İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

AC	AKD sonrasında oluşan değerlerden DC haricindekiler
C_{uv}	I çerçevesine AKD uygulanmış hali
DC	AKD sonrasında oluşan en baskın değer olan sol üst köşe elemanı
F	Kameranın konumu
$F(h,t,k)$	Ekran üzerindeki bir pikselin yol düzlemindeki konumunu bulan fonksiyon
h	Kameranın yerden yüksekliği
$Hız_{t2-t1}$	Belirli zaman aralığındaki ortalama hız
I_{xy}	Görüntünün (x,y) koordinatındaki piksel değeri
k	Görüntü merkez noktasının görüntü alt çizgi merkezine uzaklığı
$m(x,y)$	Makro bloğun ortalama renk değeri
N	Cismin yol düzlemindeki konumu
O	Cismin ekran düzlemindeki konumu
t	Görüntü alt çizgisinin ortasının kamera direği dibine olan uzaklığı
T	Doku süzgecindeki eşik değeri
$v(x,y)$	Makro bloğun varyansı
ΔKonum	Belirli zaman aralığındaki yer değiştirme
Δt	Belirlenmiş zaman aralığı

KISALTMA LİSTESİ

B	Bi-directional predicted frames
AKD	Ayrık Kosinüs Dönüşümü
EM	Expectation Maximization
GOP	Group Of Picture
GKM	Gauss Karışım Modeli
HMM	Hidden Markov Model
I	Intra-compressed frames
TAKD	Ters Ayrık Kosinüs Dönüşümü
IEC	International Electrotechnical Commission.
ISO	International Organization for Standardization
MPEG	Motion Picture Experts Group
MV	Motion Vector
P	Forward predicted frames
RLE	Run Length Encoding
ROI	Region Of Interest

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖNSÖZ

En başta ve öncelikle danışman hocam Sayın Prof. Dr. Ahmet Coşkun Sönmez beye yüksek lisans programı boyunca beni yönlendirdiği, bilgisini ve değerli görüşlerini bana aktardığı beni her daim motive ettiği için çok teşekkür ederim.

Tez konusu olarak seçtiğim trafik kontrol kameralarıyla sağlanan görüntüden trafik parametreleri çıkarımı konusunu inceleyebilmem için gerekli verileri sağlayan İstanbul Büyükşehir Belediyesi Trafik Kontrol Merkezi yetkililerine, Ayrıca MPEG Parser programını kullanmamıza izin verdiği için Sayın Dr. Fatih Porikli beye teşekkürü bir borç bilirim.

Çalışmalarımı yürütebileceğim güzel bir ortamı oluşturan ve her aşamada desteğini esirgemeyen YTÜ Bilgisayar Mühendisliğinin değerli akademik personeline en derin şükranlarımı sunarım.

Hayatım boyunca sen koşmana bak arkanda biz varız diyen aileme teşekkürlerin en büyüğü…

ÖZET

İstanbul'un trafiğini tespit etmek amacıyla kurulan İstanbul Büyükşehir Belediyesi Trafik Kontrol Merkezi'ne bağlı İstanbul'un önemli noktalarına yerleştirilen sayıları 2006 yılının sonu itibariyle 150'ye kadar çıkan trafik kontrol kameraları mevcuttur. Bu kameraların başında bulunan operatörler, görüntüleri devamlı izleyerek acil bir durumda gerekli yerlere çağrı yapmakta, oluşan trafik sıkışıklığını not etmektedirler. Bu iş insan bağımlı olarak gerçekleştirildiğinden insan gücü masrafı ve insandan kaynaklanan bazı hatalı sonuçlar olabilmektedir.

Tez çalışması boyunca geliştirilen sistemle yoğunluk durumunu, yoldan geçen araçların ortalama hızları gibi planlamada önem arz eden iki parametre çıkartılmıştır. Bu sayede kamera görüntüleri otomatik işlenip trafik planlaması için gerekli parametreler elde edilmiştir.

Araçların ortalama hızları, MPEG sıkıştırma biçimi içinde saklı bulunan hareket vektörleri kullanılarak hesaplanmıştır. Bu sayede karmaşıklığı yüksek olan optik akış çıkarımı, düşük maliyetli algoritmayla gerçeklenebilmiştir. Bu hareket vektörleri geometrik dönüşüme sokularak yol düzlemindeki gerçek hareket miktarları hesaplanmıştır.

Yol yoğunluğunun tespitinde önemli bir adım olan arka plan çıkarımı, olasılıksal bir model olan Gauss Karışım Modeli ile gerçeklenmiştir. Ön plan ise, temel fark işlemi ve süzgeçlerle işaretlenmiştir. Daha sonra, araçların kapladığı alan, yolun kapladığı alana oranlanarak yoğunluk bilgisi elde edilmiştir.

Geliştirilen sistemin performansı, Trafik Kontrol Merkezine ait 10 değişik kameradan elde edilen toplam 200 dakikalık MPEG kayıtları kullanılarak ölçülmüştür. Buna göre hız çıkarımının her türlü çevre şartlarında %10 hatanın altında çalıştığı, yoğunluk çıkarımının ise gece ve ışık miktarının oldukça hızlı değiştiği bulutlu günler dışında istenen performansı sağladığı görülmüştür.

Anahtar Kelimeler: Trafik Akışı, Hız Ölçme, MPEG, Trafik Parametreleri, Araç Yoğunluğu, Geometrik Düzeltme, Arka Plan Çıkarımı

ABSTRACT

Metropolitan Municipality of Istanbul establishes a foundation called “Traffic Control Centre”. This foundation has nearly 150 traffic cameras (in 2006) at the important points of Istanbul for watching the traffic situation. Operators watch this cameras scene continuously, they declare situation if undesirable event occurs. This system is occur waste of manpower, and cause of depending manpower, there may be a lot of errors.

During thesis, developed system extract two most important parameter from traffic cameras are vehicle density and average vehicle velocity. By this mean we provide data for traffic planning automatically.

Average vehicle velocity is extracted by motion vector which exist in MPEG format. By this mean optical flow is found by less complexity algorithm. Geometric correction was implemented to this motion vector for finding real motion.

Background estimation which is the most important step for finding average vehicle density was calculated by probability based model that called Gaussian Mixture Model. For calculate foreground, basic subtraction to current image from background image and some noisy filter was implemented. Ratio of foreground and background was declared average vehicle density.

For measuring the performance of developed system, used nearly 200 minutes traffic cameras record which is belong 10 different cameras. Velocity extraction performs every environmental situation under %10 error rate, but average vehicle density extraction is not successful at night, or variable light conditions as cloudy days.

Keywords: Traffic Flow, Velocity Measurement, MPEG, Traffic Parameters, Vehicle Density, Geometric Correction, Background Estimation

1. GİRİŞ

Günümüzde trafik akışını tespit etmek için kullanılan yöntemlerden en popüleri yolun alt ya da kenarına konumlandırılan ve yollardan geçen araçları sayma yetisine sahip cihazlar olan metal detektörlerle geliştirilen sistemler olmasına karşılık, kamera takip sistemlerinin de önemli sayılabilecek avantajları vardır. İstenen trafik parametrelerinin (yol yoğunluğu, akış hızı) kameralı sistemlerle bulunabilmesi, kameraların metal detektörlere göre daha ucuza kurulması ve idame edilmesi aynı zamanda arızaya karşı daha dayanıklı olmaları, bütün bunların yanında kameraların metal detektörlere göre daha fazla bilgi taşıması (metal detektörler sadece metal yoğunluğunu ölçerken kameralarla yolun görüntüsünü elde ediyoruz) kameralı sistemlerin en önemli avantajlarındandır.

Trafiğin, özellikle şehir içi bağlantı yollarındaki araç trafiğinin yönetilmesi diğer bir ifadeyle mevcut yolların nasıl dizayn edileceği, yapılacak yolların nereleri nerelere bağlayacağı kararının verilebilmesi için doğruluğundan emin olduğumuz oldukça yüksek miktarda trafik durum verisine (yolların değişen zamana göre araç yoğunluğuna ve akış hızına) ihtiyaç duyulmaktadır. Bütün bu verilerin toplanması için stratejik noktalara algılayıcılar (metal detektörü ya da kameralar) konulmalı ve bu algılayıcıların ürettiği veriler toplanmalıdır. Bütün bu verileri değerlendirmek için insan gücünün kullanılması yerine geliştirilecek otomatik değerlendirme yazılımları sayesinde bu zahmetli iş kolayca hem de daha yüksek doğrulukta bilgisayarlara yaptırılabilmektedir.

Trafik parametrelerinin trafik takip kameralarıyla bulunmasını kapsayan çalışmalar genelde sabit kameralarla yapılmıştır. Sullivan'ın geliştirdiği sistem araçları 3 boyutlu uzayda bulup takip etmektedir. Bu sayede 2 boyutta arka planı ya da birbirlerini örten araçları ayırt etmeyi başararak performansı arttırmıştır. Beymer ve arkadaşları yaptığı çalışmada ise özellik tabanlı bir izleme algoritması uygulamışlardır. Resimdeki bazı özel noktaların kullanılması sayesinde yöntemin karmaşıklığı azaltılıp çevre faktörlerinden etkilenmesi azaltılmıştır. Koller'in önerdiği yöntem ise kontur tabanlı ve Kalman filtreleriyle yörünge kestirimli bir izleme algoritmasıdır. Bu çalışmada araçların yörüngeleri çıkartılıp, dinamik ağlarla yolların durumları sınıflandırılmıştır. İzleme tabanlı sistemlerin en önemli dezavantajı izleme algoritmasının performansı genel başarıyı direk etkilemesidir. Işıklandırma koşulları, araçların hızlarında meydana gelecek ani değişiklikler ve takip edilen araçların birbirini örtmesi ya da araya başka bir cisim girmesi sistemi başarısız kılmaya yetecektir. Cuchiara'nın önerdiği yöntemde şehir içi trafik görüntülerinden kural tabanlı bir sistemle araçlar bulunmaktadır.

Çalışmada 6 trafik durumu tanımlanmış ve test edilmiştir. Shuming trafik kazalarını tespit etmek için hareketin olduğu alanların piksel değerlerini kullanan parametresiz regresyon metodu kullanmıştır. Maurin optik akış, Kalman filtreleri ve damlacık birleştirme tekniklerini kullanan çok seviyeli bir yaklaşım kullanmıştır. Yu trafik bilgilerini sıkıştırılmış videoları kullanarak bulmuştur. Hareket eden bloklarının toplam bloklara oranından yola çıkarak geliştirilen izleme tabanlı bir yöntemdir. Porikli ise MPEG sıkıştırma formatının içinde saklı bazı özelliklerden (hareket vektörü, AC katsayısı, DC katsayısı..vb) oluşturduğu özellik vektörünü 5 ayrı durumu belirten HMM (Hidden Markov Model)'de eğitip sonuç bulmuştur. Bu çalışmada yol durumu boş, az yoğun, akışkan, yoğun, tıkalı gibi 5 ayrı durumdan biri kullanıcıya bildirilmektedir. Bu yapılan çalışmaların çoğu özel tasarlanmış kameralar kullanılarak ve karmaşıklığı çok yüksek olan algoritmalarla gerçekleştirilmiştir.

2. KAPSAM

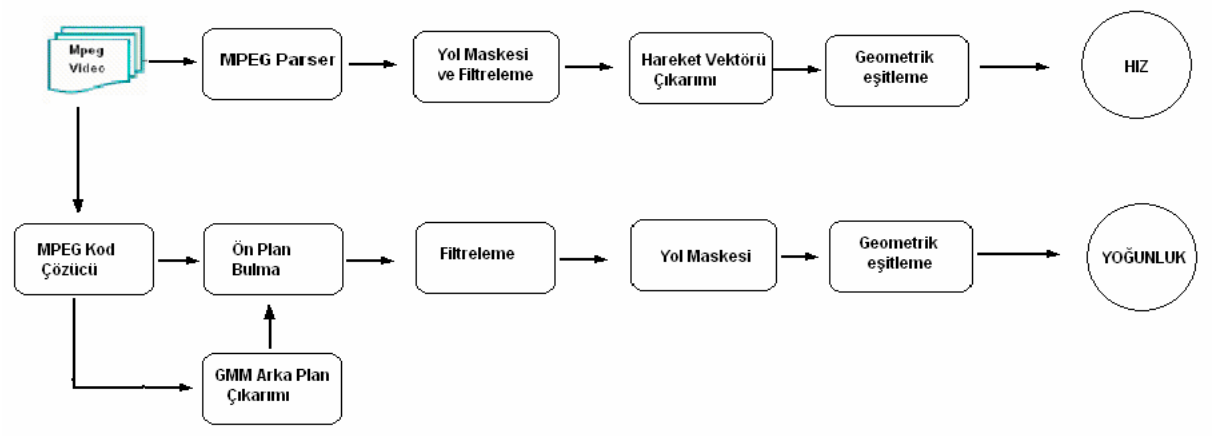
Trafik parametreleri, Beymer ve arkadaşlarının yaptığı çalışmada birim zamanda geçen araç sayısı, araçların ortalama hızları, birim alandaki araç-yol oranı, araçlar arasındaki ortalama mesafe şeklinde tanımlanmıştır. Bu çalışmada ise söz konusu 4 parametreden ikisinin çıkarımı üzerinde durulmaktadır. Bunlar araçların ortalama hızları ve birim alandaki araç-yol oranı şeklindedir. Bu iki parametre aralarında belli ölçüde korelasyona sahip olmasına rağmen ayrık parametreler gibi düşünülüp tamamen ayrı metotlarla çıkarımı yapılmaktadır.

Bu çalışmada, karmaşıklığı düşük, değişen çevre şartlarından (ışık miktarı, kameraların konumu) olabildiğince az etkilenen, doğruluğu yüksek bir trafik parametre çıkarım yöntemi önerilmiştir. Daha önceki birçok çalışmada yapıldığı gibi araçları tek tek izlemek yerine, sıkıştırılmış haldeki görüntülerin bazı özelliklerini kullanarak ayrılmış her bir yol için ortalama hız değerleri ve arka plan çıkarımıyla da araç yoğunluğu bulunmaya çalışılmıştır. Günümüzde çoğu kayıtlar Motion Picture Experts Group (MPEG) formatında tutulduğundan çalışmamız MPEG formatındaki video kayıtlarını işleyecek şekilde geliştirilmiştir.

Video işlerken yapılacak işlemleri sıkıştırılmış veri üstünde gerçekleştirmemiz bize karmaşıklığı azaltma yönünden oldukça avantaj sağlamaktadır. Aksi takdirde videoların önce kod çözücülerle çözülmesi gerekir. Aynı zamanda sıkıştırılmış veride saklı bulunan renk özelliği, doku özelliği, kenar özelliği ve bazı uzaysal özelliklerin yanında çalışmamızda çok önemli yeri olan hareket eden blokları tutan yer değişim matrisi gibi bazı önemli bilgiler hâlihazırda mevcutken bu verilerin kullanılmaması düşünülemez.

Geliştirilen sistemle, sıkıştırılmış veride bulunan hareket vektörlerine geometrik düzeltme uygulanarak yol düzleminde birim zamanda ne kadar hareket belirttiği ve buna bağlı olarak hız bilgisi elde edilmeye çalışılmıştır. Ayrıca piksel düzleminde her bir pikselin renk bilgisi olmaksızın sadece ışık bilgisini kullanılarak arka plan öğretmensiz öğrenen bir sınıflandırma modeli olan Gauss Karışım Modeli (GKM) ile çıkarılmakta, böylelikle yol üzerinde hareket eden araçlar işaretlenmektedir. Bu sayede yol yoğunluğunu bulmak kolaylaşmaktadır.

Geliştirilen sistemin akış şeması Şekil 2.1'deki gibidir. Yapılan işlemler birbirinden bağımsız hız ve yoğunluk bulma olarak iki ana başlık altında incelenebilir.



Şekil 2.1 Önerilen sistemin akış şeması

Geliştirilen sistemde trafik planlaması için en çok ihtiyacımız olan 2 parametre birbirinden bağımsız ayrı işlemlerle hesaplanmaktadır. Yoğunluk bulma işlemi şu 5 ana görevden oluşmaktadır.

- MPEG kayıtlarının çözülmesi
- Arka Plan Çıkarımı
- Ön Planın Bulunması
- Filtreleme
- Geometrik Düzletme

Hız bulunurken izlenen adımlar ise şu şekildedir.

- MPEG Ayrıştırma
- Yol Maskesi ve Filtreleme
- Hareket Vektörü Çıkarımı
- Geometrik Eşitleme

3. MPEG AYRIŞTIRICI

Görüntüleri çözülmüş düzlemden ziyade sıkıştırma düzleminde işlemenin bize kazandıracağı zaman tasarruf sistemimizin en çarpıcı yanıdır. Zaman tasarrufunun yanında alınan enstantanenin karakterini belli eden yüzey doku bilgisinin ve kenar bilgisinin doğrudan okunması bize avantaj sağlamaktadır. Aynı zamanda sıkıştırma algoritması gereği görüntüyü oluşturan 16*16'lık blokların yer değişimini gösteren hareket matrisinin düşük karmaşıklı bir algoritmayla hesaplanabilmesi, amacımıza ulaşmamızda bize yardımcı olacak en önemli unsurlardan biridir.

Sıkıştırılmış veride işlem yapmanın bazı kısıtları da elbette vardır. Sıkıştırmada kullanılan Ayrık Kosinüs Dönüşümü (AKD) yüzünden uzaysal korelasyon piksel olarak değil de kodlanan blok olarak yapılacağından bir boyut azalması söz konusu olacaktır. Diğer bir ifadeyle hareket vektörü hesaplanırken resimleri oluşturan 16*16'lık bloğu bir piksel gibi düşünmek zorunda kalacağız. Bu yüzden bizim en önemli özeliklerimizden birisi olan Hareket Vektörü (MV) bize tam olarak optik akışı veremez. Bu vektörün bazı yanlış eşleşmelerden ve yuvarlamalardan dolayı oluşan hataları da içermesi beklenir. Bu yüzden MPEG sıkıştırmalı kayıtların hiçbir zaman bu tür hatalardan muaf olması düşünülemez.

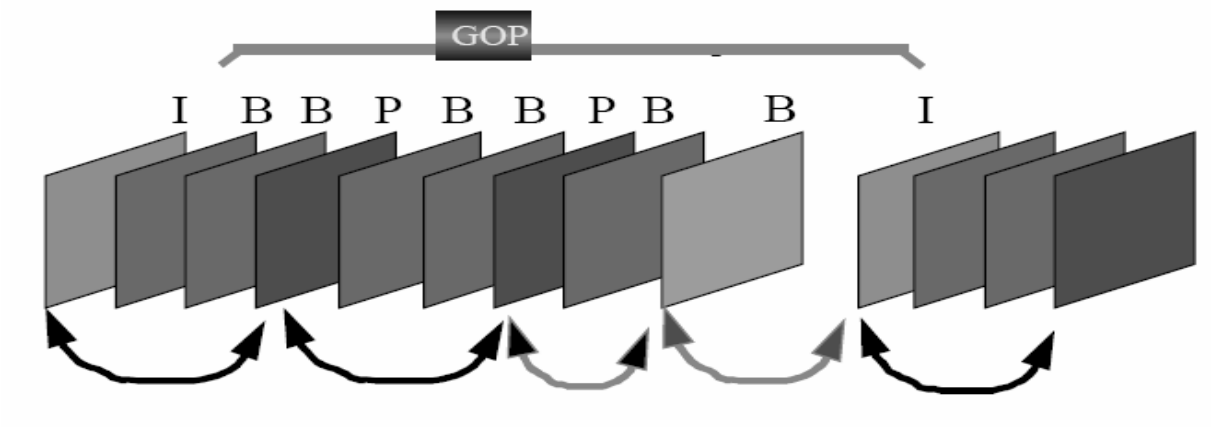
3.1 MPEG Nedir?

Ses ve görüntü verilerinin sayısal ortamda sıkıştırılması ve geri açılması prensiplerini belirleyen bir standarttır. Bu standart adını Hareketli Görüntü Uzmanları Birliği (**M**oving **P**icture **E**xperts **G**roup; **MPEG**)'nin baş harflerinden almaktadır. Aynı zamanda ISO/IEC'nin sayısal olarak kodlanmış ses ve görüntü temsili için standart geliştirmekten sorumlu çalışma birliğidir.

Sıkıştırma işleminde, görüntü kareleri arasındaki değişimler analiz edilir ve bir MPEG kodlayıcı (encoder) ile dosya boyları yaklaşık 1/200 oranında sıkıştırılır. MPEG, bilgisayar ve yayın (tv ve radyo) sektörleri tarafından en geçerli standartlardan biri olarak kabul edilmiştir. Mevcut MPEG versiyonları; MPEG-1, MPEG-2, MPEG-3 ve MPEG-4 'tür. Çalışmada konu edilen versiyon MPEG-1/2 standardıdır.

MPEG sıkıştırmasının altında yatan basit gerçek her bir çerçeve için uzaysal veri azaltma ve peşi sıra takip eden çerçeveler arası ise zamansal veri azaltmadır. AKD tabanlı sıkıştırma

uzaysal veri azaltmayı, hareketli kısımların kodlanması ise zamansal veri azaltmayı sağlamaktadır. MPEG sıkıştırma algoritması gelen görüntüleri I (intra-compressed), P (forward predicted) ve B (bi-diractional predicted) adında 3 çerçeve şeklinde kodlamaktadır. Bu üç tür çerçeve içinde I çerçeveleri aslında geçmiş ya da gelecek hiçbir referans resme ihtiyaç duymadan resmi oluşturabilecek verilere sahiptir. Diğer bir ifadeyle bu çerçeveler, o anlık resmi olduğu gibi zamansal bir sıkıştırma uygulamadan saklamaktadır. P ve B çerçeveleri ise hareket bilgilerini tutmaktadırlar. I çerçeveleri hareket ile ilgili hiçbir bilgi tutmazken, kendi renk ve doku bilgilerinin yanında kendisinden sonra gelen B ve P çerçevelerinin renk doku bilgilerinin de referanslarını tutmaktadır. P çerçeveleri kendinden hemen önce gelen I ya da P çerçevelerini referans alarak resmi oluşturur. B çerçeveleri ise bir önceki ve bir sonraki I ve ya P çerçevelerini referans alarak o anlık resmi oluşturur. Yani B çerçevelerinde her iki taraftan da (geçmiş ve gelecek) referans alarak, o anlık resmi oluşturabilecek veriler bulunmaktadır.

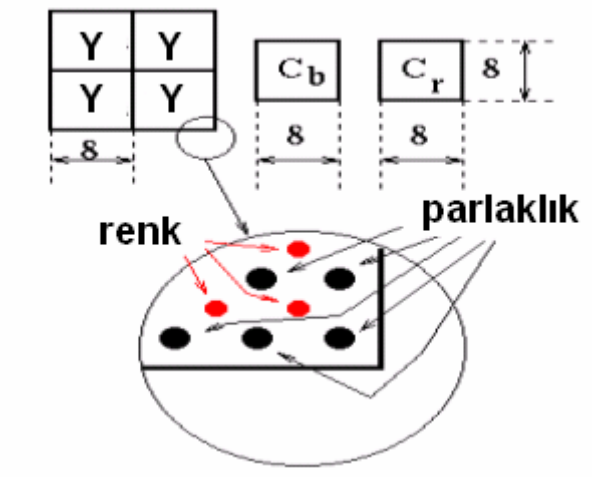


Şekil 3.1 GOP'un içeriği

I, B ve P çerçevelerinin değişik sayılarda bir araya gelmeleriyle Group Of Picture (GOP) oluşur. Bir GOP'ta bulunan çerçeve türleri ve her bir tür çerçevenin nasıl dizileceği videoya göre değişmektedir. Bu çerçeveler en az yer kaplayacak şekilde dizilmektedir. En yaygın olarak bulunanlar 12 ya da 15 çerçeveden oluşan GOP'lardır. Yine en yaygın dizilim şekli IBBPBBPBBPBB şeklindedir. Her bir GOP'ta yalnız bir I çerçevesi olurken diğer türler muhtelif sayıda olabilmektedir. Yukarıdaki dizilimden sonra bir sonraki GOP'un I çerçevesi gelir ve videonun sonuna kadar bu şekilde GOP'lar sıralanır.

MPEG kodlamada çerçeveler Şekil 3.2'deki gibi 16*16'lık makro bloklara bölünmüştür. Her

bir makro blok 4 adet 8*8'lik parlaklık bloğu (luminance,Y bileşeni) ve 2 adet 8*8'lik renk bloğu (chrominance, Cb ve Cr bileşeni) içermektedir. Tabii bu oran kodlamanın cinsine göre değişiklik gösterir. Bahsi edilen kodlamaya 4:2:0 makro blok yapılı kodlama denir. Eğer renk özelliklerinde de kaybolmaması istenirse yani bir makro blok 4 adet ışık ve Cr ve Cb değerlerinden de 2'şer olmak üzere 4 adet de renk bloğu içerdiği zaman buna 4:2:2 makro blok yapılı kodlama denir. MPEG 1-2 türden kayıtlar daha çok 4:2:0 yapısında bulunmaktadır.

