VİDEO GÖRÜNTÜLERİ İÇİNDEN HAREKETLİ NESNE AYIKLANMASI VE İZLENMESİ

Banu GÖKTAŞ DİLEK

Yüksek Lisasn Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç.Dr. Ahmet ÇINAR

MAYIS-2012

T.C.

FIRAT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

VİDEO GÖRÜNTÜLERİ İÇİNDEN HAREKETLİ NESNE AYIKLANMASI VE İZLENMESİ

YÜKSEK LİSANS TEZİ

Banu GÖKTAŞ DİLEK

Ana Bilim Dalı: Bilgisayar Mühendisliği

Programı: Yazılım

Tez Danışmanı: Yrd. Doç.Dr. Ahmet ÇINAR

MAYIS, 2012

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

VİDEO GÖRÜNTÜLERİ İÇİNDEN HAREKETLİ NESNE AYIKLANMASI VE İZLENMESİ

YÜKSEK LİSANS TEZİ
Banu GÖKTAŞ DİLEK
06229103

Ana Bilim Dalı: Bilgisayar Mühendisliği
Programı: Yazılım

Tez Danışmanı: Yrd. Doç.Dr. Ahmet ÇINAR

Tezin Enstitüye Verildiği Tarih: 22.05.2012

MAYIS, 2012

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

VİDEO GÖRÜNTÜLERİ İÇİNDEN HAREKETLİ NESNE AYIKLANMASI VE İZLENMESİ

Yüksek Lisans Tezi
Banu GÖKTAŞ DİLEK
06229103

Tezin Enstitüye Verildiği Tarih : 22 Mayıs 2012
Tezin Savunulduğu Tarih : 22 Mayıs 2012

Tez Danışmanı : Yrd. Doç. Dr. Ahmet ÇINAR(F.Ü)
Diğer Jüri Üyeleri : Doç.Dr. Servet TUNCER(F.Ü)
Yrd. Doç. Dr. Galip AYDIN(F.Ü)

MAYIS, 2012

ÖNSÖZ

Sayın Yrd. Doç. Dr. Ahmet ÇINAR'a, tez çalışmasının gerçekleştirilmesinde gerekli yönlendirici desteği sağladığı için teşekkür ederim. Ayrıca tez çalışmam boyunca beni yürekten destekleyen anneme, babama ve sevgili eşime teşekkür ederim.

Banu GÖKTAŞ DİLEK

ELAZIĞ - 2012

İÇİNDEKİLER

Sayfa No

ÖZET

Bilgisayarla görmenin önemli konularından biri olan nesne takibi, video kaydı içerisinde hareket etmekte olan nesnelerin hareket yörüngelerini tahmin etme problemidir. Nesne takibi, başta güvenlik sistemleri olmak üzere, tıpta hastalıklı hücre izlenmesi ve tanınması, askeriyede insansız araç yapılması ve akıllı silah yapılması gibi daha birçok alanda kullanılmaktadır.

Literatürde, nesne takibi ile ilgili birçok yöntem bulunmaktadır. Tez çalışmasında nesne takibi için iki farklı algoritma kullanılmıştır. Birincisi, nesnenin istatistiksel renk dağılımını kullanan parametrik olmayan bir yöntem olan ortalama kayma algoritması; ikincisi ise optik akış tahmini olarak da adlandırılan hız tahmininde iki görüntü dizisi kullanan farksal tekniklerinden biri olarak kullanılan Horn ve Schunck algoritmasıdır.

Çalışmada nesne takibinde karşılaşılan problemlere çözüm için önerilen yöntemler sınıflandırılmıştır. Kullanılan algoritmalar hakkında detaylı bilgi verilmiştir. Matlab aracı kullanılarak, iki farklı algoritma ile nesne takibi yapılmıştır. Elde edilen veriler analiz edilerek karşılaştırılmıştır. Karşılaşılan problemlere çözüm önerisinde bulunulmuştur.

Anahtar Kelimeler: Nesne Takibi, Ortalama Değer Kayması, Optik Akış, Video İşleme, Matlab ile Görüntü Analizi.

SUMMARY

Object tracking, which is one of the subjects of computer vision, is the problem of predicting the trajectories of moving objects in a video recording. Object tracking is used in many areas such as security systems for surveillance, medicine for tracking and recognition of the diseased cells, and in the military appllications for the construction of unmanned vehicles and smart weapons.

In the literature, several methods for object tracking are discussed. In this thesis two different algorithms for object tracking are used. The first one is the mean shift algorithm, which is a non-parametric method that uses the color distribution statistics of an object. The second algorithm is a differential technique called Horn and Schunck algorithm, which is also called as optical flow algorithm that uses two image sequences for speed estimation.

In this study, the methods for the solution of the problems encountered in object tracking are classified. Detailed information about the algorithms used in this thesis is given. We have developed several Matlab programs for object tracking using the aforementioned algorithms. The data obtained from these programs are compared and analyzed. And the results of these analyis are used to suggest solutions for the problems encountered during this study.

Key Words: Object Tracking, Mean Shift, Optical Flow, Video Processing, Image Analysis with MATLAB

ŞEKİLLER LİSTESİ

Sayfa No

KISALTMALAR

AR-GE	: Araştırma ve Geliştirme
AMISE	: Asymptotic Mean Integrated Squared Error
YDT	: Yüksek dereceden terimler
MATLAB	: MATrix LABoratory
IPT	: Image Processing Toolbox
RGB	: Red, Green, Blue
Fps	: Frame Per Second

SEMBOLLER LİSTESİ

A_i : Sesin i aralığında şiddetidir

$F(x),\ G(x)$: Ardışık görüntüler

f^k : Anahtar çerçeve

$H_i(k)$: Film çerçevesi

I : Görüntü

$Ix,\ Iy,\ It$: Pikselin x,y yönünde t anındaki kısmi türevleri

K : Kalman kazancı

$K(\bullet)$: Çekirdek fonksiyonu

Ki : Film çerçevesi dizisi

M : Ölçme matrisi

$p_k,\ q_k$: Renk histogramları

$P(FD)$: İki çerçeve arası farkın standart sapmasıdır

$\sin(x)$: Sinüs fonksiyonu

$S(i)$: İki film çerçevesi farkı

$s_t^{(n)}$: Şartsal durum yoğunluğu

Th : Eşik değeri

$U,\ V$: Videodan elde edilmiş yatay ve dikey hızlar

$W_t^{(n)}$: Gaussian hatası

$W^t,\ N^t$: Beyaz gürültü

Q^t : W gürültüsünün kovaryansı

X^t : Nesnenin durum bilgisi dizisi

$\overline{X^t}$, $\overline{\Sigma^t}$	: t anındaki durum ve kovaryans tahminleri
V_x, V_y	: Pikselin x,y yönünde hız vektörleri
$\vec{v}$	:Hız vektörü
$\pi_t^{(n)}$	: Örnekleme olasılığı
π	: Pi sayısı
μ, σ	: $K(\bullet)$ çekirdek fonksiyonunun merkez değeri ve bant genişliği
$\approx$	: Yaklaşıklık
$\frac{d}{dh}$	: Türev
$\sum$	: Toplam sembolü
Ω	: Görüntü üzerinde ilgilenilen alan
∇	: Gradyan operatörü

1. GİRİŞ

Günümüzde işletmelerin yoğun teknoloji kullanımlarının artmasıyla birlikte multimedya araçlarına olan talep miktarı da artmıştır. Video, tüm medyayı içerisinde bulunduran bir multimedya verisidir. Videolar; ses, görüntü ve metin verilerine sahiptir. Video kamera veya tarayıcı ile elde edilen görüntü sayılaştırılarak analiz edilmesi ve sonucunda görüntüden bilgi elde edilmesi ve yorumlanması görüntü işleme ve bilgisayarlı görü alanlarının temel amacıdır. Birçok AR-GE(Araştırma ve geliştirme) çalışmasında, kapalı devre televizyon sistemi kullanan güvenlik uygulamalarında, tıbbi ve uydusal görüntülerde bu video parçacıkları analiz edilir. İstenen verilere ulaşmak için analiz edilecek veri sahip olduğu özelliklere göre (ses şiddeti, çerçeve içerik bilgisi veya renk yoğunluğu vb.) analiz işlemleri uygulanır.

Kamera kullanılarak elde edilen ardışık video çerçevelerindeki hareketli bir nesnenin zaman boyunca, çerçeve içerisindeki konum bilgisinin belirlenmesine nesne takip işlemi denir. Video kaydı üzerinde gerçekleştirilen bu basit takip işlemi zamanla çok geniş bir kullanım alanına sahip oldu. Otomatik olarak hastalıklı hücre tespiti, insansız araç kullanımı, kapalı devre araç veya insan izleme güvenlik sistemleri, askeri uygulamalar kapsamında hareketli bir hedefin takip edilerek imha edilmesi, akıllı silahların geliştirilmesi nesne izleme uygulamalarından yalnızca birkaç tanesidir.

Ortalama değer kayma algoritması ile gerçekleştirilen nesne takibi yöntemi, gerçek zamanlı çalışabilen, olasılığa dayalı hızlı bir takip yöntemidir. Yöntem, takip edilecek nesnenin ve nesnenin bulunduğu ortamın renk dağılımına göre çalışmaktadır.

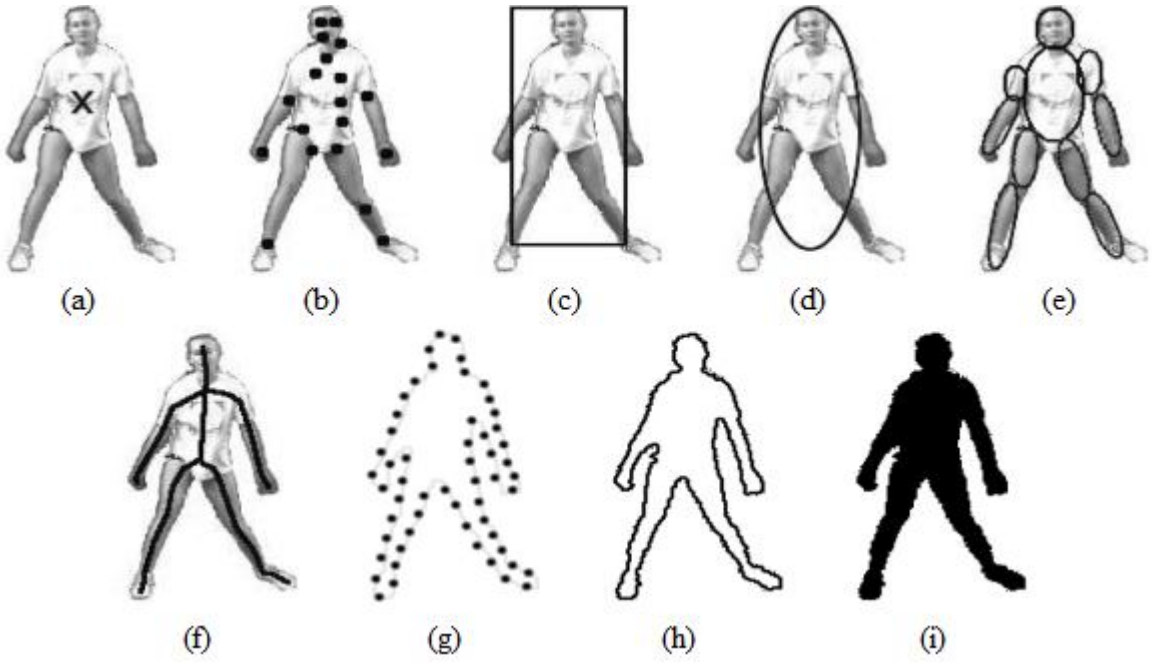
Nesne takibi için kullanılan diğer yöntem ise Optik Akış yöntemidir. Optik Akış yöntemi, hareket kestirimi, nesne takibi, video stabilizasyonu (titreşim, hareket giderme), imge mozaikleme, stereo vision ile derinlik bilgisi eldesi gibi operasyonlarda sıklıkla kullanılan temel bir araçtır.

Tez çalışmasında, gerçek zamanda çalışan bir kameradan elde edilen görüntülerde hareketli nesneleri seçerek takip edebilecek bir program geliştirilmiştir. Geliştirilen program, ortalama değer kayması ve optik akış metoduna dayalıdır ve Matlab aracı ile analiz uygulanmıştır.

1.1. Nesne Takibi

Nesne takibi, video kaydı içerisinde hareket etmekte olan bir nesnenin hareket yörüngesini tahmin etme problemidir. Diğer bir deyişle, bir nesne takip edicinin, ardışık video görüntüleri içerisinde hareket etmekte olan nesneleri algılayarak her bir nesneye eşsiz bir etiket verme işlemidir [1].

Nesne takip işlemlerinde önemli olan geçerli çerçevede nesnemizin konumunun belirlenmesi ve diğer çerçevedeki nesneler ile veri ilişkisinin bulunmasıdır. Takip edilen nesne şekil ve model bilgisiyle sunulur. Nesnenin şekli onun hareketini sınırlar. Örneğin; takip edilecek nesne sadece bir nokta ile ifade edilirse, o zaman nesne hareketlerini ifade etmek için basit bir dönüştürme modeli yeterli olabilir. Oysa nesnenin elips gibi geometrik bir şekle sahip olduğu bir durumda, parametrik hareket modelleri (projektif dönüşüm modelleri) kullanılabilir. Bu sunum şekilleri yardımıyla geometrik şekli ve sınırları net olan nesnelerin hareketleri modellenebilir. Sınırları net olmayan nesneler içinse siluet veya kenar bilgisi, daha tanımlayıcı bir sunum sağlayabilir. Ayrıca, bu tür nesnelerin hareketlerini modellemek için parametrik veya parametrik olmayan modeller kullanılabilir [2]. Şekil 1.1'de nesne sunum teknikleri gösterilmektedir.



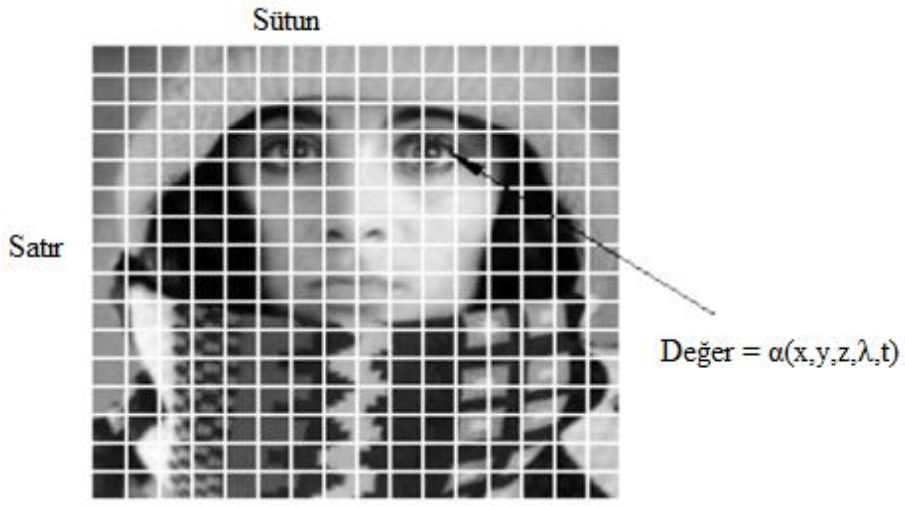
Şekil 1.1. Nesne sunum teknikleri

2. MATERYAL VE METOT

2.1. Görüntü İşleme

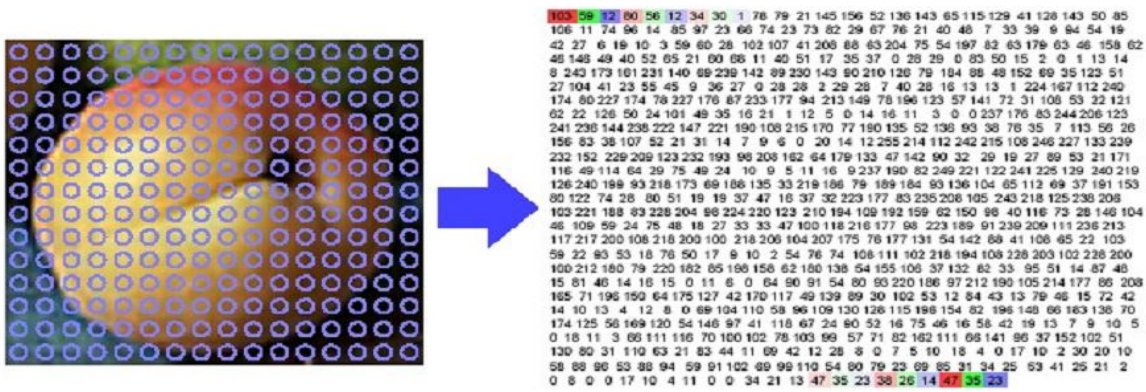
Görüntü, iki değişkenin bir fonksiyonu olarak tanımlanır. $A(x, y)$ gibi bir fonksiyonda x ve y koordinatları gösterir. A ise Görüntüye ait bir özellik fonksiyonudur. Görüntü bölgelerden oluşur. Görüntü işlenirken seçili bölgelere bazı operasyonlar yapılır. Örneğin görüntünün bir bölümü bulanıklaştırılırken, diğer bölümü parlaklaştırılır.

Sayısal görüntü 1 ve 0' lardan oluşur. $A(x, y)$ fonksiyonu örneklenerek 2 boyutlu a[m,n] matrisi elde edilir. M ve n satır ve sütunlarının kesiştiği her bölgeye piksel denir. Pikseldeki "z" değeri derinliği, "λ" değeri rengi ve "t" değeri zamanı gösteren bir fonksiyondur. Şekil 2.1'de sayısal bir görüntüde piksellerin aldığı değerler gösterilmektedir.



Şekil 2.1. Sayısal görüntü koordinatları

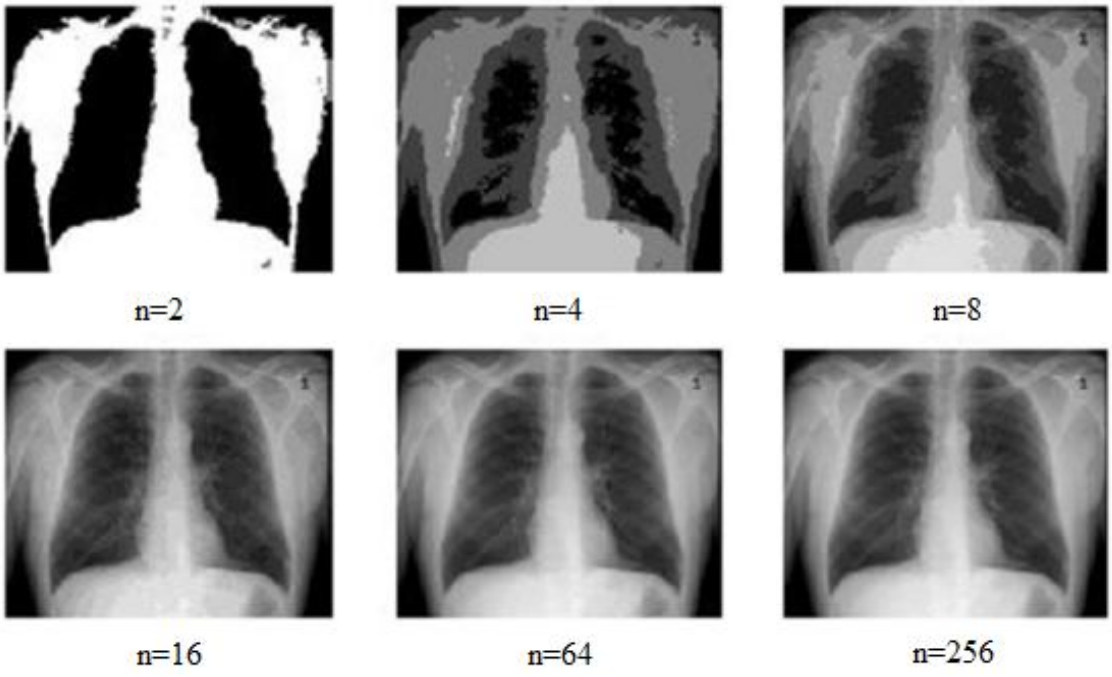
Görüntü 256 farklı renk yoğunluğundan oluşur. 0 beyazı, 255-n diğer renkleri temsil eder. Her piksel bir sayısal değere karşılık gelir. Şekil 2.2'de her bir pikselin almış olduğu renk kodları gösterilmektedir.



Şekil 2.2. Gerçek görüntünün sayısala dönüştürülmüş hali

2.1.1. Niteliklendirme

Görüntüyü gri formata çevirdiğimizde her pikseldeki 0 değeri beyazı, 1 değeri ise siyah rengi temsil eder. 0 ve 1 arasında 256 renk yoğunluğu bulunmaktadır. Görüntü analiz edilirken, çözünürlüğü $n=2^b$ ile formüle edilerek görüntülenir. b' nin sayısal değeri arttıkça gri tonlama artar ve düzgün görüntü elde edilir. Oluşan görüntü binary formattadır. Şekil 2.3'de görüntünün farklı çözünürlükte gri tonlamalarıdır.

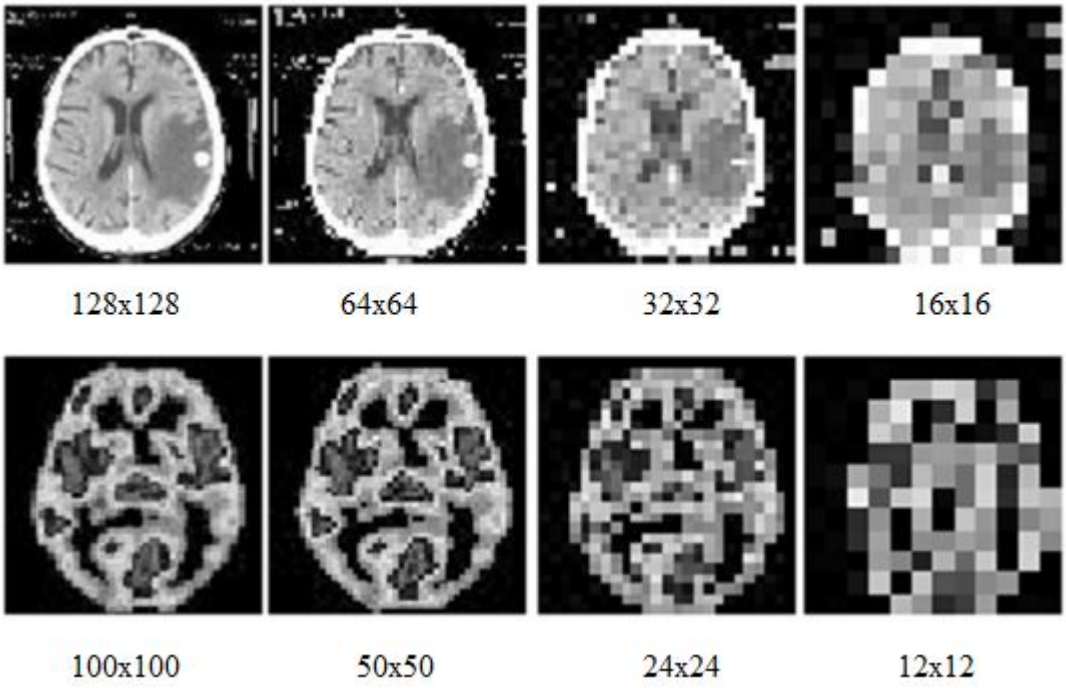


Şekil 2.3. Gri seviyede görüntü

2.1.2. Uzaysal Çözünürlük

Uzaysal çözünürlük, görüntüde seçilebilen en küçük nesnenin boyutudur[3]. Sayısal görüntüde çözünürlük, piksel boyutuyla sınırlıdır; yani bir nesnenin boyutu pikselden daha küçük olamaz. Görüntüleme sisteminin gerçek çözünürlüğü en başta sensörün ani görüş alanıdır. Bu, çok kısa zaman içinde ani olarak görüntülenen alanın büyüklüğüdür. Piksel boyutu örnekleme uzaklığı ile tayin edilir.

Görüntü pikselleri fiziksel olarak büyütülüp küçültülür. Detayı arttırmak için pikseller n*n kez arttırılır. Şekil 2.4'te piksellerin boyutunun arttırılması ile görüntü detaylandırılır.



Şekil 2.4. Görüntünün çözünürlüğünün arttırılması

2.1.3. Koyuluk Kontrast

Kontrast, imgedeki açık ve koyu renkler arasındaki farkların daha çok veya az olması şeklinde ifade edilebilir. Şekil 2.5'te farklı tonlardaki görüntünün farklı kontrast değerleri gösterilmektedir.

Şekil 2.5. Kontrast değiştirme

Her bir piksel pozitif bir değerle toplandığında parlaklık artar, negatif bir değerle toplandığında parlaklık azalır. Eğer kontrast arttırılmak istenirse pikselin sahip olduğu değer, pozitif bir değer ile çarpılır.

2.1.4. İndekslenmiş Görüntü

İndeksleme, her görüntüe belirli anahtar kelimeler atanması işlemidir. Sonuçta görüntüleme sisteminin (imaging system) kullanıcıları, ilgili görüntüye, atanan indeksler yardımıyla ulaşabileceklerdir. En önemli ve üzerinde en çok zaman harcanılan adımdır.

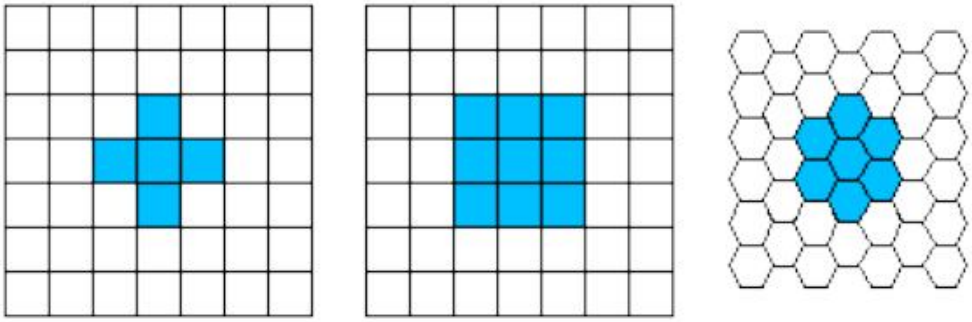
Görüntüler indekslenirken kullanılacak boyutların seçimi önemli bir yer tutmaktadır. Sınıflama yapılırken görüntü çözünürlüğü, görüntü boyutu, görüntünün yatay ya da dikey olması gibi parametreler kullanıldığında, bu parametreleri sağlayan çok sayıda kayıt olacağı için sağlıklı bir gruplandırma yapılamamaktadır. Bu tarz boyut seçiminin eksik kaldığı bir nokta da görüntünün, görüntü bakımından niteliklerini ihmal etmesidir. Bu yüzden görüntüler indekslenirken renk kodlarını kullanmak daha etkin bir yöntem olmaktadır. Herhangi bir görüntünün her bir noktasının renk değeri indeks boyutu olarak kullanılabilir. Bu da 0 ile 16.000.000 arasında farklı renk değeri alabilir.

2.1.5. Görüntü Operasyonları

Görüntü işlemede, işlemler ya noktadan noktaya, ya komşu çevreye ya da tüm görüntü elemanlarına göre uygulanır.

2.1.5.1. Komşuluk İlişkileri

Görüntü işlemede komşuluk ilişkileri çok önemlidir. Komşuluk ilişkileri ise ancak görüntü örnekleme yapılarak elde edilebilir. Temel örnekleme yöntemleri şekil 2.6'da sunulmaktadır. Dikdörtgensel örneklemede görüntünün üzerinde dikdörtgensel bir ızgara olduğu düşünülür. Altıgensel örneklemede ise görüntünün altıgenlerden oluşmuş parçalar içerdiği düşünülür. Dikdörtgensel örnekleme, donanımsal açıdan daha rahat gerçekleştirilebilindiği için daha çok tercih edilir.



Şekil 2.6. 4'lü, 8'li ve 6'lı komşuluk ilişkileri

Piksel temelli komşuluk işlemlerinde ise ilgili pikselin yeni değeri, komşu piksellerin değerleri de dikkate alınarak hesaplanır. Hangi komşuların nasıl kullanılacağı seçilen piksel komşuluk işlemine bağlı olarak belirlenir[4].

İmge üzerinde yapılacak uzamsal işlemlerde genellikle komşuluk ilişkileri dikkate alınır. Bunun nedeni ise tek başına fazla bir anlam ifade etmeyen bir pikselin, komşuları ile birlikte daha fazla bilgi içermesidir(örneğin; kenar bilgisi gibi).

