi sıfırdan oluşturulur.

2.3.4. Nesnenin Kaybolması

"2.3.1 Nesne Özniteliklerinin Eşleştirilmesi" bölümünde anlatıldığı gibi, her karede hesaplanan MKKD'nin güvenilir olması durumunda nesnenin takibi devam

etmektedir. Güvenilir olmayan MKKD hesaplandığında ise Kalman Filtre ile tahmin edilen veriler üzerinden takip devam ettirilmektedir.

Nesnenin görüntüde olması durumunda, anlık gürültülerden dolayı MKKD değerinin düşük çıkma ihtimaliyle Kalman takip yapılmıştır. Ancak nesnenin gerçekten görüntüden çıkmış olma durumunda da MKKD düşük hesaplanacaktır. (Bkz: Tablo 2.1, Tablo 2.2, Şekil 2.16)

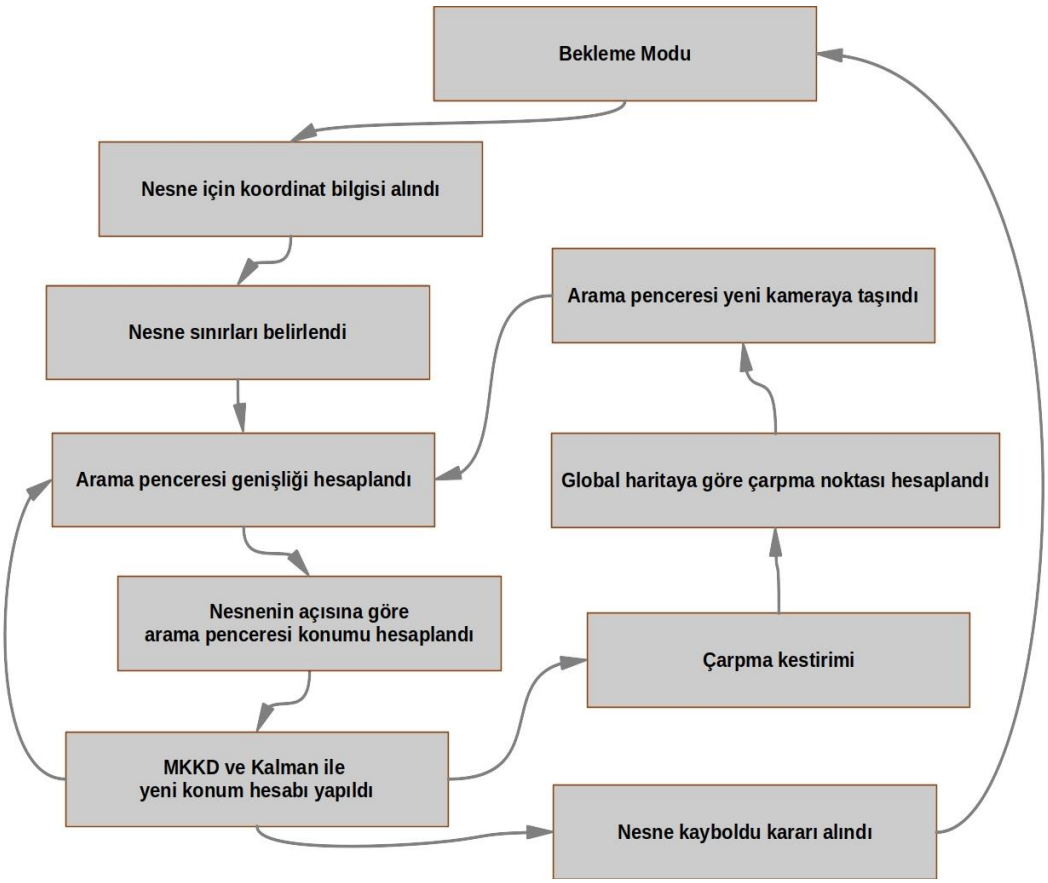
Anlık hatalardan dolayı düşük hesaplanan MKKD değeri, kısa bir süre sonra tekrar yüksek hesaplanmakta, ancak nesnenin gerçekten görüntüden çıkması durumunda MKKD değeri kalıcı olarak düşük hesaplanmaktadır. MKKD güvenilirlik eşiği değişimlere adapte olabilmek için son MKKD ortalamalarıyla güncellenmektedir. Güvenilirlik eşiği, nesnenin görüntüden çıkması durumunda elde edilen düşük MKKD'ye adapte olmadan önce nesnenin yok olduğu kararının verilmesi gerekmektedir.

MKKD dizisi boyutu kadar süre boyunca (FPS/2) güvenilir olmayan MKKD hesaplanması durumunda nesnenin kaybolduğu kararı verilir ve sistem bekleme moduna geçer. Nesnenin görüntünün içindeyken kör noktaya girme ihtimali de bulunduğu için nesnenin yalnızca kamera sınırlarında kaybolması şartı aranmamaktadır.