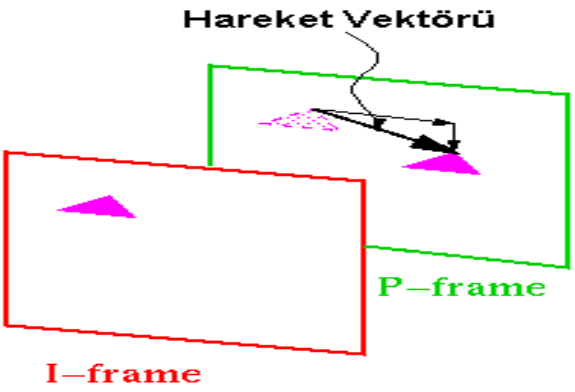


Şekil 3.2 Makro blokların içeriği

İnsan gözü ışık miktarına karşı çok hassas olduğu için Y bileşeninde herhangi bir azaltma yapılmamıştır. Fakat renk bileşenleri için aynı şeyi söylemek zordur. Her bir 2*2'lik bloğun yani 4 pikselin renk değeri, bu değerlerin ortalaması olan tek bir renk değeri şeklinde saklanmasıyla renk blokları oluşturulmuş olur. Hareket vektörleri işte bu makro bloklar bazında çıkarılacaktır. Diğer bir ifadeyle hareket vektörleri söz konusu 16*16'lık blokların yer değişimlerini göstermektedir. Bir I çerçevesinin makro blokları kendi resmini oluşturan bilgileri taşırken, P çerçevesinin makro blokları kendinden hemen önce gelen çerçevenin ne kadar değiştiği bilgisini taşımaktadır. İşte hareket vektörünü çıkarmak bu gerçeğe dayanmaktadır.

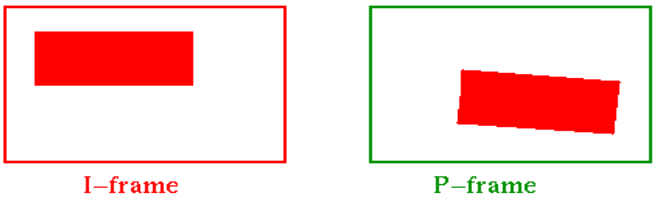
MPEG kodlamanın nasıl çalıştığına biraz daha somut örnekler üstünden bakalım. Küçük üçgen Şekil 3.3'deki gibi I çerçevesinin sol üstünde durduğunu varsayalım. Bir sonraki enstantanede üçgenin sağ alta doğru biraz hareket ettiğini düşünelim. Hareket vektörü üçgeni oluşturan blokların bir sonraki çerçevede x ve y koordinatlarında nasıl bir değişim yaptıklarını

tutan vektördür ve bu vektör I çerçevesinde bulunmayıp sadece B ve P çerçevesinde bulunmaktadır. Söz konusu hareket vektörü MPEG verilerinin bir parçasıdır. Hareket vektöründeki değerler pozitif ve ya negatif olabilirler. Pozitif değerler x ekseninde sağa, y ekseninde aşağı manasına; negatif değerler ise x ekseninde sola, y ekseninde yukarı manasında gelmektedir. Hareket vektörünün değeri [-64 +63] aralığındadır. Bu da bir makro bloğun en fazla 64 piksel x ekseninde, 64 piksel y ekseninde hareket ederse blokların eşleşebileceği, aksi halde blok eşlemesi yapılamayacağı anlamına gelmektedir.

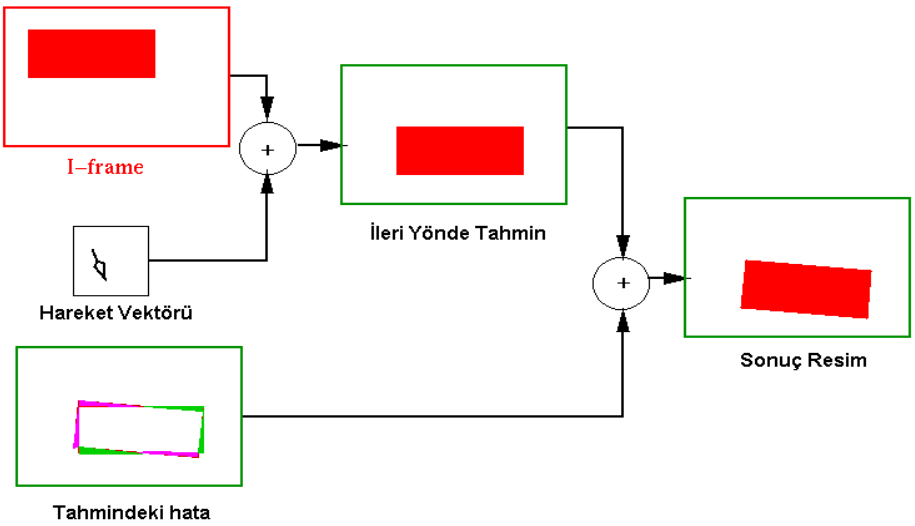


Şekil 3.3 Hareket vektörü

Sadece hareket vektörü çoğu zaman bir sonraki çerçevedeki resmi tam olarak anlatmaya yetmez. Bazen söz konusu obje yer değiştirmenin yanında dönme hareketi de yapar. Bu durumda devreye hata miktarı girmektedir.



Şekil 3.4 Dönme hareketine örnek



Şekil 3.5 Hata payının etkisi

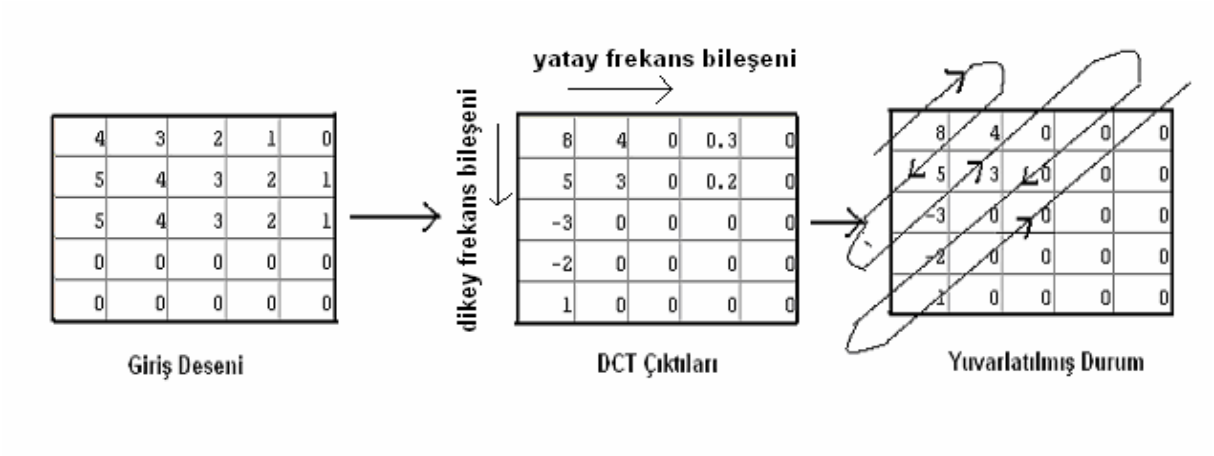
Şekil 3.4'deki duruma bakacak olursak dikdörtgenin hareketinin sadece x ve y koordinatlarındaki yer değişimi olarak verilemeyeceği açıkça görülmektedir.

Söz konusu hareket sadece hareket vektörüyle temsil edilirse dönme hareketi temsil edilemeyeceğinden yanlış resim oluşturulur. Fakat hata değerleri de çerçevelerde saklandığından hareket vektöründen sonra hata vektörü de resme eklendiğinde sonuç resmini bulmuş oluruz. Burada önemli olan husus şudur: Hata vektörünün boyutu olabildiğince küçük olmalıdır ki sıkıştırma yüksek verimde olabilsin. Eğer birbirini peşi sıra takip eden çerçeveler arası hata vektörü çok büyürse bu durumda algoritma gereği kodlama değiştirilecek ve GOP'un içine artık daha fazla I çerçevesi konacaktır.

Blokların uzaysal eksenden frekans eksenine geçirilmesi, sinyalleri bağımsız frekans bantlarına bölen AKD ile gerçekleşir. AKD katsayıları giriş deseninin konumsal frekansını verir. Oluşan AKD katsayıları 11 bitlik çözünürlüğe göre kuantize edilir. Bu kuantize işlemi sonunda ara değerler 11 bite göre sınırları ve aralıkları belirlenmiş sayılara yuvarlandığı için biraz veri kaybı söz konusu olur. Böylelikle uzaysal sıkıştırma işlemi gerçekleşir. Bu işlem sadece I çerçeveleri oluşturulurken yapılır. B ve P çerçevelerinde uzaysal piksel değeri bulunmayıp sadece ileri ya da her iki yönde hareketi kestiren yer değişim vektörü bulunur. AKD işlemlerinin matematiksel formülü aşağıdaki gibidir.

$$C_{uv} = \frac{1}{N} K^2 \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I_{xy} \cos\frac{\pi u(2x+1)}{2N} \cos\frac{\pi v(2y+1)}{2N} \qquad (3.1)$$

N sayısı AKD'si alınacak kare giriş deseninin boyutunu gösterir. Şekil 3.6'daki örneğimiz için düşünürsek N=5 olacaktır. u,v parametreleri ise oluşan çıktı deseninin indislerini tutmaktadır. 5*5'lik girişe karşı yine 5*5'lik sonuç üretileceğinden $(u,v = 0,...,N-1)$, olmalıdır. x ve y parametreleri ise giriş deseninin indislerini göstermektedir. Çarpan olan K'nın değeri ise $u,v = 0$ ise $K = \frac{1}{\sqrt{2}}$, diğer durumda ise K=1 olmalıdır. Söz konusu formülü (3.1) Şekil 3.6'daki giriş verisine uygularsak oluşacak olan çıktılar yine Şekil 3.6'dadır.



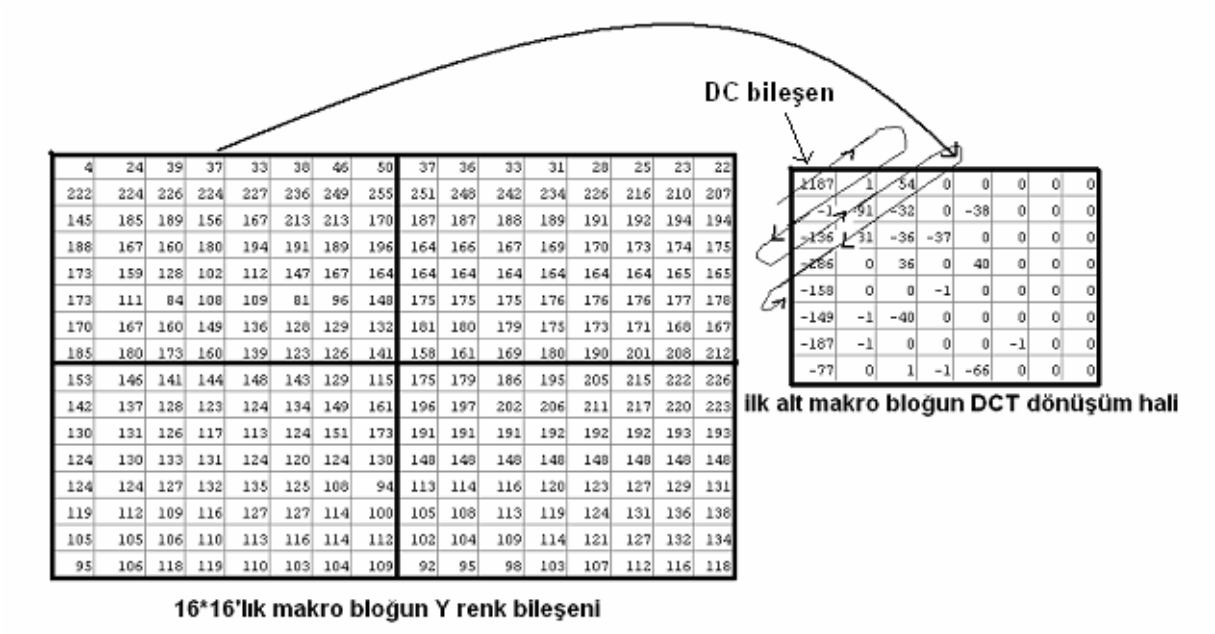
Şekil 3.6 AKD çıktılarının kodlaması

AKD dönüşümünün çıktıları değerlendirilirken sol üst köşe noktası en baskın değerler, sağ alt köşenin ise küçük değerlerle dolu olduğu görünmektedir. Sol üst değerler giriş desenini en güçlü şekilde ifade eden frekans değerleri olduğu için buradaki değerler daha değerlidir. Ondan dolayıdır ki giriş desenini en güçlü ifade eden sol üst noktadaki yani $u,v = 0$ olduğu bileşenin değerine DC parametresi, diğerlerine ise AC parametresi denilmektedir. Bu durumda örneğimiz için DC değeri 8 iken geriye kalan 24 değer AC değerlerini oluşturmaktadır. Formül (3.1)'e bakacak olursak $u,v = 0$ olduğu durumda $\cos(0) = 1$ olacağından DC değeri her zaman giriş verisinin bütün elemanlarının toplamının bir katı olacağı açıktır. Bu durumda DC değeri için giriş verisinin ortalamasıdır denilebilir. Yani DC parametre bir nevi N*N'lik blokların kuantalanmış halidir. Çıktılar yuvarlatıldıktan sonra zig-zag şekilde okunup önce Huffman ardından da Run Length Encoding (RLE) algoritmasıyla sıkıştırılır. Yukarıdaki örnek durum incelendiğinde sıkıştırılmamış saf veriler " 8 4 5 -3 3 0 0 0 0 -2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0" şeklinde olacaktır. RLE algoritması en uzun zinciri kodlayacağından ve kodlarken önce kaç kez geldiği sonra piksel değeri formatında olduğundan, söz konusu matris "8 4 5 -3 3 *4 0 -2 1 *14 0" şekline çevrilecek ve veri bu

şekilde saklanacaktır.

3.2 I Çerçevelerinin Kodlanması

Bir I çerçevesi kodlanırken resim öncelikle 16*16'lık makro bloklara ayrılır. Her bir makro bloğu oluşturan piksellerin YCbCr renk uzayındaki değerlerinden Y, yani ışık değerleri kodlanırken makro blok kendi içinde eşit 8*8'lik 4 alt makro bloklara bölünür ve her bir alt makro blok ayrı ayrı AKD fonksiyonundan geçirilip oluşan çıktılar zig-zag sıralamasıyla sıralanıp Huffman kodlama ve RLE algoritmasıyla sıkıştırılır.



Şekil 3.7 I çerçevesindeki Y bileşeni kodlanması

Şekil 3.7'de görülen işlem her bir alt makro blok için yapılır. Dolayısıyla YCbCr renk uzayında kodlanan resmin Y bileşeninde herhangi bir kayıp söz konusu olmaz.

MPEG kodlamada resimlerin ışık değerleri 8*8'lik bloklar halinde alt makro bloklara bölünüp AKD dönüşümüne sokulduklarından her bir blok için 8*8=64 adet frekans bileşeni elde edilir. Her bir alt makro bloğun AKD dönüşüm sonucunda elde edilen 64 değerden sadece sağ-üst değeri yani 1.satır 1.sütün elemanı kullanılarak oluşan matrisi ters AKD (TAKD) dönüşümüne sokarsak elde edeceğimiz resme DC bileşeni, diğer bütün elemanların teker teker kullanılarak elde edilen resimlere ise AC bileşeni denir. Bu durumda her çerçeve için 1 adet DC, 63 adet ise AC resmi elde edilir. Burada bahsi gecen Ters Ayrık Kosinüs Dönüşümü

(TAKD) (3.2)'deki gibi tanımlıdır.

$$I_{xy} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \frac{1}{N} K^2 C_{uv} \cos \frac{\pi u (2x+1)}{2N} \cos \frac{\pi v (2y+1)}{2N} \tag{3.2}$$

değişkenlerin tanımı (3.1)'deki gibidir.

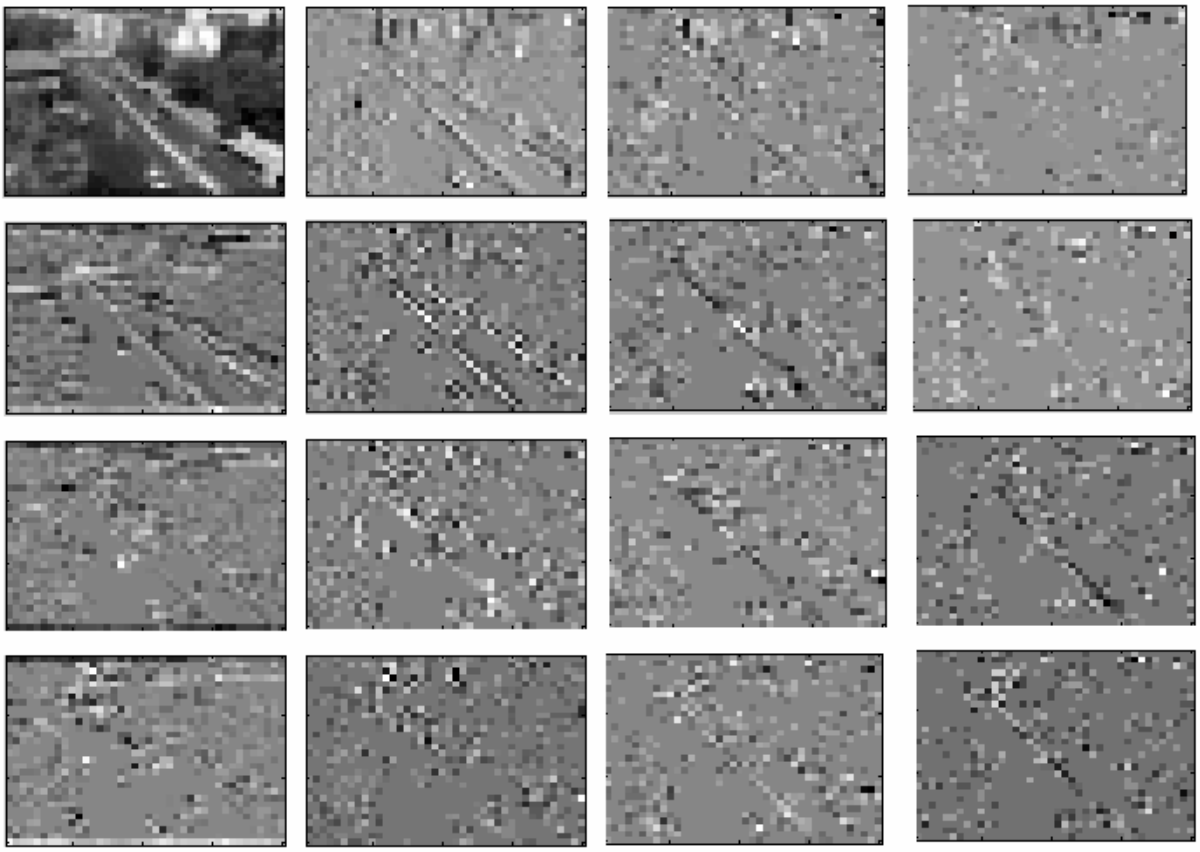
Şekil 3.8'deki örnek resme göre elde edilen 1 adet DC bileşen ve AC bileşenlerin anlamı en yüksek 15 tanesi Şekil 3.9'daki gibidir.

Şekil 3.8 Trafik kameralarından bir görüntü (240x320 piksel)

Şekil 3.9'da sağ-üstteki resim DC bileşen resmi adını almaktadır. Bu resme dikkatli bakıldığında orijinal resmin 8*8 bloklar halinde kuantalamış hali olduğu anlaşılır.

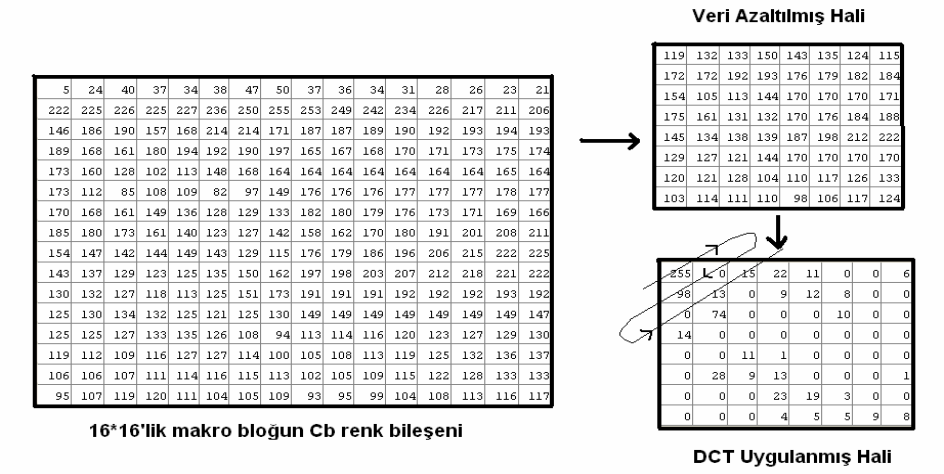
Şekil 3.9'daki her bir resim, AKD dönüşümü uygulamış matrisin hangi elemanı kullanılarak TADK uygulamışsa o konuma yerleştirilmiştir. Bu durumdan yola çıkarak özellikle AC bileşenlerinin düşük indisli resimleri, yüzeydeki değişimleri yani kenar bilgilerini oldukça iyi yansıttığı görülmektedir. Çünkü AC bileşenler, orijinal resmin konumsal frekans güçlerini ve doğrultularını vermektedir. Bu durumda düşük indisli frekanslar resmi daha güçlü ifade eden fakat ayrıntı içermeyen özellikleri saklarken, indisler arttıkça saklanan bilginin resmin genelini ifade etme gücü azalırken ayrıntıyı gösterme gücü artmaktadır. Diğer bir ifadeyle AC

değerler en belirgin doku ve kenar bilgilerini bulundururlar. Bu bilgilerin MPEG kodlama içinde halihazırda bulunması çalışmamız için çok önemlidir. Çünkü zamanın bizim için çok kritik olduğu bu çalışmada kenar ve doku özelliği bulma gibi karmaşıklığı yüksek algoritmaları uygulamak yerine bu değerlerin kullanılması kaçınılmazdır.



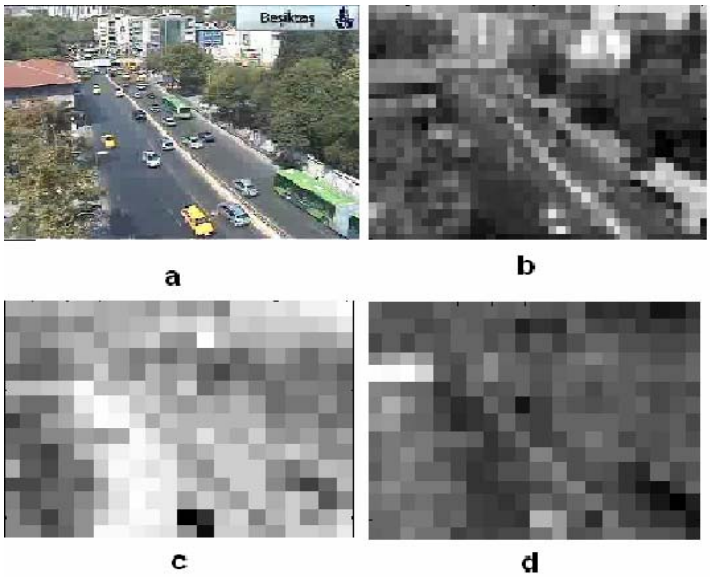
Şekil 3.9 DC ve AC bileşenleri

Y bileşeni kodlanırken herhangi bir veri kaybı olmamasına karşın Cb ve Cr bileşenler kodlanırken veri kaybı söz konusudur. Şekil 3.10'da özetlendiği gibi bir makro blok, alt makro bloklara ayrılmadan veri azaltılması uygulanır. Her bir 2*2'lik piksel grubu tek bir renk değeri olarak temsil edilir. Bu 4 pikselin renk değerlerinin ortalaması alınarak hesaplanır. Böylelikle 16*16'lık makro bloğu 8*8'lik bloklarla çevirmiş olduk. Veri azaltılmış bu blok AKD fonksiyonuna sokulur. AKD sonucu zig-zag sıralamasıyla okunur ve Huffman kodlamasıyla kodlandıktan sonra RLE sıkıştırmasıyla sıkıştırılıp kodlanır. O halde resmi oluşturan bir makro blok 4 tane 8*8'lik Y bileşen değeri, 1 tane 8*8'lik Cb bileşen değeri ve yine 1 tane 8*8'lik Cr bileşen değerlerinin AKD sonucunun zig-zag sıralamasıyla sıralanıp Huffman ve RLE sıkıştırma uygulanmış haliyle saklanır.