2.1.5.2. Filitreleme

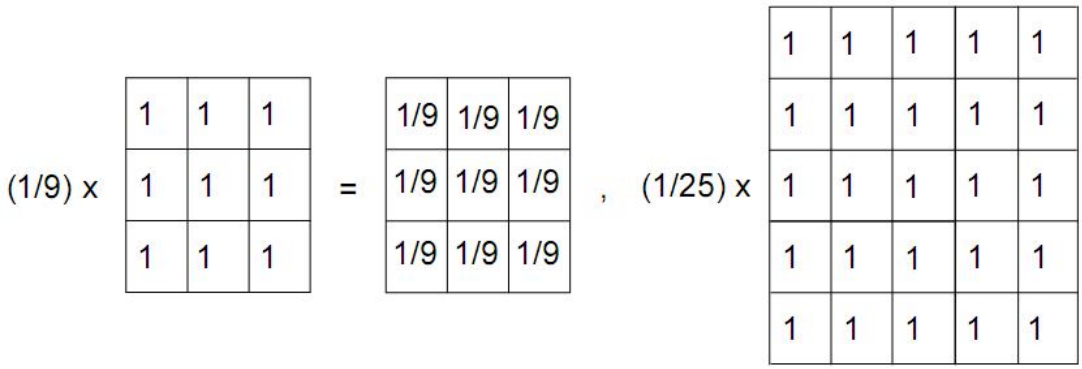
Filitreleme, görüntüde yer alan farklı fiziksel özellikler arasındaki ayrımı artırarak bir görüntünün görsel yorumlanabilirliğini artırmaktır. Bunu gerçekleştirmek için ise çeşitli sayısal filtreleme matrisleri kullanılır. Görüntüdeki farkların vurgulanması, kenar çizgilerinin vurgulanması ya da giderilmesi işlemleri için farklı sayı matrisleri kullanılmaktadır [5].

Sayısal filtreleme yönteminde her bir pikselin yeni gri renk tonları hesaplanmaktadır. Piksellerin yeni gri tonları yalnızca ortaya çıkarılacak detaya bağlı değil komşu piksellere

de bağlıdır. Uzaysal frekans filtreleme de denilen bu işlemde, bir görüntüde istenilen detayı ortaya çıkarabilmek için; yüksek, orta ve düşük frekanslı filtrelerden birisi kullanır [6].

Yüksek frekansları vurgulayan ve düşük frekansları bastıran filtrelere yüksek geçirgenli filtreler denir. Benzer olarak orta ve alçak geçirgenli filtreler de vardır.

Alçak geçirgenli filtreleri uygulamanın en basit yolu uzaysal komşuluk ortalaması ile yapılır. Örneğin şekil 2.7'de bir alçak geçirgenli filtre, orijinal görüntünün her pikselinin çevresindeki piksellerin ortalanması ve bu ortalamanın işlenmiş görüntüde piksel gri renk tonu düzeyi olarak kullanılması ile uygulanabilmektedir.



Şekil 2.7. Alçak geçiren filtre örneği

Basit bir yüksek geçirgenli filtre ise, orijinal görüntüden alçak geçirgenli filtre ile filtrelenmiş bir görüntünün çıkarılması ile ya da merkezdeki piksel için pozitif, etrafını çevreleyen pikseller için negatif ağırlıklara sahip bir nokta yayılım fonksiyonu kullanılarak döndürülmesi ile oluşturulabilir. Dönüşümde kullanılan kutu (Kernel), her bir pikseli etrafındaki piksel değerleri ile ortalamada kullanılan bir sayı matrisidir. Matristeki elemanlar, belirli pikseller yönünde bu ortalamayı ağırlıklandırmak için kullanılmaktadır.

Yüksek geçirgenli bir filtreleme örneği şekil 2.8'de görülmektedir. 3x3 boyutlu filtre kutusunun ortasına karşılık gelen yeni piksel değeri, kutudaki sayılara karşılık gelen her komşu pikselin bu sayılarla çarpılıp toplanması ve filtredeki sayıların toplamına bölünmesi ile elde edilir [7].

Yeni görüntüde orijinal görüntüdeki yüksek değerli pikseller daha yüksek; düşük değerlikli pikseller daha düşük olarak hesaplanmıştır.

3	6	6	7	7
3	8	5	5	7
3	3	8	5	7
2	3	3	8	6

Orjinal veri

-1	-1	-1
-1	16	-1
-1	-1	-1

Yüksek geçirgenli filtre (3x3)

3	6	6	7	7
3	8	5	5	7
3	3	8	5	7
2	3	3	8	6

Filtre sonrası veri değerleri

((-1*8)+(-1*5)+(-1*5)+(-1*3)+(16*8)+(-1*5)+(-1*3)+(-1*3)+(-1*8))/(-1+-1+-1+-1+16+-1+-1+-1+-1= (128-40)/(16-8)=int(88/8)=11

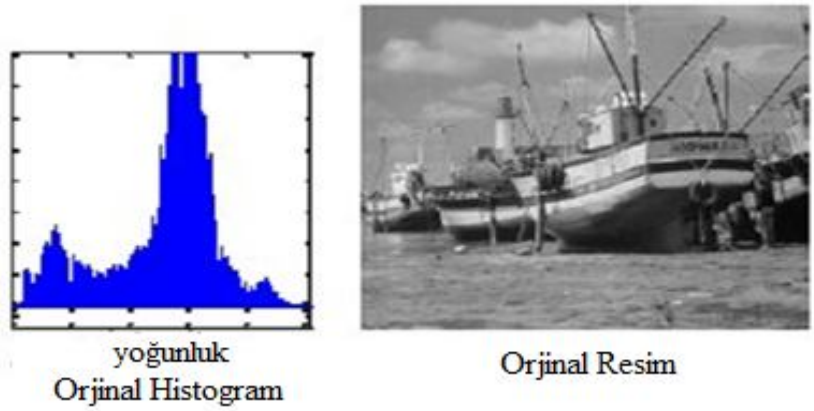
Şekil 2.8. Yüksek geçirgenli filtre örneği

Görüntüdeki sınırların belirginleştirilmesi için sıfır toplamlı doğrusal filtrelerde kullanılabilmektedir. Bu filtrelerde katsayılar toplamı sıfırdır. Filtre zenginleştirilecek detaya (sınır) bağlı olarak düşey ya da yatay doğrultuda geçirilir. Filtrenin hangi doğrultuda geçirileceği hangi doğrultulardaki cisimlerin zenginleştirileceğine bağlıdır [8]. Düşey doğrultu için tasarımlanmış 3x3 boyutlu kutu filtre algoritması aşağıda verilmiştir.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

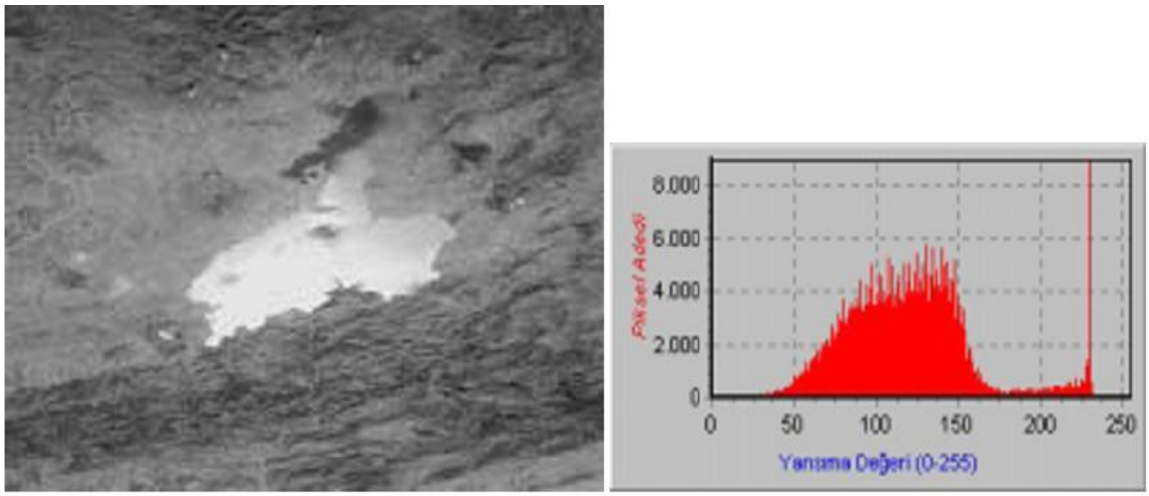
2.1.5.3. Histogram

Histogram, görüntüyü oluşturan bütün parlaklık değerlerini grafiksel olarak gösterir. Histogramda parlaklık değerleri (0-255) *x* ekseni boyunca, bulunma sıklığı (frekans) ise *y* ekseni boyunca gösterilir [9]. Şekil 2.9'da gri formatta bir görüntünün histogramı gösterilmektedir.

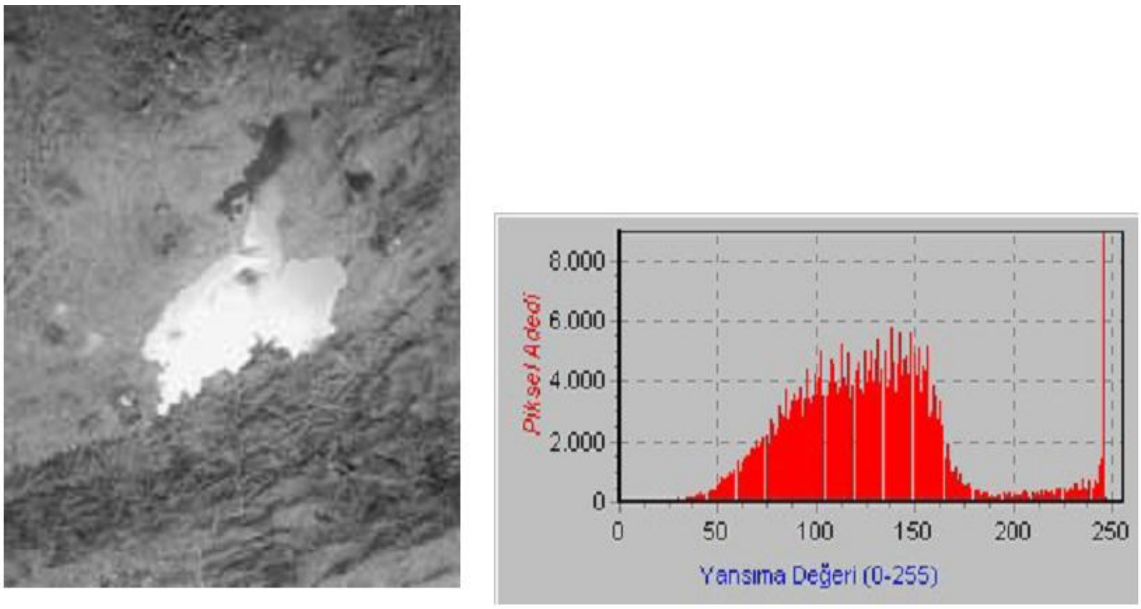


Şekil 2.9. Bir görüntünün histogramı

Histogramda grafiksel olarak gösterilen sayısal değerleri kullanarak, görüntüde çeşitli iyileştirmeler yapılabilir. Görüntüde kontrast ve detay iyileştirmenin farklı teknik ve metotları vardır. En basit iyileştirme metodu lineer kontrast gerilimidir(lineer contrast strecth). Bu yöntemde histogramdaki en alt ve en üst değerler belirlenir ve bütün aralıkları doldurmak için bu sıralar gerilir [10]. Örneğin histogramdaki minimum değer 84, maksimum değer 153 olsun. Yansıma değerleri 0-255 arasında olacak şekilde görüntünün bütün piksel değerleri değiştirilir. Böylece parlak tonlu alanlar daha parlak, koyu tonlu alanlar daha koyu olacak şekilde görüntünün kontrastı artırılmış olur ve görsel yorumlama kolaylaşır. Şekil 2.10 ve 11'de gösterildiği gibi görüntüyü daha parlak ve daha koyu yaptığımızda histogram iyileştirilebilir.



Şekil 2.10. Görüntü ve histogramı



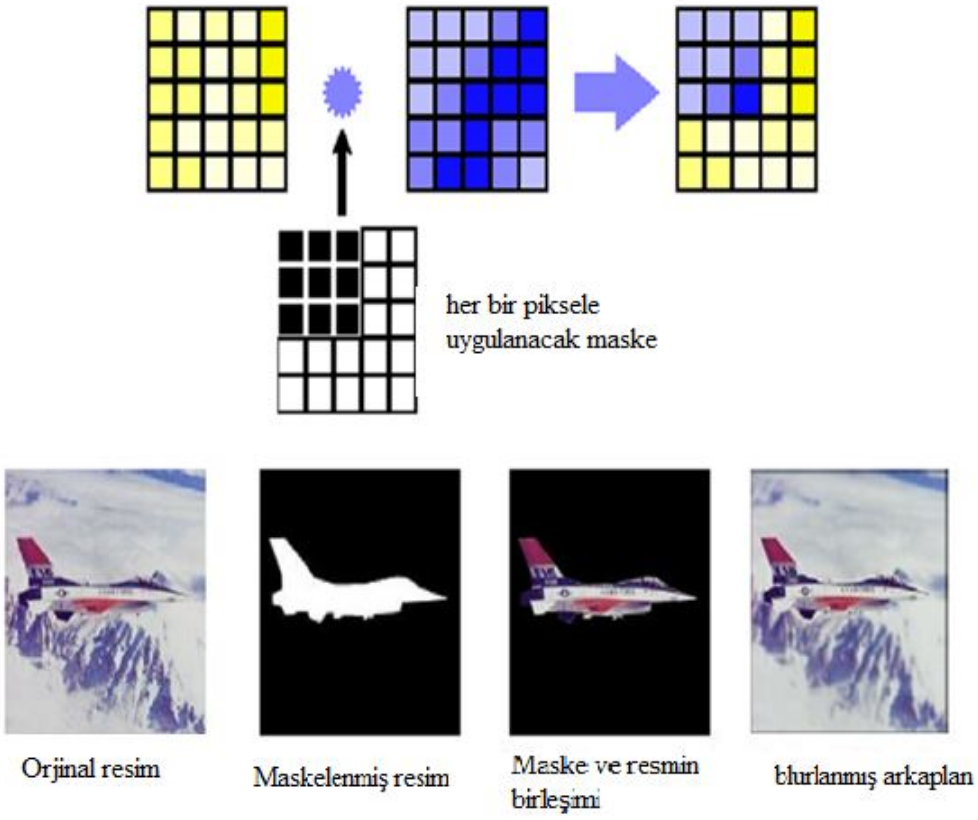
Şekil 2.11. Liner gerilmiş görüntü ve histogramı

Bu yöntem, her görüntüyü iyileştirmez. Yalnızca tüm pixellerin belli araklıkta renk değerine sahip olduğu görüntülerde etkilidir.

Görüntü gri ve tonlarından oluşuyorsa sorun yok, ama eğer renkli bir görüntü üzerine bu yöntem uygulanacaksa ve RGB değerleri ile işlem yapılıyorsa o zaman renkleri ayrıştırıp, kırmızı, mavi, yeşil renklerinin her biri için ayrı ayrı histogram dengeleme yapılmalıdır.

2.1.5.4. Maskeleme

Bazı bölümleri, çalışmanın dışında tutmak için, görüntünün bazı bölümleri kapatılabilir. Bu işleme örtme ya da maskeleme denir. Maskeler, görüntünün verilen komuttan etkilenmemesini veya bazı görüntü bölümlerinin gizlenmesini sağlar. Şekil 2.12 ile her bir piksele uygulanacak maske örneği ve alınan bir görüntünün maskelenmiş halleri gösterilmektedir.



Şekil 2.12. Maskeleme işlemleri

2.2. Video Veri Madenciliği

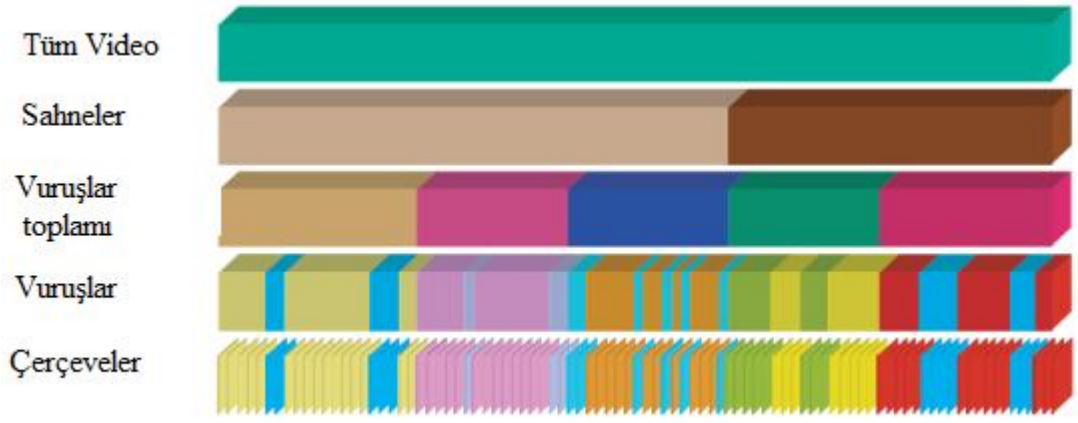
Haber bülteni analizleri, askeriye, video, eğitim, kültürel, web aramaları, reklam, suç önleme, coğrafi bilgi sistemleri içerisinde geniş bir video veritabanı bulundurur.

Bilgisayar kullanıcılarının web üzerinde mevcut olan videolardan, istediklerini çekebilmeleri, içlerinden bilgi alabilmeleri, arama yapabilmeleri ve veri madenciliği yapabilmeleri için gelişmiş teknolojilerin geliştirilmesi gerekmektedir. İşe yarar bilgiler veriler içerisinde olsa da verilerin çok büyük bloklar halinde olması ve bunları işleyebilmek için güçlü araçların gerekmesi bu verileri insanların kullanımına sunmayı zorlaştırmaktadır. Video verileri heterojen yapıdadır. Hem görüntüyü, hem sesi, hem de metni içinde bulundurabilir. Homojen veriler üzerinde analiz her zaman daha kolaydır. Heterojen veriler analiz edilirken önce belli sınıflara ayrılır ve sınıf olarak incelenir(Görüntü, ses ve metin verisi olarak).

Video veri madenciliği, video verileri yönetiminde ve doğru bilgiyi alabilme de önemli bir rol oynar. Görsel olarak benzer çerçeveler, vuruşları (fotoğraf, çerçeve) oluşturur. Benzer vuruşlar birleşerek sahneleri oluşturur. Sahneler de birleşerek videoyu oluşturmaktadır. Burada anahtar çerçeveler önem kazanır. Video veritabanından bilgi keşfinde yapısal olmayan bilgilerde gerektiğinden, bilginin keşfi için nesnelerin ve parçaların arasındaki ilişkilerin belirlenmesi gerekir. Genelde videonun kalitesini artırmak için ilk ön işleme yapılır. Daha sonra önemli özellikleri elde etmek için bu video dosyaları üzerinde çeşitli dönüştürmeler yapılır. Video verisinden ses şiddeti, frekansı, Görüntü histogramı, film uzunluğu, süresi, görüntünün kontrastı ve parlaklığı gibi ölçeklenebilir özellikler çıkarılabilinir. Elde edilen özelliklerle, belirli örnek verileri keşif için veri madenciliği teknikleri kullanılarak madencilik yapılabilir. Elde edilen bu örnek sonuçlar en son bilgiyi elde etmek için değerlendirilir ve anlamlandırılır. İçerik tabanlı geri alma, görüntüyü anlama, veri madenciliği, video tanımlama ve veritabanı alanlarını birleştirme video veri madenciliğinin yöntemleridir.

2.2.1. Video Veri Madenciliği Adımları

Bütün bir videodan veri madenciliği yapılamayacağı için ilk adım gelen videoyu bölümlere ayırmaktır. Video, birbirleriyle ilişkisel küçük parçalara ayrılır. Şekil 2.13 ile bir video parçasının alt bölümleri gösterilir.



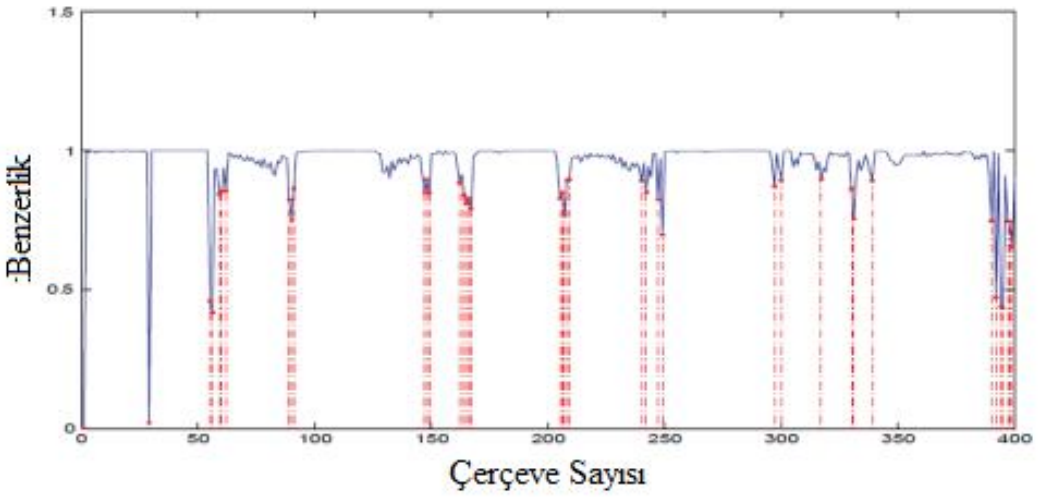
Şekil 2.13. Film yapısı

Video, sahnelerden; sahneler, vuruşlardan ve vuruşlar, çerçevelerden meydana gelir. Vuruşlar arası geçişi bulabilmek için vuruş sınır bulma teknikleri kullanılır. Ardışık

çerçeveler arası benzerlikler ölçülür. Bunun için histogramlar hesaplanır ve çerçevelerin benzerliği için bir eşik değeri seçilir.

$$S(i) = \sum_{k \in bins} \min\left(H_i(k) - H_j(k)\right) \tag{2.1}$$

$S(i)$, ardışık iki çerçeve arası benzerliği gösteren değerlerin tutulduğu bir dizidir. Belli bir eşik değerinden küçük $S(i)$ değerleri için vuruş sınırı belirlenir. Ani hareket bulunduran video sahnelerinde vuruş sayısı fazladır. Haber programı, talk show gibi bir sunucu ve birkaç konuktan oluşan video sahnelerinde ise vuruş sayısı azdır. Şekil 2.14'te alınan bir video parçasında ardışık çerçeveler arası benzerlikler $S(i)$ ölçülerek bu videodaki vuruş sayısı elde edilmiştir.



Şekil 2.14. Vuruş sınırlarının bulunması 40 vuruş bulunmuştur

Sonra anahtar çerçeveler hesaplanır. Her bir vuruşun orta çerçevesi, anahtar çerçeve olarak *Ki* isimli bir diziye atılır. Vuruşlar ise *Si* dizine atılır. *Ki* ve *Si* tek tek karşılaştırılır. *Ki*'dekine benzer çerçeveler ihmal edilerek, farklı olanlar diziye eklenir. Algoritma şöyledir:

Orta çerçeve, ilk çerçeve gibi seçilir;

$$Ki \leftarrow \left\{ f^{\left[(a+b)/2\right]} \right\}$$

$$for\ j = a\,tob$$

$$if\ \max\left(s\left(f^j,\ f^k\right)\right) < ThAf^k \in K_i$$

$$then\ K_i \leftarrow K_i U\left\{f^j\right\}$$

Burada *S(i)* dizisindeki çerçeveler eşik değeri(*Th*) ile karşılaştırılır. Benzer olmayan çerçeveler *K*(*i*) dizisine eklenir.

2.2.1.1. Benzer Çerçeveleri Tespit Etmek

Video veri madenciliğinin ilk adımı videoyu birbirleriyle ilişkisel olacak şekilde küçük parçalara ayırmaktır. Video bölümlemesi veya benzer çerçeve seçimi bir çerçeveden diğerine geçerkenki değişimleri kapsar. İki çerçeve arasındaki bu değişimler kesme veya parçalama olarak adlandırılır. Bir görüntü (vuruş) tek bir kameradan ana değişiklikler olmadan alınan çerçevelerin akışı şeklinde tanımlanabilir.

Bu görüntüleri seçebilmek için çeşitli teknikler geliştirilmiştir [12-13]. En çok kullanılan ortak özellik görüntüleri renk bilgilerine göre gruplandırmaktır. Bu konu üzerindeki son çalışmalar renk gruplandırması konusunda bir sıkıntı olmadığını göstermiştir. Renkleri kullanmanın avantajı uygulanabilirliği, vektörel alanlarının tanımlanabilirliği ve gerçek zamanlı uygulama ile grafik oluşturulabilirliğidir. Bu yolla verilen video okunur, numaralandırılır ve renk tanımlamalarına göre küçük parçalara bölünür. Bir grup görüntü $S = \left\{f\left(m\right),\ f\left(m+1\cdots f\left(n\right)\right)\right\}$ şeklinde gösterilebilir. Burada m ve n ilk ve son görüntünün ve aradakilerin indeksi şeklindedir.

2.2.1.2. Anahtar Görüntü Analizi

Görüntülerin analizi yapıldıktan sonra bilginin düzenli alınabilmesi ve görüntüler arasında bağlantı kurulabilmesi için anahtar görüntü analizinin yapılması gerekir. Anahtar görüntü seçimiyle çeşitli görüntüler arasından bize gerekli olan bilgileri alabilmemiz ve gereksiz olanları ayırabilmemiz kolaylaşır. Anahtar çalışmada görüntü ilk, orta ve son olarak üç parçadan oluşur. Anahtar görüntü olarak üç görüntü seçilmesinin nedeni özellikleri doğru olarak ve en az hatayla temsil edebilmektir.

2.2.1.3. Görüntüyü Parçalama Tekniği ve Bilgiyi Elde Etmek

Görüntüyü bölmek gerekli ve gereksiz olabilecek görüntü bilgilerini ayırmak gibidir. Bu gruplandırmadaki problem bir ön bilgi olmaksızın video içeriğini gerekli olabilecek bilgi şeklinde ayırabilmektir. Videoları parçalama metotları bölme metodu, sıradüzen metodu, yoğunluk-tabanlı metodu, parmaklık (grid) metodu ve model-örnek tabanlı metotlar olarak ayrılabilir. Bu çalışmada sıradüzen metodu görüntülere uygulanmaktadır. Veri toplama işi ancak benzer görüntüler, çerçeveler bölümlendiğinde düşünülebilir. Bu teknik kural tabanlı veya görüntülerin görselleştirilmesidir. Daha sonra çerçeveler görselleştirilip gruplandırıldıktan sonra görüntülerden veri madenciliği yapılır.

2.2.2. Gerçek Zamanda Nesne Tanıma İşlemi

Gelişen teknoloji ile birlikte çok büyük görüntü ve video arşivleri ortaya çıkmış, bu arşivlere etkin ve hızlı bir şekilde erişmek büyük önem kazanmıştır. Gerçek zamanda bu arşivlerden nesne tanıma işlemi yapılarak istenilen verilere daha hızlı ve kolay erişim sağlanmıştır.

Nesne herhangi bir videoda yüz tanıma veya parmak izi, iris gibi nesne tanıma işlemi olabilir. Nesne tanımada kullanılacak birden fazla yöntem vardır. Gereksinimlere göre bu yöntemlerden biri veya birkaçı kullanılabilir. Birden fazla yöntemi bir arada kullanmak sonuçları kesinleştirmek için gerekli olabilir. Bu yöntemler her zaman doğru sonuçları vermeyebilir, bundan dolayı kullanım alanına göre yüksek başarı sağlayanlar seçilmelidir. Başarı oranının yanı sıra tanıma işleminin gerçekleşmesi için gereken sürede yöntemlerin seçilmesinde dikkate alınmalıdır. Gerçek zamanlı (real-time) tespit yapmak gerektiğinde yöntem seçimine çok daha fazla dikkat edilmelidir.

Genel olarak bu sistemlerin çalışma prensibi; her yöntemin kendine ait girdi cihazıyla alınan verilerin analiz edilip daha önceden girilmiş değerlerle karşılaştırılıp eşleştirilmesine dayanmaktadır. Bilgisayarların birim zamanda yaptığı işlem sayısının sürekli artması göz önüne alındığında eldeki veriler ile anlık olarak alınan örneğin karşılaştırılma hızı da gittikçe artmaktadır. Saniyeler içinde yüz binlerce veriyi karşılaştırıp doğru sonuçları veren sistemler günümüzde çeşitli alanlarda kullanılmaktadır.

Nesne tanıma işleminin genel çalışma prensibi iki adımdan oluşmaktadır. Birinci adımda tanınacak kişinin ilgili yönteme ait bilgiler gerekli araçlar vasıtasıyla bilgisayar ortamına aktarılıyor. Bu bilgiler yine yönteme özel algoritmalar sayesinde analiz ediliyor ve kişiyi tanımlayacak parametreler bu bilgiler içinden seçilerek veritabanına kayıt ediliyor. İkinci adım ise kişinin kimlik doğrulama isteğidir. Bu adımda sisteme aynı araçlar vasıtası ile girilen bilgiler genellikle kayıt sisteminde uygulanan aynı algoritmayla analiz edilip veritabanındaki bilgilerle karşılaştırılıp eşleştirmelere bakılıyor. Eğer eşleşme varsa kişinin kimliği onaylanmıştır aksi halde sistemde bir sorun yoksa kişi iddia ettiği kimliğe sahip değildir.