BÖLÜM 3. TEST VE UYGULAMA

Çalışma kapsamında kavramsal tasarımı geliştirilen metodun farklı koşullarda testleri yapılmıştır. Elde edilen sonuçlara göre metot tasarımında güncellemeler yapılmıştır.

Algoritma akışı Şekil 3.1'de gösterilmiştir.



Şekil 3.1. *İşlem Akış Diyagramı*

3.1. Uygulama

Çalışma çıktısı çalışan bir yazılım paketi haline getirilmiştir. Geliştirilen kod platform ve mimari bağımsızdır. Uygulamada kullanılan bileşenler Tablo 3.1'de paylaşılmıştır.

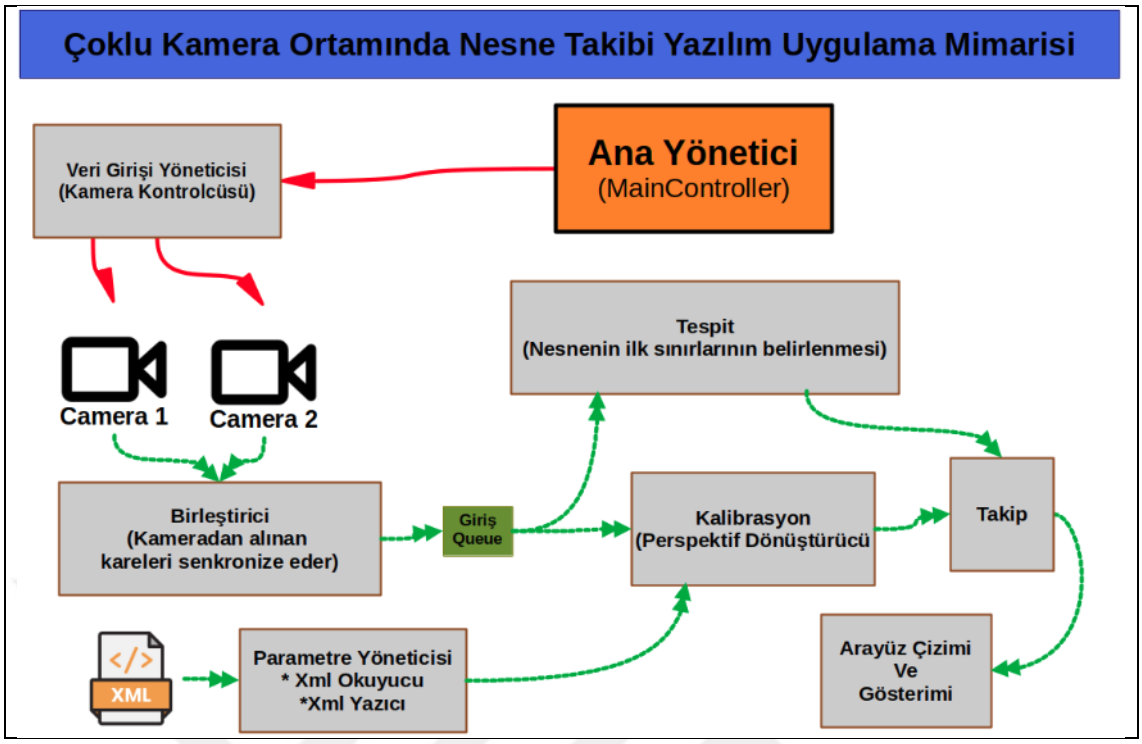
Tablo 3.1. Uygulamada kullanılan yazılım bileşenleri

Kullanılan yazılım dili ve standardı	C++ 11
Kullanılan IDE	Eclipse CDT 4.16.0
İşletim Sistemi	Ubuntu 20.04.3 LTS x64
Kullanılan 3. parti kütüphaneler ve versiyonları	Boost 1.73.0 OpenCV 4.1.5 QT pthread gcc/g++ 9.3.0

Geliştirilen yazılım nesne yönelimli, çoklu thread olarak geliştirilmiştir. Kameralardan görüntü okumayı yapan bir thread, veriyi birleştirip threadsafe bir queue üzerinden ana işleyici thread'e göndermektedir.

Kameralardan görüntü, OpenCV kütüphanesi desteği ile okunmuştur. FPS sabitlemenin yapılabilmesi için, kare okumaların arasında otomatik olarak bekleme yapan bir thread hazırlanmıştır. Bu thread, görüntünün kameradan çekilme süresini (yani usb'den gelen verinin aktarım süresini) hesaplayarak, belirlenen FPS değerini sağlayabilmek için beklemektedir. Yazılımda veri akış diyagramı Şekil 3.2'de paylaşılmıştır.