Şekil 3.10 I Çerçevesindeki renk bileşenlerinin kodlanması

Şekil 3.11'de orijinal resmin DC bileşeninin YCbCr uzayında kodlanan resimleri görünmektedir. Görüldüğü üzere Y bileşeninin çözünürlüğü daha yüksekken (30x40), Cb ve Cr bileşeninin çözünürlüğü 2 boyutta da 2 kat daha düşüktür(15x20).



Şekil 3.11 a)Orijinal resim b)DC_y katsayısı (30x40) c) DC_cb katsayısı(15x20) d)DC_cr katsayısı(15x20)

3.3 P Çerçevelerinin Kodlanması

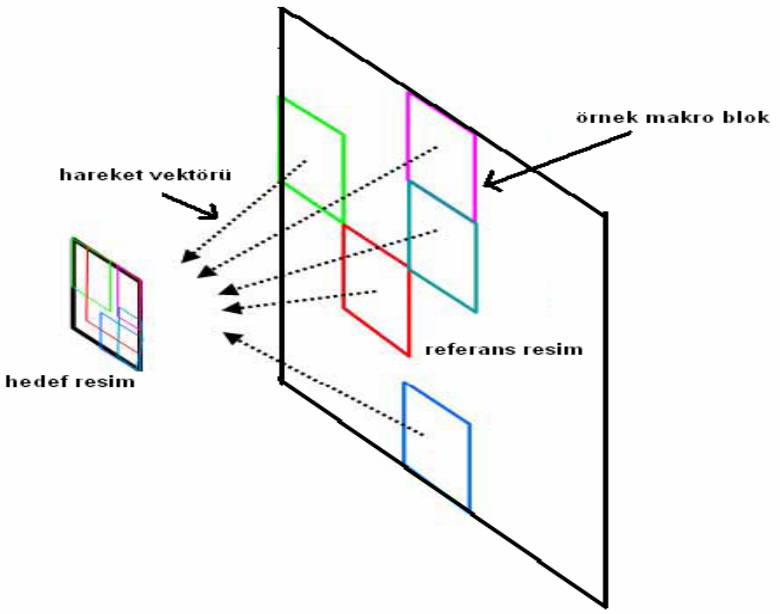
P çerçevelerinin içeriği I çerçevelerinde olduğu gibi konumsal piksel değerlerinden

oluşmayıp, makro blokların referans resme göre göreceli olarak piksel bazında x ve y eksenlerinde yaptığı hareketlerden oluşmaktadır. Zamansal sıkıştırmada birbirini takip eden makro bloklar arasında olabildiğince iyi kestirim yapabilmek oluşacak hataların azaltılmasını sağlar. Piksel değerlerinin yerine AKD katsayılarının kullanılması da zaten bu kestirim performansını arttırmak içindir. AKD katsayıları olabildiğince az yuvarlanırsa bu daha iyi bir makro blok eşlemeyi sağlayacaktır, fakat bu durumda da sıkıştırma oranı düşük olacaktır.

Zamansal sıkıştırma herhangi bir andaki çerçevenin kendinden önce gelen bir ve ya daha önceki çerçevelerle modellenebileceği varsayımını kullanmaktadır. İşte bu MPEG spesifikasyonu hareket bilgilerinin P ve B çerçevelerinde nasıl tutulacağını belirlemektedir. Makro blok kodlaması sebebiyle, bir çok uygulama mevcut resim ve referans resmindeki yanlış blok eşleşmelerini en aza indirecek blok eşleme tabanlı bir teknik kullanmaktadır. Diğer yandan MPEG hareket vektörleri hiçbir zaman gerçek hareketi gösteremezler, fakat bloklar arası en iyi eşleşmeler temsil edildiğinden gerçek harekete çok yakın olan yer değişim matrisi (MV) elde edilir.

Bir P çerçevesinin içerisine bakıldığında resmi oluşturan her bir makro bloğun 2 boyutlu koordinat ekseninde kaç piksel ilerlediği bilgisi görülür. Şekil 3.12'de görüldüğü üzere bazı makro blokların verilen hareket vektörü uygulandığında hedef resimde üst üste binmeler oluşabilir. Hareket etmeyen makro bloklar 0 değerini alırken daha önce bulunmayıp yeni gelen makro blokların değerleri I çerçevelerinde olduğu gibi kodlanır. Bu durumda hareket vektöründe söz konusu makro bloğu temsil eden değer çok büyük olacaktır. Zaten hareket vektörünün bileşenleri [-64,63] aralığında olmak durumundaydı. Bu aralığın dışında bir değer geçersiz bir hareket olduğu anlamına gelir.

Hareket vektörleri her bir makro bloğun hareketini gösterdiğinden 320x240 çözünürlüğe sahip görüntüden 15x20 çözünürlükte biri x ekseninde diğeri y ekseninde olmak üzere 2 adet hareket vektörü elde edilir. Örnek bir hareket vektörünün içeriği Şekil 3.13'deki gibidir.



Şekil 3.12 Yer değişim matrisinden hedef resmi oluşturma

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	-4	3	2	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	6	3	5	4	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	7	1	3	5	0	0	0	0	0	0	0	0	0
0	0	0	0	0	-12	-13	0	-5	11	-9	5	0	0	0	0	0	0	0	0
0	0	0	0	0	0	-13	-1	-15	15	5	5	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	-1	-15	30000	0	0	0	9	0	6	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	12	30000	7	7	7	1	0
0	0	0	0	0	0	0	0	0	0	-11	30000	19	-21	8	13	30000	9	9	3
0	0	0	0	0	0	0	0	0	0	30000	-21	11	21	0	0	2	9	10	30000
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30000	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-1	3	2	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	5	3	5	5	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	5	1	4	5	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-3	-1	0	-3	9	-3	7	0	0	0	0	0	0	0	0	0
0	0	0	0	0	-5	-16	-11	13	5	5	0	-1	0	0	0	0	0	0	0
0	0	0	0	0	0	-1	-9	30000	0	0	0	11	0	7	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	13	30000	9	11	11	1	0
0	0	0	0	0	0	0	0	0	-9	30000	19	-21	9	15	30000	13	13	-1	0
0	0	0	0	0	0	0	0	0	0	30000	-17	-5	21	0	0	2	12	13	30000
0	0	0	0	0	0	0	0	0	0	0	-13	0	0	0	0	0	0	0	30000

Şekil 3.13 Örnek bir hareket vektörü a)MV_y b)MV_x

Bu matrisin içeriğine dikkat edecek olursak bazı aykırı değerler göze çarpmaktadır. İçerdiği değer 30000 olan indisler aslında o makro bloğun önceki çerçevede olmadığını yeni ortaya çıkan blok olduğunu belirtir. Diğer değerler ise makro bloğun söz konusu eksende bir sonraki çerçevede kaç piksel ileri ya da geri gittiğini gösterir.

4. HAREKET VEKTÖRLERİ

Hareket vektörleri çalışmamızın temelini oluşturan en önemli bileşendir. Hareket vektörleri her ne kadar da teorikte hareket eden blokları gösterse de pratikte her zaman böyle olmamaktadır. Çoğu zaman kullanmak istediğimiz hareketlerin yanında bir takım gürültüler de barındırmaktadır. Bu bölümde anlatılacak olan işte bu gürültülerin etkisini nasıl en aza indirebileceğidir.

4.1 Eşleşmeyen Blok Süzgeci

Hareket vektörleri okunduktan sonra ilk yapılması gereken iş yeni oluşan makro bloklar için atanan büyük hareket değerlerini temizlemek olmalıdır. Vektördeki bu değerler normalde hareket göstermeyen daha önceki çerçeveyle hiçbir bağlantısı olmayan yeni makro bloklardır. Bu tür bloklar kodlanırken bir hareket olmadığı, aksine yeni bir blok geldiğini göstermek amacıyla çok yüksek sayılarla kodlanırlar. Bu işlemi (4.1)'deki basit bir süzgeçle yapılır.

$$MV(x,y) > 63 \quad veya \quad MV(x,y) < -64 \quad ise \quad MV(x,y) = 0 \qquad (4.1)$$

Burada MV hareket vektörünü gösterirken x ve y'ler matrisin indisleridir. Bir hareket vektörünün 63'ten büyük ve -64'ten küçük değer alamayacağı için, (4.1)'deki süzgeç kullanılmıştır.

Şekil 4.1 a)I(t) b)I(t+1) c)MV

Eşleşmeyen blok değerleri temizlendikten sonra hareket vektörünün (MV) aldığı görüntü Şekil 4.1c'deki gibidir. Bu görüntüdeki kırmızı değerler pozitif, mavi değerler negatif değerlerdir. Renk çizelgesine göre ara değerler renklendirilmiştir. Şekil 4.1a ve 4.1b'de peşi sıra gelen 2 çerçeve görünmektedir. Bu çerçeveler arasındaki söz konusu hareketi Şekil

4.1c'nin oldukça iyi verdiği söylenebilir. Şekil 4.1c'de yeşil renk hareketsiz alanları gösterirken kırmızı tonlar yukarı doğru, mavi tonlar ise aşağı doğru hareketi belirtir.

4.2 İlgi Alanı Süzgeci

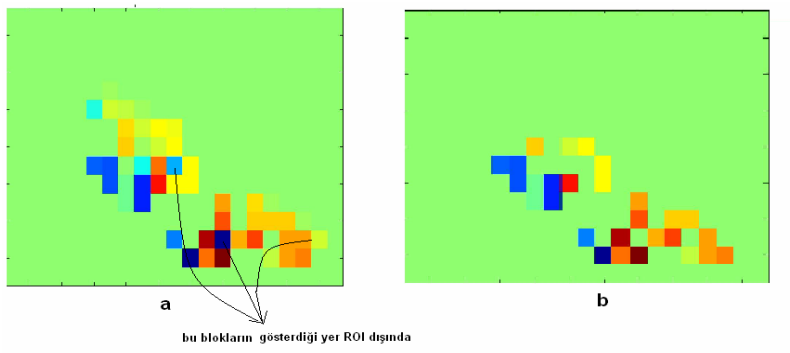
Şekil 4.1c'ye dikkatli bakıldığında bazı yanlış verilerin olduğu görülebilmektedir. Mesela MV(13,14) değerinin Şekil 3.13'te -21 olduğu görülmektedir. Yani aşağı yönde bir hareket olduğu belirtilmektedir. Fakat o değerin çevresine baktığımızda bütün blokların yukarı yönde bir hareketi olduğu görülmektedir. Resmin doğasına göre böyle bir şey mümkün değildir. Çünkü o makro bloğun işaret ettiği alan araç akışının yukarı yönde olduğu alandır. O halde daha önceden hazırladığımız ve elle etiketlediğimiz (Şekil 4.2) yol süzgeçlerini kullanarak yolun hangi kısmındaysak o kısımdaki aykırı verileri tespit edebiliriz. İşlenen görüntüde bizim ilgilendiğimiz alan olan Region Of Interest (ROI) içinde kalan hareket vektörü bileşenleri hangi yolun maskesi içinde kalıyorsa o kısımdaki davranışı sergileyip sergilemediğine bakılır. Eğer sergilemiyorsa değeri 0 yapılır. Böylelikle ROI dışındaki yerlerde (iki yolu birbirinden ayıran banket, yaya yolu) oluşacak hareketlerden de kurtulmuş oluruz. Uygulanan süzgeç işleminin matematiksel ifadesi (4.2)'deki gibidir.

$$\begin{aligned} M1(x,y)=1 \quad & ve \quad MV(x,y)>0 \quad \Rightarrow \quad MV(x,y)=0 \\ M2(x,y)=1 \quad & ve \quad MV(x,y)<0 \quad \Rightarrow \quad MV(x,y)=0 \\ M1(x,y)=0 \quad & ve \quad M2(x,y)=0 \quad \Rightarrow \quad MV(x,y)=0 \end{aligned} \tag{4.2}$$

Burada M1 yolun sağ tarafının maskesi, M2 sağ tarafının maskesi, MV ise hareket vektörü olmaktadır. M1 ve M2, 0 ve 1 değerleri taşımaktadır. Eğer içeriği 0 ise o indise karşılık gelen resmin pikseli yolun dışında, 1 ise yolun içindedir anlamına gelmektedir.

Şekil 4.2 a)Sol yol maskesi (M1) b)Sağ yol maskesi (M2)

(4.2)'deki süzgeç hareket vektörlerine uygulandığında hareket vektörünün ilk hali ve sonraki hali Şekil 4.3'daki gibidir.

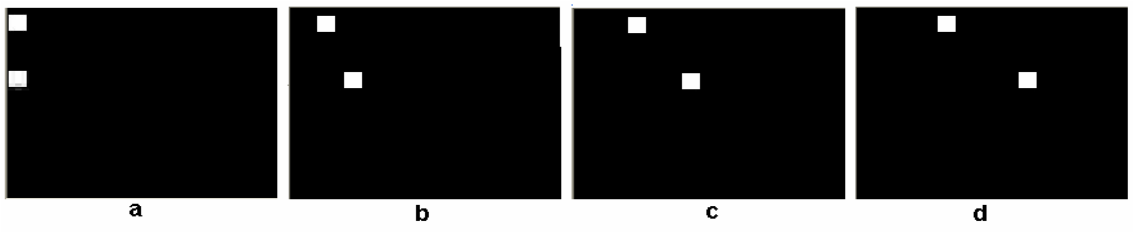


Şekil 4.3 a)Süzgeçten önceki MV b)Süzgeçten sonraki MV

4.3 Doku Süzgeci

Hareket vektörü, araçların hızını yansıtmayan çeşitli gürültülerden eşleşmeyen blok süzgeci ve ROI süzgeci ile temizlenmiş oldu. Fakat bu sonuç bile yeterince iyi değildir. Hareket vektörünün içinde araçların hareketini belirtmeyen bazı hareketler yine söz konusudur.

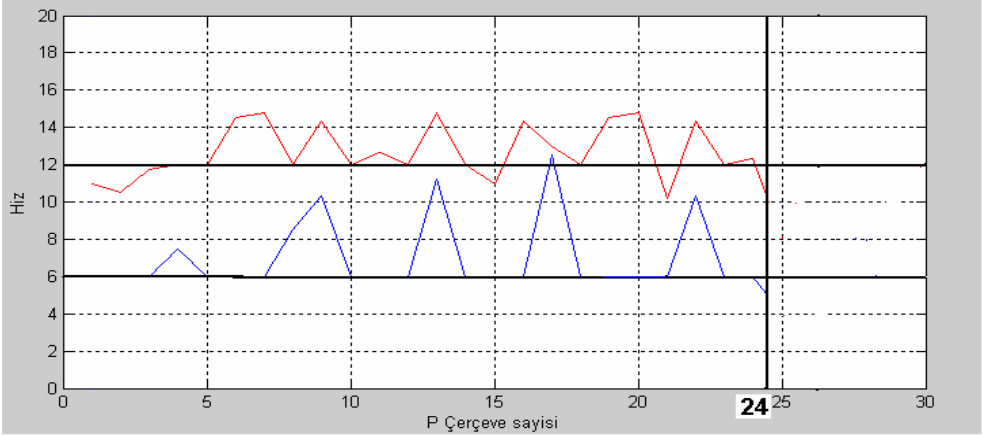
Hareket vektörünün ifade ettiği değerlerin gerçek hareketle ne kadar örtüştüğü, hata toleransının ne olduğu, hatalardan nasıl kurtulabileceğimiz örnek bir yapay video üzerinden araştırılmıştır.



Şekil 4.4 a)I(0) b)I(30) c)I(60) d)I(90)

Hazırlanan yapay video Şekil 4.4'teki gibidir. Değişik zamanlarda videodan alınan çerçeveler yukarıdaki gibidir. Videoda 20*20'lik iki beyaz kareden üstte olanı her bir çerçevede 1 piksel sağa, altta olanı ise her çerçevede 2 piksel sağa doğru kaymakta ve bu hareket 90 çerçeve boyunca devam etmektedir. Karelerin harekete başladığında sol üst noktalarının koordinatları

(10,0) ve (80,0) olurken hareketin bittiği yerdeki sol üst noktalarının koordinatı (10,90) ve (80,180) olur. Videodan okunan hareket vektörlerinin zaman içinde değişiminin grafiği, en basit ön işleme adımı olan eşleşmeyen blok değerleri temizlendikten sonra Şekil 4.5'teki gibi olmaktadır.



Şekil 4.5 Filtre uygulamadan önce x ekseninde algılanan hareket vektörleri

Örnek videomuz saniyede 30 çerçeve olacak şekilde kaydedilmiştir. Bu durumda videomuz toplam 3 saniyelik bir kayıttan ibarettir. Çerçeve dizilişine baktığımızda karşımıza 15 çerçevelik GOP çıkmaktadır. Çerçeve IBBPBBPBBPBBPBB şeklindedir. Bu durumda örnek video 90 çerçeve içerdiği için 6 GOP'tan oluşmaktadır. Kullandığımız MPEG parser sadece P çerçevesindeki hareket vektörlerini bulduğundan bir GOP için 4 hareket vektörü çıkartılır. Bunlar 1-4, 4-7, 7-10, 10-13 çerçeveleri arasındaki hareketlerdir. Dolayısıyla yukarıdaki grafik birbirini takip eden üç çerçeve arasındaki hareket vektörüdür. Bir GOP'un son P çerçevesinin ondan sonra gelecek diğer GOP'un I çerçevesine dönüşmesi için gereken makro blok hareketleri bu durumda elde edilemeyecektir. O halde her bir GOP için 1 tane eksik hareket vektörü elde edilir. Bu durumda video boyunca toplam yer değiştirmede hata oluşur. Bu hatayı telafi etmek için GOP'un son 3 çerçevesinde de hızın değişmeyeceğini, daha önce okuduğumuz 4 hareket vektörünün ortalamasına eşit olduğunu düşünerek işleme devam edilir. Bu durumda toplam yer değiştirmenin şu formülle hesaplanması gerekmektedir.

$$\Delta Konum_{gerçek} = \frac{N_p + 1}{N_p} \times \sum \Delta Konum_{anlik} \qquad (4.3)$$

Burada N_p bir GOP'un içinde bulunan P çerçevelerinin sayısını vermektedir.

Fakat algılanan anlık hızlarda dalgalanmalar söz konusudur. Grafikteki verilerin her biri peşi sıra gelen 3 çerçevede meydana gelen yer değişimler olduğuna göre beklenen değerler yavaş giden kare için 3, hızlı giden kare için 6'dır. Fakat hareket vektörleri gerçek değerlerinin iki katıyla kodlandığını düşünürsek bu durumda olması gereken değerler 6 ve 12'dir. Fakat okunan değerlerin ortalaması sırasıyla 6,8947 ve 12,514'dür. Ayrıca algılanan yer değişimlerinin olması gereken nokta olan 6 ve 12'ye Manhattan uzaklıkları (4.4) ise üstteki ve alttaki kare için sırasıyla 1,0547 ve 1,2780'dir.

$$hata = \frac{1}{N}\sum_{i=1}^{N} | A_i - B | \tag{4.4}$$

Çizelge 4.1 Yapay videodan okunan hareket vektörlerinin süzgeçten önceki hatası

	1.Kare	2.Kare
Ortalama	6,8947	12,514
Ortalamadaki % hata	%14,9	%4,2
Manhattan Uzaklığı	1,0547	1,278
Uzaklıktaki % hata	%17,58	%10,65

Kabul edilebilir hata %10 olduğuna göre Çizelge 4.1'e bakacak olursak hataların kabul edilemez olduğu görülür. Hataya sebep veren etmenler incelenirken hareket eden cisme ait olmayan zemine ait siyah kısımdaki bazı makro blokların başka konumdaki zeminlerle eşleştirildiği anlaşılır. Örneğin 170. çerçevede algılanan hız vektörleri Şekil 4.6'daki gibidir. Beyaz karenin hareketini temsil eden bloklar -6 ve -12 değeri ile belirtilirken, yanlış zemin eşleşmesinden kaynaklanan bloklar -19'luk bir hareket belirtmektedir. Tabiatıyla bu da hızın yanlış bulunmasına ya da hızda bir dalgalanmaya sebebiyet verecektir.

Hataya elverişli bu tür alanlar daha çok yüzeyin doku özelliğinin baskın olmadığı tek tip yüzeylerde gözükür. Bizim problemimiz için düşünecek olursak araçların olmadığı düz zeminde, yani yol çok geniş alanları kapladığı ve bu geniş alanın içindeki blokların çoğu birbiriyle neredeyse aynı olduğu için uniform yapıya sahip makro blokların birbirine benzeme

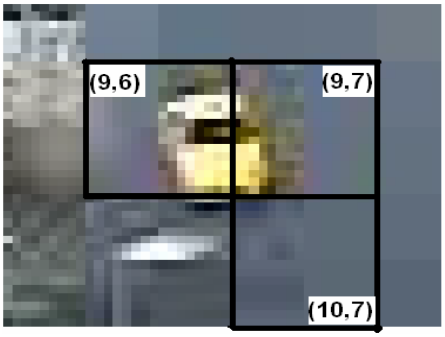
ihtimali oldukça yüksek olmaktadır.

0	0	0	0	0	0	0	0
0	-19	-6	-6	0	0	0	0
0	-19	0	-6	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	-19	-12	-12	0
0	0	0	0	-19	0	-12	0
0	0	0	0	0	0	0	0

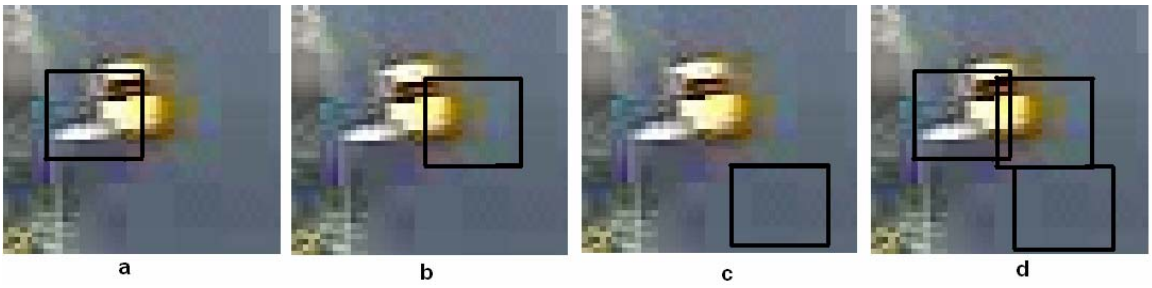
Şekil 4.6 170. çerçeveden bozulmaya sebep olan bölüm

Bunun en güzel örneği, Şekil 4.7'de hareket vektöründeki MV(9,6) elemanında görülebilir. MV(9,6) ve MV(9,7) elemanları sarı renkteki taksinin hareketini gösterirken onun hemen altındaki MV(10,7) elemanı da boş yolu göstermektedir. Bu karede hareket olmamasına rağmen 13 piksel aşağı 5 piksel sağa hareket ettiği hareket vektöründe gözükmektedir ($MV_y(10,7)$= -13, $MV_x(10,7)$= -5). Bu makro blok, düz herhangi bir deseni olmayan uniform bir blok olduğu ve onun gibi bloklar yol üzerinde çok olduğu için yanlış blokla eşleştirilmiştir. Şekil 4.8'de ele alınan üç makro bloğun eşleştirildiği yerler görünmektedir. Bunu engellemek için yeni bir süzgeç daha kullanılmalıdır.

Tasarlanacak süzgeç doku özelliği baskın olmayan düz desensiz yapısı olan makro blokların varsa hareket vektörlerindeki değerleri sıfırlayıp araç akış hızına etki etmesini önlemeye yaramaktadır. O halde yüzey dokusunun uniform olup olmamasına göre adı geçen makro bloğun hareketinin dikkate alınıp alınmayacağı belirlenmelidir. Bu durumda makro bloğun varyansı belirleyici olur.