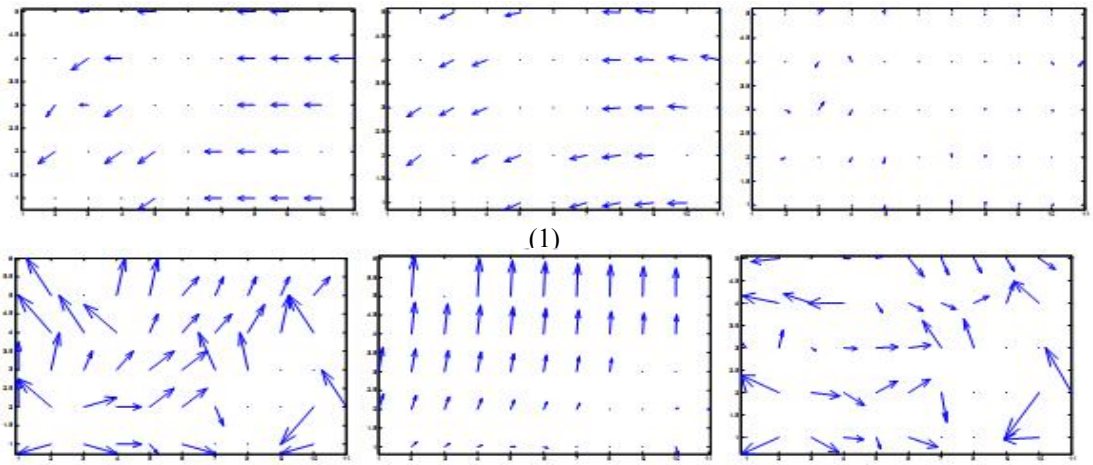
2.2.3. Videonun İçerik Özelliklerine Göre Sınıflanması

Videoda, iki önemli hareket vardır. Global hareket; kameranın ileri, geri gitmesi sağa, sola dönmesi ve yakınlaşıp uzaklaşma hareketi global harekettir. Yerel hareket ise video içindeki nesnelerin kameraya doğru hareketleridir. Yatay ve dikey alanda bu hareketlerin hızlarının farkı hesaplanarak videonun dinamik mi yoksa statik mi olduğu hakkında bilgi edinebiliriz.

$$U = a1 * x + a2 * y + b1$$
$$V = a3 * + a4 * y + b2 \qquad (2.2)$$

U ve V, videodan elde edilmiş yatay ve dikey hızlardır. a; global, b yerel harekettir. Hızların tüm vuruşlar için toplamı bulunup video içeriği hakkında bilgi elde edilir. Şekil 2.15'te hareket vektörlerine göre bir videonun içeriğinin nasıl olduğu belirlenebilir.

$$\in_j = \sum_{k \in \text{hareket blokları}} \sqrt{\left(u_k' - u_k\right)^2 + \left(v_k' - v_k\right)^2} \qquad S\ M\ C_i = \sum_{j \in S_i} \varepsilon_j \qquad (2.3)$$



(1)

(2)

Şekil 2.15. Hareket yönlerine göre video içeriğinin bulunması (1) durgun bir video (2) çok hareketin olduğu bir video

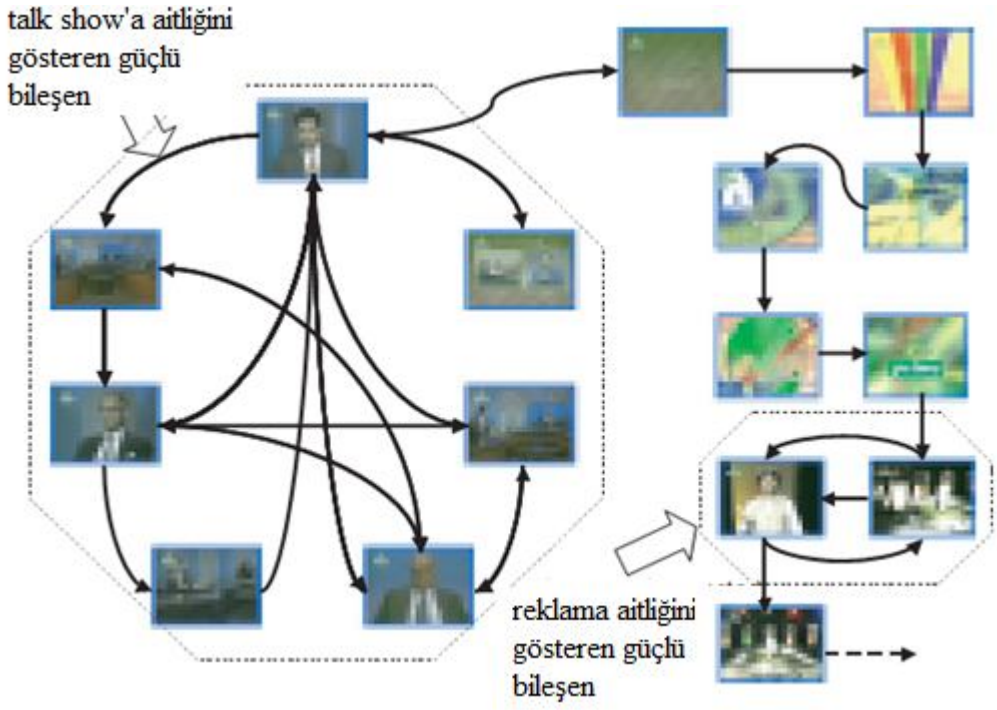
Ayrıca sahneler arası ses şiddetine, frekansına bakılarak da görüntü içeriği hakkında bilgi elde edilebilir. Aşağıdaki denklem ile ses sinyalinin enerjisi hesaplanabilir:

$$E = \sum_{i \in \text{aralık } (50\,ms)} (Ai)^2 \tag{2.4}$$

Burada A, örneğin 50 ms gibi küçük bir zaman aralığında(i) indekslenmiş ses örneğidir. Ses korku filmlerinde aniden yükselir; komedi ve dramda ise genelde statik bir tonda ses kullanılır.

Videoda kullanılan renk seviyesi ve parlaklık da videoyu sınıflamamıza yardımcı olur. Korku ve dramda kontrast yüksek, parlaklık düşüktür. Karanlık çok yoğun kullanılır. Komedide kontrast düşük, parlaklık fazladır.

Vuruşların süresi ve renk yoğunluğu yardımı ile istenilen çerçeveleri bulmak da mümkündür. Talk showlarda konuklar ve sunucu vardır. Vuruşlar belirlendikten sonra benzer vuruşlar bazı algoritmalara tabi tutularak vuruş bağlantı grafiğinde gösterilir. Bu durum şekil 2.16 ile açıklanabilir. Benzer vuruşlar arasında döngüler vardır. Sunucunun konuşması her zaman kısadır, konuklarınki ise uzun ve renklidir. Sunucu vuruşu bu süreye bakılarak bulunabilir. Ayrıca sahneler arasında bazen reklamlar da olabilir. Reklamlar ile show arasında bir boş çerçeve vardır. Ayrıca reklamlar arasında döngü olmaz, çünkü çerçeveler birbirine pek benzemez, döngü yoktur. Çok kısa sürelidir.



Şekil 2.16. Talk show ile reklam vuruşlarının ayrılması

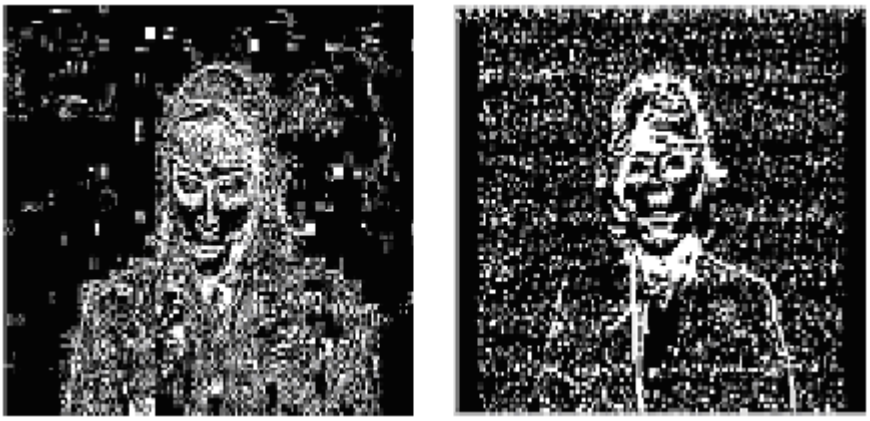
2.2.4. Videoda Dinamik Nesnelerin Bulunması

Video madenciliğinde amaç, nesnelerin, karakterlerin ve sahnelerin hareket sıklığına göre istenilen veriyi, heterojen veriden çekmektir. Güvenlik amaçlı izleme sistemlerinde çevrenin video kayıtları tutulur. Sonra bu kayıtlar incelenerek nesnelerin hareketleri bulunur.

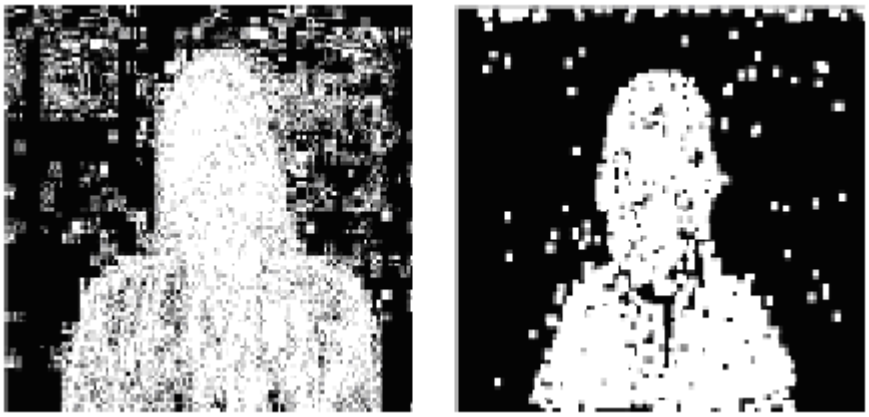
Genelde izleme sistemlerinde nesne hareketlerini bulmak için arka planı eleme yöntemi kullanılır. Bu yöntemde ilk video okunarak vuruş ve çerçevelere çevrilir. İki çerçeve arası fark hesaplanır. Böylece farklı çerçeveler belirlenir. Şekil 2.17'de iki çerçeve farkı alınan görüntü elde edildikten sonra ön plan nesneyi bulmak için şekil 2.18'de arkaplan eleme işlemi uygulanır.

$$P(FD) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(FD-\mu)^2}{2\sigma^2}\right) \tag{2.5}$$

Burada FD iki çerçeve arası farktır ve bu formül FD'nin standart sapmasıdır.



Şekil 2.17. İki çerçeve arası farkın bulunması



Şekil 2.18. Arka planın elenmesinden sonraki durum

Şekil 2.19. Post işleminin uygulanması

Çerçeve farkları sıfıra uydurulur ki arka plan elenebilsin. Arka plan elenip kaydedilir. Arka plan ve statik görüntüler kaydedildikten sonra ön planda bulunan dinamik nesneler bulunur. Hem nesnede hem arka plan bölgesinde gürültüler oluşur. Bunları önlemek için post işlemi uygulanır. Gürültüyü temizlemek için Median filtre

çalıştırılır. Şekil 2.19 ile görüntü gürültüden temizlenir. Filitrelemeden sonra elde edilen görüntü maskelenir (binary görüntü). Son olarak elde edilen görüntüdeki pikseller belli bir eşik değerinden büyükse orijinal görüntü çerçevelerindeki pikseller ile yer değişir. Şekil 2.20 ile piksellerin yer değişmesi ile oluşan piksel değişiklikleri bize hareketi gösterir. Bu işlem ile hareketli nesne bulunur.

Şekil 2.20. Nesne tanıma işlemi sorası

Görüntüde bulunan hareketli nesnelerin sayısı ise bir değişken değer ve sayaç ile tutulur. Görüntü baştan sona taranır. Nesne önceden kayıtlı mı değil mi sayaca bakılır. Değilse değişken sayı değeri arttırılır ve nesne sayaca eklenir. Bazen nesneler birbirini kapatır ve tek nesne gibi görünür. Bu yüzden nesne sayısını bulmak % 100 başarılı olmaz.

2.3. Nesne Takibi

Görsel niteliklerin karmaşık ortamlarda sağlam ve güvenilir bir şekilde takip edilmesi gerçekten üstesinden gelinmesi gereken zor bir işlemdir. Kapalı devre izleme ve görüntüleme, algılamaya dayalı kullanıcı arabirimleri, akıllı odalar ve video sıkıştırma gibi gerçek zamanlı uygulamalar hareketli nesnelerin takip edilmesine gereksinim duyarlar. Buna ek olarak gerçek zamanlı uygulamalarla takip edici sistem kaynaklarının çok az bir yüzdesini kullanmalı ve kalanı ise ön işleme aşaması ya da tanımlama, yörünge yorumlama ve ilişki kurma gibi yüksek seviyeli işlemler için ayrılmalıdır. Takip edicinin getireceği yoğun hesaplama külfeti kritik uygulamalar için bu noktada önemli olmaktadır. Bu yüzden öncelikle nesne takip yöntemlerini detaylı olarak inceleyelim.

2.3.1. Nesne Takip Yöntemleri

Hareketli nesnenin tespiti, doğal sahnelerde meydana gelen ani ışık ve hava durumu değişimi ve karışıklığa neden olan tekrar eden hareketler (rüzgarda salınan ağaç yaprakları) gibi dinamik değişikliklerden dolayı güvenilir bir şekilde gerçekleştirilmesi zor olan bir problemdir[19]. Hareketli nesne tespiti için sıklıkla kullanılan üç yaklaşım vardır: geçici fark [14-15] optik akış [16] ve arka plandan çıkarma [17-18]. Geçici fark alma yöntemi değişken alanlarda iyi çalışır, fakat gözetlenen alanın tümüyle ilgilenir. Ayrıca fark alma yönteminin gürültüden fazla etkilenmesi nedeniyle gerçek zamanlı işlemlerde az tercih edilmektedir. Optik akış, kameranın o anki konumundan bağımsız bir şekilde hedefi belirler. Zorluğu karmaşık hesaplamalar gerektirmesi ve gerçek zamanda icra edilebilmesi için özel donanımlara ihtiyaç duymasıdır. Pek çok gözetleme sisteminde otomatik öğrenilen arkaplana dayalı hareket algılama yaklaşımı kullanılmaktadır.

Arka plandan çıkarma yöntemi, dışarıdan gelen dinamik değişimleri tam olarak algılayamaz. Arka plandan çıkarma yöntemi, gerçek zamanda gözetleme uygulamalarının yanında [21], video kodlama uygulamalarında da kullanılır [22].

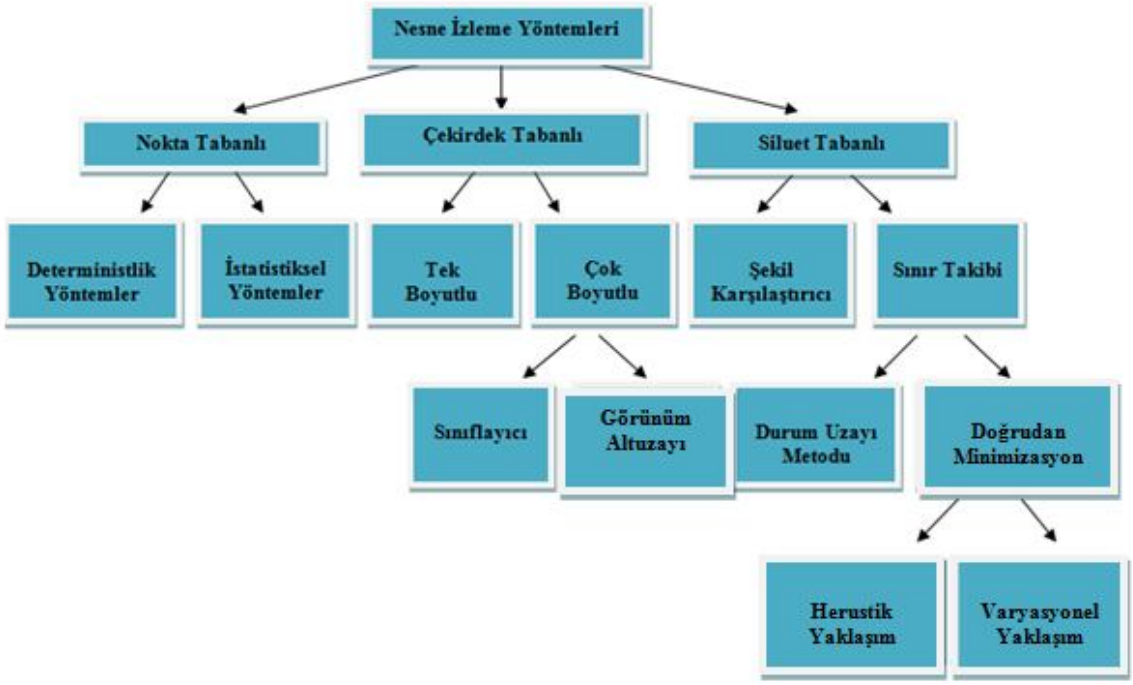
Hemen hemen tüm nesne takip algoritmaları, nesne hareketlerinin ani değişimlere değil de yumuşak hareketlere sahip olduğunu varsayar. Ayrıca, nesne hareketlerinin sabit hız veya ivmeye sahip olduğu düşünülürse, buna ilaveten, nesnelerin sayıları, büyüklükleri, görünümleri veya sahip oldukları fiziksel şekilleri önceden belirtilirse, nesne takip algoritmalarının karmaşıklığı azaltılır. Aksine çok sayıda nesne varsa ve takip edilecek nesne, diğer nesneler ile engelleniyorsa; gürültülü bir görüntü veya parlaklık görüntüyü engelliyorsa algoritmalar biraz zorlaşır. Nesne takibinde karşılaşılan zorlukları sıralamamız gerekirse [23]:

- Üç boyutlu gerçek dünya verilerinin iki boyutlu görüntü alanına yansıtılmasıyla meydana gelen bilgi kaybı
- Görüntü üzerinde meydana gelen gürültüler
- Nesne hareketlerinin karmaşık oluşu
- Nesnelerin ayırt edilebilir fiziksel bir yapıya sahip olamayışları
- Nesne görünümünün bir kısmının veya tamamının engellenmesi (kapatma)
- Karmaşık nesne şekilleri

❖ Ortamdaki ışık miktarının değişimi

❖ Gerçek zamanlı uygulamaların gereksinimleri

Buna göre üç temel nesne takip yöntemi bulunmaktadır: 1) nokta tabanlı; 2) çekirdek tabanlı; 3) siluet tabanlı [20-38]. Şekil 2.21 ile Nesne takip yöntemleri gösterilebilinir.



Şekil 2.21. Nesne takip yöntemleri [24]

2.3.1.1. Nokta Tabanlı Nesne Takibi

Bu takip yönteminde takip edilen her bir nesne tek bir nokta ile ifade edilir. Bu yöntem ile güncel imgede takip edilecek her bir nesneye bir nokta aktarıldıktan sonra bu noktalar ile önceki imgede tespit edilen noktalar arasındaki veri bağı ilişkisinin doğru bir şekilde oluşturulması beklenir. Bu problemin çözümü için şu iki adım sırayla gerçekleştirilir:

1) Güncel imgede nesne yakalama (her biri nokta ile ifade edilir)

2) Bu noktalar ile önceki imgede tespit edilen noktalar arasındaki nokta benzerlik değerlerinin hesaplanması.

Nokta benzerliğinin hesaplanması işlemi, özellikle nesne görünümünün kaybolması, yanlış nesne yakalanması, nesnenin imgeye ilk girişi veya çıkışı gibi durumlarda karmaşık bir hale dönüşür. Bu alandaki yöntemler genellikle iki alt kategoride incelenir [24].

a) Deterministlik Yöntemler

Deterministlik yöntemler, ardışık imgelerde bulunan nesnelerin birbirlerine bağlanma maliyetlerini tanımlar. Nesneler arasındaki bağlanma maliyetinin tanımlanması için nesne hareketleri üzerinde aşağıda belirtilen sınırlamalar göz önüne alınır [24]:

- **Yakınlık:** Bir imgeden diğer bir imgeye geçerken nesne pozisyonlarının önemli ölçüde değişmeyeceğini ima eder.
- **Maksimum hız:** Nesnelerin hızları üzerinde bir üst sınır değeri tanımlanır ve sadece nesnelerin etrafında dairesel bir komşuluk içerisinde kalan muhtemel nesne adaylarının benzerlik değerleri göz önüne alınır.
- **Küçük hız değişimi (yumuşak hareket):** Nesnenin hız yönünün önemli bir ölçüde değişmeyeceğini ima eder.
- **Genel hareket:** Küçük bir komşuluktaki nesnelerin hızlarının benzer olması için sınırlama uygulanır. Bu sınırlama çoklu noktalar ile sunulan nesneler için uygundur.
- **Katılık:** 3-boyutlu dünyadaki nesneler genellikle katı bir biçime sahiptir. Bundan dolayı güncel bir nesne üzerindeki herhangi iki nokta arasındaki mesafenin değişmeyeceği düşünülür. Bahsedilen sınırlamalar sadece deterministlik yöntemlerde değil aynı zamanda istatistiksel yöntemlerde de kullanılabilir.

b) İstatistiksel Yöntemler

Video algılayıcılarından elde edilen ölçümler her zaman gürültüye sahiptir. Ayrıca, nesne hareketleri birtakım istenmeyen etkilere maruz kalır. İstatistiksel yöntemler, nesnenin durum (pozisyon, hız ve ivmesinin) tahmini boyunca ölçüm değerlerini ve model belirsizliklerini göz önüne alarak nesne takibi problemini çözmeye çalışır. Bu yöntemler pozisyon, hız ve ivme gibi nesne özelliklerini modellemek için durum uzay yaklaşımını kullanırlar [25]. Ölçümler genellikle imgelerdeki nesne pozisyonlarını içerir. Bu noktadan sonra istatistiksel yöntemlerin imge içerisindeki nesnelere ait durum tahminini nasıl formülleştirdiğini ve hangi çözümleri sunduğunu detaylı bir şekilde incelenir. Buna göre problemin tanımı için Kalman filtreleme yöntemi ve parçacık filtreleme yöntemi kullanılır. [26].

İmge üzerinde hareket eden bir nesnenin istatistiksel olarak durum tahmin problemi şu şekilde formüle edilebilir [27]. Takip edilecek nesnenin durum bilgisi X^t :

t=1,2…..(örneğin pozisyon) şeklinde bir dizi olarak tanımlanır. Zaman boyunca durum üzerindeki dinamik değişim denklem (2.6) ile ifade edilir:

$$X^t = f\left(X^{t-1}\right) + W^t \tag{2.6}$$

$W^t : t = 1, 2, \ldots$ beyaz gürültüdür. Ölçüm verisi ile durum değişkeni arasındaki ilişki ise denklem (2.7) ile ifade edilir.

$$Z^t = h\left(X^t\right) + N^t \tag{2.7}$$

N^t'de beyaz gürültüdür ve W^t den bağımsızdır. İstatistiksel yöntemlere dayalı nesne takip edicilerin temel amacı, t anına kadarki tüm ölçüm değerlerini göz önüne alarak X^t durum değişkenini tahmin etmektir. Bir başka deyişle, sonrasal olasılık yoğunluk fonksiyonunu (posterior probability density function) $p\left(X^t \middle| Z^{1\cdots t-1}\right)$ elde etmektir. Teorik olarak en uygun çözüm, problemi iki adımda çözen tekrarlamalı Bayes filtresi yöntemini kullanmaktır. Bu adımlar tahmin ve düzeltme adımıdır. Tahmin adımı, dinamik bir eşitlik kullanır ve güncel durumun $t-1$ anındaki öncesel olasılık yoğunluk fonksiyonunu (prior probability density function) $p\left(X^t \middle| Z^{1\cdots t}\right)$ hesaplanır. Daha sonra sonrasal yoğunluk fonksiyonunun hesaplanabilmesi için güncel ölçümün maksimum olabilirlik fonksiyonunu $p\left(Z^t \middle| X^t\right)$ kullanılır. Buna göre sonrasal yoğunluk fonksiyonu denklem (2.8)'de gösterildiği gibi hesaplanır.

$$p\left(X^t \middle| Z^{1\cdots t}\right) = \frac{P\left(X^t \middle| Z^{1\cdots t-1}\right) p\left(Y^t \middle| X^t\right)}{p\left(Y^t \middle| Z^{1\cdots t-1}\right)} \tag{2.8}$$

Denklem (2.8)'deki $p\left(Y^t \middle| Z^{1\cdots t-1}\right)$ normalizasyon katsayısıdır. Ekranda sadece tek nesnenin olması durumunda, nesnenin durum bilgisi bahsedilen iki adım kullanılarak rahatlıkla tahmin edilebilir. Diğer taraftan, ekranda birden fazla nesnenin olması durumunda, elde edilen ölçümler ile ilgili nesneler arasında gerekli bağlantıların kurulma ihtiyacı ortaya çıkar. Bu nedenle çoklu nesnelerin takibi problemi için veri bağı ve durum tahmini problemlerinin birleştirilmiş bir çözümüne ihtiyaç duyulur.

Tek nesnenin olduğu bir durumda, eğer f^t ve h^t fonksiyonları doğrusal ve nesnenin başlangıç durumu ve gürültü değeri Gaussian dağılımına sahipse, o zaman en uygun çözüm Kalman filtre tarafından elde edilebilir. Eğer nesnenin başlangıç durumu ve sistem gürültüsü Gaussian dağılımına sahip değilse o zaman en uygun çözüm Parçacık filtreleme yöntemiyle elde edilir.

❖ Kalman Filtreleme Yöntemi

Kalman filtreler, durum vektörü bir Gaussian dağılımına sahip lineer sistemlerin durum tahminlerini gerçekleştirmek için kullanılır [28]. Tahmin ve düzeltme gibi iki adımdan oluşur. Tahmin adımı, değişkenlerin yeni durumu tahminini yapabilmek için durum modelini kullanır. Şöyle ki:

$$\overline{X^t} = DX^{t-1} + W \tag{2.9}$$

$$\overline{\sum{}^t} = D\sum{}^{t-1} D^t + Q^t \tag{2.10}$$

$\overline{X^t}$ ve $\overline{\sum{}^t}$, t anındaki durum ve kovaryans tahminleridir. D, t ile $t-1$ anındaki durum değişkenleri arasındaki ilişkiyi tanımlayan durum dönüşüm matrisidir. Q, W gürültüsünün kovaryansıdır. Benzer bir şekilde, düzeltme adımı güncel gözlem değerini nesne durumunu güncellemek için kullanır.

$$\overline{K^t} = \sum{}^t M^t \left[M \sum{}^t M^t + R^t \right]^{-1} \tag{2.11}$$

$$X^t = \overline{X^t} + K^t \left[\underbrace{Z^t - MX^t}_{V} \right] \tag{2.12}$$

$$\sum{}^t = \overline{\sum{}^t} - K^t M \overline{\sum{}^t} \tag{2.13}$$

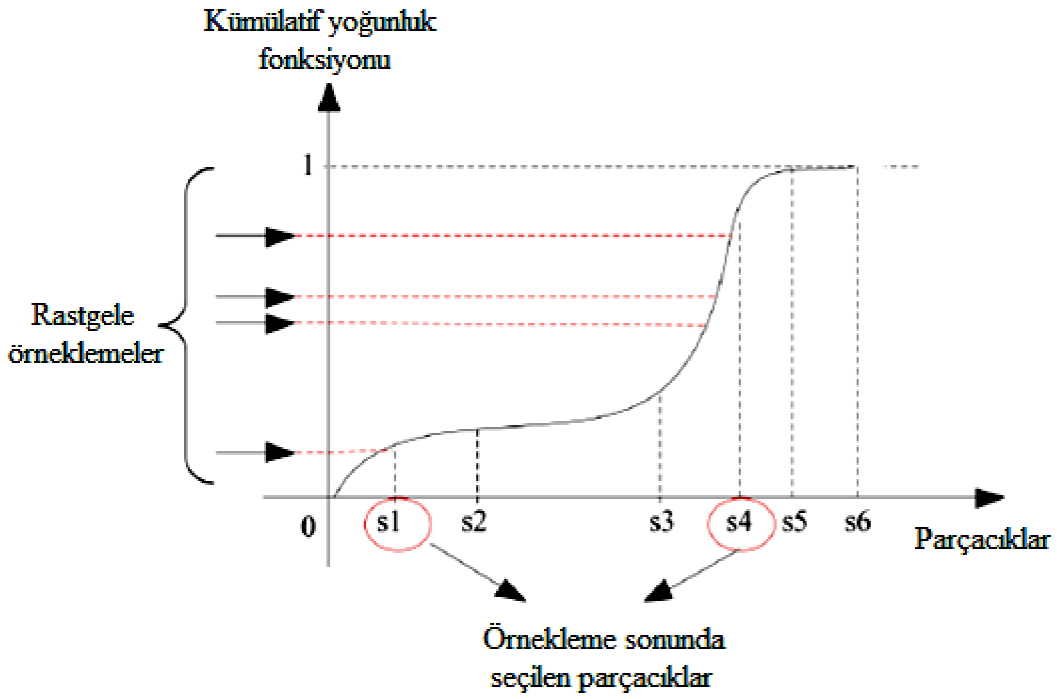
Yukarıdaki denklemlerde V yenilik olarak adlandırılır ve M ölçme matrisidir. K, durum modellerinin yayılımı için kullanılan kalman kazancıdır. Dikkat edilmesi gereken nokta şudur ki; güncellenen X^t durumu hala bir Gaussian dağılımıdır. Bu durumda f^t ve h^t fonksiyonları doğrusal değildir ve Taylor seri açılımları kullanılarak fonksiyonlar

doğrusallaştırılabilir. Bahsedilen bu filtreleme tekniği literatürde genişletilmiş kalman filtre (extended kalman filter) olarak bilinir.