Kameradan gelen görüntünün diğer thread'e aktarılması, parametrelerin okunması, anne thread'in çocuk thread'leri oluşturması aşamalarında thread'ler arası haberleşme ve veri aktarımı yapılmaktadır. Bu aşamada mücadele durumu (race condition) hatalarının önlenmesi için mutex ile thread-safety önlemleri alınmış, atomic veri yapıları kullanılmıştır.



Şekil 3.2. *Çoklu Kamera Ortamında Nesne Takibi Yazılım Uygulama Mimarisi*

3.2. İşlem Yükü İyileştirmesi

Çalışmada kullanılan yöntemler yapay sinir ağları veya bilgisayarlı görü tabanlı özellik çıkarımı gibi çok fazla işlem yükü barındıran yöntemler değildir. Temelde tek bir nesnenin takibine odaklı olarak, belirli bir konumda, düşük boyutlu matrisler üzerinde algoritmalar çalıştırılmıştır. Bu sayede işlem yükü düşük bir metottur.

640*480 çözünürlükte 20 FPS ile çalışmıştır. GPU kullanılmamıştır. CPU kullanımı "Intel® Xeon(R) CPU E3-1270 V2 @ 3.50GHz × 8" işlemci üzerinde ortalama %7 olarak gözlemlenmiştir. Bu işlemci kullanımı yapay sinir ağı destekli metotlara göre çok düşüktür. Örneğin bir segmentasyon ağı olan U-net'in aynı işlemci üzerinde 512*512 çözünürlükte çalıştırılması durumunda 0,000132223 FPS hızda çalışabileceği görülmektedir. (U-Net Inferance Time ms. Intel Xeon E3-1231 v3: 7563ms [18])

Gelişmiş metotlara karşın, bu çalışmada sunulan düşük karmaşıklıktaki bu takip metodu ile nesnenin konumundan, boyutundan, şeklinden, renginden bağımsız herhangi bir açıda montajı yapılmış kamera üzerinde takip yapılabilir. Bu algoritmada takip edilen nesnenin ne olduğuyla ilgili her hangi bir bilgiye ihtiyaç bulunmadığı için başka parametrelere bağımlılık bulunmaz, bu sayede genel kullanıma daha açık olacaktır.

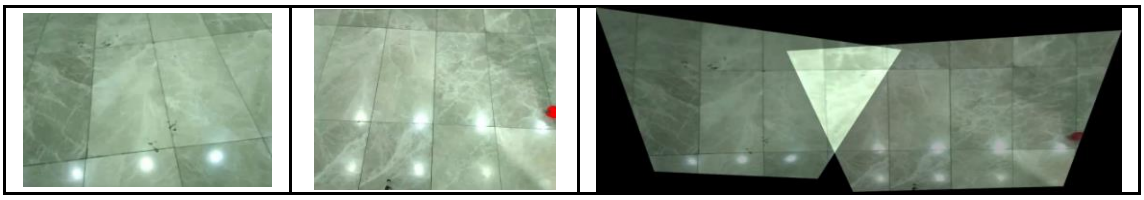
Algoritmanın çalıştırılabilmesi için ek ihtiyaçlar, GPU, TPU gibi ek donanımlar bulunmamaktadır.

3.3. Test Kurulumu

Geliştirilen algoritmanın testi için Raspberry Pi Camera Module v2 kullanılarak senkron video kayıtları alınmış ve USB Webcam'ler ile canlı görüntü üzerinde çalışılmıştır. Şekil 3.3'te paylaşılan test ortamında kameralar yerden 1.2 metre yükseklikte konumlandırılarak, kameraların arasında 1 metre mesafe ile çalışılmıştır. Bu kurulum farklı ışıklılıkta farklı zemindeki ortamlarda tekrarlanmıştır.

Şekil 3.3. *Test Düzeneği Örneği*

Düzenek üzerinden elde edilen görüntüler ve elde edilen global harita Şekil 3.4'te paylaşılmıştır.



Şekil 3.4. *Test Düzeneğinden Elde Edilen Kamera Görüntüleri Ve Oluşturulan Kuş Bakışı Global Harita*

3.4. Elde Edilen Sonuçlar

Yapılan testlerde farklı şekillerde ve renklerde nesneler, görüntüye farklı açılardan farklı hızlarda atılmış (veya görünmeyen bir iple çekilmiştir). Nesnenin tespiti için bilgisayar faresi ile seçilip, nesne sınırları belirlenmiş ve takip edilmiştir. Takip devam ettikçe (arama verisi güvenilir olsa da, güvenilir olmayıp Kalman Tahmini kullanılsa da) nesnenin hareket rotası kayıt altına alınmış ve görüntü üzerinde çizimi yapılmıştır. Aynı zamanda kare numarası ile yeni koordinat bilgileri csv formatlı dosyaya kaydedilmiştir.