Şekil 4.7 Takip edilen 3 makro bloğun resim I(t)'deki konumları



Şekil 4.8 Takip edilen üç makro bloğun I(t+1)'deki konumları a)(9,6) b)(9,7) c)(10,7) d)Toplu gösterim

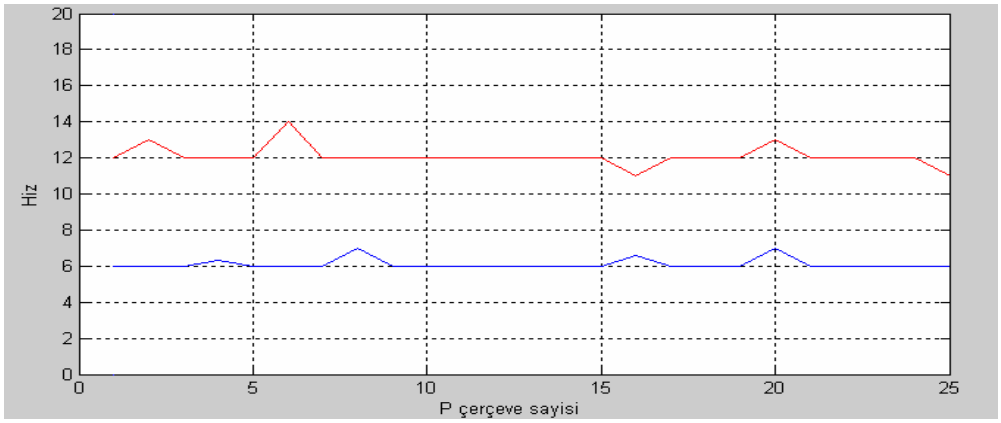
$$m(x,y) = \frac{1}{16 \cdot 16} \times \sum_{i=1}^{16} \sum_{j=1}^{16} B(x,y)_{i,j} \tag{4.5}$$

$$v(x,y) = \frac{1}{16 \cdot 16} \times \sum_{i=1}^{16} \sum_{j=1}^{16} (B(x,y)_{i,j} - m(x,y))^2 \tag{4.6}$$

$$MV(x,y) <> 0 \quad ve \quad v(x,y) < T \quad \Rightarrow \quad MV(x,y) = 0 \tag{4.7}$$

Burada m ortalama değeri, B analiz edilen makro bloğu, V varyansı, MV hareket vektörünü, T ise eşik değerini göstermektedir. Çalışmamızda T değeri 400 seçilmiştir.

Geliştirilen süzgeci yapay videomuz üzerine uygularsak beyaz karemizin x ekseni boyunca yer değişim vektörü Şekil 4.9'daki gibi olacaktır.



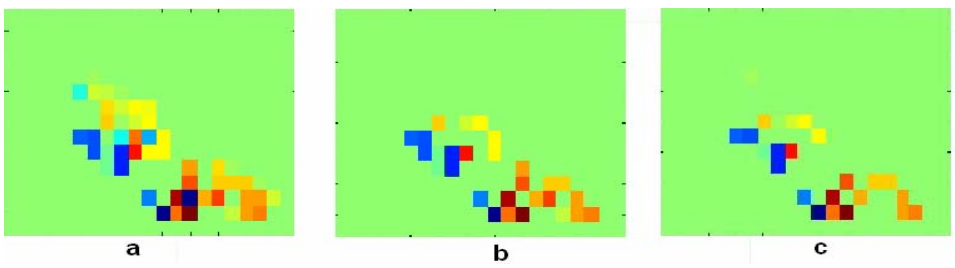
Şekil 4.9 Doku süzgecinden sonra x ekseninde algılanan hareket vektörü

Şekil 4.9'daki verilerin ortalaması 6,11 ve 12,08, ortalama mutlak hata ise 0,32 ve 0,74 olarak bulunmuştur. Bu değerlerden yola çıkarak yapay videomuz için 90 çerçeve boyunca yapılan toplam hareket %2 hata ile , anlık olarak yapılan hareket ise %6,1 hata ile bulunabilmektedir. Yapay verilerde yüksek başarı sağlanırken gerçek dünya verilerinde aynı şeyi söylemek oldukça zordur. Varyansın eşik değerinin bulunmasındaki zorluklar ve hesapta olmayan önceden tahmin edilemeyen gürültüler sebebiyle başarımız düşecektir. Başarı ölçütü olarak Manhattan uzaklık ölçütü kullanılmıştır. Olması gereken hızla ölçülen hızların mutlak farkları toplanıp veri sayısına bölünmüştür. (4.6)

Çizelge 4.2 Yapay videodan okunan hareket vektörlerinin süzgeçten sonraki hatası

	1.Kare	2.Kare
Ortalama	6,11	12,08
Ortalamadaki % hata	%1,83	%0,6
Manhattan Uzaklığı	0,32	0,74
Uzaklıktaki % hata	%5,3	%6,1

Örnek kayıtlar üzerinde süzgeç denendiğinde Şekil 4.7'de işaretlenmiş 3 makro bloğun (4.3,4.4,4.5)'deki formüllere göre varyansı hesaplandı ve sırayla V(9,6)=2883, V(9,7)=1089, V(10,7)=81 çıkmıştır. Alınan deneysel değerlerden uniform dağılıma sahip (10,7) bloğunun hareket vektörünün yanlış olduğu anlaşılmaktadır. Şekil 4.10c'de doku süzgecinin oluşturduğu çıktı görünmektedir. Böylelikle hareket vektörümüz hızı hesaplamak için uygun hale gelmiştir.



Şekil 4.10 a)Aykırı değer süzgeci sonucu b)ROI süzgeci sonucu c)Doku süzgeci sonucu

5. GEOMETRİK DÜZELTME

Kameralar ile alınan görüntü ile gerçek görüntü birbirinin aynısı değildir. Her şeyden evvel 3 boyutlu bir dünya dan aldığımız görüntüyü kamerayla 2 boyuta indirgiyoruz. Bundan ötürü görüntüde gözüken objelerin gerçek dünyadaki konumunu eğer hareket ediyorsa gerçek dünyadaki hızını bulabilmek için bazı dönüşümler yapmamız gerekir. Bu dönüşümlerin genel adı geometrik dönüşümlerdir.

Şekil 5.1 a)45. çerçevenin görüntüsü b)240. çerçevenin görüntüsü

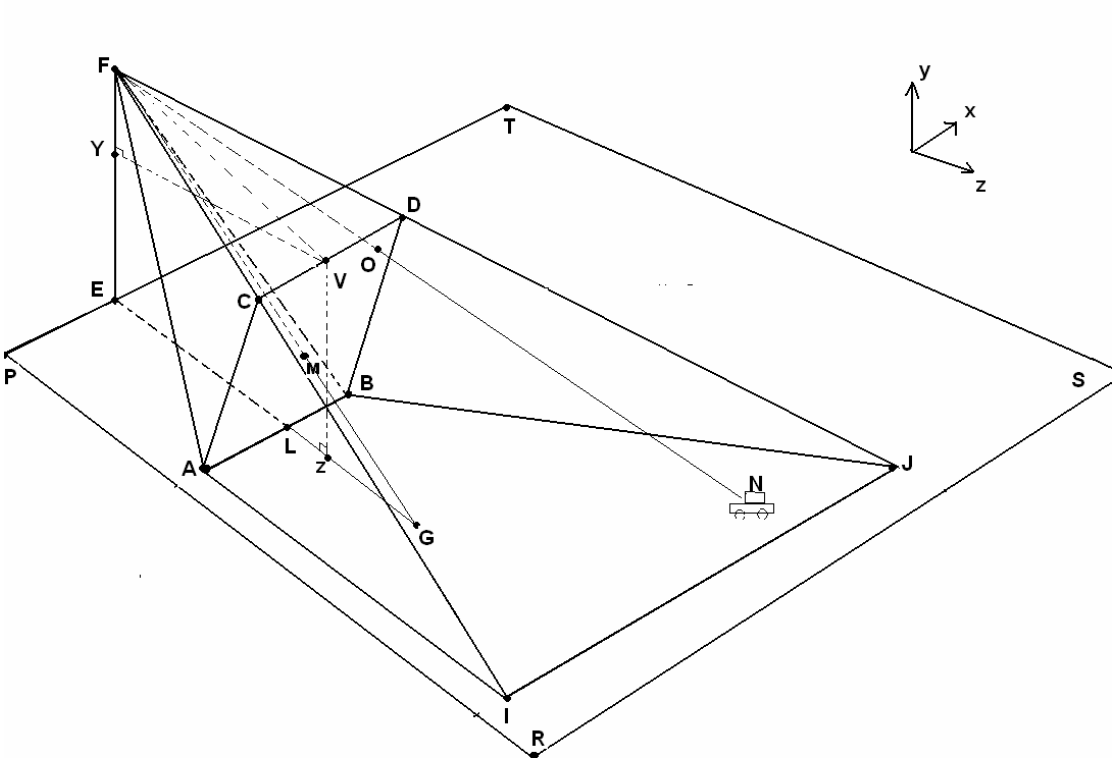
Geometrik dönüşüme ihtiyaç olduğunun en güzel kanıtı Şekil 5.1'dir. Görüş alanına yeni giren otobüs kameraya yakınken boyu 111 pikselken, uzaklaştıkça boyu küçülmekte ve görüş alanının sonlarına doğru boyu 27 piksele kadar düşmektedir. Burada yapmamız gereken kamera koordinatını gerçek dünya koordinatına çevirmektir. 2 boyutlu görüntüden 3 boyutlu görüntüyü elde edebilmek için aynı enstantanenin 2 farklı kamera açısından görüntüsü olmalı ya da aynı görüş açısından farklı ışıklandırma altında 2 farklı görüntüsü olmalıdır. Geliştirilen sistemin kullanılabilirliği açısından bu 2 çözümü de uygulamak imkansızdır. Çünkü mevcut duruma göre halihazırda bir kameranın yanına bir başkasını koymak mümkün değildir. Aynı zamanda bu kameralar dışarıda olduğu için ve görüş alanları çok geniş olduğu için ortamı farklı ışık altında ışıklandırmamız olanaksızdır. Bu imkansızlıklar yüzünden 2 boyuttan 3 boyuta geçmek yerine başka çözümler üzerinde duruldu. Her ne kadar geometrik düzeltmeyi tam manasıyla sağlayabilmemiz için görüntüyü 3 boyuta dönüştürmemiz gerekse de bazı kabulleri kullanarak 2 boyuttan yine 2 boyuta bir dönüşümle amacımıza ulaşabiliriz.

Araçların yoğunluğunu bulmak istediğimizde araçların yükseklikleri, yol yoğunluğunu

bulmak için işe yarayacağı kesinlikle söylenemez. O halde araçların ya da herhangi bir objenin yükseklik bilgisini kullanmadan ya da onları sabit yükseklikli objeler gibi düşünerek de yoğunluğu ve hızı tespit edebiliriz

5.1 Geometrik Düzeltmenin Altyapısı

Şekil 5.2'de PRST yol düzlemi, ABDC kamerada oluşan görüntü düzlemi, ABJI kamera görüntüsünün yol düzleminde karşılığı, F kameranın konumu, E kamera direği dibinin konumunu göstermektedir.



Şekil 5.2 Geometrik düzeltme hesaplaması

V |CD| kenarının orta noktası, |VY| |EF|'ye dik bir doğru, |VZ| PRST düzlemine dik bir doğru, M kamera düzleminin orta noktası,G kamera düzleminin orta noktasının yol düzlemindeki karşılığı, L kamera düzleminin alt kenarının orta noktası, N aracın bulunduğu nokta ve O ise N noktasında bulunan aracın kamera düzlemindeki karşılığını göstermektedir. Yapılan hesaplamalarda kamera üzerindeki herhangi bir konumda görünen bir noktanın yol düzlemindeki gerçek koordinatlarını bulabilmek için 3 uzunluk değerini bilmemiz gerektiği bulunmuştur. Bunlar kameranın yerden yüksekliğini gösteren |EF| doğrusu, görüş alanının en

alt çizgisinin ortasının E noktasına olan uzaklığını gösteren |EL| doğrusu ve kamera üzerindeki tam orta noktanın yol üzerindeki konumunun görülen en alt çizginin orta noktasına uzaklığı olan |GL| doğrusu bilinmelidir.

Yapılması gereken dönüşümü daha kolay bulabilmek için E noktası koordinat eksenlerinin merkezi olarak kabul edilmiştir. E(0,0,0) şeklinde tanımlanmıştır. |EF|=h verildiğini düşünürsek F(0,h,0) noktasını gösterecektir. |EL|=t verildiğinde L(0,0,t) noktasını gösterir. |LG|=k olarak verildiğinde ise G(0,0,t+k) noktasını gösterecektir. ABCD düzleminin denklemini hesaplayabilmemiz için öncelikle bu düzlemdeki doğrusal olmayan 3 noktanın koordinatlarını bilmemiz gerekmektedir. L noktasının koordinatları belli olduğuna göre V noktasının koordinatları belirlendiğinde bunun yanında L noktasının x ekseninde 1 birim ötesinde hayali bir X noktası oluşturarak düzlemin denklemi yazılır. V noktasının koordinatları için öncelikle |FV| doğrusunun PRST düzlemiyle yaptığı açı belirlenmelidir.

GFE açısına α dersek

$$\tan(\alpha) = \frac{|EG|}{|EF|}, \text{ olduğundan } \tan(\alpha) = \frac{t+k}{h} \text{ oradan da } \quad \alpha = \arctan(\frac{t+k}{h}) \tag{5.1}$$

FLE açısına β dersek

$$\tan(\beta) = \frac{|EL|}{|EF|}, \text{ olduğundan } \tan(\beta) = \frac{t}{h} \quad \text{oradan da} \quad \beta = \arctan(\frac{t}{h}) \tag{5.2}$$

LFG açısına φ dersek , $\varphi = \alpha - \beta$ olarak bulunur (5.3)

Kamera görüntü düzlemindeki bütün simetrik noktaların kamera odağına uzunlukları eşit olacağından |FV|=|FL| ve M orta nokta olduğundan |LM|=|MV| olacaktır. O halde LFG ve MFV açıları eşit açı olacaktır.

O halde EFV açısına θ dersek $\theta = \alpha + \varphi$ olduğundan $\theta = 2\alpha - \beta$ olur. (5.4)

FVY açısına λ dersek, FYV dik üçgeninden $\lambda = 90 - \alpha - \varphi$ olur. (5.5)

(5.4) ve (5.5)'i kullanırsak istenen açıyı bilindik değerlerle ifade edebiliriz.

$\lambda = 90 - 2\alpha + \beta$ bulunmuş olur. (5.6)

(5.1) ve (5.2)'deki formülleri kullanırsak istenen açıyı şöyle yazabiliriz.

$$\lambda = 90 - 2\arctan(\frac{t+k}{h}) + \arctan(\frac{t}{h}) \quad (5.7)$$

FEL dik üçgeninden $|FL| = \sqrt{|EF|^2 + |EL|^2}$ olduğundan $|FL| = \sqrt{h^2 + t^2}$ (5.8)

|FL| ve |FV| kamera odağına göre simetrik olduğundan |FL|=|FV| ve

$$|FV| = \sqrt{h^2 + t^2} \quad (5.9)$$

FVY üçgeninden $|FY| = |FV| \times \sin(\lambda)$ (5.10)

Ve $|YV| = |FV| \times \cos(\lambda)$ (5.11)

Artık V noktasının koordinatlarını bulabilecek açı ve uzunluk değerlerini elde ettik. Bu durumda V noktasının koordinatları;

X ekseninde hep 0 noktası üzerinde bulunduğundan $V_x = 0$

Y ekseninde konumu h-|FY| olduğundan $V_y = h - |FV| \times \sin(\lambda)$ olur. (5.9)'u kullanırsak

$V_y = h - \sqrt{h^2 + t^2} \times \sin(\lambda)$ olur.

FYV üçgenini düşünürsek z ekseninde |YV| kadar bir ilerleme olmuş, o halde $V_z = |FV| \times \cos(\lambda)$ olur, (5.9)'u kullanırsak $V_z = \sqrt{h^2 + t^2} \times \cos(\lambda)$ olarak bulunur.

O halde V(0, $h - \sqrt{h^2 + t^2} \times \sin(\lambda)$, $\sqrt{h^2 + t^2} \times \cos(\lambda)$) olarak V'nin koordinatları bulunmuş olur.

X noktasını L noktasından 10 birim +x yönünde itelenmiş bir nokta olarak düşünürsek X(10,0,t) olarak koordinatlar belirlenmiş olur.

Üç noktası belirlenmiş düzlemin denklemi aşağıdaki formüle göre bulunmaktadır.

A,B,C düzlemdeki doğrusal olmayan 3 nokta ise herhangi bir P(x,y,z) noktası varsa bu durumda aşağıdaki determinant 0 olmalıdır.

$$\det\begin{pmatrix} P-A \\ B-A \\ C-A \end{pmatrix} = 0 \tag{5.12}$$

$$\text{o halde} \quad \det\begin{pmatrix} x & y & z-t \\ 10 & 0 & 0 \\ 0 & h-\sqrt{h^2+t^2}\times\sin(\lambda) & \sqrt{h^2+t^2}\times\cos(\lambda)-t \end{pmatrix} = 0 \tag{5.13}$$

olması gerekir. Bu determinantı çözdüğümüzde ABDC düzleminin denklemini

$$y.(\sqrt{h^2+t^2}\times\cos(\lambda)-t)-(z-t).(h-\sqrt{h^2+t^2}\times\sin(\lambda))=0 \tag{5.14}$$

şeklinde buluruz. Görüldüğü gibi düzlem x'e göre bağımsızdır.

Kamera düzleminin en alt ve en üst noktaları arasındaki mesafe olan |LV|'yi hesaplamak için her iki noktanın koordinatları yeterlidir. O halde

$$|LV| = \sqrt{(L_x - V_x)^2 + (L_y - V_y)^2 + (L_z - V_z)^2} \tag{5.15}$$

şeklinde bulunabilir. Bu noktaları kamera görüntüsü üzerinde düşünürsek 1 pikselin gerçekte ne kadar uzunluğu gösterdiği belirlenebilir. Ekran görüntüsünde 240 satır piksel olduğunu düşünürsek ki çalışmamızda veriler 240x320 çözünürlüktedir. Bu durumda

$$pikseluzunluk = \frac{|LV|}{240} \text{ olur.} \tag{5.16}$$

Resmin üzerindeki referans noktamız en alt satırın ortasıdır. Bu nokta (0,0) noktası olacak şekilde yukarı doğru ve sağa doğru artmakta, sola doğru ise negatif değerlere inerek azalmaktadır. Bu duruma göre alışageldik referans noktası sol üst köşe olarak verilen bir pikselin (i,j) yeni referans noktasına göre dönüşümü gerekmektedir. Bu işlemi aşağıdaki gibi

yapabiliriz.

$$i_{yeni} = (satir\_sayisi - i) * pikseluzunluk$$

$$j_{yeni} = (sutun\_sayisi / 2 - j) * pikseluzunluk \quad (5.17)$$

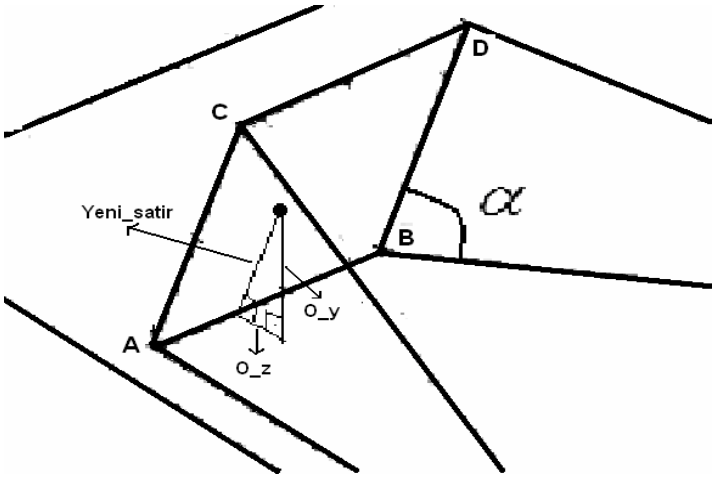
Bu dönüşüm yapıldıktan sonra ekran üstündeki satır ve sütun istediğiz referans ve metrik ölçüye çekilmiş oldu. Fakat ABDC düzlemine henüz aktarılmadı. Koordinatını bulmaya çalıştığımız noktaya O dersek, ABDC düzlemine bu koordinatları aktarmak için kamera düzleminin yol düzlemiyle yaptığı açıyı bulmalıyız.

FEG bir dik üçgen olduğundan FGE=$90 - \alpha$ ve GML bir dik üçgen olduğundan GLM=α olur.

ABDC düzleminde |AB| doğrultusundaki hareket x eksenini doğrudan etkilediği, buna karşı diğer eksenleri etkilemediği için

$$O_x = j_{yeni} \quad (5.18)$$

olur.



Şekil 5.3 Kamera görüntü ekseni koordinatı

ABCD düzlemi α açısıyla yol düzlemini kestiği için |AC| doğrultusundaki hareket hem y hem z eksenini etkileyecektir. Y eksenine etkisi açının sinüsü ile ilişkili olduğundan

$$O_y = i_{yeni} \times \sin(\alpha) \tag{5.19}$$

olacaktır.

Z eksenindeki etkisi ise açının kosinüsü ile ilişkili olacağından ve referans noktasına olan uzaklık z ekseninde t olduğundan noktanın ABDC düzlemindeki z koordinatı

$$O_z = t + i_{yeni} \times \cos(\alpha) \tag{5.20}$$

şeklinde olur.

Ekran üstünde koordinatı verilen bir O noktasının yol düzlemindeki karşılığını bulabilmek için |FO| doğrusunun denklemini bulup bu doğrunun, denklemi y=0 olan yol düzlemini kestiği nokta araştırılır. Piksel olarak koordinatı verilen görüntüdeki bir noktanın kamera düzlemindeki karşılığı bulunması (5.17), (5.18), (5.19) ve (5.20)'deki gibidir. Bu formüllerle O'nun koordinatları elde edildikten sonra |FO| doğrusunun denklemi bulunur.

A(a,b,c) ve B(d,e,f) gibi iki noktası bilinen bir doğrunun denklemi

$$\frac{x-a}{d-a} = \frac{y-b}{e-b} = \frac{z-c}{f-c} \text{ dir.} \tag{5.21}$$

Bu durumda F(0,h,0) ve $O(O_x,O_y,O_z)$ ise doğru denklemi

$$\frac{x}{O_x} = \frac{y-h}{O_y-h} = \frac{z}{O_z} \text{ olur.} \tag{5.22}$$

Bu denklemin ABDC düzlemini kestiği noktaya N dersek, bu noktanın koordinatlarını bulmak için bu noktaları düzlem denkleminde yerine koymak gerekir. Yol düzleminin denklemi y=0 olduğundan

$\frac{N_x}{O_x} = \frac{N_z}{O_z} = \frac{-h}{O_y-h}$ olur, buradan da

$$N_x = \frac{-h \times O_x}{O_y - h},\ N_y = 0,\ N_z = \frac{-h \times O_z}{O_y - h} \tag{5.23}$$

şeklinde bulunur.

Böylelikle resimde koordinatları (i,j) olan bir noktanın ABDC düzlemindeki karşılığı formül (5.18), (5.19) ve (5.20)'de hesaplanan O noktası, O noktasının da yol düzleminde karşılığı (5.23)'de hesaplanan N noktası olarak bulunmuş oldu. O halde ekranda (i,j) konumundaki bir pikselin kamera direk dibi referans alınarak yol düzlemindeki karşılığına (x,y) dersek $(i, j) \xrightarrow{F(h,t,k)} (x, y)$ dönüşümü yapan fonksiyon (5.18), (5.19), (5.20), (5.21), (5.22) ve (5.23)'de gösterildiği gibidir. Söz konusu matlab kodu Ek-1'dedir.

Bu hesaplamanın tersini yapmak da mümkündür. Yani verilen bir N noktasının kamera görüntüsündeki koordinatını bulabiliriz. Bunun için |NF| doğrusunun denklemi yazılır ve bu denklemin ABDC düzlemini kestiği noktanın koordinatları bulunur. Bu koordinatlar basit dönüşümlerle resim satır ve sütununu gösterecek hale dönüştürülür.

F (0,h,0) koordinatında olduğu, N ise yol düzleminde ve yol düzlemi y=0 denklemiyle belirtildiği için olması gereken koordinata N(a,0,c) dersek bu iki noktadan geçen doğrunun denklemi (5.21)'den

$$\frac{x}{a} = \frac{y - h}{-h} = \frac{z}{c} \tag{5.24}$$

olur. Bu denklemin (5.14)'deki ekran düzlemi denklemiyle kesiştiği yere O dersek

$$O_z = \frac{c.t.\tan(\alpha) - h.c}{c.\tan(\alpha) - h},\ O_y = (O_z - t).\tan(\alpha),\ O_x = \frac{a.O_z}{c} \tag{5.25}$$

formülüyle bulunur. Ekran çözünürlüğünü 240x320 olarak kabul edersek, ekran düzlemi üzerinde olan koordinatların ekran koordinat sistemindeki karşılığı (i,j) ise bu değerler şöyle ifade edilir.

$$j = 160 + \frac{O_x}{pikseluzunluk},\quad i = 240 - \frac{\sqrt{(O_z - t)^2 + (O_y)^2}}{pikseluzunluk} \tag{5.26}$$

Söz konusu denklemlerin matlab kodu Ek-2'dedir.

Parametreleri bizim tarafımızdan kontrollü olarak belirlenen kameradan çekilen örnek görüntü Şekil 5.3a'daki gibidir. Bu resmi bildiğimiz parametre değerleriyle düzelttiğimiz zaman oluşan yeni resim ise Şekil 5.3b'dedir.