❖ Parçacık Filtreleme Yöntemi

Kalman filtresinin bir dezavantajı, durum değişkenlerinin Gaussian dağılımına sahip olma zorunluluğudur. Böylelikle, kalman filtresi gaussian dağılıma sahip olmayan sistemler için zayıf bir tahmin edicidir [27]. Bu dezavantaj parçacık filtreleme yöntemiyle çözülebilir [29]. Parçacık filtrede, t anındaki şartsal durum yoğunluğu π ağırlığına sahip N tane parçacık içeren örnek küme ile ifade edilir. Her bir parçacık $\left\{s_t^{(n)} : n = 1,\ldots, N\right\}$ ve her bir parçacığın ağırlığı ise $\pi_t^{(n)}$ (örnekleme olasılığı) olarak gösterilir. Ağırlıklar parçacığın önemi veya onun gözlenme frekansı olarak tanımlanabilir. Her bir $\left(s_t^{(n)},\ \pi_t^{(n)}\right)$'nin hesaplama maliyetinin azaltılması için birikmiş (katlanmış) bir ağırlık $c^{(n)}$ de aynı zamanda hafızaya yüklenir $\left(c^{(N)} = 1\right)$. T anındaki yeni örnekler $t-1$ anındaki $S_{t-1} = \left\{\left(s_{t-1}^{(n)},\ \pi_{t-1}^{(n)},\ c_{t-1}^{(n)}\right) : n = 1,\ldots N\right\}$ örnekleme şemasıyla elde edilir. En genel örnekleme şeması aşağıda bahsedildiği gibi önem örnekleme şemasıdır. Bu şema, seçme, tahmin ve düzeltme olmak üzere üç temel adımın tekrarlı bir şekilde gerçekleşmesi prensibine dayanır [25]. Bu adımlar şu şekilde gerçekleşir:

(1) Seçme adımı: S_{t-1}'den N tane rastgele $\hat{s}_t^{(n)}$ örnek seçilir. Seçme işlemi yapılırken $r \in [0,\ 1]$ olmak kaydıyla rastgele bir değişken üretilir ve $c_{t-1}^{(j)} > r$ ve $s_t^{(n)} = s_{t-1}^{(j)}$ şartlarını sağlayan en küçük j indeksine sahip örnekler seçilir. Bu adımdaki gerçekleşen operasyon Şekil 2.22'de görüntülenmiştir. Bu şekli açıklamak gerekirse nesnenin rastgele olasılık yoğunlukları olsun ve bunların her birinin toplam olasılık yoğunluk değerine bölünmesi ile kümültatif yoğunluk fonksiyonu oluşur. Kümültatif yoğunluk fonksiyonuna grafikde denk gelen parçacıklardan $(s1,\ s2\cdots)$ olasılık yoğunluk dağılımı en yüksek olanlar seçilir, düşük olanlar ise yok edilir.



Şekil 2.22. Parçacık filtreleme yönteminin seçme adımı

(2) Tahmin adımı: Seçilen her bir $\hat{s}_t^{(n)}$ örneği için $s_t^{(n)} = f\left(s_t^{(n)},\ W_t^{(n)}\right)$ ile yeni bir örnek üretilir. Burada $W_t^{(n)}$ sıfır ortalamaya sahip bir Gaussian hatasıdır ve f negatif olmayan bir fonksiyondur $\left(f(s) = s\right)$.

(3) Düzeltme adımı: $s_t^{(n)}$ örneklerine ait $\pi_t^{(n)}$ ağırlıkları z_t ölçümleri kullanılarak hesaplanır. Bunun için $\pi_t^{(n)} = p\left(z_t \middle| x_t = s_t^{(n)}\right)$ eşitliği kullanılır. Burada $p(\bullet)$, Gaussian dağılımı ile modellenebilir.

Elde edilen yeni S_t örnekleri $\in_t = \sum_{n=1}^{N} \pi_t^{(n)} f\left(s_t^{(n)},\ W\right)$ eşitliğinde kullanılarak yeni nesne pozisyonları hesaplanabilir. Parçacık filtre tabanlı nesne takip edicilerin başlangıç değerleri, örnekleme dizisini kullanan sistemlerin eğitimiyle veya ilk ölçümler $\left(s_0^{(n)} \sim X_0\right)$ kullanılarak gerçekleşir. Sistem ilk ölçümler kullanılarak başlatılırsa, her bir örneğin ağırlık değeri $\pi_0^{(n)} = \frac{1}{N}$ şeklinde eşit olarak dağıtılır. Ayrıca en iyi parçacık örneklerinin korunması için düşük ağırlıklı olanların elenmesi gerekir. Bunun için ek bir örnekleme

algoritmasına ihtiyaç duyulur. Burada dikkat edilmesi gereken nokta, sonrasal yoğunluk fonksiyonunun Gaussian olmak zorunda olmayışıdır.

2.3.1.2. Çekirdek Tabanlı Nesne Takibi

Çekirdek tabanlı nesne izleyiciler, parametrik olmayan tahmin ediciler gurubu içerisinde yer almaktadır. Parametrik olmayan sistemlerde sabit bir fonksiyon yapısı söz konusu değildir ve bir tahmin gerçekleştirileceği zaman dağılıma ait tüm veri değerleri göz önünde bulundurulur. Bunun yanında parametrik sistemlerde sabit bir fonksiyon yapısı ve sabit parametre değerleri bulunmaktadır [25].

Çekirdek tabanlı nesne takibi yönteminin amacı, takip edilecek nesneyi basit bir geometrik şekil içerisine alarak şekil içerisinde kalan görünüm bilgisinin olasılık yoğunluk dağılımını elde etmek ve bu dağılımı ardışık video imgeleri boyunca takip edebilmektir [30]. Olasılık yoğunluk dağılımı, takip edilecek nesnenin her pikselinin diğer pikseller üzerindeki etkisini ifade eder [31]. Bu etki, çekirdek yoğunluk fonksiyonu kullanılarak yumuşatılır.

Çekirdek tabanlı nesne takibi işlemini anlayabilmek için öncelikle bir nesnenin görünümüne ait histogram sunumunu ve böyle bir sununum sahip olduğu dezavantajları bilmek gerekir. Takip edilecek nesneye ait görünüm bilgisinin histogramı oluşturulmak istenildiği zaman öncelikle görünüm veri kümesinin kaç eşit parçaya (“bin” olarak adlandırılır) bölünmesi gerektiği ve bu parçaların başlangıç ve bitiş noktalarının hangi değerler olacağı belirlenmek zorundadır. Bu zorunluluklar histogram sunumunun cazibeliğini azaltır ve bu sınırlamaların olmadığı çekirdek yoğunluk fonksiyonlarının kullanılmasına neden olur [31].

Histogramdaki her bir parçanın başlangıç ve bitiş noktalarına olan bağımlılığını kaldırmak için çekirdek yoğunluk tahmin edicileri her bir veri noktasını merkez kabul ederek veriyi belirlenen çekirdek fonksiyonundan geçirir ve her bir veri noktası için bir yoğunluk değeri elde edilir. Böylelikle çekirdek fonksiyonunun yumuşak veya sertliği yapılan tahmininin yumuşak veya sert olmasına neden olur. Bu sayede histogram sunumu kullanıldığı zaman elde edilen dezavantajlar ortadan kalkmış olur [32].

$K(\bullet)$ çekirdek fonksiyonunun giriş parametresi denklem (2.14)'te ifade edildiği gibi hesaplanır.

$$U = \frac{x - \mu}{\sigma} \tag{2.14}$$

Formüldeki μ ve σ simgeleri ile $K(\bullet)$ çekirdek fonksiyonunun merkez değerini ve bant genişliğini ifade etmektedir. Literatürde en sık kullanılan çekirdek fonksiyonu Gaussian olarak bilinir [32].

Çekirdek tabanlı nesne takibi işlemi tek boyutlu ve çok boyutlu veriler üzerinde gerçekleştirilebilir. Buna göre, takip edilecek nesnenin görünüm bilgisi denklem (2.15)'te ifade edildiği gibidir.

$$X_i = \left[X_{i1}, X_{i2}, \ldots, X_{id}\right]^T, \quad i = 1, \ldots, n \tag{2.15}$$

a) Tek Boyutlu Çekirdek Yoğunluk Tahmini

Tek boyutlu veri kümesinin olasılık yoğunluk dağılımını hesaplanmak için Denklem (2.16)'da belirtilen formül kullanılır. Buna göre, veri elemanının dizisi üzerindeki katkısı (olasılık yoğunluk değeri veya yoğunluk tahmini) şu şekilde hesaplanır:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2.16}$$

Bu formülde $K(\bullet)$ çekirdek fonksiyonu x merkezi noktasına ve h bant genişliğine sahiptir. Burada $\int K(t)\,dt = 1$ eşitliğinin sağlanma zorunluluğu vardır.

b) Çok Boyutlu Çekirdek Yoğunluk Tahmini

Çok boyutlu bir veri kümesinin olasılık yoğunluk dağılımı Denklem (2.17) kullanılarak hesaplanır.

$$\widehat{F_h}(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^d} K\left(\frac{x - x_i}{h}\right) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^d} K\left(\frac{x_1 - x_{1d}}{h}, \cdots, \frac{x_d - x_{id}}{h}\right) \tag{2.17}$$

Burada kullanılan bant genişlikleri $h=\left(h_1,\ldots,h_d\right)^T$ açık bir şeklinde yazılabilir. Buna göre denklem (2.17)'nin güncel hali denklem (2.18) ile ifade edilebilir.

$$\widehat{f_h}\left(x\right)=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_1,\ldots,h_d}K\left(\frac{x_1-x_{i1}}{h_1},\ldots,\frac{x_d-x_{id}}{h_d}\right) \tag{2.18}$$

Uygun bant genişliği değerini elde etmek için sık kullanılan genel bir yöntem denklem (2.19 ve 2.20)'de belirtilir. Bu eşitlikler yardımıyla dağılımın (Asymptotic Mean Integrated Squared Error) değeri minimum seviyeye indirilir. Bu değer takip edilecek nesneye ait görüntü verisinden elde edilir. Bant genişliğinin belirlenmesinde değerinin kullanılması, veri kümesindeki tüm karakteristik özelliklerin korunması anlamına gelir.

$$H_{optimal}=\arg_{\min}\ AMISE \tag{2.19}$$

$$AMISE\left(h\right)=\int_{-\infty}^{\infty}E\left(f_h\left(y\right)-f_h\left(y\right)\right)^2dy \tag{2.20}$$

Dağılımın değerinin hesaplanmasında kullanılan f fonksiyonu bilinmeyen yoğunluğu ifade eder ve $\hat{f}$ simgesiyle de f fonksiyonunun n. örneğine dayalı tahmin bilgisi ifade edilmektedir. Bununla birlikte E simgesi beklenen değeri ifade eder.

2.3.1.3. Siluet Takibi

Takip edilmesi istenilen nesneler basit geometrik şekiller ile tanımlanamayan el, baş, omuz gibi karmaşık geometrik şekillere sahip olabilir. Siluet tabanlı yöntemler bu nesneler için doğru bir şekil tanımlayıcısı sağlarlar. Siluet tabanlı nesne takip edicilerinin asıl hedefi, önceki imgeler kullanılarak üretilen nesne modelini güncel imge içerisinde bulmaktır [33]. Bu modeller nesnenin renk histogramı, nesne kenarını veya sınır şeklini kullanır. Siluet tabanlı nesne takip edicileri "şekil karşılaştırıcılar" ve "sınır takip ediciler" olarak adlandırılan iki alt kategori altında incelemek mümkündür. Şekil karşılaştırmalı yaklaşım, güncel imge içerisinde nesne siluetini arar. Diğer taraftan sınır izlemeli yaklaşım ise, bazı enerji fonksiyonlarının minimizasyonunu veya durum uzay modellerini kullanarak nesnenin güncel imgedeki yeni yerini belirlemeye çalışır. Kategorileri şöyle açıklayabiliriz;

a) Şekil Karşılaştırıcılar

Şekil karşılaştırıcı yöntemlerin çalışma prensibi şablon karşılaştırmalı nesne takip yöntemleri ile benzerdir. Bu yaklaşımda, nesne siluetinin güncel imgeden bir sonraki imgeye aktarıldığı varsayılır. Güncel imgedeki nesne silueti ve onunla ilişkili nesne modeli bir sonraki imgede aranır. Arama işlemi, bir önceki imgede elde edilen nesne silueti ve modeli ile güncel imgede yakalanan nesneler arasındaki benzerlik değerleri hesaplanarak gerçekleştirilir. Bu nedenle, silueti net olmayan (katı olmayan) nesneler doğru ve istikrarlı bir şekilde takip edilemez. Genellikle nesne modeli için kenar haritası kullanılır. Nesne modelinin her imgede nesneyi tam ifade edebilmesi için görünüş değişikliklerini göz önüne alarak yeniden hesaplanması gerekir. Bu güncelleme işlemi, özellikle katı olmayan nesne hareketlerindeki ışık değişimi ve bakış noktası gibi nesne takip problemlerinin üstesinden gelebilmek için gereklidir [34].

b) Sınır Takibi

Şekil karşılaştırma yöntemlerinin aksine sınır takibi yöntemleri güncel imgedeki sınır çizgilerini başlangıç kabul ederek bir sonraki imgede yeni nesne sınırlarını arar. Sınır takibi algoritmalarının uygun sonuç vermesi için güncel imgedeki nesne ile bir sonraki imgede bulunan nesne alanlarının kesişimi olmak zorundadır. Sınır değerlendirme algoritmaları iki alt kategoride incelenebilir. İlk yaklaşım, sınır biçim ve hareketini modellemek için durum uzay modellerini kullanır. İkinci yaklaşım, gradient azalma gibi minimizasyon tekniklerini kullanarak sınır enerjisini minimize eder [35].

(1) Durum Uzay Modelli Kullanarak Nesne Takibi

Nesneye ait durum vektörü, nesne sınırının biçim ve hareketi şeklinde tanımlanır. Nesne sınırına ait sonrasal olasılık yoğunluk değerinin maksimum olması için sınıra ait durum vektörü her bir imgede güncellenir. Sonrasal yoğunluk, öncesel yoğunluğa ve güncel olabilirliğe bağlıdır. Güncel olabilirlik değeri, nesne sınırları ile gözlenen kenar değerleri arasındaki mesafe kullanılarak hesaplanabilir [36].

(2) Sınır Enerji Fonksiyonunun Minimizasyonu ile Nesne Takibi

Sınır değerlendirme kontekstinde, bu bölümde işlenen sınır takip ediciler ile bir önceki bölümde incelenen bölütleme yöntemleri arasında bir paralellik bulunmaktadır. Nesne

bölütleme ve nesne takip yöntemleri sınır enerji fonksiyonunu greedy veya gradient azaltma yöntemlerini kullanarak minimize eder. Sınır enerjisi iki farklı yöntem ile tanımlanabilir [37]: 1) geçici gradient(optik akış) yöntemi; 2) nesne ve arka plan alanından üretilen görünüm istatistikleri yöntemi.

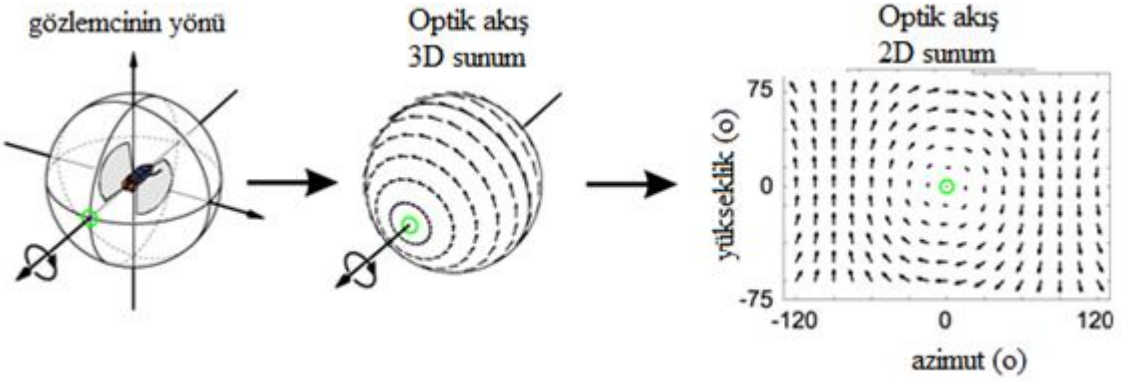
Geçici görüntü gradientini kullanarak sınır takip yöntemi, optik akış hesaplaması üzerine yapılan kapsamlı bir çalışmayla ile motive edilir. Optik akış sınırlamaları denklem (2.21) ile belirtilen sabit parlaklık sınırlaması kullanılarak üretilir.

$$I^{t+1}(x,\ y) - I^{t}(x-u,\ y-v) = 0 \tag{2.21}$$

Burada I görüntü, t zaman, (u, v) ise x ve y yönündeki optik akış vektörleridir. Optik akış yöntemine alternatif olarak, bir imgeden diğerine geçerken nesnenin iç ve dış alanları hakkında hesaplanan istatistiksel tutarlılık değeri kullanılarak nesne takibi gerçekleştirilebilir. Bu yaklaşım sınırın güncel imgede başlangıç değerlerinin belirlenmesi gerekliliğine ihtiyaç duyar.

2.4. Optik Akış Metodu

Üç boyutlu sahnede gözlemci hareket ettiğinde, görüntüde meydana gelen gözle görünür hareket, "Optik Akış" olarak adlandırılır [39]. Optik akış, görüntüdeki hareketin yön ve hızını tanımlar. Optik akış, görüş alanında nesnelerin hareketi olarak da düşünülebilir. Örneğin, bir otoyolda trafik levhasına doğru yaklaşıldığında nesne sürekli büyüyecektir ve kenarları görüş sahasının dışına hareket etme eğiliminde olacaktır. İşareti geçtikten sonra arkaya bakıldığında ise küçüldüğü ve kenarların büzüştüğü izlenir. Nesnenin görüntüsünün kenarlarının küçülüp büyümesiyle üç boyutlu uzaydan iki boyutlu düzleme izdüşümü alınmaktadır. Kenarlar hareket ettikçe görüş açısındaki diğer nesnelere göre bir hızı olacaktır. Optik akış tekniği bu hızın şiddetini ve istikametini bulma faaliyetidir. Şekil 2.23'te hareketli nesnenin etrafındaki optik akış alanı gösterilmektedir.



Şekil 2.23. Hareketli bir gözlemcinin çevresindeki optik akış vektörleri [44].

Optik akış hesaplaması için birçok teknik ileri sürülmüştür ve günümüzde hala diğerleri ortaya çıkmaya devam etmektedir. Optik akış teknikleri aşağıda bahsedilen üç gruptan birine bağlı olarak sınıflandırılabilir:

1. Fark Teknikleri: Uzay-zamansal şiddet türevlerinden görüntü hızını hesaplar.
2. Frekans-temelli Teknikler: Hız ayarlı filtre çıkışındaki enerji/faz bilgisini kullanırlar.
3. Eşleştirme Teknikleri: Az sayıda görüntüden (genellikle iki veya üç görüntü dizisinden) değişik görüntü özelliklerini eşleştirerek görüntü yer değişimlerini hesaplar.

Bu üç grup yaklaşım arasında uygulama ve performans farklılıkları olmasına karşın genelde birçok bakımdan denk olarak düşünülürler. Genel yapısı bakımından bu teknikler üç işleme aşamasına dayanarak incelenebilir:

1. Görüntü üzerinde ilgilenilen işaret yapısını elde edebilmek ve işaret/gürültü oranını arttırmak için alçak-geçiren veya bant-geçiren filtre kullanarak ön-filtreleme veya yumuşatma,
2. Hızın normal bileşenlerini hesaplamak için temel ölçümlerin hesaplanması; örneğin uzay-zamansal türevlerin elde edilmesi,
3. İki boyutlu akış alanını üretebilmek için ön filtreleme ve temel hesaplamaların entegrasyonunun yapılması.

Bu tez çalısmasında genel fark tekniği temeline dayanan Horn ve Schunck optik akış modeli ve yerel fark tekniği temeline dayanan Lucas ve Kanade optik akış modeli incelenmiştir.

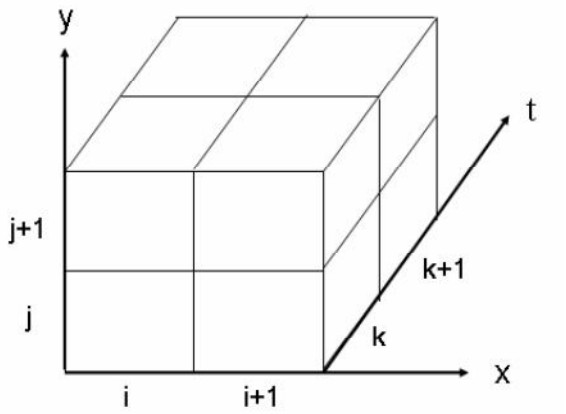
2.4.1. Optik Akış Fark Teknikleri

Optik akış, görüntü dizilerindeki yerel türevler üzerine kurulu yerel görüntü hareketi yaklaşımıdır. İki boyutlu görüntüde yerel türevler hesaplanarak her görüntü pikselinin ardışık görüntüler arasında ne kadar ilerlediği belirlenir. Optik akış, gözlenen görüntülerin uzaysal düzenlemesi ve bu düzenlemenin değişim oranı hakkında yararlı bilgiler vermektedir. Hareket eden örüntüler görüntü parlaklığında zamansal değişimlere neden olmaktadır ve tüm bu zamansal şiddet değişimlerinin sadece harekete bağlı olduğu varsayımı yapılmaktadır[40].

Hangi optik akış metodu kullanılırsa kullanılsın, her zaman görüntü şiddet türevleri hesaplanmak zorundadır. *Ix*, *Iy*, *It* şiddet türevlerinin sürekli olması hız tahmininde önemlidir. Bu nedenle, türev hesaplamaları yapılırken hepsi aynı zamanda görüntü üzerindeki aynı noktaya işaret etmelidir.

Optik akış hesabında yaklaşık fark hesabı için birçok yöntem kullanılmaktadır. Burada bahsedilen fark hesabı yöntemi, sekiz ardışık ölçüme sahip bir küpün merkez noktasındaki, *Ix*, *Iy*, *It* türev tahminlerini veren bir set kullanmaktadır.

Sekiz ardışık ölçüm arasındaki uzay ve zamansal ilişki Şekil 2.24'de gösterilmektedir. Küpün merkezindeki görüntü parlaklığının üç kısmi türevi, küpün dört paralel kenarı boyunca alınan farkların ortalamasından tahmin edilir. Kolon *j* indeksi *x* doğrultusuna, satır i indeksi *y* doğrultusuna karşılık gelir ve *k* indeksi zaman doğrultusunda ilerlemektedir.



Şekil 2.24. Türev hesaplamalarında kullanılan sekiz ardışık ölçüme sahip küp

Hesaplanan türev tahminlerinin her biri, küpteki bitişik ölçümlerden alınan dört farkın ortalamalarıdır:

$$I_x \approx \frac{1}{4}\left(I_{i,\,j+1,\,k} - I_{i,\,j,\,k} + I_{i+1,\,j+1,\,k} - I_{i+1,\,j,\,k} + I_{i,\,j+1,\,k+1} - I_{i,\,j,\,k+1} + I_{i+1,\,j+1,\,k+1} - I_{i+1,\,j,\,k+1}\right)$$

$$I_y \approx \frac{1}{4}\left(I_{i+1,\,j,\,k} - I_{i,\,j,\,k} + I_{i+1,\,j+1,\,k} - I_{i,\,j+1,\,k} + I_{i+1,\,j,\,k+1} - I_{i,\,j,\,k+1} + I_{i+1,\,j+1,\,k+1} - I_{i,\,j+1,\,k+1}\right) \quad (2.22)$$

$$I_t \approx \frac{1}{4}\left(I_{i,\,j,\,k+1} - I_{i,\,j,\,k} + I_{i+1,\,j,\,k+1} - I_{i+1,\,j,\,k} + I_{i,\,j+1,\,k+1} - I_{i,\,j+1,\,k} + I_{i+1,\,j+1,\,k+1} - I_{i+1,\,j+1,\,k}\right)$$

Görüntünün uzay ve zamansal türevlerinin maskeleri aşağıdaki şekilde yazılmaktadır:

$$\begin{array}{lll} M_x = \frac{1}{4}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} & M_y = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} & M_t = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \\ I_x = M_x^*\left(I_1 + I_2\right) & I_y = M_y^*\left(I_1 + I_2\right) & It = Mt^*\left(I_2 - I_1\right) \end{array} \quad (2.23)$$

Burada uzunluk birimi, her görüntü çerçevesindeki grid mesafesidir; zaman birimi, görüntü çerçeveleri arasındaki periyottur.

Optik akış yöntemlerinde V_x ve V_y hız bileşenlerinin Laplasiyenleri de hız tahmininde kullanılmaktadır. Laplasiyen hesabında kullanılan güvenilir bir yaklaşım aşağıdaki denklemle verilmektedir:

$$\nabla^2_{V_x} \approx K\left(\overline{V}_{x(i,\,j,\,k)} - V_{x(i,\,j,\,k)}\right) \quad ve \quad \nabla^2_{V_y} \approx K\left(\overline{V}_{y(i,\,j,\,k)} - V_{y(i,\,j,\,k)}\right) \quad (2.24)$$

Burada yer alan yerel ortalamalar, V_x ve V_y , (2.25) denkleminde belirtilmektedir:

$$\begin{aligned} \overline{V}_{x(i,\,j,\,k)} &= \frac{1}{6}\left\{V_{x(i-1,\,j,\,k)} + V_{x(i,\,j+1,\,k)} + V_{x(i+1,\,j,\,k)} + V_{x(i,\,j-1,\,k)}\right\}\cdots \\ &\quad + \frac{1}{12}\left\{V_{x(i-1,\,j-1,\,k)} + V_{x(i-1,\,j+1,\,k)} + V_{x(i+1,\,j+1,\,k)} + V_{x(i+1,\,j-1,\,k)}\right\} \\ \overline{V}_{y(i,\,j,\,k)} &= \frac{1}{6}\left\{V_{y(i-1,\,j,\,k)} + V_{y(i,\,j+1,\,k)} + V_{y(i+1,\,j,\,k)} + V_{y(i,\,j-1,\,k)}\right\}\cdots \\ &\quad + \frac{1}{12}\left\{V_{y(i-1,\,j-1,\,k)} + V_{y(i-1,\,j+1,\,k)} + V_{y(i+1,\,j+1,\,k)} + V_{y(i+1,\,j-1,\,k)}\right\} \end{aligned} \quad (2.25)$$

Hız bileşenlerinin Laplasiyen hesabı, komşu piksellerin ağırlıklı ortalamalarından bir noktadaki değerin çıkarılması ile elde edilir.

2.4.1.1. Horn ve Schunck Optik Akış Modeli

Hız alanının nedensel olarak sadece iki boyutlu şiddet bilgisinden tahmin edilmesi mümkün değildir. Nesne hareketinin görsel sensör tarafından algılanması sonucu iki

boyutlu düzlem üzerinde meydana gelen görünür hız, optik akış olarak adlandırılır ve kullanılan yöntemler ile optik akış alanı, hız alanına yakınlaştırılır[41].

Görüntü üzerinde yer alan $r=(x,\ y)$ noktasındaki parlaklığın $I=(x,\ y,\ t)$ olduğu varsayımı yapılmaktadır. Görüntüde belirli bir noktanın parlaklığı zamanla sabittir. Ayrıca nesnenin parlaklığı önceki ve sonraki görüntü çerçevelerinde sabit kalacaktır.

$$(I_x,\ I_y)*(V_x,\ V_y)=-I_t \quad veya \quad \nabla I.\vec{v}=-I_t \tag{2.26}$$

Şiddet zamana göre değişiyorsa nesnenin hareket tespiti imkansızlaşır; pikselin bir sonraki görüntü dizisindeki yeri hesaplanamaz. Şiddet değişmezliği varsayımı, "Hareket Kısıtlama" denkleminin temellerini oluşturur. Denklem (2.26).

Yerel görüntü parlaklığı yapısından tam görüntü hızı elde edilemese de "Hareket Kısıtlama" doğrusuna normal gelen görüntü hızı hesaplanabilir.

Optik akış, ek bileşenler olmadan komşu noktalardan bağımsız olarak görüntü üzerinde bir noktada hesaplanamaz; çünkü her görüntü noktasındaki hız alanı iki bileşene sahipken harekete bağlı görüntü düzlemindeki bir noktanın görüntü parlaklığındaki değişim sadece bir bileşen sağlar.