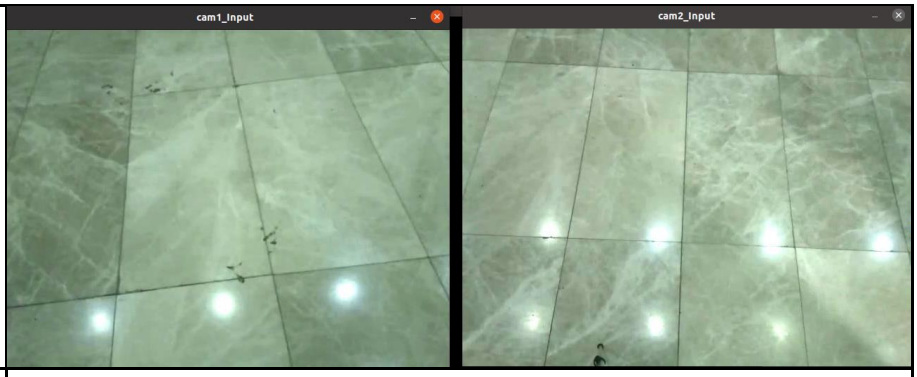
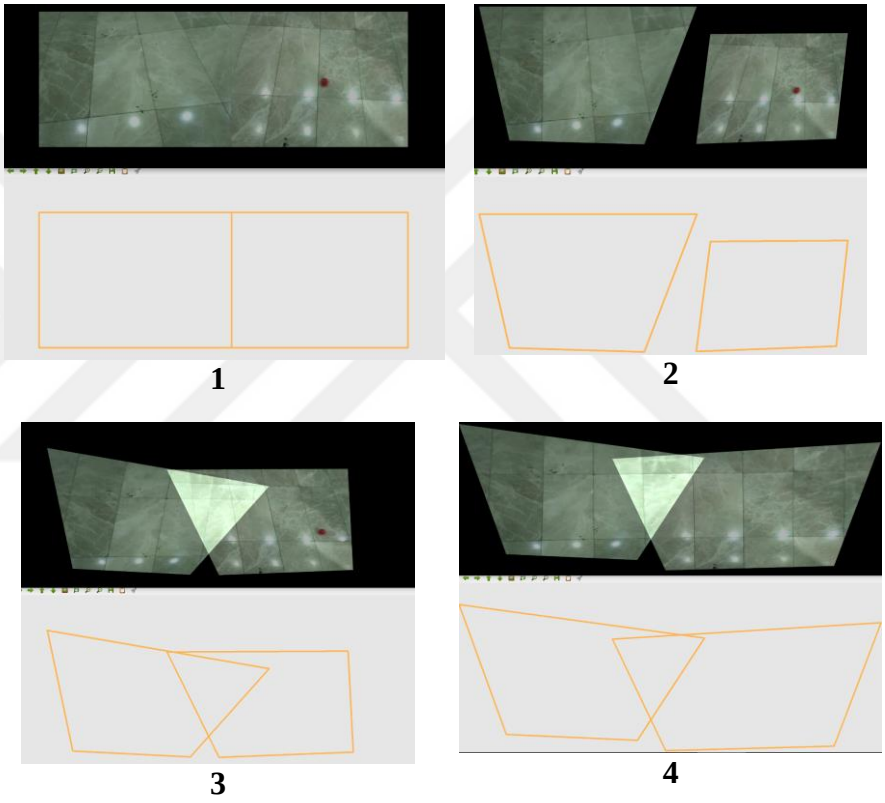
3.4.1. Başarı testleri