Şekil 5.3 Uygulanan geometrik düzeltmenin örnek üstünde gösterilişi

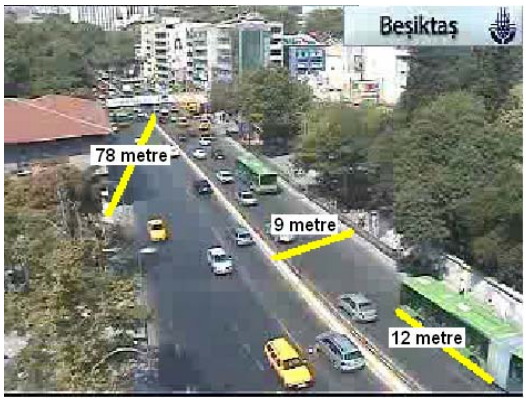
5.2 Kamera Parametrelerinin Belirlenmesi

Geometrik düzeltmenin asıl amacı olan ekran üzerindeki verilen (i,j) noktasının gerçek düzlemde düştüğü (N_x,N_z) noktasına dönüştürülmesi ancak ve ancak yukarıda açıklanan h,t,k parametrelerinin bilinmesiyle mümkün olacaktır. Projenin uygulanabilirliği düşünüldüğünde bu parametrelerin öğrenilmesinde zorluklarla karşılaşılmaktadır. Kamera yüksekliği ve açısı kameraları kuran firmadan öğrenilebilirse de bunlar yeterli olmayacaktır. Kamera yüksekliği ile h'ı, açıyla da (5.1)'i kullanarak (t+k)'yı öğrenmiş olmak bize yetmeyecektir. Bu durumda uygulamanın kullanılabilirliğini arttırmak için bu parametrelerin başka bir yoldan bulunup bulunamayacağı üzerinde durulmuştur.

Çalışmada yapılan çözüm bir ters-mühendislik yaklaşımıdır. Trafiği kontrol eden kameraların görüntüleri incelenerek gerçekte uzunluğunu ne olduğunu bildiğimiz noktaları kullanarak, acaba hangi h,t,k parametrelerini uygularsak gerçek uzunluğu elde edebiliriz sorusuna cevap aranmıştır. Üç bilinmeyen olduğuna göre (h,t,k) bu üçlüye bağlı üç denklem yazarak bu parametreleri nümerik metotlarla bulunabilir. O halde yol düzleminde doğrusal olmayan üç doğrunun gerçek uzunlukları bilindiğinde bu 3 uzunluğu belli bir hata payına göre sağlayacak h,t,k parametreleri çözüm uzayında kör arama algoritmasıyla aranıp sonuca ulaşılabilir.

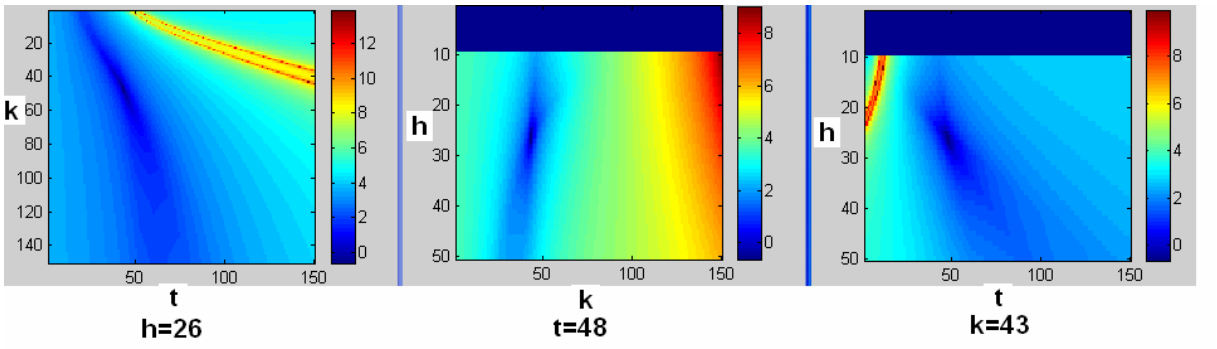
Şekil 5.4 Beşiktaş kavşağının uydu görüntüsü

İhtiyacımız olan gerçek uzunluk bilgilerine Şekil 5.4'teki gibi Google Earth programı kullanılarak ulaşılmıştır. Bu program sayesinde istediğimiz kavşağın uydu görüntülerine ulaşıp istenilen noktalar arası oldukça hassas uzunluklar elde edilebilir. Bunun yanında görüntüden geçen aracın modelini tespit edip, bu araçların uzunluğunu kataloglardan öğrenmekte mümkündür. Parametreleri bulmak için Beşiktaş kamerasında kullandığımız ölçüler Şekil 5.5'teki gibidir.



Şekil 5.5 Parametreleri bulmak için kullanılan ölçüler

İki uzunluk uydu fotoğrafındaki ölçülere göre alınırken diğer ölçü İETT tarafından kullanılan körüklü otobüslerin, körükten otobüsün başına kadar olan mesafe öğrenilip kullanılmıştır.



Şekil 5.6 Değişen parametre değerlerine göre logaritmik hata grafikleri

Söz konusu üç değeri en küçük yapacak parametre değerlerini bulabilmek için değişik parametre değerleriyle toplam hatalar hesaplanmıştır. Şekil 5.6'daki grafiklerde koyu mavi renk 0 hatayı temsil ettiğine göre hatanın en küçük olduğu değerin (26,48,43) olduğu görülmektedir. Bu sayılar çalışmamızda kör arama tekniğiyle bulunmuştur. Çözüm uzayındaki bütün üçlüler denenmiş ve en az hata veren üçlü çözüm olarak atanmıştır. Çözümün aranacağı uzay h=[10,50], t=[0,150], k=[0,150] ve arama adımımız ise 1 metre seçilmiştir. Eğer hassasiyeti cm mertebesine indirirsek çözüm uzayı üssel büyüyeceğinden kör arama etkisiz kalabilir. Bu durumda nümerik metotlarla en az hata veren üçlü rahatlıkla bulunabilir.

Şekil 5.7 a)İlk durum b)Geometrik düzeltme sonucu

Bulunan parametreler örnek resme uygulandığında oluşan geometrik düzeltilmiş resim Şekil 5.7b'deki gibi olmaktadır.

6. HIZ ÇIKARIMI

Trafiğin durumunu belirleyen en önemli parametrenin hız olduğu kabulünden yola çıkarak bu parametrenin tam bulunmasıyla yolun durumu hakkında en genel bilgi edinilmiş olunacaktır. Önerilen sistem geçen araçları takip edip her birinin hızını bulmak yerine ilgi alanında oluşan ortalama hızı vermektedir. Bu yöntemin iyi tarafı yol ile ilgili daha genel herkesin işine yarayacak bilgiyi daha az karmaşıklıkla bulmasıdır. Kötü yanı ise istenen araca odaklanıp onun hızını bulamamasıdır. Diğer bir kötü yanı ise önerilen adımlar sonucu bulunan hız sadece hareketli blokların hızlarıdır. Buradaki kötü durum eğer 3 şeritli bir yolun 2 şeriti kapalı ise ve son şerit boşsa ve bir araç yüksek süratle gelip son şeritten geçiyorsa, bu durumda sistem hareket edenin yani son şeritten hızla giden aracın hızını verecek diğer iki şeritte bekleyen araçları sanki yokmuş gibi düşünecektir. Bu durumu engellemek için yolun yoğunluğu başka bir işlem olarak bulunur ve yoğunluk bilgisine göre hız tekrar şekillendirilir.

6.1 Geometrik Düzeltmenin Hıza Etkisi

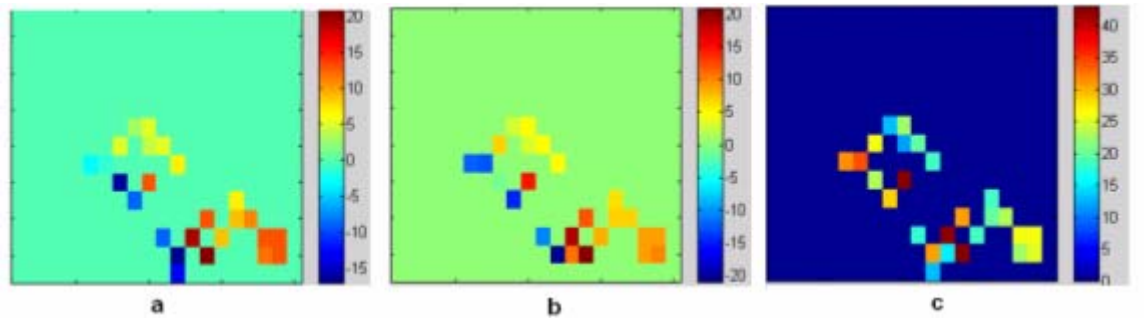
Geometrik düzeltme, hareket vektörlerinden tutulan makro blokların hızlarının gerçek ölçülere dönüştürülmesinde kullanılmıştır. Gerçek hızı hesaplamak için hareket tespit edilen makro bloğun merkezinin geometrik düzeltme sonucu oluşan konumuna ilk konum, hız vektörü eklendikten sonra oluşan yeni konumunun geometrik düzeltme uygulandıktan sonraki konumuna ise ikinci konum denir. Bu iki konum arasındaki fark gerçek yer değiştirmedir. Ekran üstündeki (i_1,j_1) noktasında (m_x ,m_y) miktarında bir hareket algılanmışsa hareketi algılanan cisim bir çerçeve boyunca ekran üstünde ($i_{1+}m_x$, ,j_1+m_y) konumuna gider. Bu durumda gerçek yer değiştirmeyi şöyle hesaplarız.

(O_{i1} ,O_{j1}) noktasına (i_1,j_1) noktasının gerçek yeri, (O_{i2} ,O_{j2}) noktasına da ($i_{1+}m_x$,j_1+m_y) noktasının gerçek yeri dersek gerçek yer değiştirme;

$$\Delta Konum = \sqrt{(O_{i1} - O_{i2})^2 + (O_{j1} - O_{j2})^2} \tag{6.1}$$

şeklinde hesaplanır. Yer değişiminin zamana bölümü ise bize anlık hızı vermektedir.

$$\nu = \frac{\Delta Konum}{\Delta t} \tag{6.2}$$



Şekil 6.1 a)Yatay hız bileşeni b)Dikey hız bileşeni c)Geometrik düzeltme sonucu hız

Şekil 6.1a ve 6.1b'de hız bilgileri ekranın alt tarafında yani kameraya yakın olan tarafta yüksek, ekranın üst tarafı yani kameraya uzak olan tarafta ise düşüktür. Fakat uygulanan geometrik düzeltme sonucu Şekil 6.1c'de görüldüğü üzere hız bileşenleri düzeltilmiş oldu.

6.2 Aykırı Değer Analizi

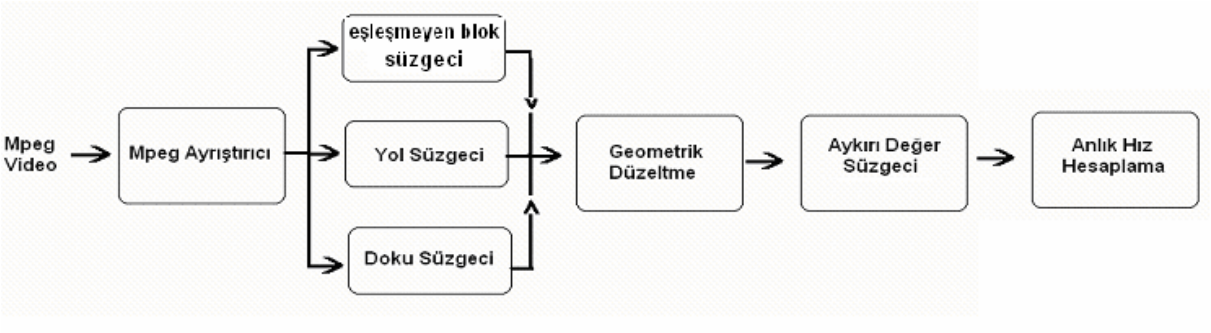
Hareket vektörlerinin belirttiği yer değişimin yol düzleminde ne kadar bir hareketi belirttiği geometrik düzeltmeyle hesaplanmış oldu. Fakat bu değerlerin içinde bazı aykırı değerler bulunmaktadır. İzlenen yola ait hareket eden bloklardan bir kısmının çok yüksek hız, bir kısmının çok düşük hız belirtmesi beklenmeyen bir durumdur. O halde bu aykırı değerler temizlenmelidir. Önerilen süzgeç (6.3)'te tanımlanmıştır.

$$\Delta Yer(x,y) < \overline{\Delta Yer(x,y)}/2 \quad yada \quad \Delta Yer(x,y) > \overline{\Delta Yer(x,y)} \times 2 \quad \Rightarrow \quad \Delta Yer(x,y) = 0 \qquad (6.3)$$

(6.3)'de görüldüğü gibi bir yola ait hareket vektörünün ifade ettiği gerçek yer değiştirme bu yola ait ortalama yer değiştirmenin yarısından daha küçük, iki katından daha büyük ise sıfırlanmalı, aksi taktirde değerlendirilmeye alınmalıdır. Bu sayede özellikle yüksek hızda araçların geçtiği yollarda daha doğru bir sonuç bulunmuştur.

6.3 Hız Çıkarım Adımları

MPEG video içindeki hareket vektörleri ayrıştırıcı sayesinde okunduktan sonra yapılacak iş vektörde bulunan gürültülerin temizlenmesidir. 4. bölümde tanımladığımız eşleşmeyen blok süzgeci, ilgi alanı süzgeci ve doku süzgecinden geçen hareket verileri işlenmek üzere alınırken diğerleri silinir.



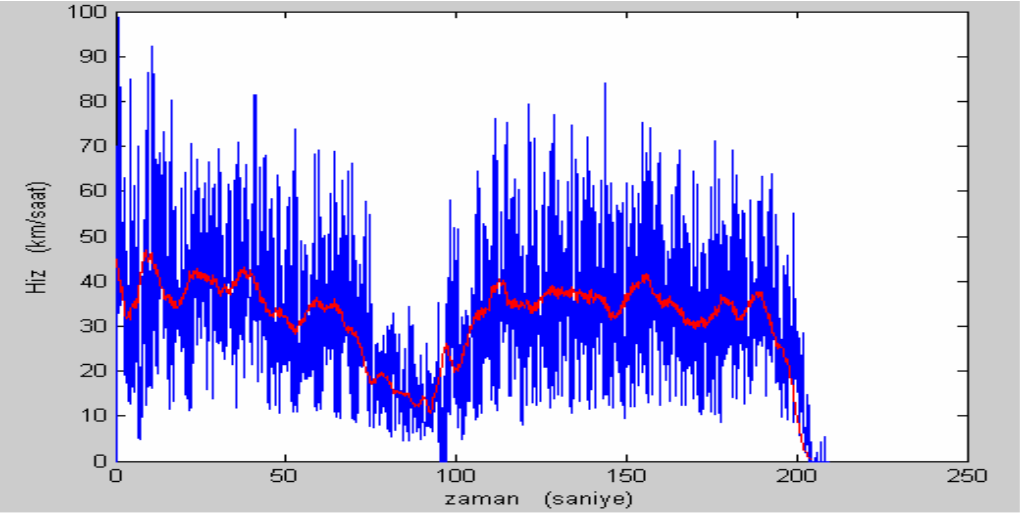
Şekil 6.2 Hız çıkarımı adımları

Süzgeçleri geçen bileşenlerin gösterdiği hareketin başlangıç ve bitiş noktaları ekran düzleminde belirlenir. Ekran düzleminde belirlenen koordinatların yol düzlemine düştüğü noktalar geometrik eşitleme fonksiyonlarıyla hesaplanır. Böylelikle anlık yer değiştirme bulunmuş olur. Anlık yer değiştirmenin (6.3)'te tanımlanan aykırı değer süzgecinden geçirilmesinden sonra o çerçevenin geçiş süresine bölümüyle anlık hızlar hesaplanır. Fakat hesaplanan anlık hızlar genelde varyansı çok yüksek diğer bir ifadeyle çok değişken verilerdir. Dolayısıyla kullanılabilir veri değildir.

Anlık hızları kullanmak yerine belirli bir zaman içinde oluşan toplam yer değiştirmenin o zamana bölümüyle oluşacak belirli periyotlu hız daha anlaşılır ve kullanılabilir olacaktır. Böylelikle ölçülen hatalı hızların etkisi azaltılır. Eğer bu süreyi çok uzun tutarsak bu sefer de yollarda oluşabilecek kısa süreli hız değişimleri fark edilemeyecektir.

$$Hiz_{t2-t1} = \frac{\sum_{i=t1}^{t2} \Delta Konum}{t_2 - t_1} \tag{6.4}$$

Geliştirilen uygulamada bu süre 2 saniye olarak alınmıştır. Beşiktaş kamerasından 3 dakika boyunca yapılan kayıtları kullanarak çıkartılan hız verileri Şekil 6.2'deki gibidir. İlk şekil anlık hızları gösterirken ikinci şekil 2 saniyelik periyotlar içinde yapılan toplam yer değiştirmenin toplam zamana bölümünü yani 2 saniyelik dilimlerdeki hızları göstermektedir.

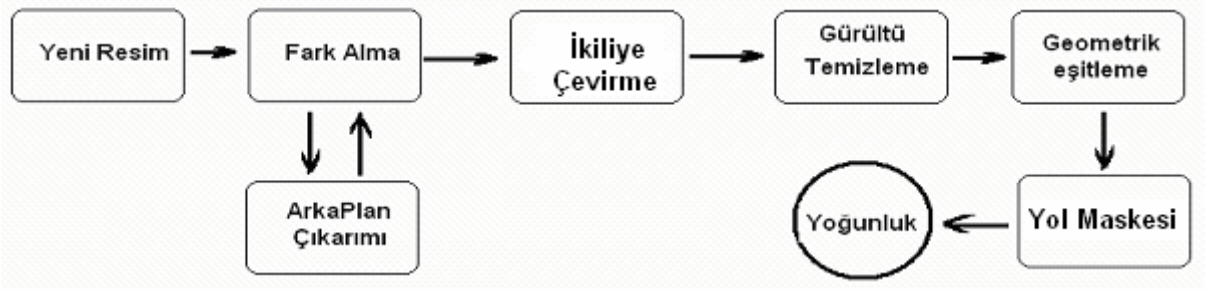


Şekil 6.2 Anlık hız ve 2 saniye periyotlu zaman dilimi ortalama hız

Anlık hızlardaki hatalar Şekil 6.2'de hemen göze çarpmaktadır. Fakat belirli 2 saniyelik periyodun hızını hesapladığımızda hatanın kompanze edilmesi daha kolay olmaktadır.

7. ARAÇ YOĞUNLUĞUNUN BULUNMASI

Yol üzerindeki araçların yolun yüzde kaçını kapladığı bilgisi planlama açısından oldukça önemlidir. Çalışmamızda yoldan geçen araçlar sayılmasa da yolun durumuyla ilgili gayet çarpıcı sonuçlar üretilebilmektedir. Yoğunluk bilgisi her ne kadar hız ile ilişkisi yüksek bir parametreyse de bu çalışmada iki parametre birbirinden bağımsız düşünülüp ayrı ayrı çıkarılmıştır.



Şekil 7.1 Yoğunluk bulma adımları

Çalışmamızda izlenen yolun önce GKM (Gauss Karışım Modeli) ile arka planı çıkarılmış, ardından basit fark işlemi yapılıp belli bir eşit değerine göre ikili (0,1) formata çevrilmiştir. En son olarak da gürültüler temizlenip araçlar işaretlenmiştir. Araçları işaretledikten sonra geometrik düzeltme sonucuna göre araçların kapladığı alan yolun kapladığı alana oranlanarak bir yoğunluk parametresi elde edilmiştir.

7.1 Arka Plan Çıkarımı

Arka plan hareketli videoda değişmeyen kısımlar olarak adlandırılabilir. Videodaki bu sabit kısımları bulunca söz konusu hareket eden objeleri algılamak, tanımak, sınıflamak oldukça kolaylaşmaktadır. Fakat arka planın zamanla değişmesi arka plan çıkarmanın tek seferlik bir iş değil de zamanla güncellenmesi gereken bir adım olduğunu göstermektedir. (7.1)'de Fg ön planı, Bg arka planı, I ise mevcut resmi göstermektedir. Arka plan çıkarımı, üzerinde oldukça fazla araştırılma yapılmış çok çeşitli metotlar geliştirilmiş bir konudur**.** Basit fark metodu, Gauss ortalama , Gauss Karışım, Kernel Yoğunluk, Mean Shift-Based, Eigenbackground metotları bilinen belli başlı arka plan çıkarım metotlarındandır.

$$Fg_i = I_i - Bg_i \tag{7.1}$$

Her ne kadar arka plan için videonun değişmeyen kısımlarıdır desek de ışıklandırma şartları, kamera osilasyonu, kıpırdayan yapraklar ve ya dalgalanan deniz gibi periyodik hareketlerle

tekrarlanan arka planların bulunması, arka planda yapılan geometrik değişiklikler (yolun kenarına kaldırım yapılması,...vs) gibi problemler karşımıza çıkan en önemli engellerdendir. Özellikle çalışmamızda karşılaştığımız bulutlar sebebiyle ışık şartlarının ani değişmesi durumunda bu değişikliği algılayacak algoritma geliştirilmesi en önemli problemimiz olarak değerlendirilmiştir. Bütün bu sayılanlar arka plan çıkarma işleminin adaptif olması gerektiğini göstermektedir.

Birçok araştırmacı elle ayarlanması gereken parametrelerin çokluğu yüzünden adaptif olmayan metotlardan vazgeçmişlerdir. Standart adaptif metoda göre arka plan gelen görüntülerin ortalamasına eşittir. Fakat bu yöntem yavaş hareket eden ya da bir süreliğine duran araçlar olduğunda onları da arka plana atacağı için çok efektif görünmemektedir. Yol dolu olduğunda araçlar arka plan olarak ayarlanacağı gibi yol açıldığında da arka planı düzeltene kadar boş olan yolu arka planla uyuşmadığından dolu gibi gösterecektir. Tüm bu sorunları çözmek için uygulanan istatistiğe dayalı GKM, her bir pikseli ayrı ayrı değerlendirip pikselin geçmişini tutmakta ve bu geçmişe göre pikselin ait olduğu sınıfı belirlemektedir.

7.2 Gauss Karışım Modeli

Gauss Karışım Modeli istatistiğe dayalı bir öğretmensiz öğrenme algoritmasıdır. Modelin basit manada yaptığı iş aynı sınıfa ait verilerin normal dağılıma uygun bir şekilde sınıf merkezi etrafında dağıldığını öngörür. Algoritma, oluşacak sınıf sayısını girdi olarak alıp bu verilerin en az varyansla toplandığı sınıf merkezlerini ve bu merkezlere ait verilerin standart sapmalarını bulur.

Modelin dayandığı matematik temel şöyledir:

$P(j)$'yi j. sınıfın dağılımdaki ağırlığı, diğer bir ifadeyle bu sınıftaki elemanların sayısının bütün eleman sayısına oranı olarak düşünürsek aşağıdaki denklemi çıkarabiliriz.

$$\sum_{j=1}^{K} P(j) = 1 \tag{7.2}$$

K uzayı ayıracağımız bölge sayısıdır.

x elemanının j. sınıfa aidiyet ihtimaline $P(x \mid j)$ dersek, bu ihtimali gauss dağılım formülüyle (7.3)'teki gibi yazabiliriz.

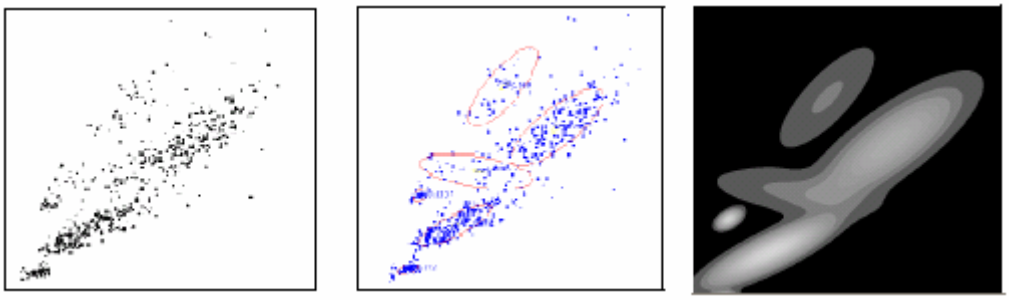
$$P(x \mid j) = \frac{1}{(2\pi)\sigma_j^2} e^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}} \tag{7.3}$$

burada x sınıflandırılacak veri, j aidiyetine bakılan sınıf numarası μ_j söz konusu sınıfın ortalaması ve $\sigma^2{}_j$ ise söz konusu sınıfın varyansını göstermektedir.