Hareket Kısıtlama denklemi hız tahmininde yetersiz kalmaktadır. Fakat şiddet örüntüsünün her noktası birbirinden bağımsız olarak hareket edebiliyorsa, hala hız tahmini iyileştirmesi yapılabilme olasılığı bulunmaktadır. Bu durumda nesne üzerinde bulunan komşu noktalar benzer hızlara sahiptir ve görüntü üzerindeki şiddet örüntülerinin hız alanı neredeyse her yerde değişmektedir. Bununla birlikte, bir nesne diger bir nesneyi kapattığında akışta süreksizlik gözlemlenir [40].

Hareket Kısıtlama denklemine getirilen ilave bileşeni tanımlananın bir yolu, bir noktadaki akış hızı ile noktayı da içeren küçük bir komşuluk üzerindeki ortalama arasındaki farkı sınırlamaktır. Eşdeğer olarak, bu sınırlamayı karşılamak için akışın *x* ve *y* bileşenlerinin toplam çerçevesel Laplasiyenleri minimize edilmektedir. *x* ve *y* yönündeki V_x ve V_y hız bileşenlerinin Laplasiyenleri aşağıdaki denklemlerle ifade edilmektedir:

$$\nabla^2 V_x=\frac{\partial V_x^2}{\partial x^2}+\frac{\partial V_x^2}{\partial y^2}$$
$$\nabla^2 V_y=\frac{\partial V_y^2}{\partial x^2}+\frac{\partial V_y^2}{\partial y^2} \tag{2.27}$$

Gözlemci düz bir nesneye paralel olarak değişirse, yüzeye dik olarak hareket eder ve sonrasında V_x ve V_y'nin ikinci kısmi türevlerinin her ikisi de kaybolur. Çünkü gerçek hız

değeri ile gözle görünür hız arasında herhangi bir sapma yoktur. Bu tür basit durumlarda, her iki denklemden elde edilen Laplasiyen sıfırdır.

Görüntü üzerindeki bir noktanın parlaklığı diğer görüntü çerçevelerinde aynı kalmaktadır. Fakat gürültü nedeni ile Hareket Kısıtlama denklemi sıfıra eşit olmayabilir. Ayrıca, hız akışı tahmininde elde edilen hızlar gerçek hız değerinden sapmış olabilir.

Bu hataları iki şekilde ele alabiliriz:

1. Görüntü parlaklığının değişim miktarı için denklemdeki hataların toplamını minimize etme:

$$E_b = I_x V_x + I_y V_y + I_t \tag{2.28}$$

2. Hız akışında yumuşatma için sapma tahminini minimize etme:

$$E_c^2 = \left(\overline{V}_x - V_x\right)^2 + \left(\overline{V}_y - V_y\right)^2 \tag{2.29}$$

Pratikte görsel sensörlerin algıladığı görüntü şiddet bilgisi, kuantalama hatası ve gürültü tarafından bozulacaktır ve bu nedenle E_b hata değerinin idealde sıfır olması beklenmez. Hız akışındaki sapma ve görüntü parlaklığındaki değişim miktarı arasındaki α^2 ağırlık faktörü ile gösterilmektedir. Şiddet bilgisinin gürültüye olan eğilimi bu faktörün belirlenmesinde önemli bir etkendir.

Sonuç olarak minimize edilmek istenen toplam hata (2.30) denkleminde verilmektedir:

$$E^2 = \alpha^2 E_c^2 + E_b^2 \tag{2.30}$$

Horn ve Schunck yöntemi, tahmin edilen hız alanı $\vec{V} = \left(V_x,\ V_y\right)$'yi sınırlamak için genel yumuşatma terimi ile gradyan kısıtlama denklemini birleştirmekte ve aşağıdaki denklemi minimize etmektedir:

$$E^2 = \int_D \left(\nabla I.\vec{V} + I_t\right)^2 + \alpha^2 \left[\left(\frac{\partial V_x}{\partial x}\right)^2 + \left(\frac{\partial V_x}{\partial y}\right)^2 + \left(\frac{\partial V_y}{\partial x}\right)^2 + \left(\frac{\partial V_y}{\partial y}\right)^2\right] dx.dy \tag{2.31}$$

D alanı görüntü boyunca tanımlanmıştır; α katsayısının genliği yumuşatma teriminin ilgili etkisini yansıtır.

a) Tekrarlama Yöntemi

Görüntü parlaklığında ve hız tahmininde karşılaşılan hataları azaltma, optik akış hız bileşenlerine $\left(V_x,\ V_y\right)$ uygun değerler bulunarak gerçekleştirilmelidir. Hata denkleminin kısmi türevi alınarak aşağıdaki (2.32) denklem elde edilir:

$$\frac{\partial E^2}{\partial V_x} = -2\alpha^2\left(\overline{V}_x - V_x\right) + 2\left(I_x V_x - I_y V_y + I_t\right) I_x$$
$$\frac{\partial E^2}{\partial V_y} = -2\alpha^2\left(\overline{V}_y - V_y\right) + 2\left(I_x V_x - I_y V_y + I_t\right) I_y \qquad (2.32)$$

Bu iki hata türevi sıfıra eşitlenerek V_x ve V_y bilşenlerine bağlı iki denklem elde edilir:

$$\left(\alpha^2 + I_x^2\right) V_x + I_x I_y V_y = \left(\alpha^2 \overline{V}_x - I_x I_t\right)$$
$$I_x I_y V_x + \left(\alpha^2 + I_y^2\right) V_y = \left(\alpha^2 \overline{V}_y - I_y I_t\right) \qquad (2.33)$$

Katsayı matrisinin determinantı, $\alpha^2\left(\alpha^2 + I_x^2 + I_y^2\right)$'a eşittir. (2.33) denklemi V_x ve V_y için çözülerek aşağıdaki denklemler elde edilir:

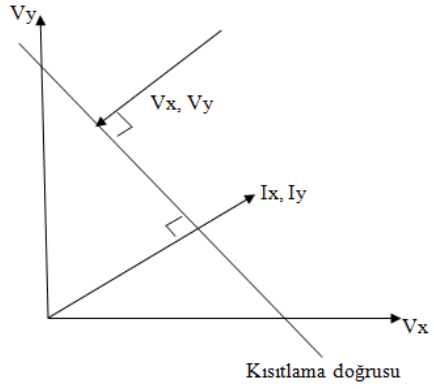
$$\left(\alpha^2 + I_x^2 + I_y^2\right) V_x = +\left(\alpha^2 + I_y^2\right)\overline{V}_x - I_x I_y \overline{V}_y - I_x I_t$$
$$\left(\alpha^2 + I_x^2 + I_y^2\right) V_y = -I_x I_y \overline{V}_x + \left(\alpha^2 + I_x^2\right)\overline{V}_y - I_y I_t \qquad (2.34)$$

Bu eşitlikler alternatif bir formda (2.35) denklemindeki şekilde yazılır:

$$\left(\alpha^2 + I_x^2 + I_y^2\right)\left(V_x - \overline{V}_x\right) = -I_x\left[I_x \overline{V}_x + I_y \overline{V}_y + I_t\right]$$
$$\left(\alpha^2 + I_x^2 + I_y^2\right)\left(V_y - \overline{V}_y\right) = -I_y\left[I_x \overline{V}_x + I_y \overline{V}_y + I_t\right] \qquad (2.35)$$

Hatayı indirgeyen akış hızı değerinin $\left(V_x \text{ ve } V_y\right)$, kısıtlama doğrusunu karşı sağ açılarda kesen doğru boyunca, kısıtlama doğrusuna karşı gelen doğrultuda bulunduğunu gösterir. Bu ilikşinin doğru üzerindeki gösterimi Şekil 2.25'te verilmektedir.

Sonuç olarak şiddet gradyanının küçük olduğu yerlerde α^2 ağırlık faktörünün önemli bir rolü olduğu görülmektedir. Bu parametre kabaca gradyanların çerçeveleri toplamının $I_x^2 + I_y^2$ tahmininde beklenen gürültü değerine eşittir.



Şekil 2.25. Kısıtlama doğrusu

Hatalar toplamını en aza indirgemek için tekrarlama denklemleri kullanılmaktadır. Horn ve Schunck yönteminde, Euler-Lagrange denklemlerini çözen Gauss-Seidel tekrarlama denklemleri ile hız akışı tespit edilmektedir:

$$V_x^{k+1} = \overline{V_x^k} - \frac{I_x\left[I_x V_x^k + I_y . V_y^k + I_t\right]}{\alpha^2 + I_x^2 + I_y^2}$$
$$V_y^{k+1} = V_y^k - \frac{I_y\left[I_x V_x^k + I_y . V_y^k + I_t\right]}{\alpha^2 + I_x^2 + I_y^2} \quad (2.36)$$

Burada k, tekrarlama sayısını; V_x^0 ve V_y^0 başlangıç hız tahminlerini (tipik olarak sıfıra ayarlanır); $\overline{V_x}$ ve $\overline{V_y}$ ise V_x^k ve V_y^k 'nın komşuluk ortalamalarını gösterir.

Yumuşatma faktörü α , pozitif bir sabittir. İki görüntü veya video dizisi arasındaki ilişkili hareket büyükse, yumuşatma faktörü için büyük bir pozitif skalar değer verilmelidir. Eğer ilişkili hareket küçükse, küçük bir pozitif skalar değer verilmelidir. Uygulamaya en iyi uyan yumuşatma faktörünü bulunması için birden fazla değer denenmesi gerekmektedir.

Ortalama hız ve elde edilen hız farkı belirli bir eşik değerinin altına indiğinde veya belirlenen bir tekrarlama sayısına ulaşıldığında, Gauss-Seidel tekrarlamasını durdurulmaktadır. Hem eşik değeri hem de tekrarlama sayısı kullanıldığı durumda ise bu koşullardan biri sağlandığında tekrarlama durur. Eşik değeri, hız tahmininde kabul edilebilir sapma miktarını belirlemektedir. Tekrarlama sayısı ise, hareket eden en büyük nesnenin kesit alanından daha büyük olmalıdır. Nesne kesitinin önceden bilinmediği durumlarda ise, tüm görüntünün kesitinin kullanılması uygun olacaktır. Birkaç farklı tekrarlama sayısı kullanılarak elde edilen sonuçlar karsılaştırıldığında, belli bir tekrarlama sayısından sonra hata değerinde küçük değişiklikler meydana geldiği görülmektedir. Bu karşılaştırmalar sayesinde daha doğru bir tekrarlama sayısı elde edilebilir.

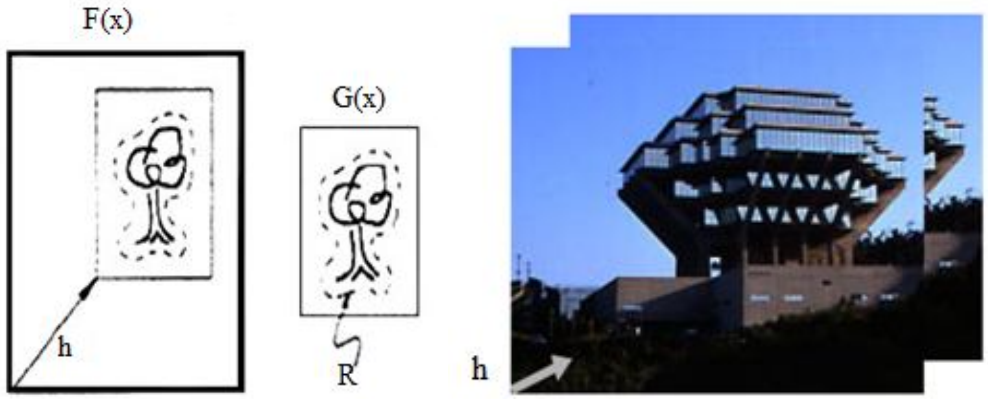
2.4.1.2. Lucas ve Kanade Optik Akış Modeli

Günümüzde Lucas-Kanade yöntemi, optik akış tahmini olarak da adlandırılan hız tahmininde iki görüntü dizisi kullanan farksal tekniklerinden biri olarak kullanılmaya devam etmektedir [39,42].

a) Kaydetme Problemi

Bilgisayarla görmede, aynı sahneyi veya nesneyi farklı zamanlarda veya perspektiflerde ve farklı koordinat sistemlerinde örnekleyerek veri setleri oluşturulur. Görüntü kaydetme, farklı veri setlerini tek bir koordinat sistemine toplama işlemidir. Farklı ölçümlerden alınan verilerin karşılaştırması veya birleştirilmesi için kayıt gereklidir.

Görüntü dizilerinde karsılaşılan kaydetme problemi örnekle şu şekilde tanımlanmaktadır; x 'in vektör olarak belirlendiği iki ardışık görüntü dizisi olarak $F(x)$ ve $G(x)$ fonksiyonları verilir. R ilgilenilen alan içerisindeki x için $F(x+h)$ ve $G(x)$ arasındaki fark hesabını küçülten fark vektörü h bulunmaya çalışılmaktadır. Durum şekil 2.26' da gösterilmektedir.



(a) (b)

Şekil 2.26. (a), (b) Görüntü kaydetme problemi

$F(x+h)$ ve $G(x)$ arasındaki tipik fark hesaplamaları şu şekildedir:

$$L1\ normu = \sum_{x \in R} \left| F(x+h) - G(x) \right|$$

$$L2\ normu = \sum_{x \in R} \left(\left[F(x+h) - G(x) \right]^2 \right)^{\frac{1}{2}} \qquad (2.37)$$

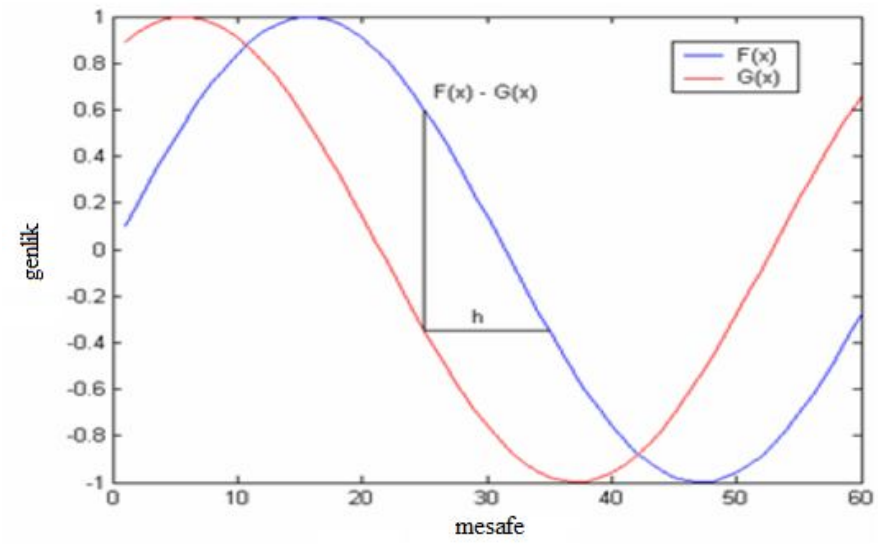
$L1$ normu, $L2$ normuna basit bir yaklaşım olarak kullanımı açısından kolaylık sağlamaktadır.

b) Kaydetme Algoritması

Öncelikle bir boyutlu kaydetme problemi için bir yaklaşım geliştirilmektedir. Daha sonra çok boyutlu görüntüler için tek boyutlu durumdan hareketle alternatif bir çözüm sunulmaktadır.

1) Tek Boyutlu Durum

Tek boyutlu kaydetme probleminde, iki eğri arasındaki (F(x) ve G(x)=F(x+h)) şekil 2.27' de yatay fark vektörü h bulunmak istenmektedir.



Şekil 2.27. Eşleştirilen iki eğri

Bu problemin çözümü, x komşuluğundaki $F(x)$'in davranışına olan lineer yaklaşıma dayanmaktadır. Küçük bir h değişimi için ayrık zamanlı türev (2.38) yaklaşıklığı ile hesaplanmaktadır:

$$F'(x) \approx \frac{F(x+h)-F(x)}{h} = \frac{G(x)-F(x)}{h} \tag{2.38}$$

$$h \approx \frac{G(x)-F(x)}{F'(x)} \tag{2.39}$$

ve buradan h değeri yaklaşıklığın sol tarafına alınırsa:

elde edilir. Bu algoritmanın başarılı olabilmesi için, bu yaklaşımın yeterli olmasını sağlayacak kadar h değerinin küçük olması istenmektedir. (2.39) yaklaşıklığında verilen *h* yaklaşımı, *x* bileşenine bağlıdır. *x* bileşeninin değişik değerlerindeki *h* değerleri tahminlerini birleştirmek için basit bir şekilde ortalamaları alınmaktadır:

$$h \approx \frac{\sum_{x} \frac{G(x)-F(x)}{F'(x)}}{\sum_{x} 1} \tag{2.40}$$

Giriş görüntüsü lineer bir fonksiyon olduğunda, (2.38) yaklaşıklığı ile verilen lineer yaklaşım doğru sonuçlar vermektedir. Tam tersine giriş görüntüsünün ikinci dereceden türevinin $|F''(x)|$ büyük olduğu yerlerde kötü sonuç verdiği kabulü yapılırsa (2.40) yaklaşıklığı ile elde edilen ortalama geliştirilir. (2.40) yaklaşıklığındaki ortalamaya, $|F''(x)|$ tahmininin her noktaya olan etkisi ters orantılı olarak ilave edilir.

$|F''(x)|$ tahmini (2.41) yaklaşıklığı ile gösterilmektedir:

$$F''(x) \approx \frac{G'(x)-F'(x)}{h} \tag{2.41}$$

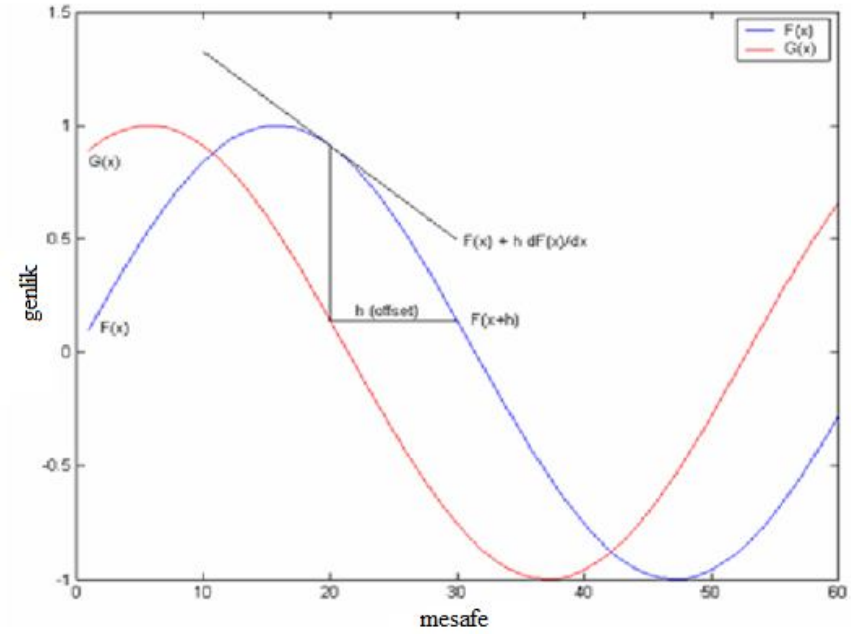
(2.41) yaklaşıklığından elde edilen $\frac{1}{h}$'ın sabit faktörü, (2.40) yaklaşıklığında ağırlık fonksiyonu olarak kullanılmaktadır:

$$W(x) = \frac{1}{G'(x)-F'(x)} \tag{2.42}$$

Sonuç olarak $|F''(x)|$ yaklaşıklığı ile verilen ağırlık fonksiyonu bu bölümde bahsedilen sezgisel yaklaşımı karşılamaktadır. Örnegin, Şekil 2.28'de gösterilen iki eğrinin kesiştiği noktada (2.39) yaklaşıklığı ile sağlanan *h* değeri "0"a eşittir ve bu durum hız tahmininde istenmeyen bir durumdur. Bununla birlikte, eğrilerin kesişim noktasında $F'(x)$ ve $G'(x)$ arasındaki fark büyük olduğu için ortalamada bu tahmine karşılık gelen ağırlık küçük olacaktır.

(2.42) ’de elde edilen ağırlık faktörünün h hesaplamasına etkisi, (2.43) yaklaşıklığı ile tekrar düzenlenmektedir:

$$h \approx \frac{\sum_x \frac{W(x)\left[G(x)-F(x)\right]}{F'(x)}}{\sum_x W(x)} \tag{2.43}$$



Şekil 2.28. Eşleştirilen eğrilerin h mesafesine göre değişimleri

h yaklaşımı elde edildikten sonra görüntü $F(x)$ hareket ettirilir ve bu işlem Newton-Raphson tekrarlaması ile tekrar edilir. İdeal durumlarda sağlanan h tahmin dizileri, en iyi h değeri ile kesişecektir. *h* tahmininde kullanılan tekrarlama aşağıda (2.44) denklemi ile verilmektedir:

$$h_0 = 0,$$

$$h_{k+1} = h_k + \frac{\sum_x \frac{W(x)\left[G(x)-F(x+h_k)\right]}{F'(x+h_k)}}{\sum_x W(x)} \tag{2.44}$$

Burada *k* tekrarlama sayısını ifade etmektedir.

2) Alternatif Yaklaşım

İki boyutlu görüntü dizilerinde önceki bölümde anlatılan h tahmini için kullanılan tekrarlama ile istenilen sonuçlar elde edilemez. Bunun nedeni, iki boyutlu görüntülerin lineer yaklaşımı ve tek boyutlu işaretin lineer yaklaşımı farklı biçimlerde meydana gelmesidir. Örnegin, (2.39) yaklaşıklığında $F'(x)$'in "0" olduğu durumlarda eğim seviyesi belirlenemeyecektir. Bu problemler (2.38) yaklaşıklığının lineer yaklaşımı ile düzeltilebilir:

$$F(x+h) \approx F(x) + hF'(x) \tag{2.45}$$

Hareket değişim mesafesi h'a bağlı olarak hatayı minimize etmek için hatanın h bileşenine göre türevi alınır:

$$h \approx \frac{\sum_x F'(x)\left[G(x) - F(x)\right]}{\sum_x F'(x)^2} \tag{2.46}$$

ve

$$\begin{aligned} 0 &= \frac{dE}{dh} \\ &\approx \frac{d}{dh}\sum_x \left[F(x) + hF'(x) - G(x)\right]^2 \\ &= \sum_x 2F'(x)\left[F(x) + hF'(x) - G(x)\right] \end{aligned} \tag{2.47}$$

(2.47) yaklşıklığı ile elde edilen h bileşeni aslında (2.43)'de bahsedilen h yaklaşıklığı ile aynıdır; sadece burada belirlenen ağırlık fonksiyonu $w(x) = F'(x)^2$'dir. Özetle bu bölümde elde edilen lineer yaklaşım yardımıyla iki veya daha çok boyutlarda görüntü hareketi tespiti yapılmaktadır. Ayrıca bu yaklaşıklık sayesinde bölen paydanın "0" olması probleminden kaçınılmış olunur; sadece $F'(x)$'in görüntünün her yerinde sıfır olduğu durumda yaklaşıklık bir sonuç vermeyecektir. (2.39) yaklaşıklığında ise $F'(x)$'in herhangi bir yerde "0"a eşit olması durumunda bir sonuç alınamayacaktır.

$$h_0 = 0$$

$$h_{k+1} = h_k + \frac{\sum_x w(x) F'(x+h_k)\left[G(x) - F(x+h_k)\right]}{\sum_x w(x) F'(x+h_k)^2} \tag{2.48}$$

Burada $w(x)$ ağırlık faktörü (2.39) yaklaşıklığında verildiği şekliyle kullanılmaktadır.

3) Performans

Bu bölümde h_k, s dizilerinin hangi koşullar altında ve ne kadar hızlı gerçek h değerinde birleştiği değerlendirilmektedir. Örnek olarak aşağıda (2.49) ile belirtilen durum ele alınmaktadır:

$$F(x) = \sin(x)$$
$$G(x) = F(x+h) = \sin(x+h) \tag{2.49}$$

Her iki kaydetme algoritması da doğru h değerine $|h| < \pi$ 'den önce ulaşacaktır. Yanlış başlangıç kaydetmeleri için en kötü durumda doğru h değerine ulaşma süresi yarım dalga boyu kadar büyük olacaktır. Görüntü yumuşatılarak yüksek uzaysal frekanslar batırılarak doğru h değeri ile kesişme süresi geliştirilir. Örneğin, yumuşatma uygulanarak görüntüdeki her piksel komşu piksellerdeki ağırlıklı ortalamalar ile değiştirilir. Fakat yumuşatmanın kötü bir tarafı ise küçük detayları söndürmesidir ve bu nedenle iki görüntü çerçevesi arasında yapılan eşleştirme daha az doğru olacaktır. Ayrıca eşleştirilmeye çalışılan nesnenin boyutu yumuşatma penceresinden daha küçük ise nesne tamamen söndürülür ve hiçbir eşleşme mümkün olmaz [41].

Alçak geçiren filtre ile yumuşatma sonucunda elde edilen görüntü bilgi kaybı olmadan düşük çözünürlükte örneklenebilir. Bu düşünceden hareketle coarse-fine stratejisi seçilerek daha doğru tahminler elde edilmektedir. Coarse-fine yöntemi, yaklaşık bir eşleştirme elde edebilmek için görüntünün daha düşük çözünürlüklü yumuşatılmış versiyonlarını kullanmaktadır.

Yumuşatma etkisi doğru h değeri ile kesişme aralığını genişletirken, ağırlık fonksiyonu yaklaşımın doğruluğunu arttırmak için çalışmaktadır ve bu nedenle kesişme süresini azalmaktadır. Ağırlık fonksiyonu kullanılmadığı durumda, diğer bir anlamda

$w(x)=1$ için $f(x)=\sin(x)$ giriş işaretine sahip (2.48) denkleminin ilk tekrarında hesaplanan h_1 farkı, mesafe yarım dalga boyuna ulaşır ulaşmaz sıfıra düşer. Bununla birlikte, (2.42) denklemindeki $w(x)$ ağırlığına sahip fark hesabının doğruluğu daha yüksektir ve h farkı, yarım dalga boyuna çok yakın sıfıra düser. Bu nedenle (2.42) denklemindeki gibi bir $w(x)$ ağırlık faktörüne sahip kesişimler büyük mesafeler için daha hızlı sonuç verecektir.

4) Uygulama

(2.48) denkleminin uygulanabilmesi için görüntü üzerinde R ilgilenilen alan boyunca $F'G$, $F'F$ ve $(F')^2$ değerlerinin ağırlıklı ortalamaları hesaplanması zorunludur. $F'(x)$ değerinin tam olarak hesaplanması mümkün değildir. Fakat Lucas ve Kanade algoritmasının uygulanabilirliği için $F'(x)$ aşağıdaki yaklaşıklık kullanılarak tahmin edilmektedir:

$$F'(x)=\frac{F(x+\nabla x)-F(x)}{\nabla x} \tag{2.50}$$

ve benzer şekilde $G'(x)$'i de bu şekilde hesaplanır. ∇x için küçük bir değer seçilir (yaklaşık 1 piksel). Birinci türevlerin hesaplaması için bazı geliştirilmiş yöntemler bulunmaktadır fakat genelde bu tür yöntemler ilk yumuşatma fonksiyonuna eşittir.