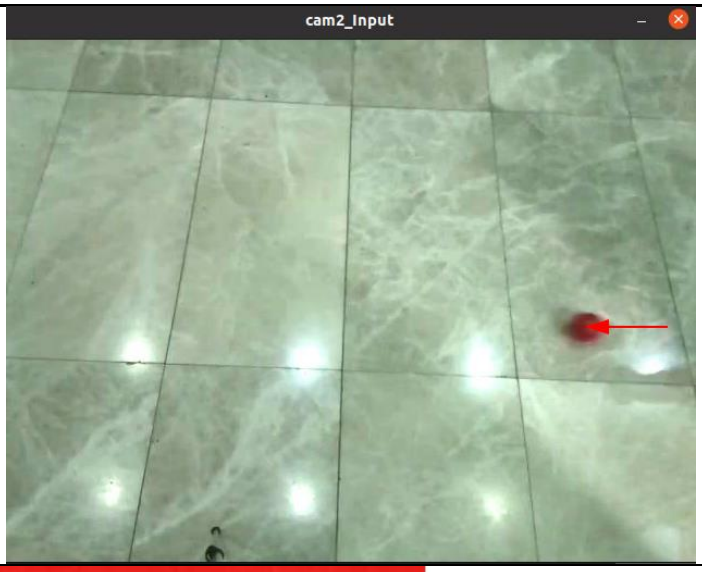
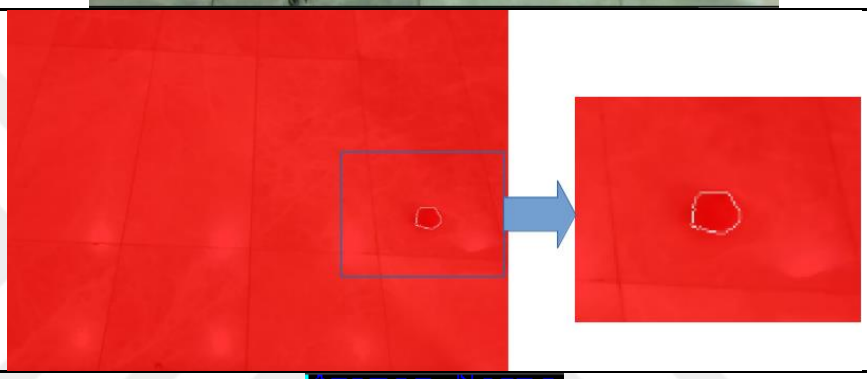
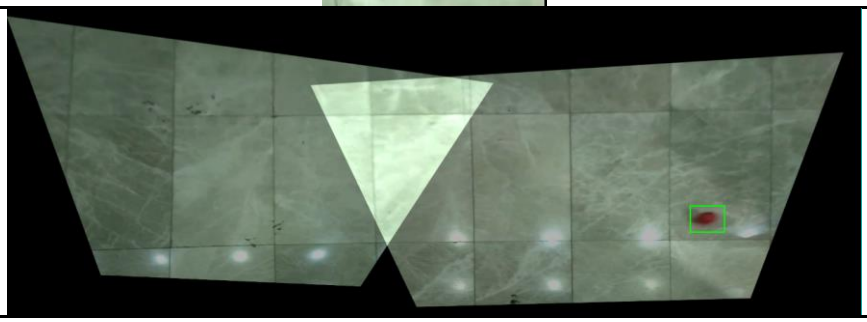
Çıktının gözlemlenmesi için farklı ışıklılık ve zemin renklerinde, farklı nesnelerle 20 adet kontrollü deney yapılmıştır. Bu deneylerde nesne her zaman 1 metre/saniye hızdan yavaş hareket ettirilmiş, yalnızca sağdan sola, soldan sağa geçiş yaptırılmıştır.

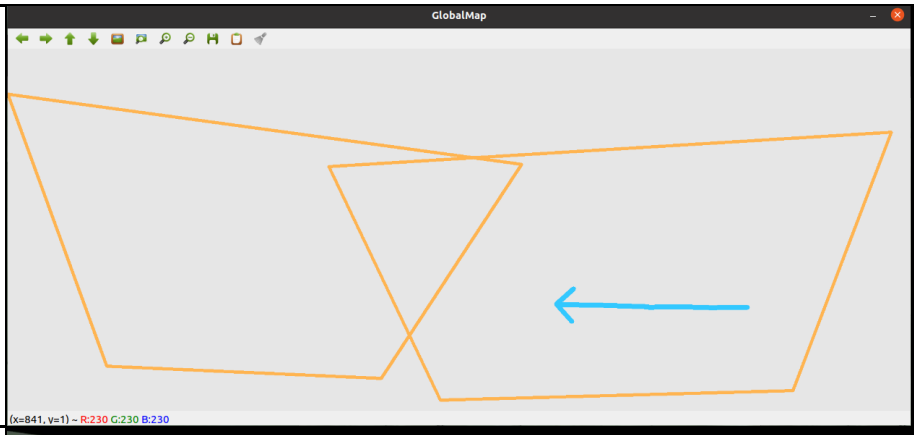
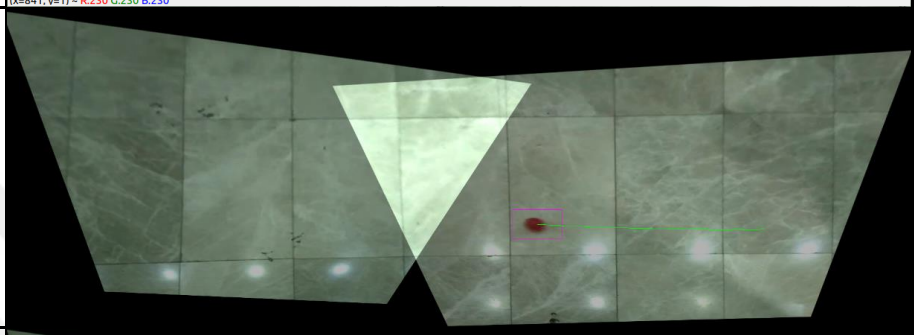
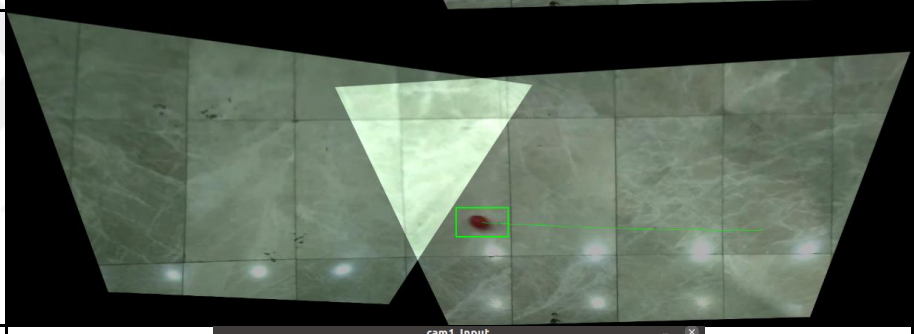
20 deney üzerinden elde edilen sonuçlar:

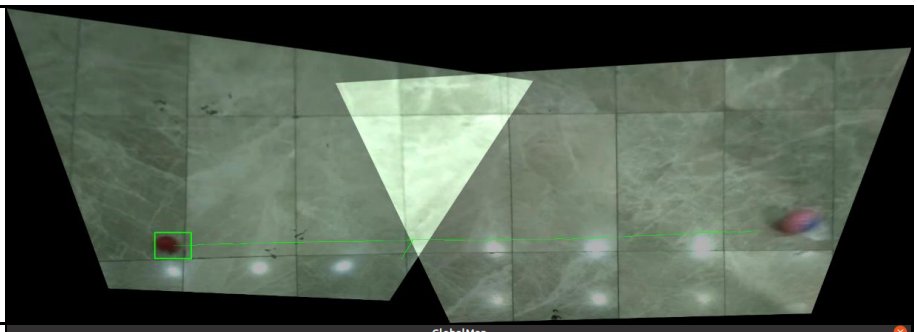
- 18'inde nesne sorunsuz tespit edilebilmiş, baştan sona sorunsuz takip edilebilmiş ve nesne görüntüden çıktığında "nesne kayboldu" mesajı görülmüştür.
- 1'inde MKKD değeri, zeminde farklı bir bölgeden güvenilir limitin üzerinde sonuç üretmiş ve nesne takip edilememiştir.
- Diğer 1'inde ise nesne görüntüden çıktığı anda başka bir bölgede MKKD yüksek çıktığı için nesnenin görüntüden çıkmadığı şeklinde yorumlanmıştır.

Tablo 3.2. Bir Nesne Takip Örneğinin Tüm Aşamaları

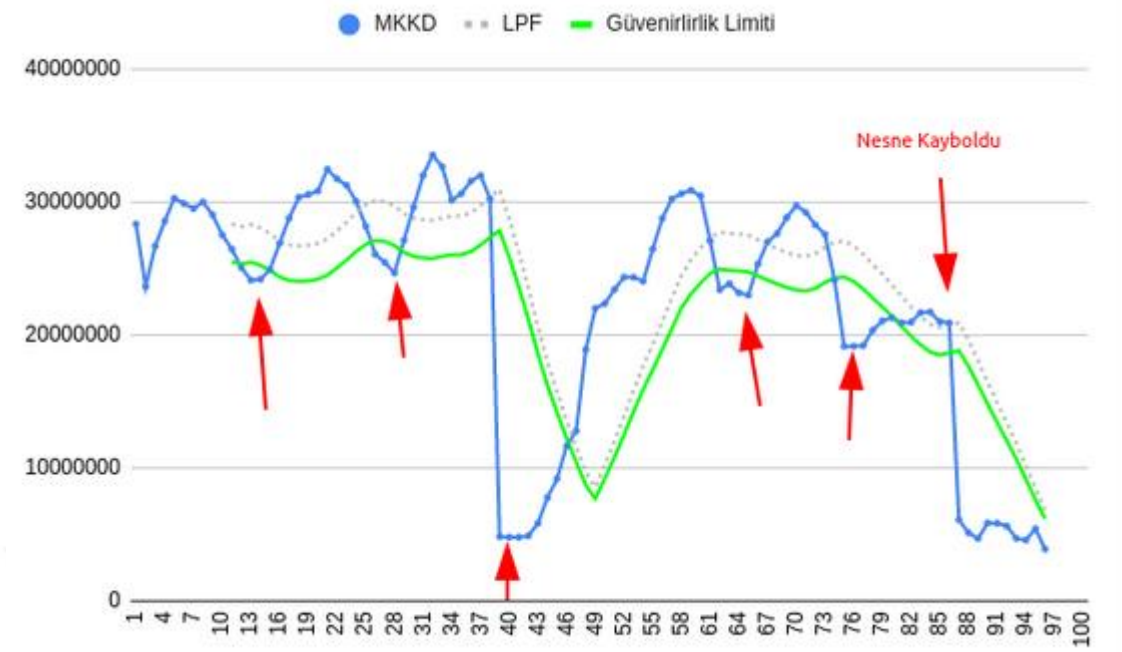
İki kamera görüntüsü ayrı ayrı alınıyor	
Kamera görüntüleri birleştirilerek kalibrasyon yapılıyor ve global harita üzerinde kuş bakışı görüntü elde ediliyor	