O halde sınıflamak istediğimiz bir x parçacığının j sınıfına hangi oranda ait olabileceği (7.4) deki gibi bulunabilir.

$$P(x) = P(x \mid j).P(j) \tag{7.4}$$

Bu durumda bütün x parçacıklarının söz konusu sınıflara ait olma ihtimali bütün x'ler için (7.4)'teki ihtimallerin toplamıdır. Bu ihtimali en yüksek yapacak sınıf ortalamalarının ve varyanslarının ne olacağı bulunduğunda sınıflama işlemi yapılmış olmaktadır. Bütün bu parametreleri bulma işlemi literatürde Expectation Maximization (EM) olarak bilinen yöntemle hesaplanır. Algoritma başlarken her sınıfın ortalama değerleri, standart sapmaları ve ağırlıkları rastgele atanır ve her bir iterasyon için bu değerler güncellenir. Örnek bir veri kümesi ve bu kümeyi GKM'nin ayırdığı uzay Şekil 7.2'deki gibidir.



Şekil 7.2 a)Örnek bir biyolojik veri b)Bu verinin GKM ile bulunan sınıf sınırları c)Verilerin sınıflara aidiyet dereceleri

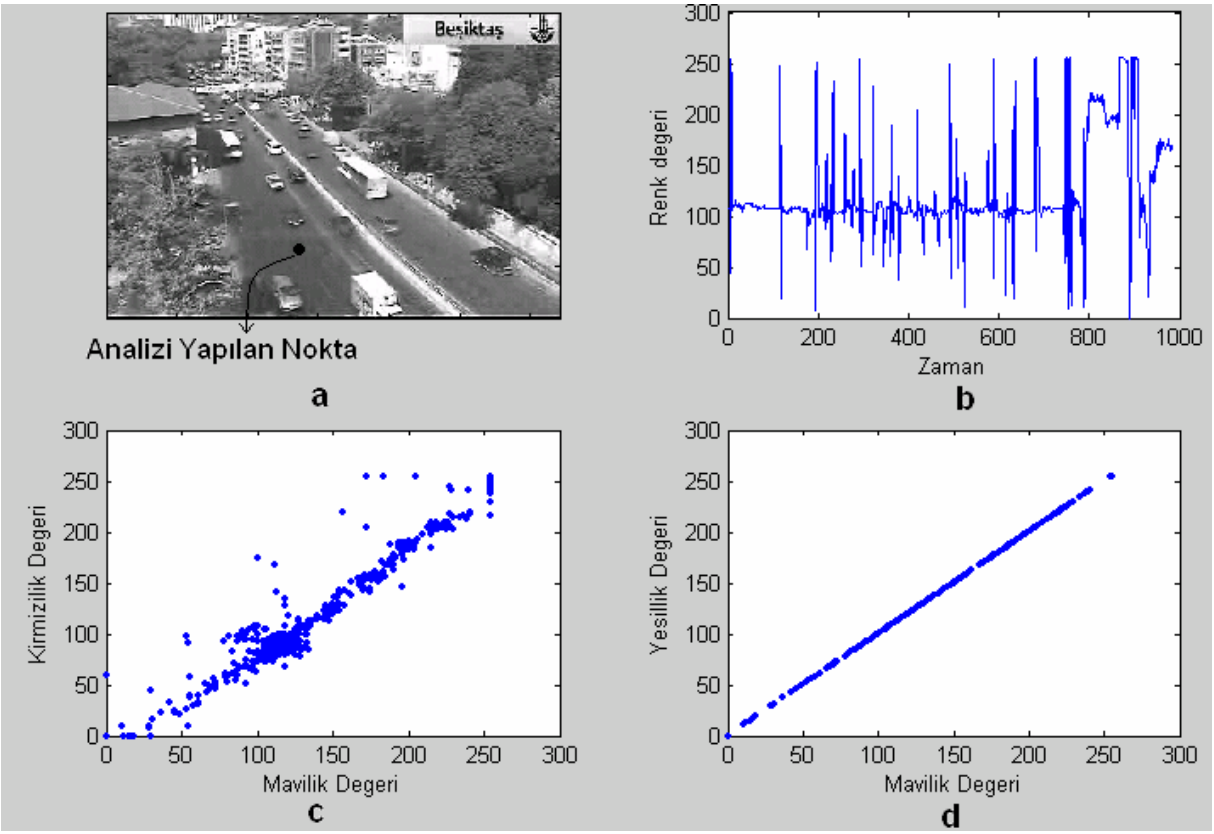
7.3 GKM Tabanlı Arka Plan Çıkarımı

Arka plan çıkarımı için Stauffer C. ve Grimson W'nin önerdiği GKM tabanlı yöntem kullanılmıştır. Bu model her bir pikselin geçmişini tutmakta ve bu geçmiş durumdan şimdiye kadarki süreç içinde pikselin aldığı renk değerleri belirlenen sınıflara ayrılmaktadır. Bu sınıflardan ağırlığı en büyük olan arka plan sınıfı olarak atanmaktadır. Ardından pikselin her yeni gelen değerine göre sınıflandırma yapılıp arka plan sınıfına mı ön plan sınıfına mı ait olduğu araştırılmakta ve ait olduğu sınıfın değerleri daha sonra açıklanacak eşitliklere göre

güncellenmektedir. Bu metodun elle ayarlanması gereken tek parametresi eşitliklerde α olarak gösterilen öğrenme katsayısıdır.

Arka plan çıkarımda optimum sınıf parametrelerini bulmak için EM algoritması kullanılmamıştır. Çünkü bu algoritma off-line işlem yapmaktadır. Yani belirli sayıdaki geçmiş bilgiler alınıp sınıflama yapılmaktadır. Bu durumda yeni bir piksel değeri elde edildiğinde yeniden sınıflama yapmak gerekmektedir. Bu da her yeni gelen çerçeve için piksel sayısınca sınıflama anlamına gelmektedir. Stauffer C.'nin önerdiği k-means benzeri on-line parametre güncelleme adımları sayesinde yeni gelen piksel bilgileri geçmiş bilgilere öğrenme katsayısı kadar etki edecek şekilde güncellenmektedir. Böylelikle her yeni gelen çerçeve için bir kez güncelleme yapılmaktadır.

Bu metodun en güçlü yanı periyodik olarak değişen arka planları saklaması ve birden çok durumu arka plan olarak sınıflayabilmesidir.



Şekil 7.3 a)Analizi yapılan nokta b)Bu noktanın zamana bağlı ışık şiddeti c)Bu noktanın kırmızı-mavi ekseninde aldığı değerler d)Bu noktanın yeşil-mavi ekseninde aldığı değerler

Şekil 7.3a'da resim üstündeki söz konusu pikselin renk değerlerini zaman ekseninde analiz edersek karşımıza Şekil 7.3b'deki şekil çıkar. Görüldüğü üzere renk bileşenlerindeki ani çıkış ve düşüşler bize arka plandan farklı bir durum geldiğini işaret etmektedir. Şekil 7.3b ve c 'de

bu pikselin renk bileşenlerinin ilk 1000 çerçevede aldığı durumun 2 boyutta görüntüsü verilmiştir. Mavi-yeşil ekseninde görüntü, tam bir düz çizgi gibi çıkarken mavi-kırmızı ekseninde bazı dağılmalar olduğu görülmektedir. GKM'nin yaptığı her bir pikseli zaman içinde değişen değerlerine göre sınıflamaktır.

Modelin matematiksel temeli şöyledir.

Deneysel olarak belirlenen (genelde çalışmalarda 3 ila 5 arası seçiliyor) sınıf sayısı (K adet) kadar sınıf ortalaması, sınıf varyansı ve sınıf ağırlıkları rasgele olarak belirlenir.

$$\{w_i, \mu_i, \sigma_i\} = \{S_i(w, \mu, \sigma) : 1 \le i \le K\} \tag{7.5}$$

burada w sınıfın ağırlık değerini, μ sınıfın ortalama değerini, σ sınıfın varyansını, K sınıf sayısını, S ise sınıf adını göstermektedir. Resmin x,y koordinatındaki pikselin t anındaki değerine X_t, resme ise I dersek bu ifadeyi,

$$\{X_1, ..., X_t\} = \{I(x, y, i) : 1 \le i \le t\} \tag{7.6}$$

şeklinde yazabiliriz. Gelen her yeni resim için yol maskesi içinde kalan bütün pikseller tek tek taranıp aidiyet fonksiyonuna sokulup en çok hangi sınıfa ait oldukları hesaplanır. Yeni gelen pikselin ait olduğu sınıf şu şekilde bulunur.

$$Sinif = \underset{i=1..K}{\arg\max} P(I_{xy} \mid S^i_{xy,t-1}) \tag{7.7}$$

(7.7)'de tanımlanan $S^i_{xy,t-1}$ ifadesi resimdeki x,y koordinatında bulunan pikselin t-1 anındaki i. sınıfını ifade eder. Bu şekilde bütün sınıflardan hangisine ait olduğu araştırılır. Fakat araştırılırken bulunan en büyük aidiyet değeri eğer 2.5 standart sapmadan daha büyük ise bu sefer hiçbir sınıfa ait olmadığına hükmedilir ve ağırlığı en küçük olan sınıf yok edilip o sınıfın yerine yeni gelen piksel değeri konulur. Bu yapılan işlemler bize K-means sınıflandırmayı hatırlatmaktadır. Yeni gelen değer eğer bir sınıfa atılırsa hemen o sınıfın değerleri şu şekilde güncellenir.

$$w_{i,t} = (1-\alpha).w_{i,t-1} + \alpha(M_{i,t}) \tag{7.8}$$

$$\mu_{i,t} = (1-p)\mu_{i,t-1} + pI_{xy,t} \tag{7.9}$$

$$\sigma^2_{i,t} = (1-p)\sigma^2_{i,t-1} + p(I_{xy,t} - \mu_t)^T (I_{xy,t} - \mu_{i,t}) \tag{7.10}$$

Eşitliklerde geçen p ifadesinin eşiti ise (7.11)'dedir.

$$p = \alpha P(I_{xy,t} \mid Sinif) \tag{7.11}$$

Bu eşitliklerde kullanılan $w_{i,t}$ i. sınıfın t anındaki ağırlığı, $I_{xy,t}$ t anında resmin x,y koordinatındaki piksel değeri, $\mu_{i,t}$ i. sınıfın t anındaki ortalaması, $\sigma_{i,t}^2$ i. sınıfın t anındaki varyansı, p değeri resmin incelenen pikselinin incelenen sınıfa aidiyetinin öğrenme katsayısıyla çarpılmış değerini göstermektedir. (7.8)'de kullanılan $M_{i,t}$ değeri ise t anında incelenen piksel i. sınıfa aitse değeri 1, değilse değeri 0 olan bir sayıdır.

Eğer incelenen pikselin hiçbir sınıfa olan aidiyeti o sınıfın standart sapmasının 2.5 katından daha az değilse, diğer bir ifadeyle söz konusu piksel hiçbir sınıfa dahil edilememişse bu durumda ağırlığı en az olan sınıf iptal edilir ve o sınıfın yerine ortalaması yeni gelen piksel olan yeni bir sınıf açılır. Bu yeni sınıfın varyansı diğer sınıfların içinde en az varyansa sahip sınıfın varyansı olarak belirlenir.

$$S_{xy,y} = \arg\min_{i=1..K} w_{xy,i} \tag{7.12}$$

$$\mu_{S_{xy,y}} = I_{xy} \tag{7.13}$$

$$\sigma^2_{S_{xy,y}} = \min_{i=1..K}(\sigma^2_{xy,i}) \tag{7.14}$$

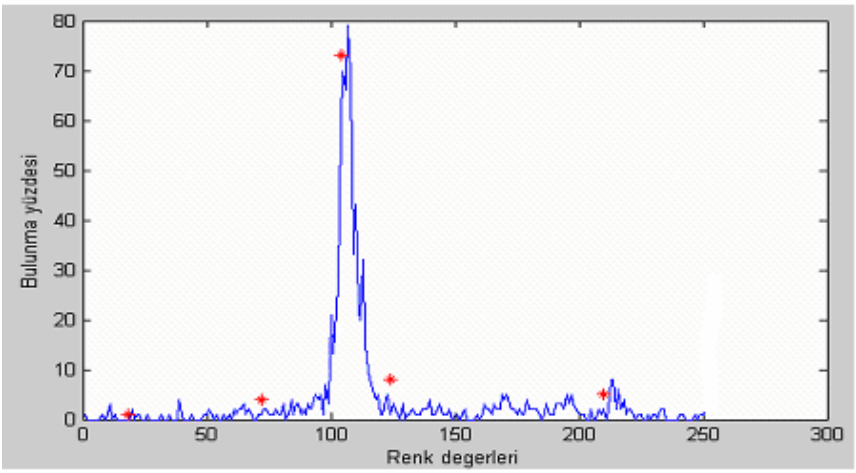
Yeni oluşan sınıfın ağırlığı belirlenirken (7.15)'deki ifade uygulanır.

$$w_{i,y} = \begin{cases} 0.05 & i = S_{xy,y} \\ w_{i,y} - \frac{1}{K}(w_{S_{xy}} - 0.05 & i \neq S_{xy,y} \end{cases} \tag{7.15}$$

bu formüle göre yeni oluşan sınıfın ağırlığına 0.05 değeri atanırken diğer sınıfların değeri de toplamları 1 olacak şekilde güncellenmektedir.

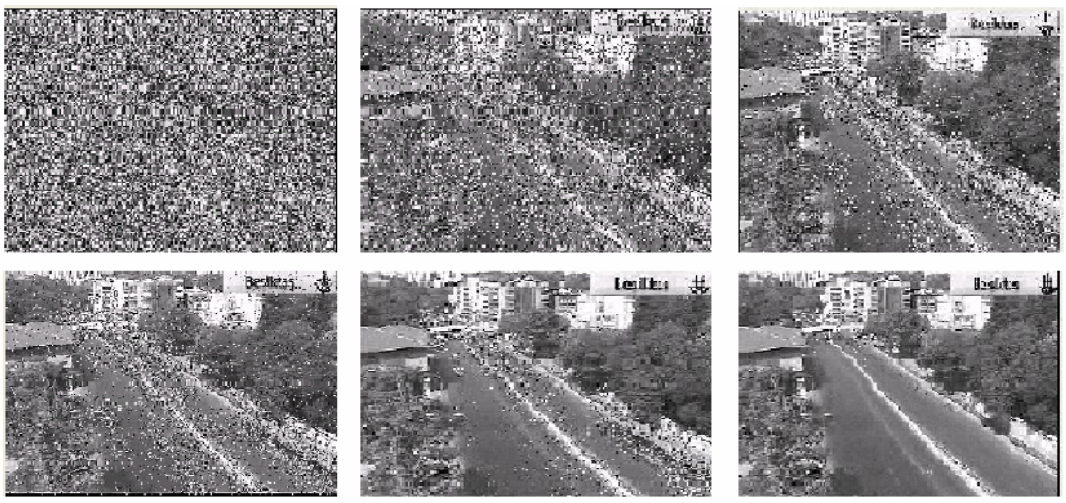
Takip edilen (185,147) koordinatlarındaki pikselin gri seviyesindeki histogramı Şekil 7.4'tedir. Yukarıda anlatılan adımlar uygulandığında bu piksel için GKM'nin oluşturduğu sınıf merkezleri ve bu sınıfların ağırlıkları yine şekil üzerinde gösterilmiştir. Kırmızı yıldız ile gösterilen yerlerin yatay ekseni sınıf merkezlerini, dikey eksenleri ise o sınıfın ağırlık yüzdelerini göstermektedir. Bu duruma göre arka plan sınıfının merkezi 104 ve bu sınıfın

ağırlığı ise 0.73 olacaktır. Ayrıca bu sınıfın varyansı 4.39 çıkmıştır.



Şekil 7.4 Takip edilen pikselin histogramı, GKM'nin bulduğu sınıf merkezleri ve bu merkezlerin ağırlıkları

Burada karşımıza çıkan bir problem de hangi sınıfın arka plan olarak atanacağı problemidir. Literatürde bir çok tanım yapılmış bulunmaktadır. Stauffer çalışmasında ağırlıkları büyükten küçüğe doğru sıralayıp ağırlık toplamının belli bir eşik değerini aşana kadar tüm sınıfların arka plan, diğerlerinin ön plan olduğunu savunmuştur. Bu sayede bir çok sınıfın arka plan olma özelliğini yakalayabileceği, diğer bir ifadeyle kırpınan yaprakta her bir durumun arka plan olarak saklanacağını savunmuştur. Pavlidis I. ise sınıfları, sınıf ağırlığını sınıf varyansına oranlayarak sıralamak ve bunlardan en yüksek belli sayıdaki sınıfı arka plan olarak atamayı önermiştir. Bu çalışmada Stauffer'in metodu tercih edilmiştir. Yapılan testlerde daha başarılı olduğu görülmüştür. Bu durumda arka plan olarak atanan sınıflar şu şekilde belirlenmiş olur.



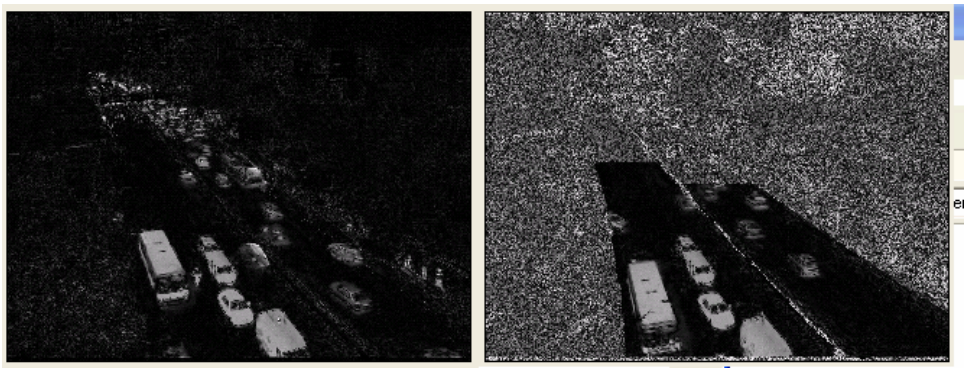
Şekil 7.5 Arka plan çıkarım algoritmasının çalışmasına bir örnek

$$i = 1..K-1 \quad için \quad w_{xy,i+1} > w_{xy,i} \rightarrow B_{xy} = \arg\min_{b} \sum_{i=1}^{b} w_{xy,i} > T \tag{7.16}$$

Burada B_{xy} ifadesi koordinatları x,y olan noktanın arka planına atılan sınıf numarasını vermektedir. Programın çalışma anı Şekil 7.5'te özetlenmiştir. Önce rasgele değerlerle başlayan arka plan, iterasyonlar ilerledikçe gerçek arka plana doğru yaklaşmaktadır.

7.4 Araçların Bulunması

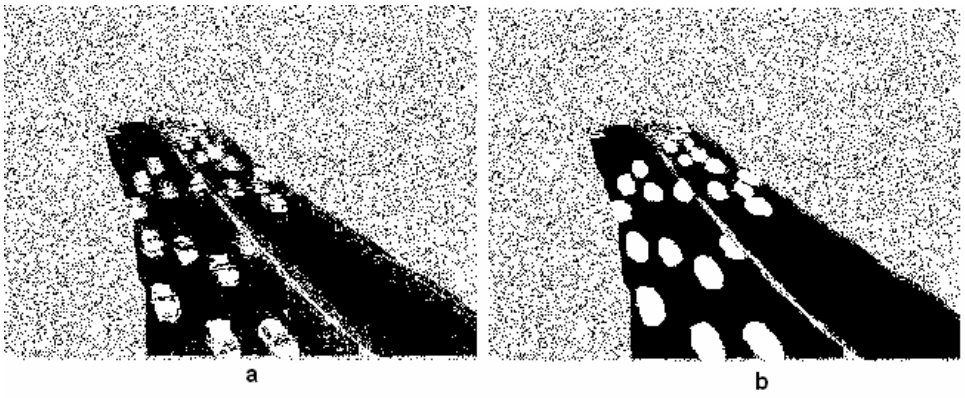
Arka plan çıktıktan sonra yapmamız gereken işlemler sırasıyla fark alma, ikiliye çevirme ve yoğunluk ölçümüdür. Arka planı ne kadar iyi bulabilirsek o ölçüde ön plana yaklaşmış oluruz. Eşitlik (7.1)'de gösterildiği gibi arka planla güncel resim arasındaki mutlak farkları alırsak ve bunu ilgi alanı süzgecinden geçirirsek Şekil 7.6 ile karşılaşırız.



Şekil 7.6 Güncel resim ile arka planın farkı

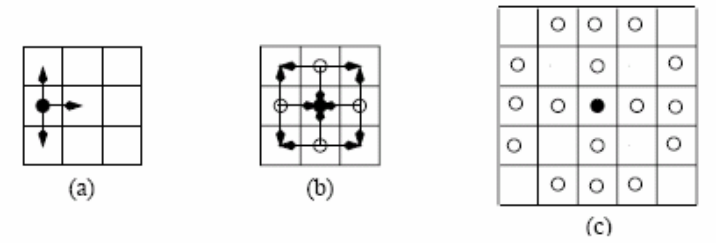
Şekil 7.6'daki resimlere basit bir ikileme işlemi uygularsak araçları tam olarak ayırt edemeyiz. Arka plan olarak atanan piksellerin bazıları çevresel gürültüler yüzünden değişime uğrayan arka plan pikselleriyle uyuşmadığından bazı yanlış eşleşmelerle karşılaşılabilmektedir.

Basit mutlak fark işleminden sonra belirlenen bir eşiği geçen pikseller 1 geçemeyenler 0 olarak etiketlendiğinde oluşan resim Şekil 7.7a'daki gibidir. Bu resim bir çok gürültüyü de içermektedir. Araçlar elle işaretlendiğinde olması gereken görüntü Şekil 7.7b'deki gibidir. Bu resimlerin arasındaki fark hesaplandığında arka plan olduğu halde ön plan algılanan piksel sayısı 670, ön plan olduğu halde arka plan algılanan piksel sayısı ise 902 olmakta böylelikle toplam yanlış bulunan piksel sayısı 1572 olmaktadır. Yolu oluşturan toplam piksel sayısının 25628 olduğunu bildiğimize göre toplam yanlış oranı bu resim için %6,13 çıkmaktadır.



Şekil 7.7 a) Hesaplanan fark b)Olması gereken fark

İkiliye çevirme işleminden sonra oluşan hatayı düşürmek için literatürde bazı yöntemler geliştirilmiştir. Bevilacqua çalışmasında bu problemi çözmek için kendisi tarafından geliştirilen yeni bir morfolojik operatör önermiştir. Bu operatör sayesinde yapısal analiz yaparak araç olması muhtemel yerleri güçlendirirken arka plan olması muhtemel yerleri söndürmektedir. Bizim yaptığımız testlerde Gauss filtrelerinden daha başarılı olduğu sonucuna varıldığından bu yöntem tercih edilmiştir.



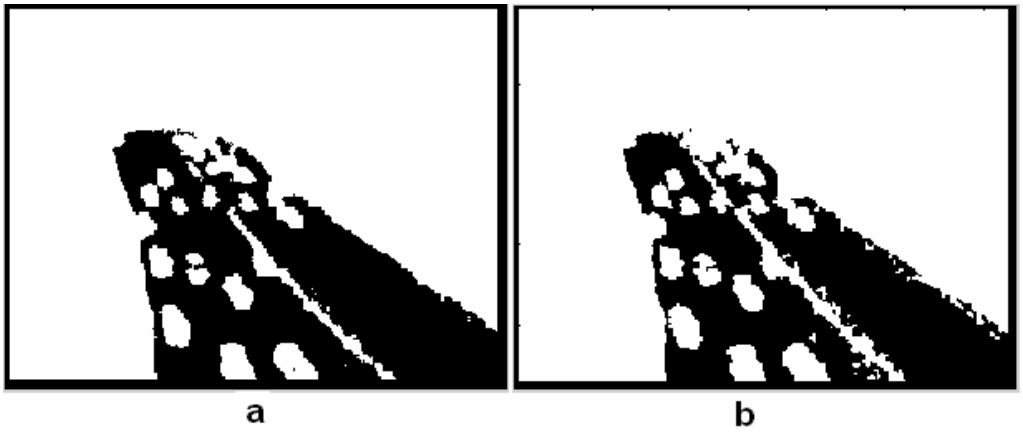
Şekil 7.8 Bevilacqua'nın tanımladığı operatörler

Yöntemin en temel adımı bir pikselin kendisi, önündeki ve her iki yanındaki piksel değerlerinin toplanmasıdır. Şekil 7.8a'da görüldüğü üzere bu 4 piksel değeri toplanacaktır. Bu işlem eğer seçtiğimiz matris 3*3'lük ise bütün kenarların ortaları için yapılacaktır. Eğer matrisimiz 5*5'lik ise Şekil 7.8c'de olduğu gibi önce en dış kareden başlayıp köşede olan pikseller hariç bütün kenarlar üzerinde bu işlem gerçekleştirilmelidir. Ardından kareyi küçültüp aynı işlemlerin 3*3'lük kare için gerçekleştirilmesi gerekmektedir. Bütün bu yapılan toplama işlemlerinin sonucu, merkez pikselin değeri olarak atanmaktadır. Bu durumda yapılan iş aslında bir operatörü resmin üzerine dolaştırmaktan farksızdır. O halde 5*5'lik matris örneği için düşünecek olursak her bir toplama değeri için toplanan pikselin değerini 1 arttırırsak oluşacak operatör matrisi Şekil 7.9'daki gibi olacaktır.