5) Çok Boyutlu Gerçekleme

(2.48) denklemi ile belirtilen tek boyutlu algoritma iki ve ikiden daha fazla boyutlar için de gerçeklenebilir. Genel anlamda amaç hata hesabı $L2$ normunu minimize etmektir:

$$E=\sum_{x\in R}\left[F(x+h)-G(x)\right]^2 \tag{2.51}$$

Burada x ve h, n-boyutlu satır vektörleridir. (2.45) yaklaşıklığına benzer şekilde aşağıdaki lineer yaklaşım yapılır:

$$F(x+h)\approx F(x)+h\frac{d}{dx}F(x) \tag{2.52}$$

$\frac{d}{dx}$, burada x'e bağlı gradyan operatörüdür. $\frac{d}{dx}$'in kolon vektörü gösterimi (2.53)'de gösterilmistir:

$$\frac{d}{dx} = \left[\frac{d}{dx_1} \frac{d}{dx_2}, \ldots, \frac{d}{dx_n} \right]^T \tag{2.53}$$

Bu yaklaşım kullanılarak, hata hesabı $L2$ normunu minimize etmek için, aşağıdaki değerleri atanmaktadır:

$$h \approx \left[\sum_x \left(\frac{dF}{dx} \right)^T \left[G(x) - F(x) \right] \right] \left[\sum_x \left(\frac{dF}{dx} \right)^T \left(\frac{dF}{dx} \right) \right]^{-1} \tag{2.54}$$

ve

$$\begin{aligned} 0 = \frac{dE}{dh} &\approx \sum_x \left[F(x) + h \frac{dF}{dx}(x) - G(x) \right]^2 \\ &= \sum_x 2 \frac{dF}{dx}(x) \left[F(x) + h \frac{dF}{dx}(x) - G(x) \right] \end{aligned} \tag{2.55}$$

Tek boyutlu durum hesaplamaları ilgili olarak önceki bölümlerde bahsedilen tekrarlama, ağırlıklandırma, yumuşatma ve coarse-fine teknikleri, n-boyutlu durumlara da aynı şekilde uygulanmaktadır. İki boyutlu durumda h tahmini için beş çarpımın $\left((G-F)F(x),\ (G-F)F_y,\ F_y^2,\ F_x^2 \ ve\ F_x F_y \right)$ ağırlıklı ortalamasının görüntü üzerinde R ilgilenilen alan boyunca toplanması zorunludur.

c) Minimize Etme

Lucas-Kanade fark yönteminde olduğu gibi bu tür farksal yöntemler, $t\ ve\ t+\delta t$ anlarında alınan iki görüntü çerçevesi arasındaki hareketi her piksel konumunda hesaplamaya çalışmaktadır. Bu yöntemler görüntü işaretinin yerel Taylor serileri yaklaşımına dayandığı için farksal olarak adlandırılırlar; diğer bir anlamda uzaysal ve zamansal koordinatlara göre kısmi türevleri kullanırlar [43]. $I(x, y, t)$ parlaklığına sahip (x, y, t) konumundaki bir piksel, iki çerçeve arasında $\delta x, \delta y, \delta t$ ile hareket ederse, aşağıdaki "görüntü kısıtlama denklemi" yazılabilir:

$$I(x,\ y,\ t) = I(x+\delta x,\ \ y+\delta y,\ \ t+\delta t) \tag{2.56}$$

Hareketin küçük olduğu varsayılırsa, $I(x,\ y,\ t)$'deki görüntü bileşeni Taylor serileri ile aşağıdaki denklemle genişletilebilir:

$$I(x+\delta x,\ \ y+\delta y,\ \ t+\delta t) = I(x,\ y,\ t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + YDT \tag{2.57}$$

Burada YDT, yüksek dereceden terimler anlamına gelmektedir ve bu terimler ihmal edilebilecek kadar küçüktür. Bu eşitliklerin devamında aşağıdaki sonuçlar elde edilmektedir:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0 \tag{2.58}$$

veya

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t}\frac{dt}{dt} = 0$$

Bunun sonucu olarak hız bileşenlerini içeren denklem elde edilir:

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \tag{2.59}$$

Burada V_x ve V_y; $I(x,\ y,\ t)$'nin optik akışı veya hızın x ve y bileşenleridir ve $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ ve $\frac{\partial I}{\partial t}$; $(x,\ y,\ t)$ pikselindeki görüntünün ilişkin yönlerdeki türevleridir. (2.59) denklemindeki türevler yerine I_x, I_y ve I_t bileşenleri yazılabilir:

$$I_x V_x + I_y V_y = -I_t \tag{2.60}$$

veya

$$\nabla I.\vec{v} + I_t = 0 \tag{2.61}$$

Bu eşitlik iki bilinmeyenlidir ve bu şekilde çözülemez. Bu durum optik akış algoritmalarının açıklık problemi olarak bilinmektedir. Optik akışı bulabilmek için bazı ek bileşenler veren başka bir eşitlik seti gerekmektedir. Lucas ve Kanade tarafından elde edilen çözüm, yerel olarak sabit bir akış kabulü yapan yinelemesiz bir yöntemdir.

$m \times m$ boyutundaki küçük bir pencere içerisinde $(m > 1)$, merkezi $(x,\ y)$ pikselinde yer alan akışın $(v_x,\ v_y)$ sabit olduğu kabulü ile bir denklem seti bulunabilir:

$$\begin{aligned} &I_{x1}v_x + I_{y1}v_y = -I_{t1} \\ &I_{x2}v_x + I_{y2}v_y = -I_{t2} \\ &\vdots \\ &I_{xn}v_x + I_{yn}v_y = -I_{tn} \end{aligned} \tag{2.62}$$

İki bilinmeyen için iki denklemden daha fazla denklem bulunduğundan sistem aşırı-kararlıdır. Bu nedenle aşağıdaki ifade yazılabilir:

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix} \tag{2.63}$$

veya

$$A\vec{v} = -b$$

Aşırı-kararlı sistemin denklemlerini çözmek için, Lucas ve Kanade optik akış tahmininde "en küçük kareler" yöntemi kullanılır:

$A^t A\vec{v} = A^t(-b)$ veya

$\vec{v} = \left(A^t A\right)^{-1} A^t(-b)$ veya

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum I_{xi}^2 & \sum I_{xi}I_{yi} \\ \sum I_{xi}I_{yi} & \sum I_{yi}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_{xi}I_{ti} \\ -\sum I_{yi}I_{ti} \end{bmatrix} \tag{2.64}$$

Bu denklemde toplamlar i = 1'den n'e kadar çalıştırılmaktadır.

Bunun anlamı, tüm üç boyut içerisinde görüntü türevleri hesaplanarak optik akış bulunabilir. $i,\ j \in [1,...,m]$ bileşenlerine sahip $W(i,\ j)$ ağırlık fonksiyonu eklenerek pencere merkezine daha fazla önem verilebilir. Bu amaçla Gauss fonksiyonları tercih edilir. Diğer fonksiyonlar veya ağırlık şemaları da kullanılmaktadır. Ayrıca yerel çevrimleri hesaplamada, görüntü bozulmalarını benzeştirmek için de optik akış modeli genişletilebilir.

Görüntü üzerinde Ω penceresi küçültülerek her küçük uzaysal komşudaki $\vec{v}$ için sabit bir modele yerel birinci dereceden bileşenlerin ağırlıklı en küçük çerçeveleri uygulanmaktadır:

$$E(v_x,\ v_y) = \sum_{x,\ y \in \Omega} W^2(x,\ y)\left[\nabla I(x,\ y,\ t).\vec{v} + I_t(x,\ y,\ t)\right]^2 \tag{2.65}$$

Yukarıdaki denklemin çözümü şu şekilde verilir:

$$\vec{v} = \left[A^t W^2 A\right]^{-1} A^t W^2 \vec{b} \tag{2.66}$$

N piksel için $(n \times n$ komşuluk, $N = n^2)$, t anında $(x_i,\ y_i) \in \Omega$ kabulü ile yukarıdaki denklemin çözümünde gerekli bileşenler aşağıdaki eşitliklerle hesaplanmaktadır:

$$\begin{aligned} A &= \left[\nabla I(x_1,\ y_1),\ldots,\nabla I(x_N,\ y_N)\right], \\ W &= diag\left[W(x_1,\ y_1),\ldots,W(x_N,\ y_N)\right], \\ b &= -\left(I_t(x_1,\ y_1),\ldots,I_t(x_N,\ y_N)\right) \end{aligned} \tag{2.67}$$

$A^T W^2 A$ tek olmayan matris ise denklemi kapalı formda çözülebilir, $A^T W^2 A$, 2x2'lik matristir:

$$A^T W^2 A = \begin{bmatrix} \sum W^2(x,\ y) I_x^2(x,\ y) & \sum W^2(x,\ y) I_x(x,\ y) I_y(x,\ y) \\ \sum W^2(x,\ y) I_y(x,\ y) I_x(x,\ y) & \sum W^2(x,\ y) I_y^2(x,\ y) \end{bmatrix} \tag{2.68}$$

Burada tüm toplamlar Ω komşuluğunda (x, y) pikseli boyunca alınmaktadır.

Stereo eşleştirmede olduğu gibi görüntü kaydetme uygulandığında, Lucas-Kanade metodu genellikle coarse-fine tekrarlama davranışına taşınır, bu tür bir yolda ilk olarak uzaysal türevler ölçek-uzayında (veya piramit) coarse ölçeğinde hesaplanır, görüntülerden biri hesaplanan bozulma ile çevrilir ve daha sonra tekrarlama güncellemeleri liner ölçeklerde başarılı bir şekilde hesaplanır.

Lucas-Kanade algoritması ve diğer yerel optik akış algoritmalarının karakteristiklerinden biri de çok yüksek akış vektörü yoğunluğu sağlamamasıdır; bir diğer değişle akış bilgisi, hareket sınırları boyunca hızlıca kaybolur ve büyük homojen alanların iç bölümleri küçük hareketleri gösterir. Bunun avantajı, gürültü varlığında karşılaştırılmalı sağlamlıktır.

2.5. Ortalama Değer Kayması Metodu

Ortalama değer kayması algoritması parametrik olmayan istatistiksel bir yöntemdir ve nokta dağılımının en yoğun olduğu yeri bulmayı sağlar. Ortalama değer kayma yöntemi, Fukunaga ve Hostetler tarafından ilk kez ortaya çıkarılmış parametrik olmayan bir öbekleme yöntemidir [45]. Yöntem, Cheng'in makalesine [46] kadar ilgi görmemiştir ve üstün özelliklerine rağmen istatistiksel literatürde de görülememiştir. Ancak yöntem, daha sonraları kesimleme ve renkli nesne takibi uygulamalarında kullanılarak tekrar gündeme gelmiştir [46-47].

Ortalama değer kayması algoritması, hedefin istatistiksel bilgilerine dayanarak renk, doku gibi görsel nitelikleri verimli bir şekilde takip eder. Bu yöntem birden çok (farklı renkli bölgelere sahip) hedefin, örneğin manzarayı, yüzü, insanları, gerçek zamanlı olarak ve hedeflerin kısmen kaybolması durumda bile sağlam bir takip sağlamaktadır. Aktif görüntüdeki hedefin koordinatı ile sonraki görüntü çerçevesindeki hedefin özelliklerine benzer en yakın bölgenin koordinatının belirlenmesine dayanır.

Buna göre:

$\{x_i(j+1)\}_{i=1\cdots n}(j+1)$ görüntüdeki piksellerin konumu,

$\{q_k(j+1)\}_{k=1\cdots m,\ m=1\sim 256}(j+1)$ görüntüdeki m adet aralıktan oluşan renk histogramı,

$\{p_k\}_{k=1\cdots m}$ m adet aralıktan oluşan hedefe ait renk histogramı.

Ayrıca $x_i(j+1)$ konumundaki piksel ile ilgili olarak $c[x_i(j+1)]$ indeksinin o pikselin rengine ait histogram aralığını belirtecek olan $c: R^2 \to \{1\cdots m\}$ fonksiyonu da tanımlanmış olsun. Hedefin varlığı için ölçüt aşağıdaki gibi tanımlanan orantı histogramıdır.

$$r_k = \left\{ \frac{p_k}{q_{k(j+1)}}, 1 \right\} : \{r_k\}_{k=1\cdots m} \tag{2.69}$$

Orantı histogramının $(j+1)$ görüntüdeki izdüşümü $r_c[x_i(j+1)]$ değerini tüm $i = 1\cdots n$ için $x_i(j+1)$ pikseli ile ilişkilendirir. Orantı histogramı hedefe ait baskın rengi vurgulayıp, dağınık ve arka plan renklerinin varlığını azalttığı için izdüşüm görüntüsü $\{r_c[x_i(j+1)]\}_{i=1\cdots n}$ hedefin varlığının uzaysal ölçütünü gösterir. İzdüşüm görüntüsündeki değerleri her piksel konumundaki ağırlık değeri olarak alırsak, hedef takip işlemi izdüşüm görüntüsündeki yerel modun konumunu belirleme işlemine dönecektir.

Görüntü koordinatları *x* ve *y*'nin (en ve boy olarak) *hx* ve *hy* ile normalize edildiğini varsayalım. Bu, ortalama kayma takip çalışmasında pencere yarıçapı olarak *h* = 1 kullanılmasını olanaklı kılar. Burada *hx* ve *hy* hedefi sınırlayan pencereye ait *x* ve *y* koordinatlarındaki yarıçaplardır. Ortalama kayma takip algoritması aşağıdaki aşamalardan oluşmaktadır.

Öncelikle *j*. Görüntüde takip edilecek hedefin konumu $\hat{y}(j)$ belirlenir, *m* adet aralıktan oluşan hedefe ait renk histogramı $\{p_k\}_{k=1\cdots m}$ hesaplanır ve sonraki görüntü $((j+1).\text{ görüntü})$ için:

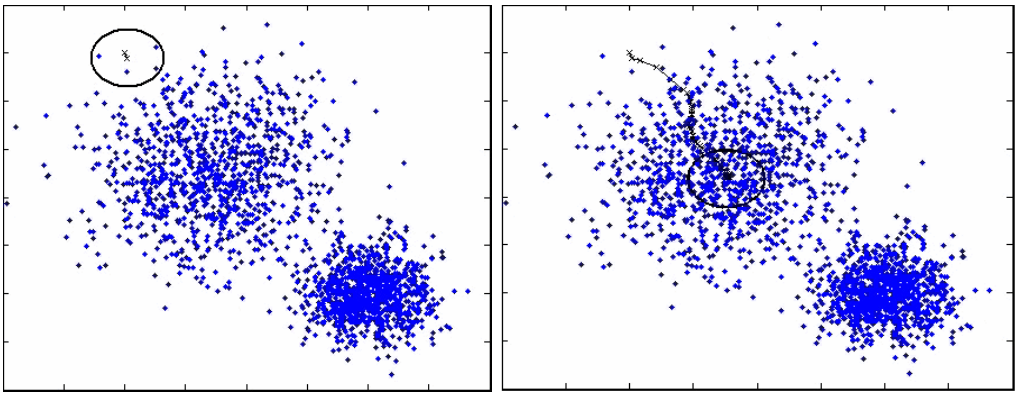
- ✓ m adet aralıktan oluşan renk histogramı $\{q_k\}_{k=1\cdots m}$ hesaplanır.
- ✓ orantı histogramı $\{r_k\}_{k=1\cdots m}$ hesaplanır.
- ✓ İzdüşüm görüntüsü $\{r_c[x_i(j+1)]\}_{i=1\cdots n}$ üretilir.
- ✓ $\hat{y}_1(j+1) = \hat{y}(j)$ ve $u = 1$ alarak ortalama kayma iterasyonlarına başlanır.

Yakınsayana kadar aşağıdaki hesaplama yapılır:

$$y_{u+1}(j+1)=\frac{\sum_{i=1}^{n} x_i(j+1)\, r_c\left[x_i(j+1)\right] K\left(\left\|\widehat{y_u}(j+1)-x_i(j+1)\right\|^2\right)}{\sum_{i=1}^{n} r_c\left[x_i(j+1)\right] K\left(\left\|\widehat{y_u}(j+1)-x_i(j+1)\right\|^2\right)}, \qquad (2.70)$$

$$u \leftarrow u+1$$

İterasyon sırasında hesaplanan yeni koordinata pencere taşınır. Ve tekrar hesaplama yapılır. Ardışık iki hesaplama arasındaki fark, belirli bir hata değerinin (toleransın) altına düştüğünde hesaplama bırakılır, yakınsama sağlanmış olur[48]. Şekil 2.29'da ortalama değer kayması yöntemi ile hareketli nesnenin izleyeceği yol tahmin ediliyor.



Şekil 2.29. Ortalama Değer kayması ile hedef nesnenin izlendiği yol

Ortalama değer kayması algoritmasının değişik versiyonları nesne ve ortam renkleri faklı iken güvenli bir yöntem olarak kullanılsa da nesne ve nesnenin bulunduğu ortamı temsil eden benzer olasılık dağılımları ortaya çıktığında yetersiz ve sınırlı kalmaktadır. Oluşan bu olumsuzluğu çözmek için kernel yoğunluk tahmini yöntemi kullanılır.

2.5.1. Kernel Yoğunluk Tahmini

Kernel yoğunluk tahmini en çok kullanılan yoğunluk tahmini yöntemidir. Verilen n adet veri noktası $\{x_i\}_{i=1\cdots n}$ ler de boyutlu öklit uzayı R^d 'de rastgele noktalar olmak üzere, $K(x)$ kerneli ile x noktası için hesaplanan çok değişkenli kernel yoğunluk tahmini şöyle tanımlanır:

$$\hat{f}(x)=\frac{1}{n\,h^d}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right) \qquad (2.71)$$

Kernel yoğunluk tahmininin kalitesi yoğunluk ve tahmin edilen yoğunluk çerçevesel hatanın ortalaması ile ölçülür. Pratikte, bununla birlikte, bu ölçütün sadece asimptotik yaklaşımı ölçülebilir. En optimum kernel, ortalama minimum toplam çerçeve hatası veren Epanechnikov kerneldir ve profili denklem (2.72) eşitliğindeki gibi verilen ve denklem (2.73) eşitliğindeki gibi yarıçap olarak simetrik bir kerneldir.

$$k_E(x) = \begin{cases} 1-x & 0 \le x \le 1 \\ 0 & x > 1 \end{cases} \tag{2.72}$$

$$KE(x) = \begin{cases} \frac{1}{2} c_d^{-1} (d+2)\left(1 - \|x\|^2\right) & \|x\|^2 \le 1 \\ 0 & \text{diğ } d \end{cases} \tag{2.73}$$

(2.73) eşitliğinde c_d d boyutlu alanın birim hacmidir. Örnek vermek gerekirse; $c_1 = 2,\ c_2 = \pi,\ c_3 = \frac{4\pi}{3}.$

Epannechnikov kerneli sınırlarda türevlenemez ve (2.74) denklemindeki profil, (2.75) denklemindeki çok değişkenli normal kernel sonuçlanır.

$$k_{N(x)} = \exp\left(-\frac{1}{2}x\right) \quad x \ge 0 \tag{2.74}$$

$$K_{N(x)} = (2\pi)^{-\frac{d}{2}} \exp\left(-\frac{1}{2}\|x\|^2\right) \tag{2.75}$$

Profil temsili de uygulnarak (2.71) eşitliğindeki yoğunluk tahmincisi tekrar aşağıdaki gibi yazılabilir.

$$\hat{f}_{h,K}(x) = \frac{c_{k,d}}{n\, h^d} \sum_{i=1}^{n} K\left(\left\|\frac{x - x_i}{h}\right\|^2\right) \tag{2.76}$$

Burada $f(x)$ yoğunluk fonksiyonu ile tanımlanan bir nitelik uzayının analizindeki ilk adım bu yoğunluğun modlarını belirlemektedir. Modlar gradyentin sıfır noktalarında $\left(\nabla f(x) = 0\right)$ bulunmaktadırlar ve ortalama kayma prosedürü yoğunluk tahminini hesaplamaya gerek kalmadan bu sıfır noktalarını bulabilen tek yöntemdir.

(2.76) denkleminde linerliğinden faydalanarak yoğunluk gradyentinin tahmini, yoğunluk tahmininin gradyenti olarak ifade edilebilir.

$$\hat{\nabla} f_{h,K}(x) = \nabla \hat{f}_{h,K}(x) = \frac{2c_{k,d}}{n\, h^{d+2}} \sum_{i=1}^{n} (x - x_i)\, k'\left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \tag{2.77}$$

Kernel profili k'nın tüm $x \in [0, \infty]$ için türevlenebilir olduğu varsayılarak aşağıdaki gibi bir fonksiyon tanımlanabilir.

$$g(x) = -k'(x) \tag{2.78}$$

(2.78) denkleminde tanımlanan $g(x)$ profiline sahip $G(x)$ kerneli ise şu şekilde tanımlanır;

$$G(x) = c_{g,d}\, g\left(\|x\|^2 \right) \tag{2.79}$$

(2.77) denkleminde $g(x)$ yerine yazılırsa:

$$\begin{aligned} \hat{\nabla} f_{h,K}(x) &= \frac{2c_{k,d}}{n\, h^{d+2}} \sum_{i=1}^{n} (x - x_i)\, g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \\ &= \frac{2c_{k,d}}{n\, h^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \right] \left[\frac{\sum_{i=1}^{n} x_i\, g\left(\left\| \frac{x - x_i}{h} \right\| \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right)} - x \right] \end{aligned} \tag{2.80}$$

(2.80) denklemi elde edilir. Burada $\sum_{i=1}^{n} g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right)$ ifadesi pozitif bir sayı olarak kabul edilir ve pratikte de bu şart kolaylıkla sağlanabilir. (2.80) denklemindeki her iki çarpım da önem taşımaktadır. İlk terim (2.77) denklemindeki x noktası için G kerneli ile tahmin edilen yoğunlukla ilişkilidir.

$$\hat{f}_{h,G}(x) = \frac{c_{g,d}}{n\, h^{d}} \sum_{i=1}^{n} g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \tag{2.81}$$

İkinci terim ise ortalama kayma yani kernel merkezi x ile G kerneli kullanılarak elde edilen ağırlıklı ortalama arasındaki faklılıktır.

$$m_{h,G}(x) = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x \tag{2.82}$$

(2.81) ve (2.82) denklemleri yerlerine yazılırsa (2.80) denklemi şu şekle döner;

$$\hat{\nabla} f_{h,K}(x) \equiv \hat{f}_{h,G}(x) = \frac{2c_{k,d}}{h^2 c_{g,d}} m_{h,G}(x) \tag{2.83}$$

Sonuç olarak ortalama kayma vektörü şu şekilde hesaplanır.

$$m_{h,G}(x) = \frac{1}{2} h^2 c \frac{\hat{\nabla} f_{h,K}(x)}{\hat{f}\, h, G(x)} \tag{2.84}$$

(2.84) ifadesinden de görüleceği gibi x konumunda G kerneli ile hesaplanan ortalama kayma vektörü, K kerneli ile elde edilen normalize edilmiş yoğunluk gradyentinin tahmini ile orantılıdır. Normalizasyon, G kerneli ile hesaplanır ve bu yüzden ortalama kayma vektörü daima yoğunluktaki maksimum artışa işaret eder. Ortalama kayma yerel gradyent tahmini ile ilgili olduğundan tahmin edilen yoğunluğun sabit noktasına doğru bir yol tanımlayabilir. Yoğunluğun modları ise böyle sabit noktalardır. Ortalama kayma algoritması ardışık olarak aşağıdaki işlemlerin yapılması ile gerçekleşir:

- $m_{h,G}(x)$ Ortalama kayma vektörünün hesaplanması
- Kernel $G(x)$'in ortalama kayma vektörü ile tekrar hesaplanması

Bu işlem, yoğunluk tahminin gradyentinin sıfır olduğu noktaların yakınında yakınsamayı garanti eder. Düşük yoğunluk değerlerine sahip bölgeler nitelik uzayı analizinde önemsenmeyecek olan alanlardır ve böyle alanlarda ortalama kayma adımları büyük olur. Benzer şekilde yerel maksimum noktaları yakınında ise ortalama kayma adımları oldukça küçüktür ve analiz daha hassas olur. Bu yüzden ortalama kayma adaptif bir gradyent yükseltme yöntemidir.

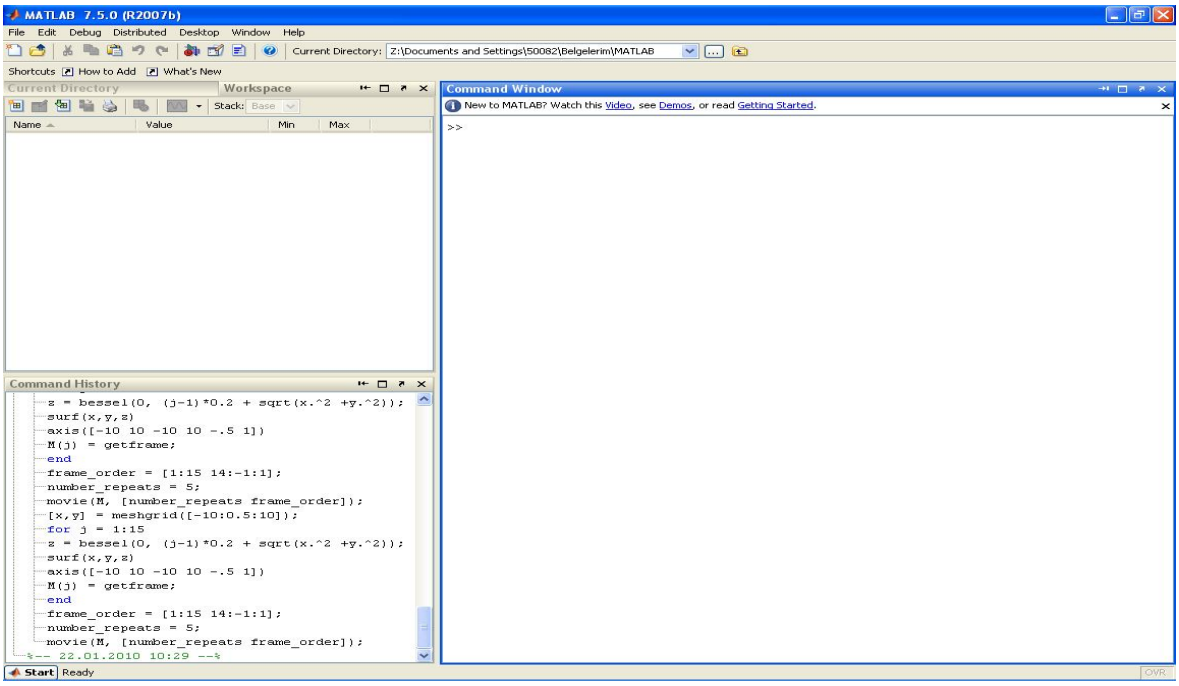
2.6. MATLAB'a Giriş

2.6.1. Temel Bilgiler

MATLAB kelime itibari ile MATrix LABoratory kelimelerinin kısaltılması ile oluşmuştur. Bu program ilk geliştirildiğinde amaç matris işlemlerinin kullanıcılar tarafından kolaylıkla yapılmasını sağlamaktır. Matlab, geliştirilmesi sonucu günümüzde basit matematiksel hesaplamalardan karmaşık analizlere varan çok çeşitli alanlarda kullanılabilir hale gelmiştir. Bu nedenle son zamanlarda Matlab özellikle bilimsel araştırmalar için tercih edilen ve popüler olarak kullanılan bir ortam haline gelmiştir.

Matlab'ın bu denli popüler oluşunun altında sunduğu çok çeşitli komutların yanısıra, grafiksel arabirime sahip oluşu, kolay alışılabilir ve kullanışlı bir ortam etkileşimi sunması, çok çeşitli alanlara hizmet eden farklı ve zengin kütüphanesinin olması yatmaktadır.

Matlab yazılımını başlatmak için Başlat menüsünde veya Masaüstünde yer alan MatLab ikonunu tıklamak yeterlidir.



Şekil 2.30. Matlab Ana Penceresi

Şekil 2.30 ile açılan bu pencere kendi içinde çok değişik işlevleri bulunan ve kullanıcının Matlab'ı rahat kullanmasını sağlayan şu pencerelerden oluşur.

• Array Editor

• Command History

• Command Window

• Current Directory

• Demos

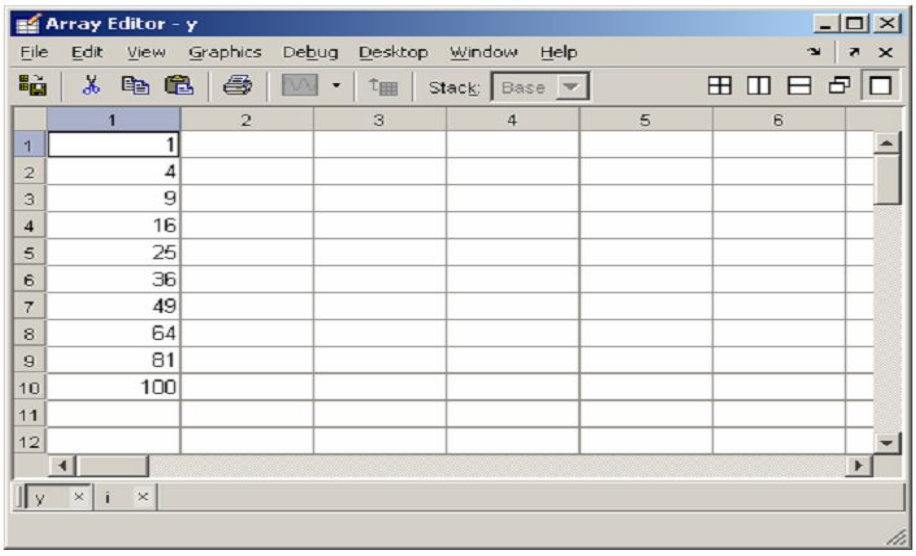
• Help

• Workspace

Bu pencereleri görevleri şu şekildedir:

2.6.1.1. Array Editor Penceresi

Kullanıcı workspace penceresindeki herhangi bir değişkenin üzerinde çift tıkladığında Şekil 2.31'de görülen Array Editor penceresi ile karşılaşır. Bu pecere yardımıyla seçilen herhangi bir değişkenin içeriği görülebileceği gibi yine aynı değişkenin içeriği bu pencere yardımıyla değiştirilebilir.



Şekil 2.31. Array Editor Penceresi

Ancak, bazı farklı tipteki değişkenlerin içeriğini değiştirmek mümkün değildir. Bu durum komut kullanılarak gerçekleştirilebilir.

2.6.1.2. Command History Penceresi

Bu pencere kullanılan tüm komutların geçmişini tutmak için kullanılır. Komut ekranından girilen her komut Matlab açıldığında geçerli zaman başlığı altındaki listeye eklenerek bu pencerede görüntülenir. Kullanıcının geçmiş komutları görmesine ve tekrar kullanmasına imkan verir.