Kamera 2’de kırmızı nesne bilgisayar faresi ile seçiliyor.	
Watershed Segmentasyon ile nesne sınırları belirleniyor	
Nesne tespiti yapılıyor ve şablon oluşturuluyor	
Arama penceresi hesaplanıyor	
Seçilen nesne takibe başlanıyor	

Seçilen nesne hareketi global harita üzerinde çiziliyor	
MKKD güvenilir değerin altında ise çerçeve pembe renkte gösteriliyor	
MKKD güvenilir değerin üstünde ise çerçeve yeşil renkte gösteriliyor	
Nesne Kamera-2 sınırlarına soldan çarpmak üzereyken arama penceresi Kamera-1'de nesnenin çıkacağı yere taşınıyor	
Nesne bir süre sonra yeni arama penceresinde beliriyor ve eşleştirilmesi yapılıyor	

Kamera-1'de takip devam ediyor (Bu örnekte kamera geçişi anında sapma oluşmuştur.)	
Nesnenin tüm hareket rotası çıkarılıyor. Nesne görüntüden kaybolunca takip sonlandırılıyor	

Tablo 3.5'te tüm aşamaları listelenen testte elde edilen MKKD grafiği Şekil 3.5'teki gibidir. MKKD'nin güvenilirlik limiti altında olması durumunda Kalman takip yapılmıştır. Kalman takip yapılan karelerde nesne pembe çerçeve ile gösterilmiş ve grafikte kırmızı ok işareti ile belirtilmiştir. Nesne 39-46 numaralı kareler arasında kameralar arası geçiş yapmakta olduğu için bir süre diğer kamerada görüntüye girmesi beklenmiştir. Bu aşamada oluşan sapma, Tablo 3.2'de rota çiziminde de görüntülenmektedir.



Şekil 3.5 *Test Düzeneğinde Elde Edilmiş MKKD Grafik Örneği*

Bu grafik tavan ışıkları yansıması bulunan parlak zeminde elde edilmiştir. Grafikte MKKD'nin salınım yaptığı alanlar, nesnenin tavan ışıklarının yansıması üzerinden geçerken hesaplanan değerler olduğu gözlemlenmiştir. Sabit renkli zeminde yapılan takiplerde grafik daha sabit ve sorunsuz elde edilmektedir.

BÖLÜM 4. SONUÇ

Bu çalışma kapsamına model bazlı nesne sınıflandırma ve tespit metodları dahil edilmemiştir. Çalışmanın temel odağı, farklı kamera koordinat sistemlerinin ortak bir koordinat sisteminde bir araya getirilmesi olmuştur. Bu aşama "global harita" oluşturulmasıdır. Bu sayede kameraların konumları bilindiği için kameralar arası hesaplamalar yapılabilir hale gelmiştir.

Nesnenin watershed algoritması desteğiyle tespit edilmesi ve şablon arama yöntemiyle takip edilmesi; takip edilecek nesneye, zemine ve saha şartlarına göre farklılık gösterebilir. Ancak global harita oluşturma metodu pek çok farklı uygulamada farklı şartlarda kullanılabilecektir.

Bu çalışmada, bir bölgeye yerleştirilmiş kameraların görüş alanlarında hareket eden nesnenin takip edilmesi sağlanmıştır. Bu problemdeki en önemli konulardan biri, kameraların yerleşim konum ve açılarından kaynaklı olarak farklı perspektif bozunumlarıyla görüntülerin oluşmasıdır.

Lambert yansıması ile kameradan elde edilen görüntüde perspektif etkisi oluşmakta ve kameralar arasında konum/açı izdüşümleri farklı hesaplanmaktadır. Bu sebeple her kameranın ayrı kalibrasyon parametreleriyle bu etkilerin geri çatılması gerekmiştir.

Geliştirilen metot ile perspektif etkileri giderilmiş, kör noktalardaki takip devamsızlıklarına rağmen nesnelerin devamlı takibi sağlanabilmiştir.