2	2	3	2	2
2	4	2	4	2
3	2	4	2	3
2	4	2	4	2
2	2	3	2	2

Şekil 7.9 Bevilacqua'nın tanımladığı matris

Bu matris operatörünü fark alınmış ikiliye çevrilmiş resmin üzerine uygularsak oluşan resim Şekil 7.10'daki gibi olmaktadır.

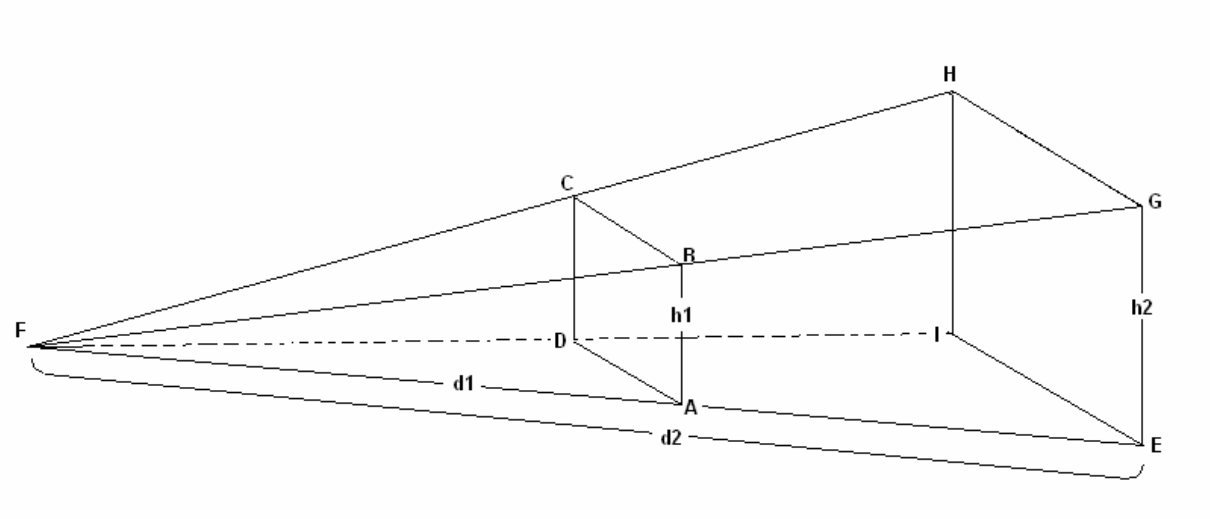


Şekil 7.10 a)Bevilacqua süzgeç sonucu b)Gauss süzgeç sonucu

Şekil 7.10'daki görüntüler her iki metot içinde en iyi sonucu veren eşik değerleri için hesaplanan çıktılardır. Bevilacqua'nın tanımladığı süzgeçle hata oranı %3,2'ye düşerken Gauss süzgeciyle %3,9'da kalmıştır.

7.5 Geometrik Düzeltmenin Araç Yoğunluğuna Etkisi

Geometrik düzeltme olmaksızın araç yoğunluğunu tam olarak hesaplamak imkansızdır. Kameraya yakın durumdaki araçlar büyük, uzaktakiler küçük göründüğünden yoğunluk tam hesaplanamayacaktır. Bunu engellemenin yolu geometrik düzeltmeyi araç yoğunluğu hesaplamaya da dahil etmektir. Bir cismi bakılan yere uzaklığı nispetinde boyutu değişmektedir. Cismin kameraya uzaklığı ile görünür boyutu doğru orantılıdır. O halde ön plan olarak belirlenmiş bir pikselin yoğunluğa etkisini hesaplayabilmek için onu kameraya olan uzaklığıyla çarpmamız yeterli olmaktadır.



Şekil 7.11 Mesafenin görünen boya etkisi

Şekil 7.11'de göre F noktasından kenarı h1 olan ABCD karesi ve kenarı h2 olan EGHI karesine bakan göz her iki kareyide aynı büyüklükte görür. O halde aralarındaki boy oranı benzer üçgenlerden (7.17)'deki gibi yazılabilir.

$$\frac{h_1}{h_2} = \frac{d_1}{d_2} \tag{7.17}$$

Bu eşitliğe göre cisimlerin boylarının birbirlerine oranının uzaklıkları oranı olduğu çok rahat bir şekilde gösterilebilir. Bu iki karenin kameradan algılanan alanlarına x dersek ve yoğunluk hesabı için araçların kapladığı alanlar hesaplanması gerektiğinden algılanan cismin uzaklığı ile alanı arasındaki ilişki (7.18)'deki gibi yazılabilir.

$$A(ABCD) \approx x.d_1^{\,2} \qquad A(EGHI) \approx x.d_2^{\,2} \tag{7.18}$$

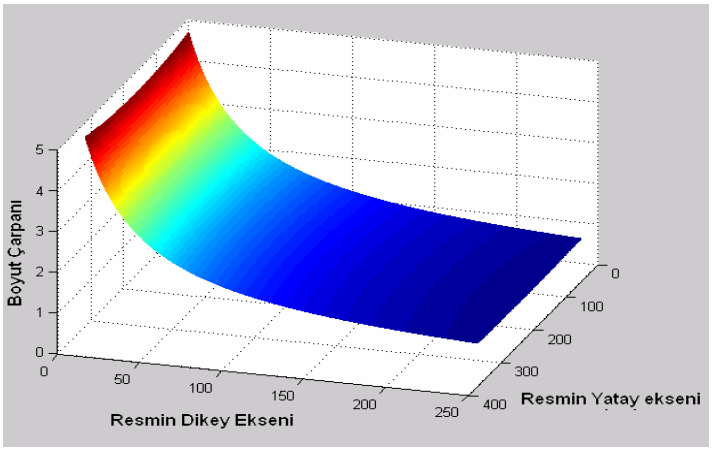
Çalışmamızda her pikselin kamera direk dibine olan uzaklığını 5. bölümde anlatılan dönüşümlerle hesaplanmıştı. O halde kameranın yüksekliğini bildiğimize göre ekrandaki her pikselin gerçek yerinin kamera odağına olan uzaklığı (7.19)'daki gibi hesaplanabilir.

$$M_{I_{xy}} = \sqrt{h^2 + X_{I_{xy}}^2 + Z_{I_{xy}}^2} \tag{7.19}$$

(7.19)'da I_{xy} resim üzerinde x,y koordinatlarına sahip noktayı, $X_{I_{xy}}$ resim üzerinde x,y koordinatına sahip noktanın yol düzlemindeki x koordinat bileşeni , $Z_{I_{xy}}$ aynı şekilde söz konusu noktanın z koordinat bileşenini temsil etmektedir. h ise kameranın yüksekliğini

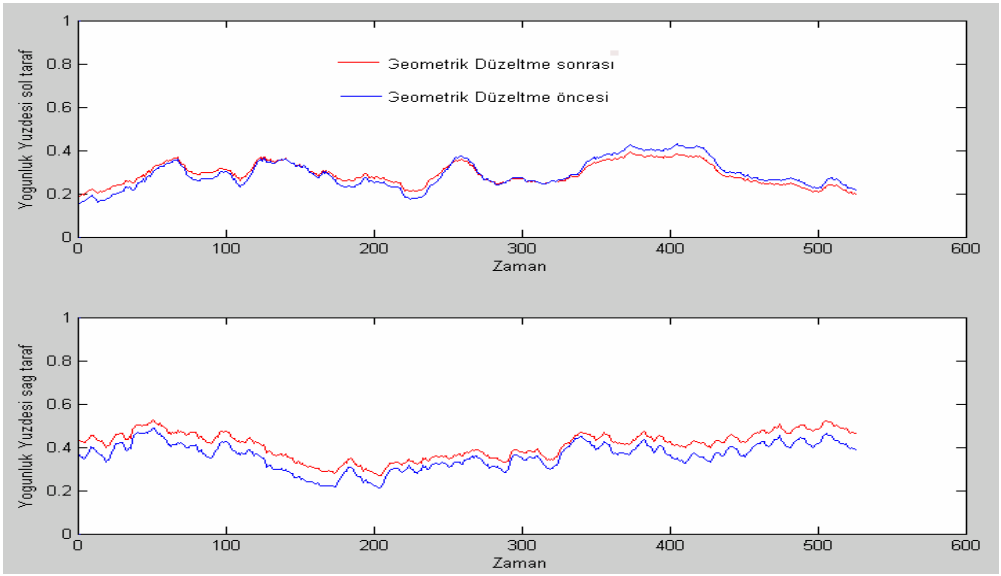
vermektedir. Bu eşitliğe göre resimde her bir piksele karşılık gelen uzaklık değerleri hesaplanıp bir matris oluşturulursa oluşan matris aşağıdaki gibi olur.

Şekil 7.12'de görüntüsü verilen matris oluşturulurken görüntü alanının tam merkezinde olan yani resim üzerinde (120,160) koordinatında bulunan pikselin ağırlığı 1 olacak şekilde normalize edilmiştir. Bu matrise göre görüntünün en altında bulunan piksel 0,58 ile en üst noktada bulunan piksel ise 4,51 ile çarpılırsa gerçek uzunluk etkisi çıkacağı anlaşılmaktadır. Şekil 7.12'deki matris uzunlun oransal katsayısını vermektedir. Çalışmamızda bize alanların oransal katsayısı gerektiğinden bu matrisin karesi alınmalıdır. O halde uzunluğun oransal katsayılarının karesini alırsak alanın oransal katsayısını elde ederiz. Bu durumda ekranın en alt satırında bulunan pikselin alan katsayısı 0,3364, ekranın en üst noktasında bulunan pikselin alan katsayısı ise 20,34 olmaktadır.



Şekil 7.12 Beşiktaş kamerası için hesaplanan boyut düzeltme matrisi

Şekil 7.13'te görüldüğü üzere geometrik düzeltme uygulandığında yoğunluk grafiği biraz fark etmiştir. Bu farkın hesaplamalarda %7 olduğu görülmüştür. Örneğin yolun sol tarafında 400. çerçeve dolaylarında yolun ekrana yakın kısmında bir yığılma olmaktadır. Dolayısıyla geometrik düzeltme olmaksızın yapılan yoğunluk çıkarımında biraz yükselmeler meydana gelmektedir. Aynı şekilde yolun sağ tarafında ise genelde ekrana uzak taraflarda bir yığılma göze çarpmaktadır. Fakat geometrik düzeltme olmaksızın yapılan yoğunluk çıkarımı yoğunluğu düşük gösterirken geometrik düzeltme sonucu yoğunluk olması gerektiği şekliyle bulunmuştur.



Şekil 7.13 Geometrik düzeltmenin yoğunluğa etkisi

8. SONUÇ VE ÖNERİLER

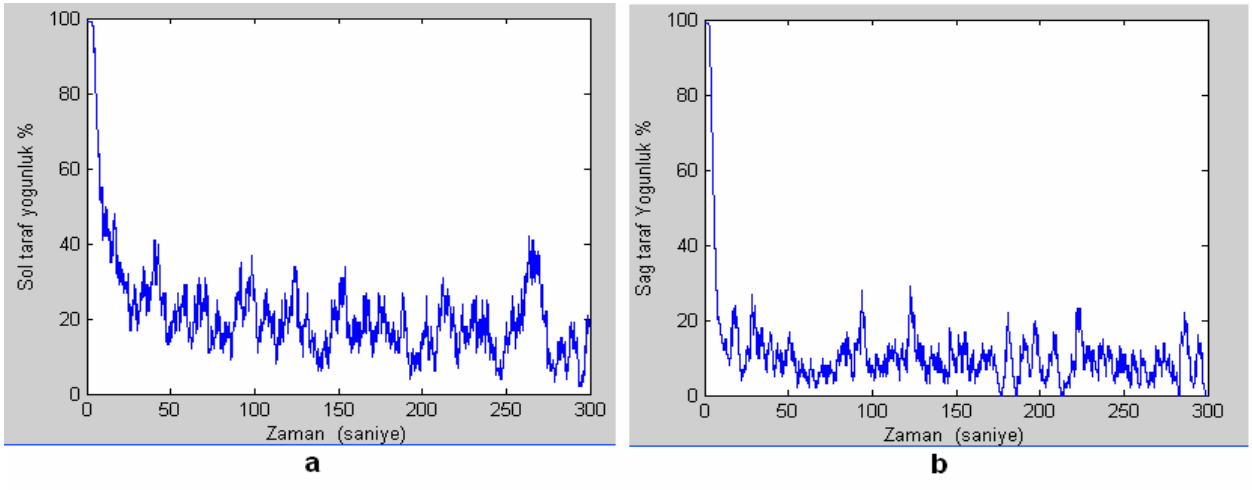
Yapılan testlerde hız bulma algoritmamız gece, gündüz, bulutlu, güneşli havalarda çevre şartlarından etkilenmeden hızları oldukça iyi bulabilmesine karşı, yoğunluk bulma algoritmamız trafiğin uzun süre tıkalı durduğu durumlarda, ışığın çok değişken olduğu ortamlarda arka planı kaybettiği için gerçek yoğunluğu bulamamaktadır. Ayrıca gece araçların farları yolun büyük bir kısmını kapladığı için bu alanlar arka planla eşleşmediğinden sanki büyük bir araç geçtiği yorumunu yapmaktadır. Yapılan testleri 2 sınıfa ayırabiliriz.

8.1 İdeal Durumda Sistemin Davranışı

Sistemimizi test ettiğimiz kameralardan alınan görüntüler ve buna paralel olarak çıkarılan hız, hareket eden blok sayısı ve yoğunluk grafikleri şu şekildedir.

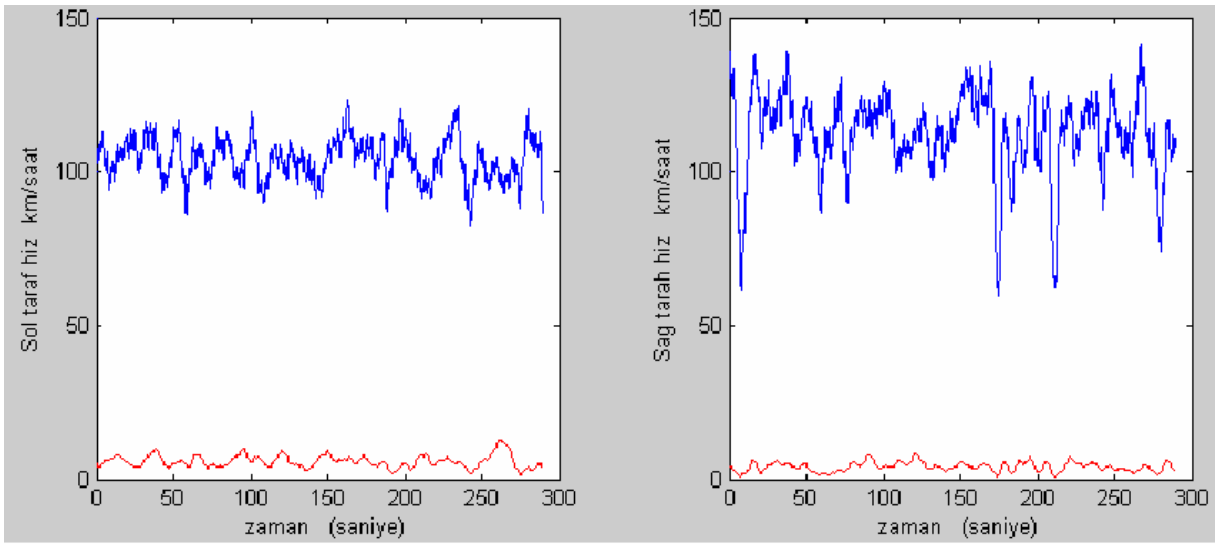
Şekil 8.1 Mecidiyeköy köprü girişi kamerası görüntüsü

Şekil 8.1'de Mecidiyeköy E-5 üzerindeki Boğaziçi köprüne girişi kontrol eden kameranın 300 saniyelik görüntüsünden bir kare görülmektedir. Yapılan testlerde bu kameranın h,t,k parametreleri h=33, t=47, k=89 çıkmıştır. Bu parametrelere göre geometrik düzeltme yapılıp istenen bilgiler çıkarılmaya çalışılırsa oluşacak grafikler şu şekildedir.



Şekil 8.2 Mecidiyeköy E-5 kamerasında algılanan yoğunluk grafiği

Grafiğe göre yoğunlukların başta %100 olarak görünmesinin sebebi arka planın henüz çıkmamış olmasıdır. Arka plan 25. saniyede bulunduktan sonra yoğunluk bilgileri düzgün çıkmıştır. Yolun genelinde sol taraf sağ taraftan daha yoğun görünmektedir. Bu bilgi grafiğe yansımaktadır.



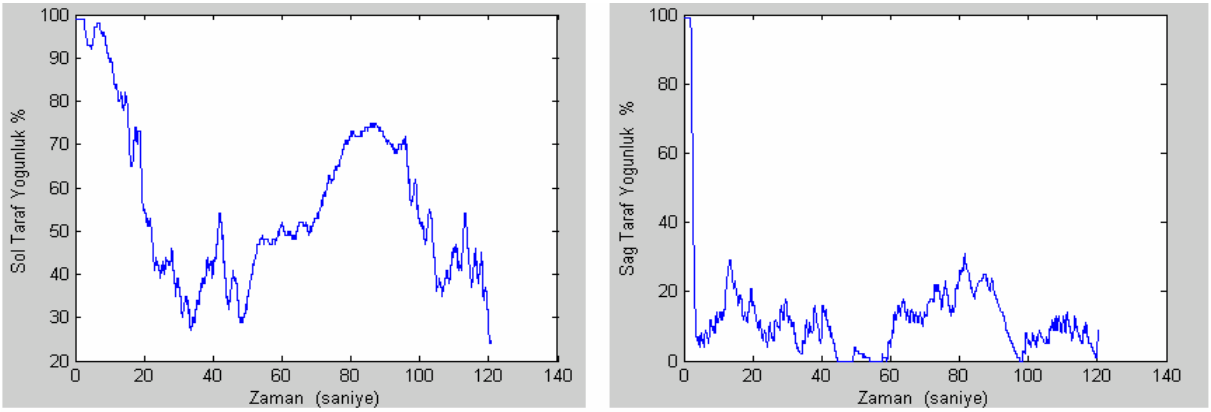
Şekil 8.3 Mecidiyeköy E-5 kamerasında algılanan hız ve hareket eden blok grafiği

Hız grafiğinde ise araç yoğunluğu fazla olan sol tarafın 300 saniye içindeki ortalaması 104 km/saat, araç yoğunluğu daha düşük olan sağ tarafın ortalama hızı ise 112 km/saat çıkmıştır. Yolun solu için hız grafiğinde bazen keskin düşüşler görülmektedir. Bunlar o sırada yoldan araç geçmediği durumları göstermektedir. Hız eğrisinin altındaki kırmızı eğri ise yolun söz konusu kısmında algılanan hareketli makro blok sayılarını vermektedir. Hız ani düşüş

gösterdiğinde hareket eden blok sayısı da düşüş göstermektedir. Çünkü bu durumda yoldan araç geçmemektedir.

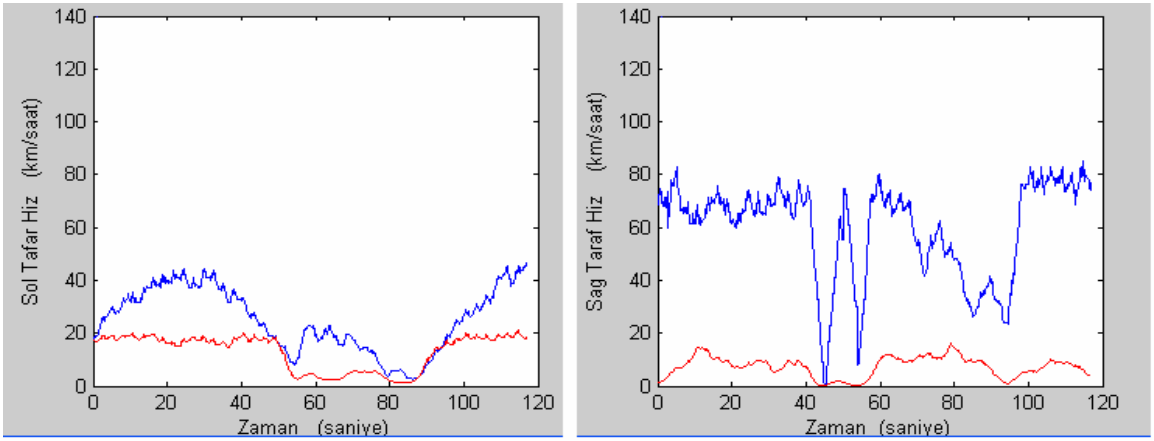
Şekil 8.4 Havaalanı kamerasının görüntüsü

Şekil 8.4'te havaalanı yolunda bulunan trafik kamerasının çektiği 120 saniyelik görüntünün bir karesi verilmektedir. Bu kamera için bulunan parametreler h=26 metre, t=52 metre ve k=47 metredir. Burada bir yol çok yoğunken diğer yönde fazla yoğunluk görülmemektedir. Daha videonun başından itibaren yol yoğunluğu sağ tarafta yaşandığı için sol taraf arka plan çıkarımını 25. saniyede yapabilmiş. Bundan dolayı 25. saniyeden sonra yoğunluk değerleri gerçek değerleri göstermektedir. Sağ taraf ise videonun genelinde akıcıdır. Bu yüzden daha 8. saniyede arka plan bulunmuştur. Sol taraf 55. saniyede tıkanmaya başlamakta ve bu durum 90. saniyeye kadar devam etmektedir. Sol tarafta ise 45-60 saniye arasında yalnızca 1 araç geçmektedir. Bu durum yolun sağ kısmının yoğunluk grafiğinde göze çarpmaktadır.



Şekil 8.5 Havaalanı kamerasında algılanan yoğunluk grafiği

Hız için de oldukça güvenli sonuçlar elde edilmiştir. Daha videonun başında sol taraf tıkalıyken yavaş yavaş açılmaktadır. Grafikte görüldüğü üzere hızda yavaşça bir artım olmaktadır. 55. saniyede hızın en alt noktaya inmesi aynı şekilde hareket eden blokların sayısında ani düşüş yaşanmasının sebebi yolun tıkanmaya başlamasıdır.



Şekil 8.6 Havaalanı kamerasında algılanan hız ve hareket eden blok sayısı grafiği

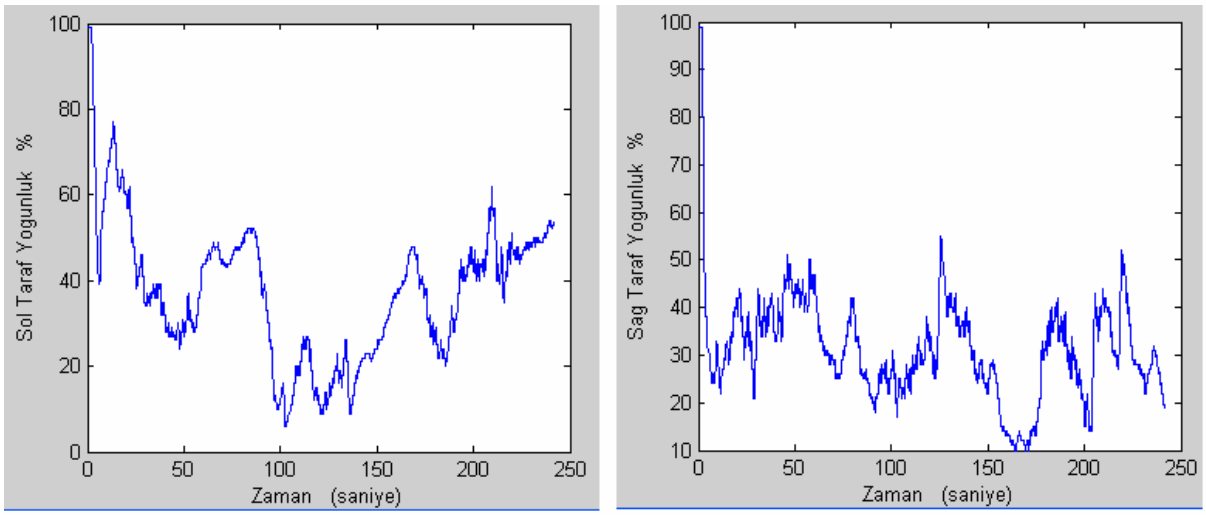
90. saniyeden itibaren yol tekrar açıldığı için hareket eden blok sayısı ve hızda yükselmeler başlamıştır.

Yolun solunda ise hiç tıkanma meydana gelmemiştir. Hızdaki ani düşüşün sebebi yoldan hiç araç geçmemesidir. Aynı şekilde bu zaman aralığında yol yoğunluk grafiğine bakılırsa zaten araç yoğunluğunun en alt seviyede olduğu görülecektir. Bu 120 saniyelik kayıt içerisinde sol tarafın ortalama hızı 25 km/saat, sağ tarafın ortalama hızı ise 59.5 km/saat olarak tespit edilmiştir.