2.6.1.3. Command Penceresi

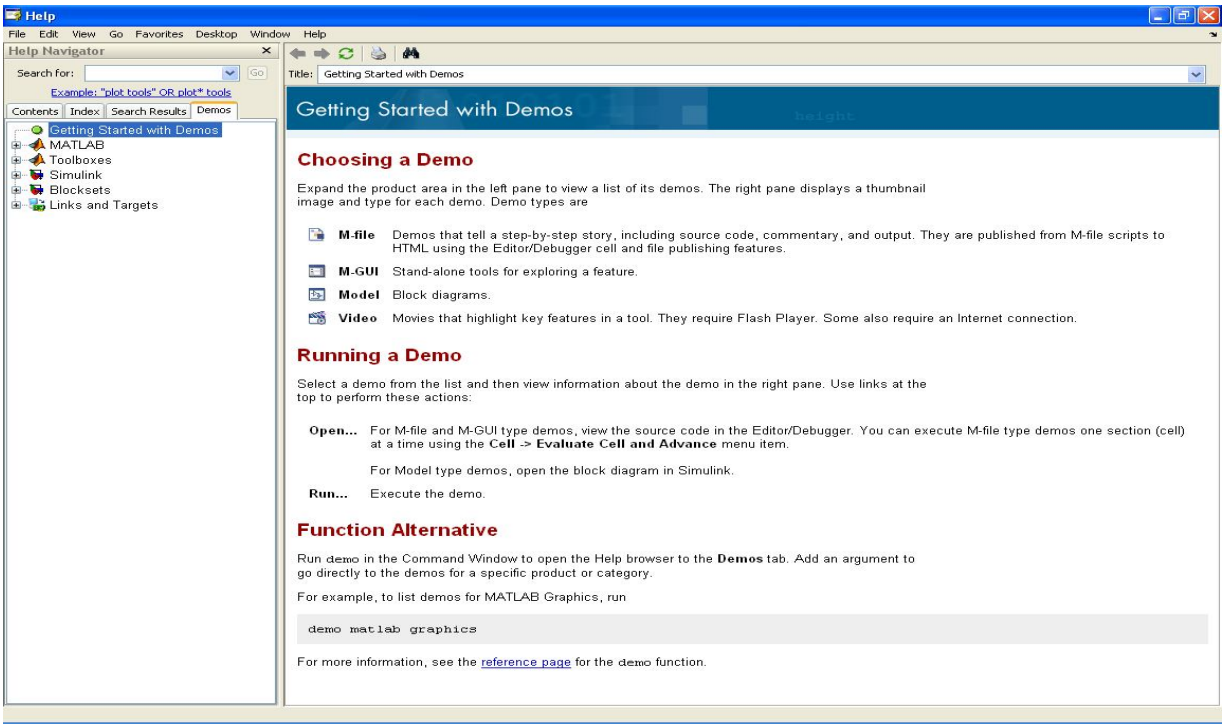
Bu pencere yardımıyla kullanıcı Matlab komutları girer. Girilen her komutun çıktısı da yine bu pencerede ve komutun girilmesinin hemen ardından görüntülenir.

2.6.1.4. Current Directory Penceresi

Matlab'ın herhangi bir anda aktif olarak kullandığı geçerli dizin yolunu değiştirmek, içinde bulunulan klasör içerisinde taşıma, kopyalama ve dosya silme gibi işlemleri gerçekleştirmek ve ya dosyalar hakkında bilgi edinmek için bu pencere kullanılır. Matlab daima geçerli bir yol üzerinden çalışır. Varsayılan olarak bu yol Matlab'ın kurulu olduğu dizin içinde yer alan Work klasörüdür. Geçerli klasör içerisinde yer alan kullanıcının tanımladığı fonksiyon dosyaları da bu alandan çağrılır. Eğer kullanılacak fonksiyonlar farklı ise bu pencere yardımıyla geçerli yol tanımı değiştirilmelidir. Ayrıca, Matlab'ın geçerli klasör yolu ana penceredeki araç çubuğunda yer alan Current Directoryu bölümünden de görülebilir veya bu yol tanımı değiştirilebilir.

2.6.1.5. Demos Penceresi

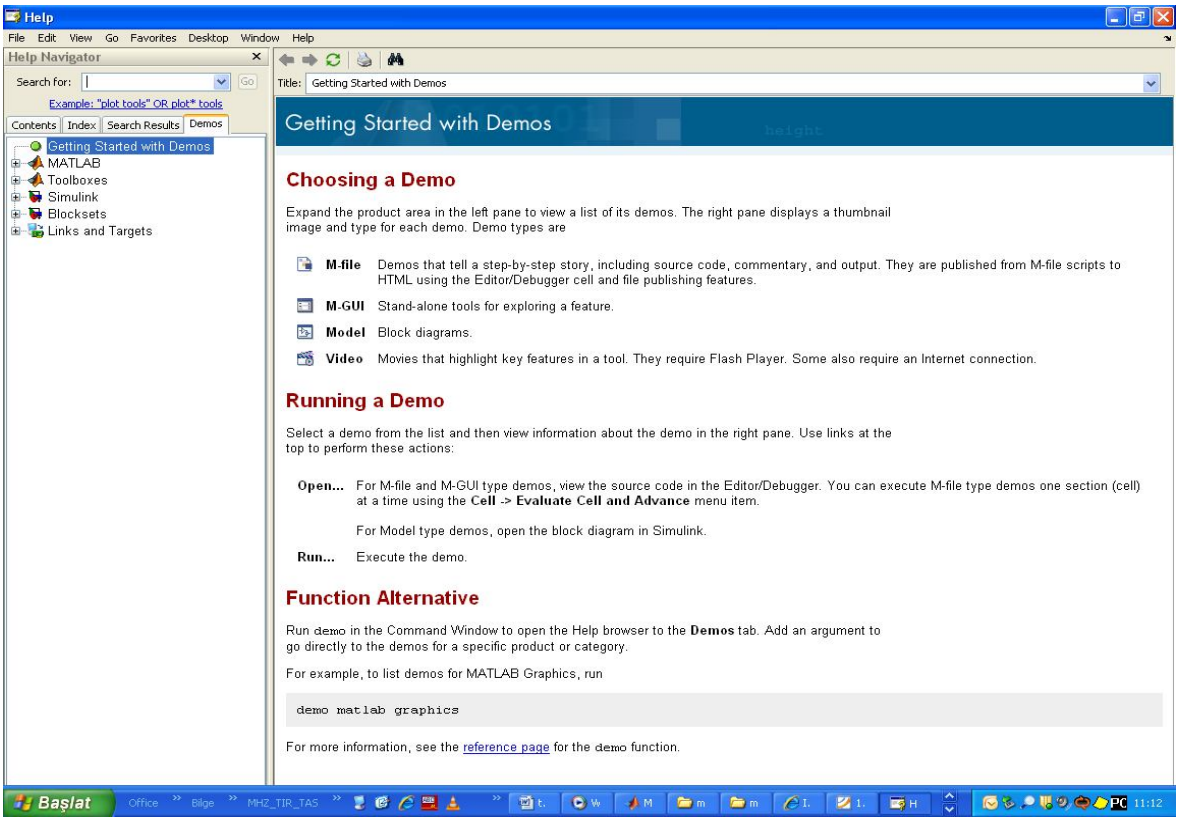
Help menüsünden Demos veya Matlab ana penceresi sol alt köşede yer alan Start butonu kullanılarak Demos komutunun verilmesi ile karşımıza Şekil 2.32'de görülen demos penceresi gelir. Kullanıcı bu pencere yardımıyla Matlab'ın kendi içinde yer alan hazır uygulamaları görebilir, kodlara bakabilir veya konu ile ilgili bilgiler edinebilir.



Şekil 2.32. Demos Penceresi

2.6.1.6. Help Penceresi

Bu pencereye ulaşmak için araç çubuğundan soru işareti simgesi tıklanabilir ya da Help menüsü kullanılarak Matlab Help komutu verilebilir. Kullanıcı Şekil 2.33'deki gibi bir ekran ile karşılaşacaktır.



Şekil 2.33. Matlab Help Penceresi

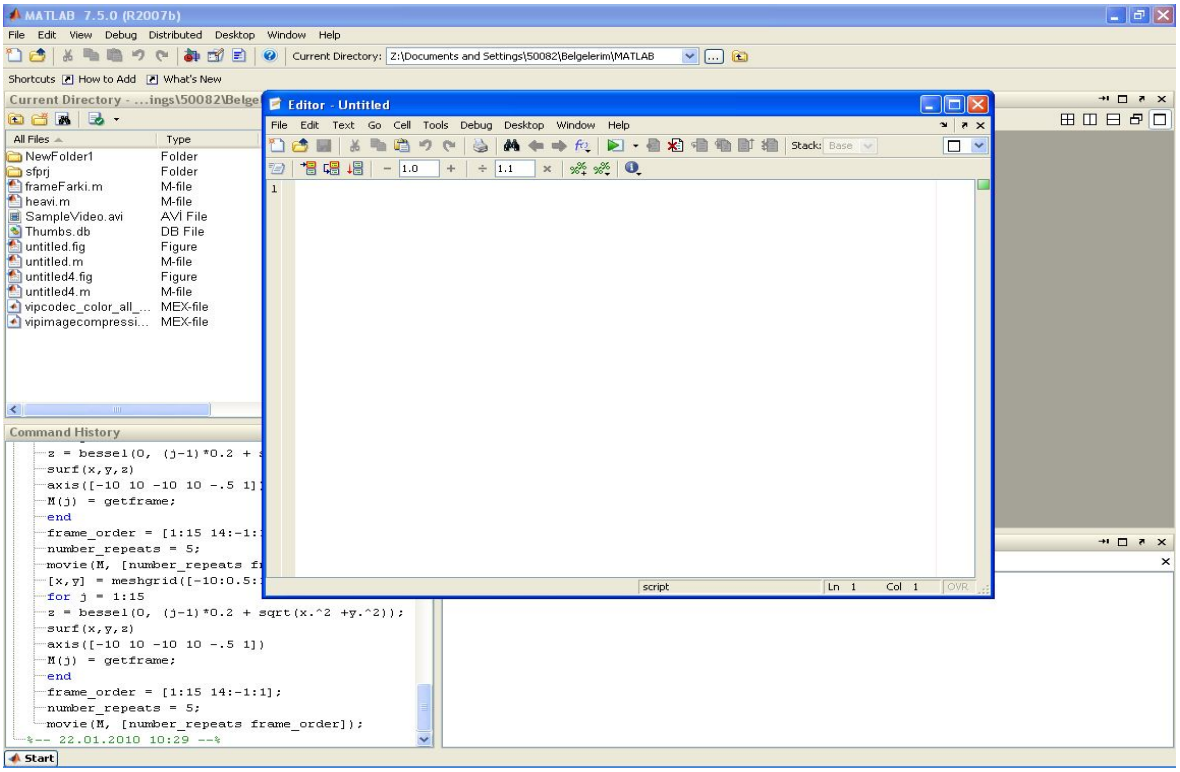
Matlab yardım penceresi ile kullanıcı sunulan çok geniş ve açıklayıcı anlatımı ile herhangi bir kaynağa gerek kalmadan Matlab kullanımı, Matlab komutları ve pek çok farklı konularda bilgiler edinebilir. Ayrıca, bu pencerenin sol tarafında yer alan Index tabı ile tüm konu başlıklarını sırasıyla görebilir ve herhangi bir konu hakkında bilgi edinebilir. Yine benzer şekilde Search tabını kullanarak hakkında bilgi edinmek istediği bir konuyu Matlab'ın kendi yardım dosyaları içerisinde aratabilir.

2.6.1.7. Workspace Penceresi

Kullanıcının işlettiği bir komuta ait değişkenler veya çıktı parametreleri daima Workspace alanına atılır. Böylece kullanıcı herhangi bir anda bu pencere yardımıyla mevcut değişkenlerin listesini görebilir. Ayrıca, kullanıcı bu pencereyi kullanarak içeriğini görmek istediği her hangi bir değişkenin üzerinde çift tıklayarak Array Editor penceresini açabilir.

2.6.1.8. Fonksiyon Dosyaları (M Dosyaları)

Kullanıcılar Matlab içinde kendilerine ait fonksiyonlar yazabilir ve kullanabilir. Bir fonksiyon yazmak için Windows'un Notepad programı kullanılabileceği gibi Matlab'ın kendisine ait "Editor" uygulaması da kullanılabilir. Bu uygulamayı açmak için komut satırından "edit" komutu verilir. Kullanıcı Şekil 2.34'teki gibi bir ekran ile karşılaşacaktır.



Şekil 2.34. Matlab Editor uygulaması ekran görüntüsü

Örneğin "hesap" isminde bir fonksiyon yazılmış olsun ve bu fonksiyon kendisine parametere olarak gönderilen sayıların toplamını hesaplasın. Bunun için Editor uygulamasında aşağıdaki komutlar yazılır.

```
function sonuc=hesap(a,b)

sonuc=a+b;

end
```

Daha sonra bu dosya Matlab'ın kurulu olduğu dizin altında yer alan work klasörüne kaydedilir. Dosyanın kaydedilme esnasında isminin fonksiyon ismi ile aynı olmasına

dikkat edilmelidir. Yani “hesap” ismi ile kaydedilir. Böylelikle work dizini altında hesap.m isminde bir m dosyası oluşacaktır. Daha sonra komut satırından aşağıdaki komutları girildiğinde hazırlanan fonksiyon kullanılarak girilen sayının çerçevesi hesaplanacaktır. Şekil 2.35’de gösterilmektedir.

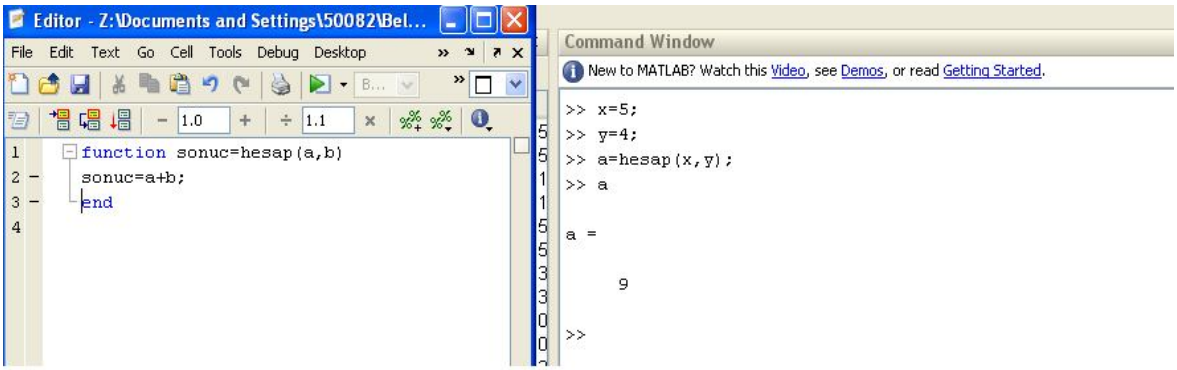
```
x=5;

y=4;

hesap(x,y)
```

veya

hesap (5,4) yazılır.



Şekil 2.35. Hesap fonksiyonun çalışması

2.6.2. MATLAB ile Görüntü İşleme

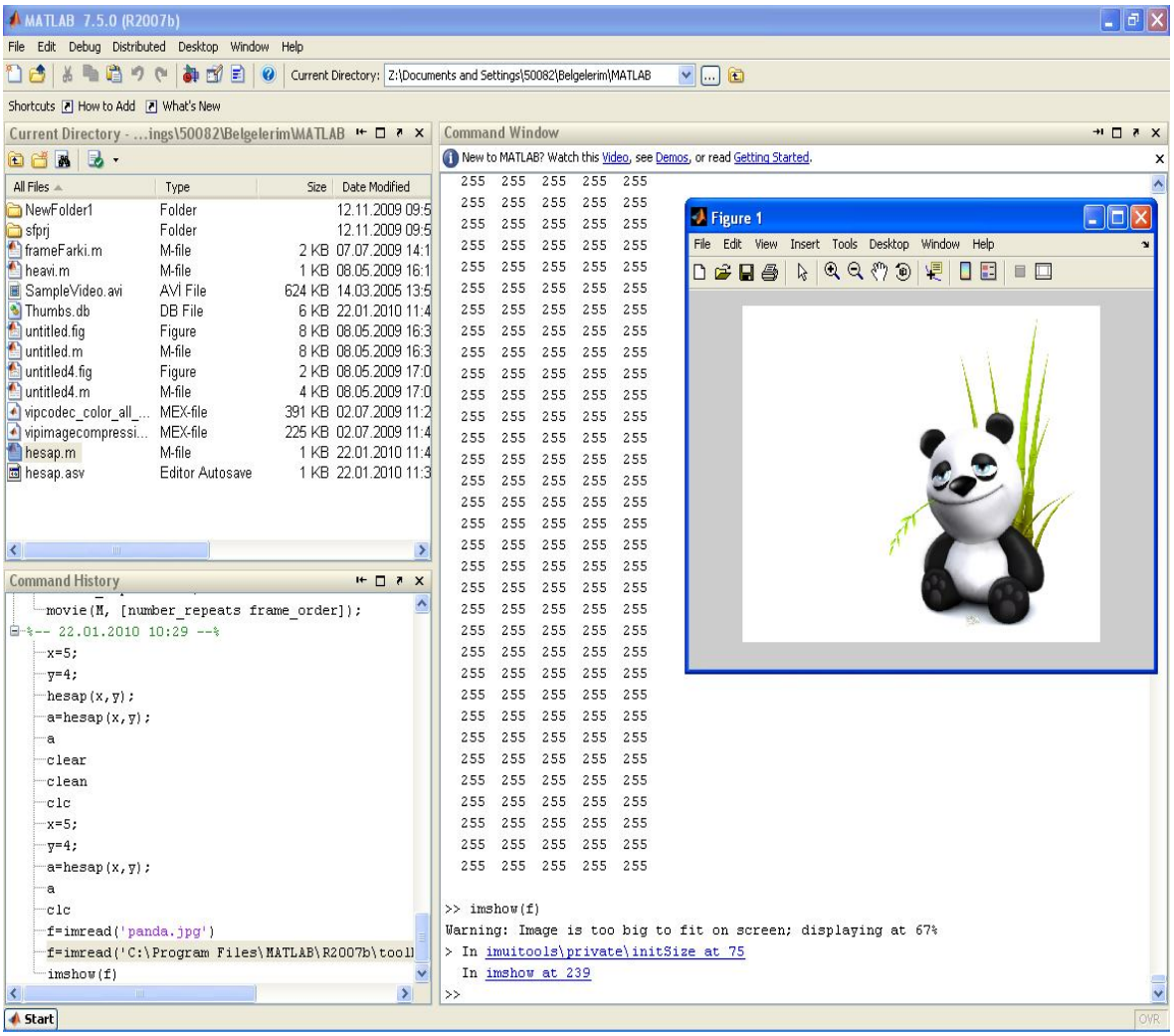
Matlab programlama dilinin eklentilerinden biri olan Image Processing Toolbox (Görüntü işleme aracı); geomatik uygulamalardan biri olan Sayısal Görüntü İşleme ile ilgili hedeflenen sonuçlara ulaşmayı sağlar. Önce bir görüntünün nasıl Image Processing Toolbox’a (IPT) atanabileceğini göstermek gerekiyor. IPT, Matlab’a görüntü atıldığında otomatik olarak ortaya çıkan bir araç. Herhangi bir JPEG görüntüyü atmak için ilk temel kod:

```
f=imread('C:\Program Files\MATLAB\R2007b\toolbox\images\imdemos\panda.jpg');
```

Görüntü bu kod ile, Matlab içersine atanır ve görüntünün tüm matrisi command window içersinde oluşmaya başlar. Görüntünün figure penceresinde görünmesi için yazılması gereken kod:

imshow(f)

Böylelikle görüntü matlab içersinde açılmış olacaktır. Şekil 2.36 ile görüntü Matlab programına yüklenir.



Şekil 2.36. Görüntü dosyasının yüklenmesi

size(f) kodu ile görüntünün boyutunu, whos f kodu ile de görüntünün detaylı mekansal çözünürlük ve radyometrik bilgilerini gösterir. Bu kodların dışında direkt olarak aşağıdaki kodla, toolbox içersinde görüntü açılabilir ve gerekli bilgiler elde edilebilir.

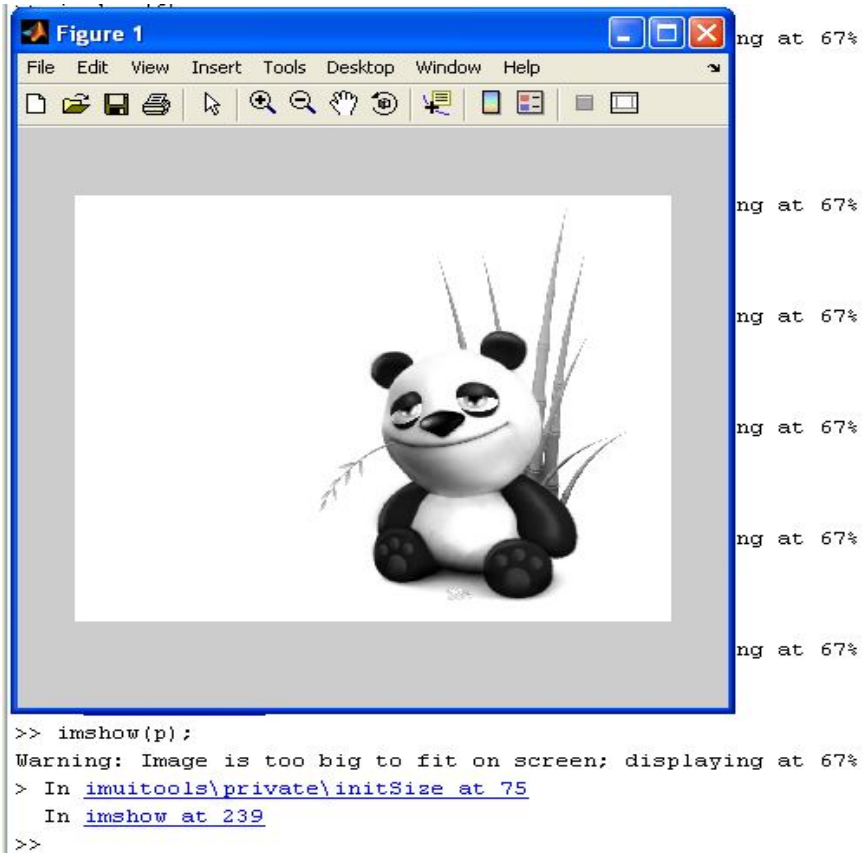
imtool(f)

Şekil 2.37. İmtool aracının kullanılması

Şekil 2.37 ile açılan pencerede, info sekmesiyle görüntü bilgilerini, pixel region sekmesi ile de, her bir pikselin yansıtım değerleri (RGB) gösterilir.

RGB görüntünün yoğunluk formatına (intensity) dönüşümü için f RGB görüntüsünü, p intensity görüntüsüne çevirecek kod ise;

```
p=rgb2gray(f);
```



Şekil 2.38. Görüntünün gri ölçeklenmesi

Çevirimden sonra görüntü gri değerlerine dönüştüğünden, kontrast ayarları da yapılabilir. Şekil 2.38 ile görüntü gri formattadır.

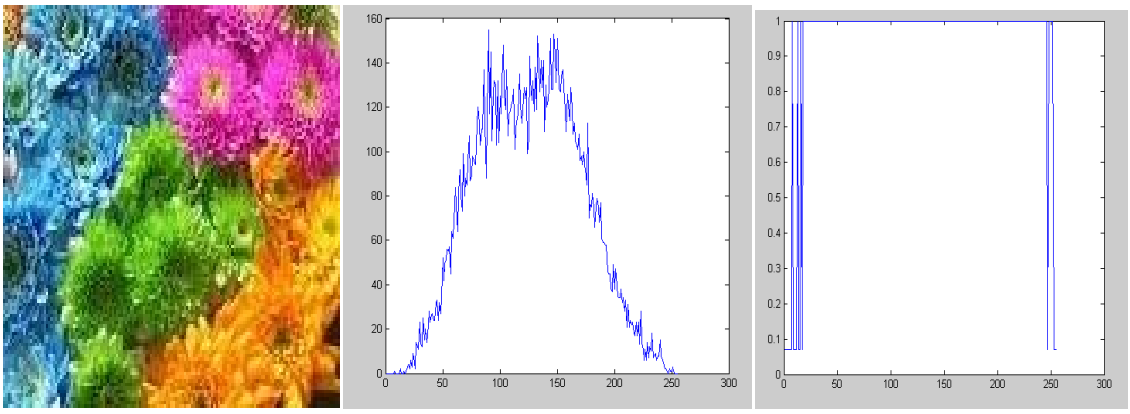
IPT dahilinde, intensity değerlerine dönüştürülmüş görüntüler ile histogram işlemleri yapılabilir. Aşağıdaki kod ile görüntünün histogramı çizilebilir. Matlab'taki grafik opsiyonları sayesinde de histogramı, bar ya da plot formatlarında da çizdirilebilir.

```
imhist(p)
```

Temel görüntü işleme algoritmalarından olan histogram eşitleme algoritması (Histogram Equalization), IPT dahilinde hazır olarak bulunmaktadır. Eldeki görüntüyü atadıktan sonra, aşağıdaki kod ile görüntüye histogram eşitlemesi geçirttirilebilir ve sonuçları histogramlarını çizdirerek karşılaştırılabilinir.

```
g=histeq(p,256)
```

Elde edilen, eşitlenmiş görüntü şekil 2.39'daki gibidir.



Şekil 2.39. Verilen görüntünün histogramı ve eşitlenmesi

Eşitlenmemiş görüntü ve eşitlenmiş görüntünün histogramları karşılaştırıldığında, algoritmanın amacına uygun olan daha düze yakın bir histogram elde edilir.

Bir diğer görüntü işleme örneği de; görüntünün kontrast dengelemesidir. (contrast adjustment)

z=imadjust(p)

kodu yardımıyla, görüntünün kontrast değerlerinin dengelenmiş bir hale dönüşmesini sağlar.

Görüntüden binary (1 bit- sadece siyah ve beyaz) görüntüler elde etmek için, görüntü segmentasyonunun (image segmentation) temeli olan thresholding (eşiklendirme) algoritmasını Matlab IPT dahilinde uygulanabilir.

level=graythresh(p);

bw=im2bw(p,level);

Görüntü ve histogramın yeni eşiklendirilmiş durumları elde edilir.

3. BULGULAR

3.1. Yapılan Çalışmalar

Tez çalışması için Intel(R) core(Tm)2 duo işlemcili 2.20GHz ve 3GB RAM'e sahip bir bilgisayar kullanıldı. Algoritmaların çalıştırılması için ise MATLAB R2011b programı kullanıldı.

Çalışmada başlangıç olarak Matlab ortamında bir video parçasının çerçeve farkları alındı. Elde edilen değerler analiz edildi.

Sonraki çalışmada gerçek zamanda çalışabilmek için webcamdan alınan görüntü içerisinde hareketli nesne tanınarak nesne takibi işlemi gerçekleştirildi. Nesne takibi için iki farklı algoritma kullanıldı; ortalama değer kayması ve optik akış algoritmaları.

3.1.1. Matlab Tarafından Videonun Tanınması

Video analizi için Matlab yazılımı kullanıldı. Aşağıda adım adım Matlab yazılımı ile bir videonun tanınması aşamaları açıklanmaktadır.

Öncelikle dışarıdan bir video parçası Matlab'a aşağıda verilen kod satırı ile tanıtılır.

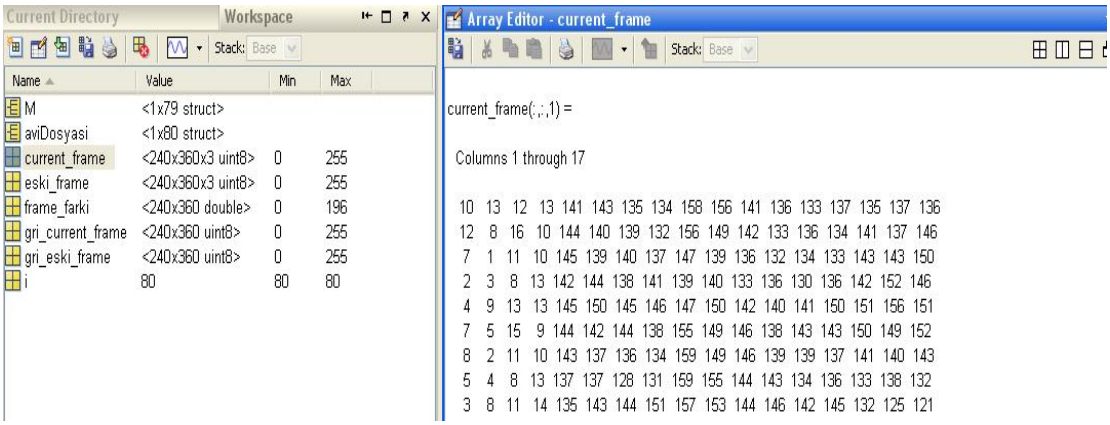
```
aviDosyasi = aviread('SampleVideo.avi');
```

Bu kod ile dışarıdan bir video parçası, aviDosyasi olarak adlandırılan değişkene atanır.