Farklı kameraların perspektif etkilerini gideren kalibrasyon çıktıları global haritada birleştirilmiş ve tüm kameraların tek bir ortak düzlemde kuş bakışı izdüşümleri oluşturulmuştur. Bu sayede nesne eşleştirmeleri basitleştirilmiş ve takipleri yapılabilmiştir.

Şablon arama için her karede nesne özniteliklerinin değiştirilmesi bir risk barındırmaktadır. Şablon arama kesin bir sonuç üretmez, bir benzerlik değeri (MKKD) üretir. MKKD nesne harici bir konumda (veya nesnenin bir bölgesinde) yüksek çıkarsa, nesne takip edilebiliyor olacaktır. Ancak öznitelik güncellenmesinde hata payı, kümülatif olarak artacaktır. Bu da nesnenin takip edilememesine sebep olacaktır.

Yapılan testlerde her karede arama şablonunun değiştirilmemesi (yani özniteliklerin güncellenmemesi) de incelenmiştir. Bu durumda nesne tespit aşamasında alınan ilk noktanın en güvenilir bilgi olduğu varsayımıyla bu şablona güvenilmiştir.

Ancak bu yöntemde, nesne açı ve ışıklılık değişimine uğradığında MKKD düşmeye başladığı için takip başarısı düşmüştür.

Bu testler sonrasında en yüksek başarı, güvenilirlik limiti formülü desteğiyle elde edilmiştir. MKKD'nin güvenilir olduğu durumda özniteliklerin güncellenmesi, güvenilir olmaması durumunda Kalman Filtre tahminiyle takibin devam etmesi yöntemi ile çalışmanın en yüksek başarısı elde edilmiştir.

KAYNAKÇA

[1] Guopu, Z. & Qingshuang, Z. & Changhong, W. (2006) "Efficient edge-based object tracking, Pattern Recognition, Volume 39, Issue 11, 2006, Pages 2223-2226"

[2] Liu W. et al. (2016) "SSD: Single Shot MultiBox Detector. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905. Springer, Cham."

[3] Sun, D. & Roth, S. & Lewis, J.P. & Black, M.J. (2008) "Learning Optical Flow. In: Forsyth D., Torr P., Zisserman A. (eds) Computer Vision – ECCV 2008. ECCV 2008. Lecture Notes in Computer Science, vol 5304. Springer, Berlin, Heidelberg."

[4] Nascimento, J. C. & Abrantes, A. J. & Marques J. S. (1999) "An algorithm for centroid-based tracking of moving objects", 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings.

[5] Li, X & Li, K & Wang, W. & Li, Y. (2010) "A multiple object tracking method using Kalman filter, The 2010 IEEE International Conference on Information and Automation, 2010"

[6] Chen, W & Cao, L & Chen, X & Huang, K. (2014) "A novel solution for multi-camera object tracking," 2014 IEEE International Conference on Image Processing (ICIP), 2014, pp. 2329-2333, doi: 10.1109/ICIP.2014.7025472.

[7] Sun, Q. & Xu, D. & Wu, Z. (2012) "Automatic calibration of multi-camera networks," 2012 8th IEEE International Symposium on Instrumentation and Control Technology (ISICT) Proceedings, 2012, pp. 75-79, doi: 10.1109/ISICT.2012.6291656.

[8] Öztürk, B. "Gerilimdeki Dalgalanmaların Aydınlatma Elemanlarının Renk Sıcaklığına Etkisi"

[9] Akarsu, V. "Geometride uzay, düşey ve yatay açılar arasındaki fonksiyonel ilişki"

[10] https://learnopencv.com/geometry-of-image-formation , erişim tarihi: 12.10.2021

[11] Gherdevich, D. & Gonizzi Barsanti, S. & Degrassi, D. (2011) "Historic and archaeological itineraries for the discovery of Friuli during the Lombard period."

[12] Constantin-Octavian, A. (2006) "3D affine coordinate transformations"

[13] Lin-Bo, L. & In-Sung, K. & Kyeong yuk, M. & Jun W. "Low-cost implementation of bird's-eye view system for camera-on-vehicle"

[14] https://opencv.org/ , erişim tarihi: 23.04.2021

[15] Beucher, S. "Watershed, Hierarchical Segmentation And Waterfall Algorithm"

[16] Liu, X. & Wang, Z. & Wang, L. & Huang, C. & Luo, X. (2021) "A Hybrid Rao-NM Algorithm for Image Template Matching"

[17] Mohinder, S. G. & Angus, P. A. "Kalman Filtering: Theory and Practice Using MATLAB, Second Edition"

[18] https://ai-benchmark.com/ranking_deeplearning_detailed.html , erişim tarihi: 26.12.2021