Şekil 8.7 Fındıklı kamerasının görüntüsü

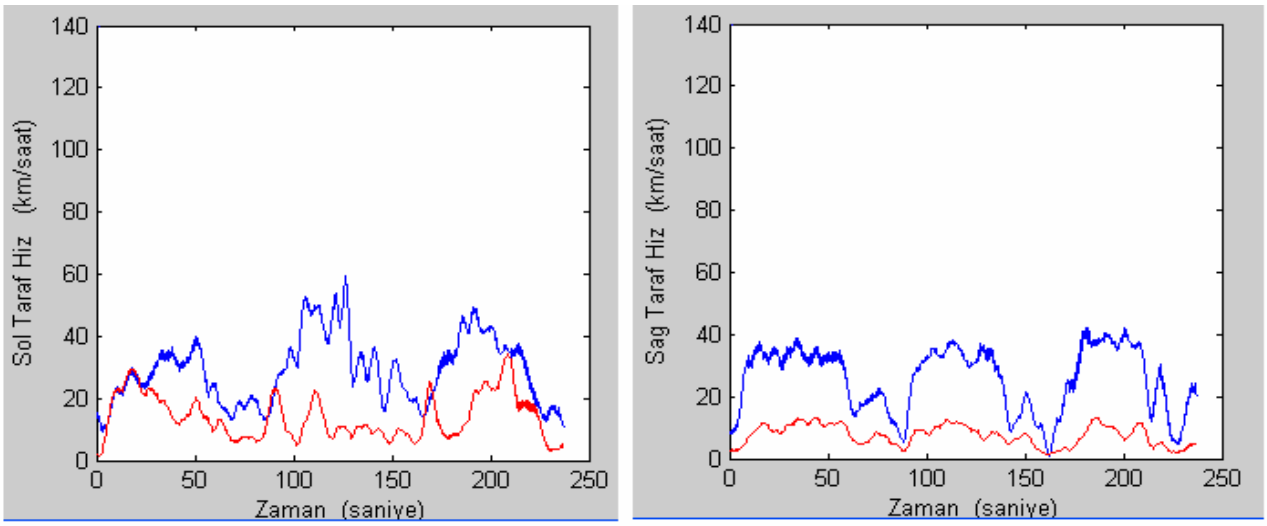
Bir diğer örneğimiz de Fındıklı kamerasından 240 saniyelik kayıttır. Bu kameranın

parametreleri h=23metre, t=61metre, k=55metre olarak bulunmuştur. Bu parametrelerle resmi düzelttiğimizde yukarıdaki görüntü karşımıza çıkmaktadır.



Şekil 8.8 Fındıklı kamerasından algılanan yoğunluk grafiği

Görüntünün başlangıç anında yolun sol tarafı yüksek oranda dolu ve tıkalı durumdadır. Bu yüzden arka plan çıkarımı yolun solu için ancak 33 saniyede gerçekleşebilmiştir. Dolayısıyla sol taraf için ancak 33. saniyeden sonraki yoğunluk değerleri doğrudur. Yolun sağı ise nispeten az yoğun olduğundan 9. saniyede arka plan çıkmıştır. Yoğunluk grafiğine baktığımızda 55. saniyeden sonra sol taraf yoğunluğunda bir artış görülmektedir. Gerçekten de yolun sol kısmı 55. saniyeden 95. saniyeye kadar çok yavaş ilerlemiştir.



Şekil 8.9 Fındıklı kamerasında algılanan hız ve hareket eden blok sayısı grafiği

Aynı doğruluk oranı hız için de söylenebilmektedir. Sol taraf için 55-90. saniye arasında hızın

azaldığı görülmektedir. Yolun sağı için hız grafiğine bakacak olursak belli aralıklarda hızda bir azalım söz konusudur. İşte o zamanlar trafik ışıklarından kaynaklanan durmalar olduğu videoyu seyredince hemen göze çarpmaktadır. Ortalama hızlara bakıldığında sol taraf 29 km/saat, sağ taraf ise 26 km/saat çıkmaktadır. Trafiğin yer yer durduğu ve yolların oldukça kalabalık olduğu bir durumda bu hızlar kabul edilebilir makul değerlerdir. Trafiğin akışkan olduğu zamanlar için ise grafiğe göre hız yaklaşık 40 km/saat çıkmıştır. Araçlar gözle takip edildiğinde bu rakamın çok makul olduğu görülmektedir.

8.2 İdeal Olmayan Durumda Sistemin Davranışı

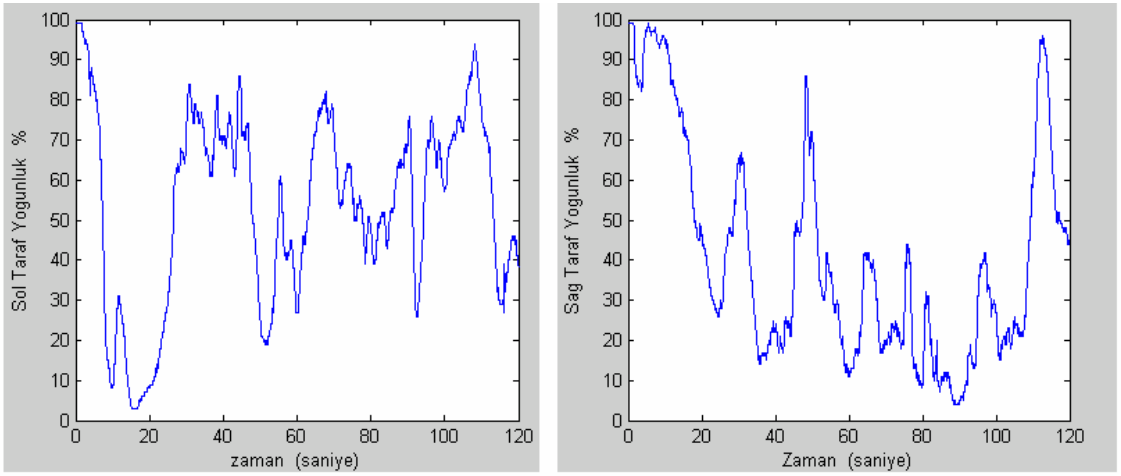
İdeal olmayan durumlar gece, araçların gölgesinin fazla olduğu, güneş ışıklarının çok değiştiği bulutlu günler olarak adlandırılabilir.

Şekil 8.10 Beşiktaş kamerasından gece alınan görüntü

Beşiktaş kamerasından gece alınan 120 saniyelik kayıt analiz edildiğinde yoğunluk grafiği Şekil 8.11'deki gibi çıkmaktadır. Grafiğe göre yolun solu oldukça yoğundur. Halbuki böyle bir yoğunluk yoktur. Tek bir araç dahi geçtiğinde farının bütün yolları aydınlatması sonucu arka planla eşleştirilmediği için ön plan olarak algılanmakta ve yoğunluğa dahil edilmektedir. Yolun sağında araçların farları kameraya değil de diğer yöne baktığı için yolun ışık durumu farlardan daha az etkilenmiştir. Bu durum yoğunluk grafiğinde de göze çarpmaktadır. Çünkü grafikte yolun sağının yoğunluğu sola göre biraz daha makul gözükmektedir.

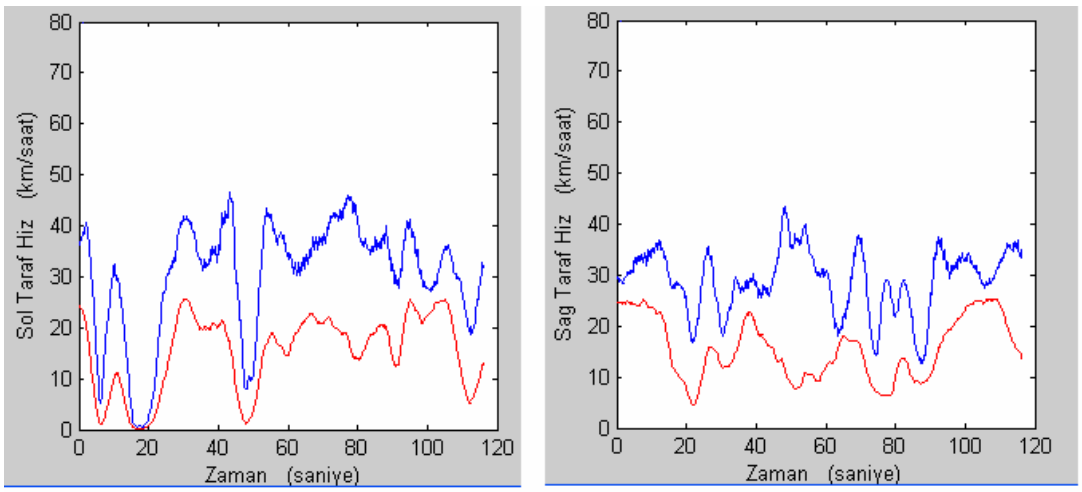
Hız grafiği ise oldukça doğrudur. Çünkü araç hızları hareket eden makro blokların hareket miktarlarına göre hesaplanıyor. Bu durumda farların aydınlattığı yollar da araçla aynı hızla gideceği için bu bir problem teşkil etmez. Yalnızca hareket eden makro blokların sayısında bir

artış meydana getirir.



Şekil 8.11 Gece çekilen görüntülerden algılanan yoğunluk grafiği

Şekil 8.12'de görüldüğü üzere sol tarafta ilk saniyelerde bir hız görülmekte, ardından bu hız sıfırlanmaktadır. Sonra yine bir artış ve azalış takip etmektedir. Video izlendiğinde bu durum aynen olduğu gözükmektedir. Yine 50. saniyelerde hızda bir düşüş yaşanmaktadır. Bunun sebebi yine o zamanda yolun boş kalmasıdır. Hareket eden makro blokların sayısında yaşanan azalma zaten yolun solu için 50. saniyelerde göze çarpmaktadır. Yapılan ölçümlerde hızların kabul edilebilir doğrulukta olduğu anlaşılmaktadır.



Şekil 8.12 Gece çekilen görüntülerden algılanan hız ve hareket eden blok sayısı grafiği

8.3 Genel Değerlendirme

Geliştirilen sistemin performansına bakılırsa trafik planlaması için gereken en önemli

parametrelerden olan hız ve yoğunluk oldukça efektif bir metotla bulunabilmiştir. Özellikle hız bulma algoritması her türlü çevre şartlarında çalışabilmesi açısından oldukça verimlidir. Kamera parametrelerini doğru bildiğimiz takdirde yüksek doğrulukta hız çıkarabilmektedir. Trafik Kontrol Merkezi'nden temin edilen 10 değişik kameraya ait toplam 200 dakikalık kayıt üzerinden denemeler yapılmış ve hız çıkarımının her türlü çevre şartlarında %10 hatanın altında çalıştığı belirlenmiştir. Yoğunluk ile ilgili olarak üzerinde çalışılması gereken konu, gece için araçların far etkisini azaltacak ya da yok edecek bir metot bulmaktır. Ayrıca arka plana ait gün içindeki ışık değişiklikleriyle değişmeyecek bazı özelliklerin çıkarılması gerekmektedir. Bu problem ışıkla değişmeyen özelliklerden biri olan kenar bulmaya dayalı bir arka plan çıkarım algoritması geliştirilebilir.

Bulunan parametrelerle trafik planlaması yapılabileceği gibi, bu parametreler bir uzman sistem yardımıyla gündelik hayatta kullandığımız boş, akıcı, tıkalı gibi ifadelere rahatlıkla çevrilip yol durumunu merak eden insanlara duyurulabilir. Bu tür modelleme bulanık mantık yardımıyla yapılabilir.

KAYNAKLAR

Azcan H. "Uzayın Analitik Geometrisi", Anadolu Üniversitesi Yayınları,Yayın No:1077, Eskişehir, 1999

Bevilacqua A. "Effective Object Segmentation in a Traffic Monitoring Application", ICVGIP 2002 conf. proc.,Ahmedabad, India, sayfa.125-130, 2002

Beymer D., McLauchlan P., Coifman B., ve Malik J., "A Real-time computer vision system for Measuring Traffic Parameters", CVPR, sayfa 495-501, 1997

Cucchiara R., Piccardi M., ve Mello M., "Image Analysis and Rule-Based Reasoning for a Traffic Monitoring System", IEEE Trans. On Intelligent Transportation Systems, vol. 1, no 2, sayfa 119-130, 2000

Drake M., Hoffmann H., Rabbah R., "MPEG-2 in a Stream Programming Language", Computer Science and Artificial Intelligence Laboratory Technical Report,, MIT, cambridge, October 22, 2005

Forsyth D., Ponce J, "Computer vision : a modern approach", Prentice Hall, London , 2003

Gürbüz Z.,"Traffic Flow Speed Detection",Yıldız Teknik Ünv. Bilgisayar Müh. Bitirme Projesi, İstanbul,2006

ISO-11712: Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to 1.5 Mbits/sec (MPEG1).

Jung Y.K., Lee K. W., ve Ho Y. S., "Content-Based Event Retrieval Using Semantic Scene Interpretation for Automated Traffic Surveillance",IEEE Trans. on Intelligent Transportation Systems, vol. 2, no 3, sayfa 151-163, 2001

Kamijo S., Matsushita Y., Ikeuchi K., Sakauchi M., "Traffic Monitoring and Accident Detection at Intersections", IEEE Trans. On Intelligent Transportation Systems, vol. 1, no 2, sayfa 108-118, 2000

Low A. "Computer Vision and Image Processing",McGraw-Hill UK,England,1991

Masaki I., "Machine-vision System for Intelligent Transportation: The Autoscope system", IEEE Trans. Vehicle Technology, vol. 40, sayfa 21-29, 1991

Maurin B., Masoud O., ve Papanikolopoulos N., "Monitoring Crowded Traffic Scenes", IEEE Int. Conference on Intelligent Transportation Systems, sayfa 19-24, 2002

Maurizio P., **"**On Using Raw MPEG Motion Vectors, To Determine Global Camera Motion", Digital Media Department HP Laboratories Bristol August, 1997

Piccardi M." Background subtraction techniques:a review", Computer Vision Research Group (CVRG) University of Technology, 2004,Sydney

Porikli, F.; Li, X**. "**Traffic Congestion Estimation Using HMM Models Without Vehicle Tracking", Mitsubishi Electric Research Laboratories, Inc., 2004, Massachusetts

Porikli F. "Real-Time Video Object Segmentation forMPEG Encoded Video Sequences", Mitsubishi Electric Research Laboratories, Inc., 2004, Massachusetts

Shuming T., Xiaoyan G., ve Feiyue W., "Traffic Incident Detection Algorithm Based on Non-parameter Regression", IEEE Int. Conference on Intelligent Transportation Systems, sayfa

714-719, 2002

Stauffer C, Grimson W, “Adaptive background mixture models for real-time tracking” Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 1999, sayfa. 246-252.

Sullivan G., ”Model-based Vision for Traffic Scenes using the Ground-plane Constraint”, Real-time Computer Vision, D. Terzopoulos and C. Brown, Cambridge University Press, 1994

Wang R., Zhang H.J., and Zhang Y.Q., ”A Confidence Measure Based Moving Object Extraction System for Compressed Domain”,IEEE International Symposium on Circuits and Systems, Switzerland, 2000

Yu X.D., Duan L.Y., Tian Q., ”Highway Traffic Information Extraction from Skycam MPEG Video”, IEEE Int. Conference on Intelligent Transportation Systems, sayfa 37-41, 2002

Zelnik L,Manor L. ve Irani M., ”Event-Based Analysis of Video”,CVPR, vol. 2, sayfa 123-130, 2001

EKLER

.

Ek 1 Ekran Düzleminden Yol Düzemline Dönüşümün Matlab Kodu

```
function [donusum,pikseluz] = ekran2yol(h,t,k)
% pikseluz 1 pikselin cm cinsinden uzunluğunu döndürür.
% donusum matrisi sözkonusu indisteki pikselin yol düzleminde düştüğü
        koordinatı saklar
% h kameranın yerden yüksekliği metre cins
% t kameranın direk dibinden görünür en alt noktanın uzaklığı metre
        cinsinden
% k görünüren alt noktadan görüntü merkezinin uzaklığı metre cinsinden

dikey=240;    %dikey çözünürlük piksel
yatay=320;    %yatay çözünürlük piksel
f.x=0;f.y=h;f.z=0;
e.x=0;e.y=0;e.z=0;
g.x=0;g.y=0;g.z=t+k;
l.x=0;l.y=0;l.z=t;
alfa=atan((t+k)/h);
beta=atan(t/h);
gama=alfa-beta;
teta=pi/2-alfa-gama;
n.x=0;
n.y=h-sqrt(h*h+t*t)*sin(teta);
n.z=sqrt(h*h+t*t)*cos(teta);
m.x=0;  %(n.x+l.x)/2;
m.y=(n.y+l.y)/2;
m.z=(n.z+l.z)/2;
kenar=sqrt(power((n.x-l.x),2)+power((n.y-l.y),2)+power((n.z-l.z),2));
pikseluz=kenar*100/dikey;
ekranaci=alfa;
for a=1:dikey
    for b=1:yatay
         piksel.y=(dikey+1-a)*kenar/dikey;
         piksel.x=(yatay/2+1-b)*kenar/dikey;
         ekran.x=piksel.x;
         ekran.y=piksel.y*sin(ekranaci);
         ekran.z=t+piksel.y*cos(ekranaci);
         duzlem.x=-h*(ekran.x)/(ekran.y-h);
         duzlem.y=0;
         duzlem.z=-h*(ekran.z)/(ekran.y-h);
         donusum(a,b,1)=fix(((duzlem.x*dikey/kenar)));
         donusum(a,b,2)=fix((1+(duzlem.z-t+1)*dikey/kenar));
    end
end
```

Ek-2 Yol Düzleminden Ekran Düzlemine Dönüşümün Matlab Kodu

```
function [tablom,pikseluz]=terstablo(h,t,k)
% resmin sol üst köşesi(1,1) elemanı olmak üzere
% a satır b sutun numarasını olmalı
% d satır c stun değerlirini döndürür.
% pikseluz 1 pikselin cmcinsinden uzunluunu döndürür.
%kameranın yerden yüksekliği metre cins
%kameranın direk dibinden görünür en alt noktanın uzaklığı metre
%görünüren alt noktadan görüntü merkezinin uzaklığı metre
dikey=240;    %dikey çözünürlük piksel
yatay=320;    %yatay çözünürlük piksel

f.x=0;f.y=h;f.z=0;
e.x=0;e.y=0;e.z=0;
g.x=0;g.y=0;g.z=t+k;
l.x=0;l.y=0;l.z=t;
alfa=atan((t+k)/h);
beta=atan(t/h);
gama=alfa-beta;
teta=pi/2-alfa-gama;
n.x=0;
n.y=h-sqrt(h*h+t*t)*sin(teta);
n.z=sqrt(h*h+t*t)*cos(teta);
m.x=0;  %(n.x+l.x)/2;
m.y=(n.y+l.y)/2;
m.z=(n.z+l.z)/2;
kenar=sqrt(power((n.x-l.x),2)+power((n.y-l.y),2)+power((n.z-l.z),2));
pikseluz=kenar*100/dikey;
ekranaci=alfa;
for c=fix(t)+1:800
    for a=-500:500;
       cc=c/10;
       aa=a/10;
       o.y=(h*cc-h*t)/(cc+h/atan(alfa));
       o.z=t+o.y/atan(alfa);
       o.x=aa*(o.y-h)/(-h);
       aa=a+501;
       tablom(c,a+501,1)=160+o.x*dikey/kenar;
       tablom(c,a+501,2)=240-sqrt(power((o.z-
t),2)+power(o.y,2))*dikey/kenar;%*kenar/dikey;
    end
end
```

Ek-3 Ekran Üstündeki İki Noktanın Gerçek Düzlemdeki Karşılıkları Arasındaki Mesafenin Bulunması

```
function [sonuc] = mesafe(h,t,k,ii1,jj1,ii2,jj2);
i1=240-ii1;     % h t k kameralarla ilgili parametreler
i2=240-ii2;     % ekran üstündeki ilk koordinat (ii1,jj1)
j1=jj1-160;     % erkan üstündeki ikinci koordinat (ii2,jj2)
j2=jj2-160;     % sonuc gerçek düzlemde iki nokta arasındaki mesafe
alfa=atan((t+k)/h);
pk=P(h,t,k);
x1=(h*j1*pk)/(h-i1*pk*sin(alfa));
x2=(h*j2*pk)/(h-i2*pk*sin(alfa));
x3=x1-x2;
y1=h*(t+pk*i1*cos(alfa))/(h-i1*pk*sin(alfa));
y2=h*(t+pk*i2*cos(alfa))/(h-i2*pk*sin(alfa));
y3=y1-y2;
sonuc=sqrt(x3*x3+y3*y3);



function [sonuc]=P(h,t,k);
alfa=atan((t+k)/h);
beta=atan(t/h);
gama=alfa-beta;
teta=pi/2-alfa-gama;
sonuc=sqrt(power((h-
sqrt(h*h+t*t)*sin(teta)),2)+power((sqrt(h*h+t*t)*cos(teta)-t),2))/240;
```

Ek-4 GKM ile Arka Plan Çıkarımı Matlab Kodu

```
% gaussian mixture
ogrenme=0.01;
pikselmu     =fix(rand(240,320,5)*255);
pikselsigma  =rand(240,320,5)*5+2;
pikselagirlik=rand(240,320,5);
arkaplan=zeros(240,320);
sol=rgb2gray(imread('besiktas_sol.bmp'));
sag=rgb2gray(imread('besiktas_sag.bmp'));
video=aviread('Besiktas.avi',1:250);
for sayac=1:1:250
  rsm=double(rgb2gray(video(sayac).cdata));
  resim=rsm(1:240,1:320);
  for i=1:240
    for j=1:320
      if (sag(i,j)==255 || sol(i,j)==255)
        carpan=2.5;
        yer=6;
        for ii=1:5
          ss=abs(double(resim(i,j))-pikselmu(i,j,ii));
          if carpan>ss/pikselsigma(i,j,ii)
            carpan=ss/pikselsigma(i,j,ii);
            yer=ii;
          end
        end
        if yer<6
          %agirliklar ayarlanıyor.
          for ii=1:5
            pikselagirlik(i,j,ii)=(1-ogrenme)*pikselagirlik(i,j,ii);
            end
            pikselagirlik(i,j,yer)=pikselagirlik(i,j,yer)+ogrenme;
            tplm=sum(pikselagirlik(i,j,:));
            pikselagirlik(i,j,:)=pikselagirlik(i,j,:)/tplm;
            % mu değeri ayarlanıyor

ss=normpdf(double(resim(i,j)),pikselmu(i,j,yer),pikselsigma(i,j,yer));
            if ss>1
               ss=1;
            end
            pikselmu(i,j,yer)=(1-
ogrenme*ss)*pikselmu(i,j,yer)+ogrenme*ss*double(resim(i,j));
            pikselsigma(i,j,yer)=(1-
ogrenme*ss)*pikselsigma(i,j,yer)+ogrenme*ss*power((resim(i,j)-
pikselmu(i,j,yer)),2);

            else
              yerr=6;kucuk=1;
              for ii=1:5
               if kucuk>pikselagirlik(i,j,ii)
                  kucuk=pikselagirlik(i,j,ii);
                  yerr=ii;
               end
              end
              pikselmu(i,j,yerr)=resim(i,j);
              kucuk=255;
              for ii=1:5
                if kucuk>pikselsigma(i,j,ii)
                   kucuk=pikselsigma(i,j,ii);
                end
              end
              pikselsigma(i,j,yerr)=kucuk;
```

```
              for i=1:5
pikselagirlik(i,j,ii)=pikselagirlik(i,j,ii)+0.25*abs(pikselagirlik(i,j,yerr
)-0.05);
              end
              pikselagirlik(i,j,yerr)=0.05;
        end
      end
    end
  end
end

kucuk=0;yer=6;
for i=1:240
    for j=1:320
        for ii=1:5
          if kucuk<pikselagirlik(i,j,ii)
              kucuk=pikselagirlik(i,j,ii);
              yer=ii;
          end
        end
        arkaplan(i,j)=pikselmu(i,j,yer);
        kucuk=0;
    end
end
colormap(gray);imagesc(arka);
```

ÖZGEÇMİŞ

Doğum tarihi	19.10.1979	
Doğum yeri	Rize	
Lise	1990-1997	Rize Anadolu Lisesi
Lisans	1999-2005	Yıldız Teknik Üniversitesi Elektrik-Elektronik Fak. Bilgisayar Mühendisliği Bölümü
Yüksek Lisans	2005-	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı.

Çalıştığı kurum

2004-2005	IDD Yazılım A.Ş.
2005 -	YTÜ Fen Bilimleri Enstitüsü Araştırma Görevlisi