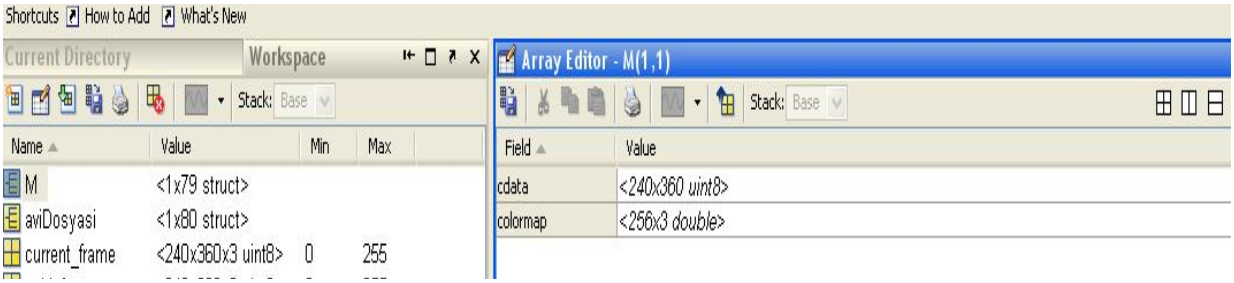
Daha sonra video parçasını çerçevelere ayırarak renk yoğunluğu analiz edilir. İki çerçeve arası fark alınır.

'SampleVideo.avi' dosyası 80 çerçeve uzunluğundadır. Çerçevelerin sırayla tek tek farkları alınır.

Elde edilen çıktılar ise şekil 3.1, şekil 3.2 ve şekil 3.3'de gösterilmektedir.

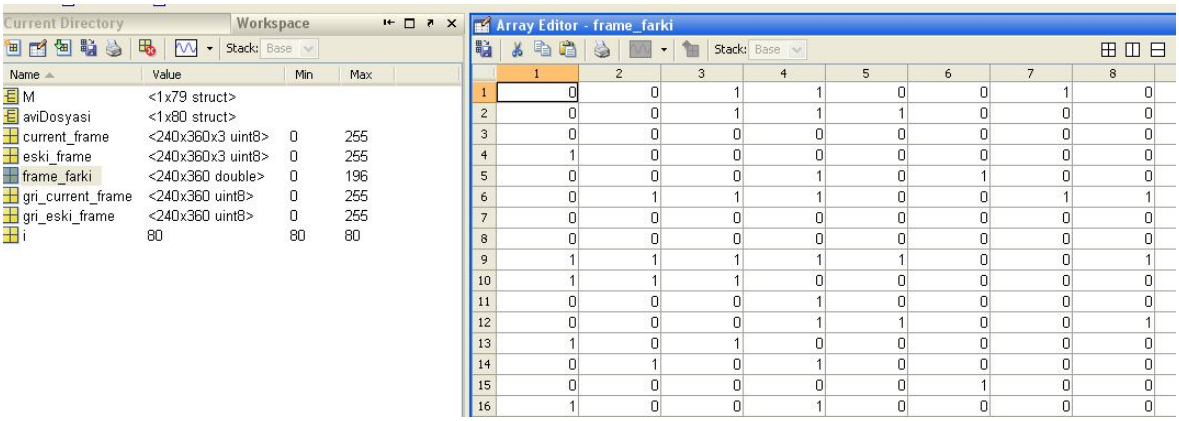


Şekil 3.1. Video parçasının bir değişkene atılması ve renk değerlerine göre matrisi(RGB)



Şekil 3.2. Tüm farkların alındığı M vektörü içeriği

M vektöründe tüm çerçevelerin farkları bulunmaktadır. Çerçeve renk değerini öğrenmek için 1x79 struct içinde herhangi bir kolona tıklanarak array editörü açılır. Burada o çerçevelere ait değerler bulunur.

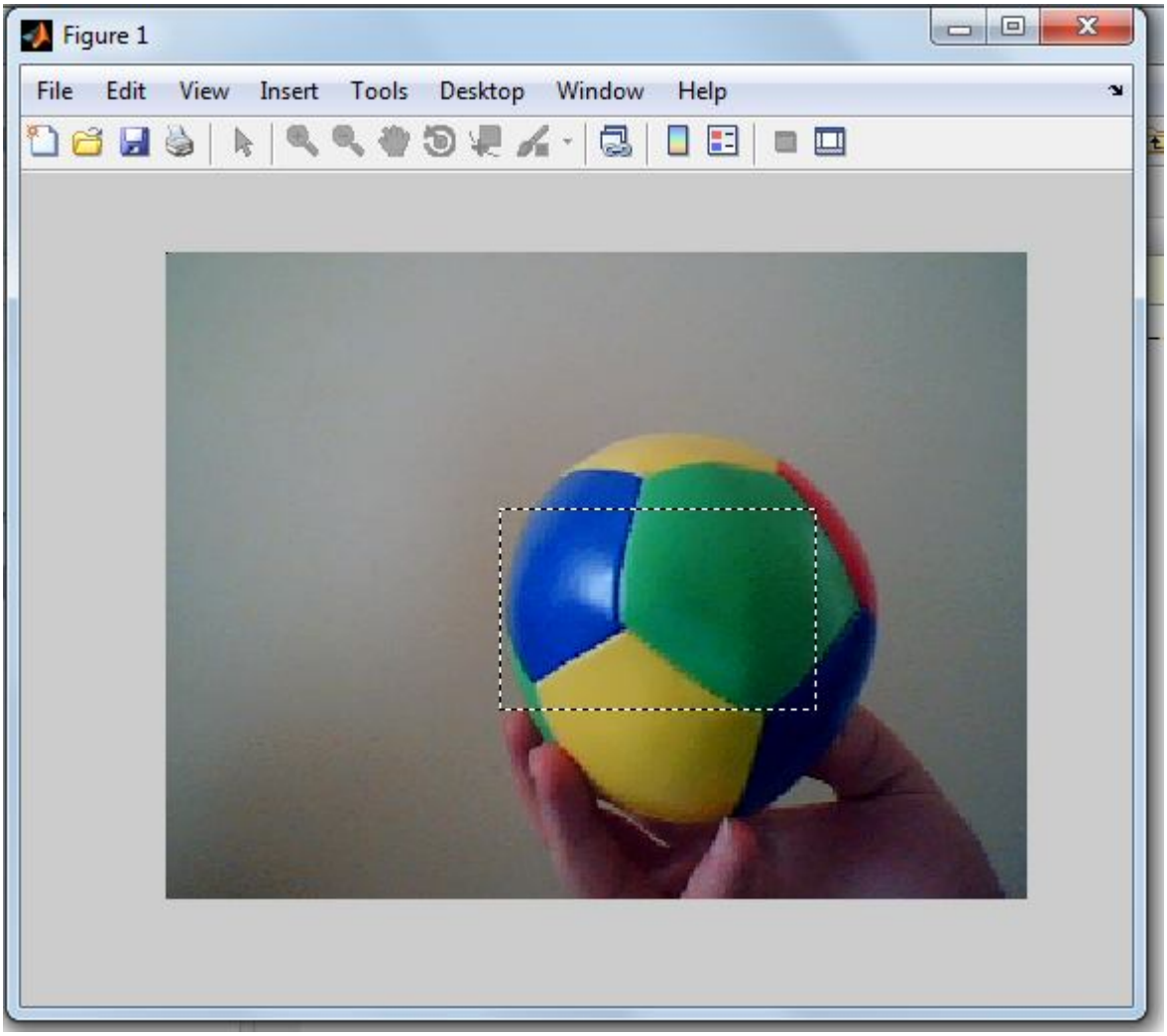


Şekil 3.3. Çerçeve farklarının elde edildiği matris

3.1.2. Video Akışında Nesne Seçimi

Video akışları içinde birçok hareketli nesne olması mümkündür. Takip edilecek nesne ilk çerçevede seçilirse, izleme işlemi daha kolaylaşır. Webcamdan alınan görüntüde takip edilecek nesneyi seçmek için, RGB formatındaki görüntünün ilk çerçevesinde nesneyi içerisine alacak dikdörtgen bir seçim çerçevesi kullanıcı tarafından çizilir. Böylece arama penceresi belirlenir ve nesneyi modellemek için istatistiksel bilgi olacak histogram(renk dağılımı) bu pencere içerisinde hesaplanır. Ayrıca hedef nesnenin pencere içindeki merkezi hesaplanır.

İlk çerçevede takip edilecek nesnenin belirlenmesi şekil 3.4' de gösterildiği gibi gerçekleşir.



Şekil 3.4. İlk çerçevede takip edilecek nesnenin seçilmesi

Programda iki farklı algoritma kullanıldı. Program başlarken algoritma seçimi kullanıcıya bırakıldı. Birinci algoritma "Ortalama Değer Kayması", ikinci algoritma ise "Optik Akış" algoritmasıdır.

Gerçek zamanda çalışabilmek için webcamdan alınan görüntü kullanılan Ortalama değer kayması Kernel yoğunluk tahmini ve Horn ve Schunk Optik akış algoritması ile izlendi. Algoritmaların çalışması için 250 çerçeve sabit tutuldu. Webcam video formatı ise "YUY2_320x240" olarak belirlendi.

3.1.3. Ortalama Değer Kayması İle Nesne Takibi

Ortalama değer kayması metodu renk dağılımına dayalı bir metoddur. Seçilen nesnenin başlangıç pozisyonu ve hareket ile yer değiştirdiği hedef pozisyonlar belirlenir. Yalnız seçili nesne ve arkaplan rengi benzerlik gösterdiğinde bu algoritma doğru çalışmayacağından Kernel yoğunluk tahmini yöntemi algoritmaya eklenmiştir. Denklem (2.71) ile bu yöntem gösterilmektedir.

Epannechnikov kerneli çalıştırılmıştır. Denklem (2.75) ile Epannechnikov kerneli gösterilmektedir.

Öncelikle pencere içerisindeki nesnenin renk dağılımından yararlanılarak nesnenin renk histogramı çıkarılır. Nesneye ait renk histogramı $\hat{q} = \left\{ \widehat{q_u} \right\}_{u=1\cdots m}$ ve $\sum_{u=1}^{m} \widehat{q_u} = 1,$ y noktasındaki hedefe ait olduğu tahmin edilen renk histogramı $\hat{p} = \left\{ \widehat{p_u} \right\}_{u=1\cdots m}$ ve $\sum_{u=1}^{m} \widehat{p_u} = 1,$ ise benzerlik katsayısı şöyle elde edilir:

$$\hat{p} \equiv p\left[\hat{p}(y),\, q \right] = \sum_{u=1}^{m} \sqrt{\widehat{p_u}(y)\ \widehat{q_u}} \tag{3.1}$$

Bu katsayı iki histogramın benzerlik ölçütü olarak kullanılır. Katsayı bire ne kadar yakınsa iki histogram birbirine o kadar benziyordur. Her bir noktanın histograma dayalı ağırlığı w(i) ise şöyle bulunur:

$$w(i) = \sqrt{\frac{\widehat{q_u}}{\widehat{p_u}(y)}} \tag{3.2}$$

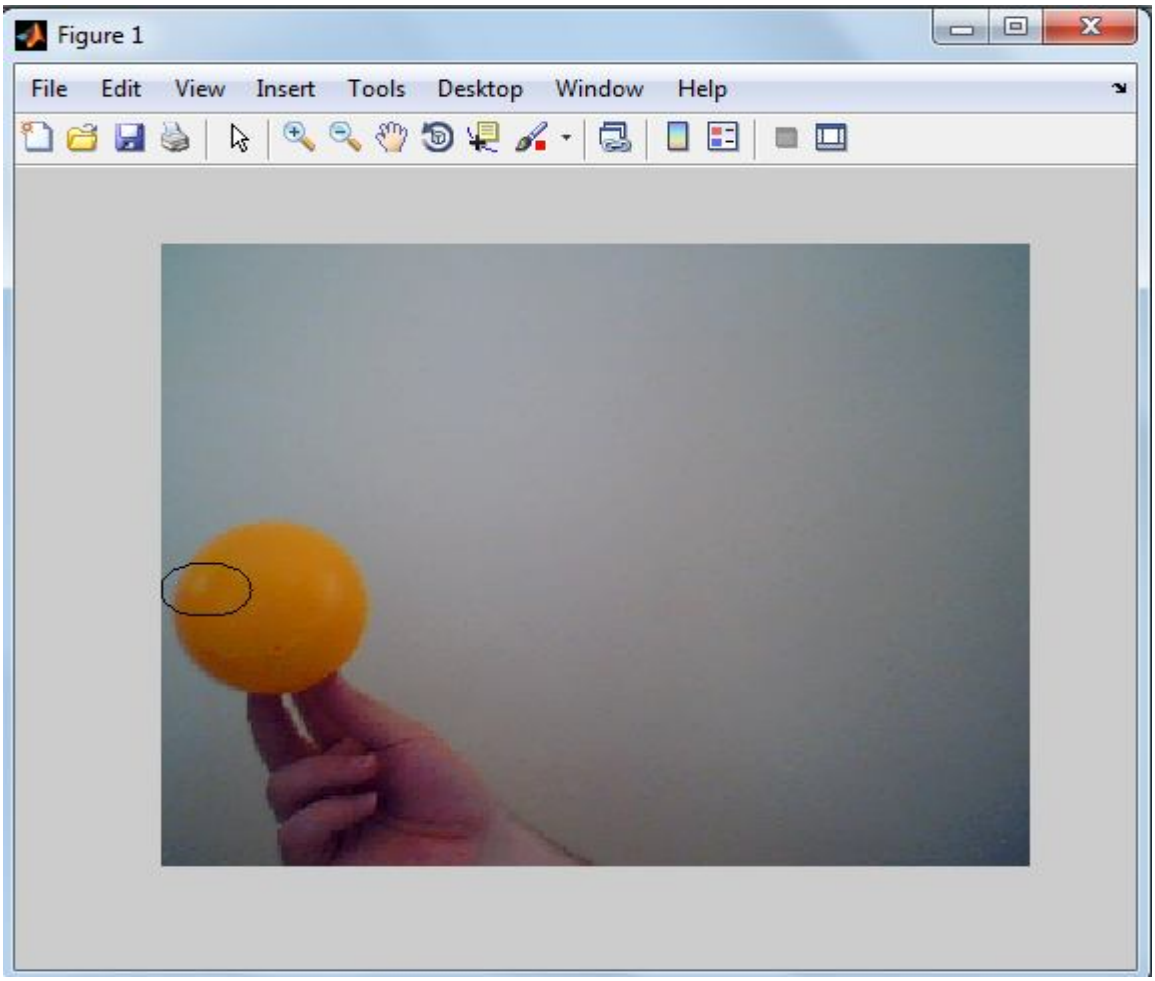
Bu ağırlık değeri her bir piksel ile ilişkilendirilerek nesne pikselleri bazında olasılık dağılımı elde edilmiş olur.

Elde edilen Kernel ağırlığı ise denklem (2.82) ile gösterilmektedir.

Programda kullanılan genel ortalama değer kayması algoritması, denklem (3.3)' de gösterildiği gibidir.

$$\text{Hesaplanan yeni yer} = \frac{\sum_{i=1}^{n} w_i g\left(\left\|\frac{y - y_i}{h}\right\|^2\right) y_i}{\sum_{i=1}^{n} w_i g\left(\left\|\frac{y - y_i}{h}\right\|^2\right)} \tag{3.3}$$

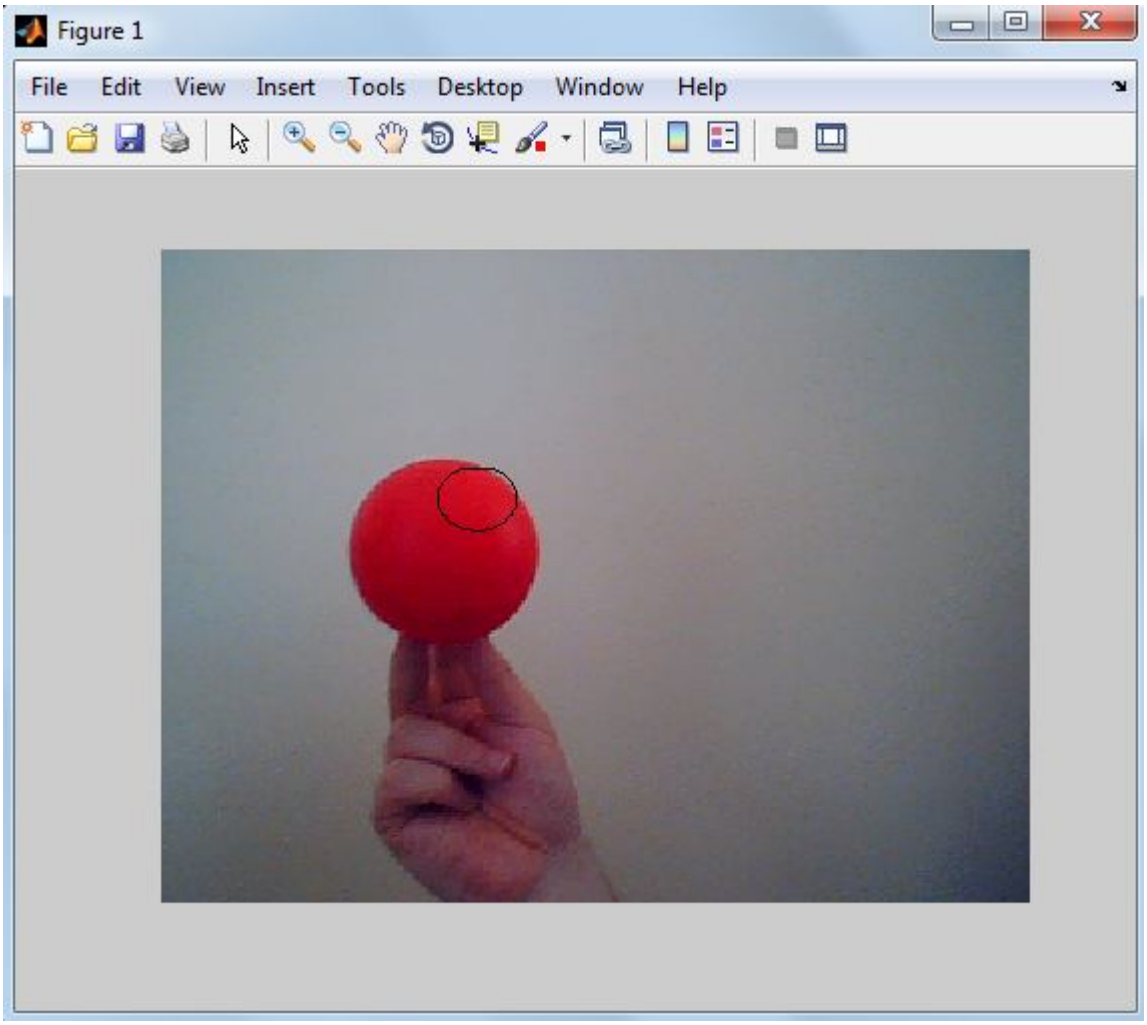
Algoritma farklı renkdeki nesneler ile çalıştırıldı. Arkaplan rengi ile top rengi aynı olan nesnenin takibi Şekil 3.5'de gösterilmektedir.



Şekil 3.5. Sarı renkli top ve arka plan aynı renkte iken nesne takibi

Hareketli nesne ve arkaplan rengi aynı olmasına rağmen takip algoritması başarıyla gerçekleşmektedir. Algoritmanın saniyede işlediği çerçeve sayısı yani fps değeri ortalama 33 fps'dir.

Hareketli nesne, arkaplan rengine zıt renk seçildiğinde de algoritma başarı ile çalışmaktadır. Şekil 3.6 kırmızı renkli topun ortalama değer kayması ile takibini göstermektedir.



Şekil 3.6. Kırmızı topun ortalama değer kayması algoritması ile takibi

Zıt renkteki nesneyi takip ederkende algoritma hızı 33fps olarak kalmıştır. Algoritma hızı farklı nesneler izlendiğinde değişmediği görülmektedir.

3.1.4. Optik Akış ile Nesne Takibi

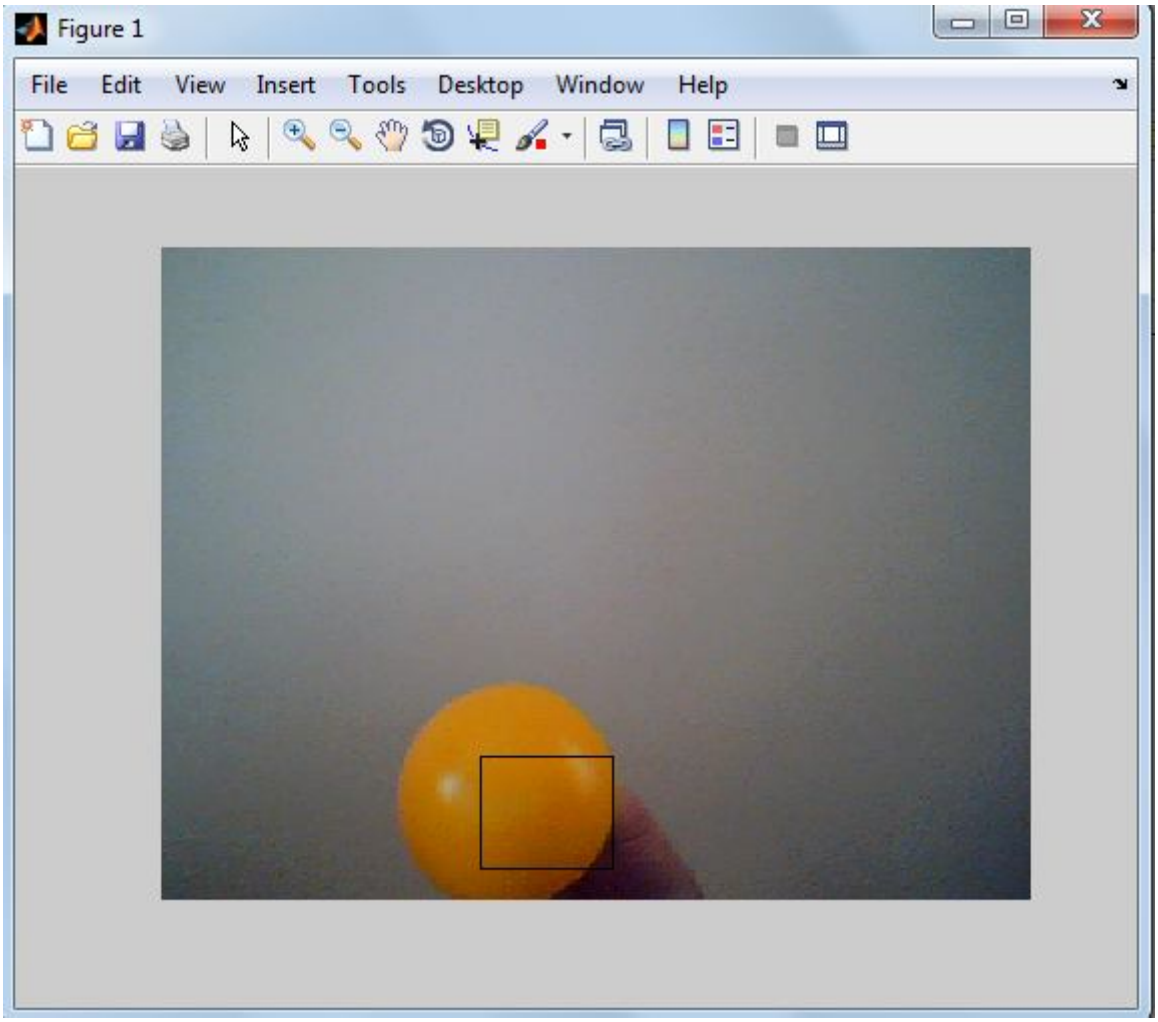
Programda Horn ve Schunk Optik Akış yöntemi kullanılmıştır. Optik akış yönteminde hangi algoritma kullanılırsa kullanılsın I_x, I_y ve I_t görüntü şiddet türevleri hesaplanır. Şiddet türevlerinin sürekli olması hız tahmininde önemlidir. Bu nedenle, türev hesaplamaları yapılırken hepsi aynı zamanda görüntü üzerindeki aynı noktaya işaret etmelidir.

Optik akış hesabında yaklaşık fark hesabı için birçok yöntem kullanılmaktadır. Burada bahsedilen fark hesabı yöntemi, sekiz ardışık ölçüme sahip bir küpün(Bknz. Şekil 2.24) merkez noktasındaki I_x, I_y ve I_t türev tahminlerini veren bir set kullanmaktadır. Hesaplanan türev tahminlerinin her biri, kübteki bitişik ölçümlerden alınan dört farkın ortalamalarıdır. Kullanılan denklem, (2.22)'de gösterilmektedir.

Türev hesaplamasından sonra V_x ve V_y hız bileşenlerinin laplasiyenleri hesaplanır. Bunun için denklem (2.25) kullanılmıştır.

Horn ve Schunk algoritmasının kullanılması için yumuşatma faktörü α değeri 100 ve eşik değeri 0.1(sapma miktarı) olarak seçilmiştir. Son olarak Horn ve Schunck yönteminde, Euler-Lagrange denklemlerini çözen Gauss-Seidel tekrarlama denklemleri ile hız akışı tespit edilir. Denklem (2.36) bu hız akışı tespitini yapmaktadır.

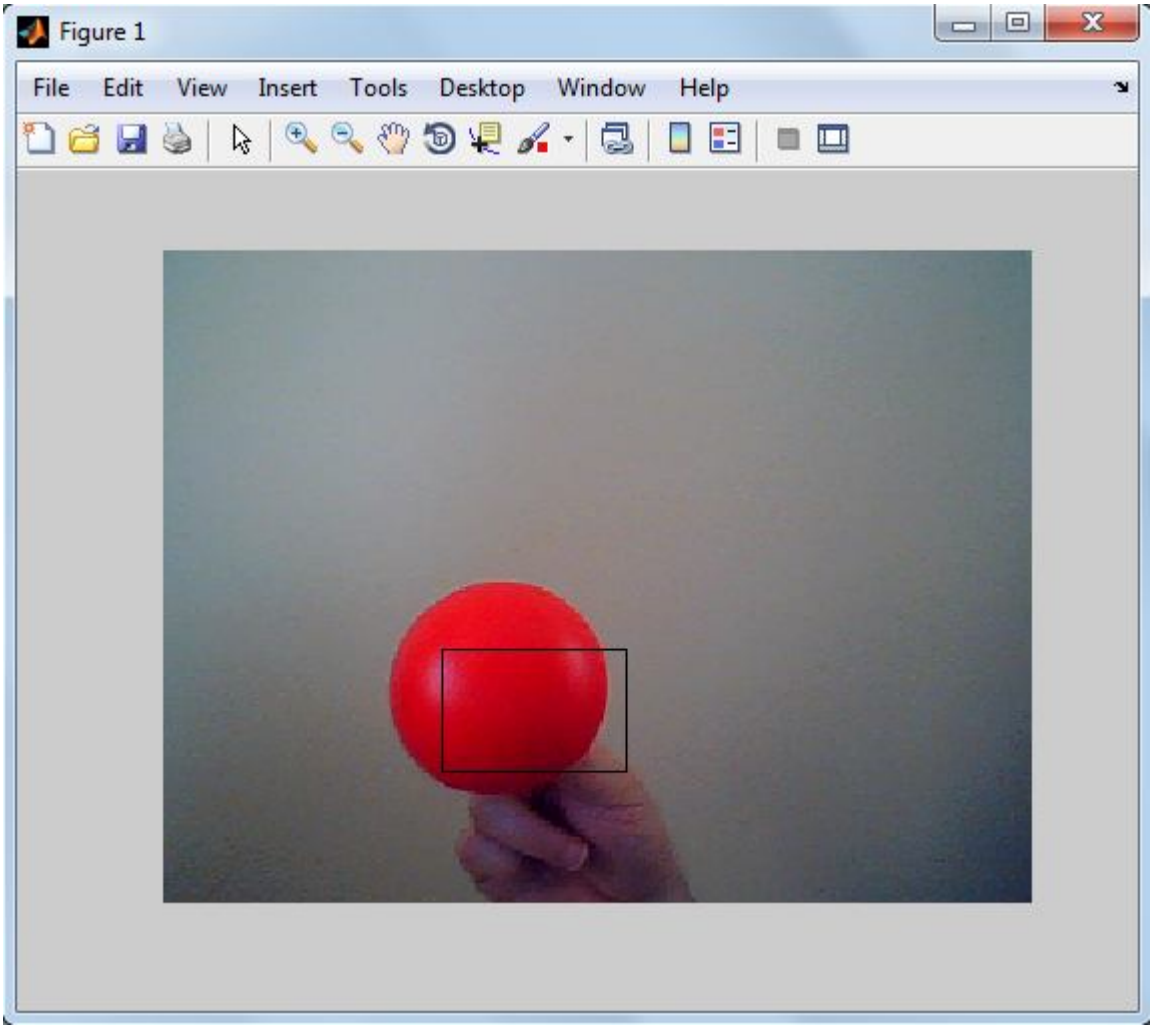
Bu algoritma için de iki farklı renkte topun takibi yapıldı. Önce sarı renkte top Horn ve Schunck algoritması ile çalıştırıldı. Ortalama değer kayması ile takipten farklılık olsun diye seçim çerçevesi dikdörtgen olarak belirlendi. Şekil 3.7'de bu algoritmanın çalışması gösterilmektedir.



Şekil 3.7. Horn ve Schunck algoritması ile sarı topun takibi

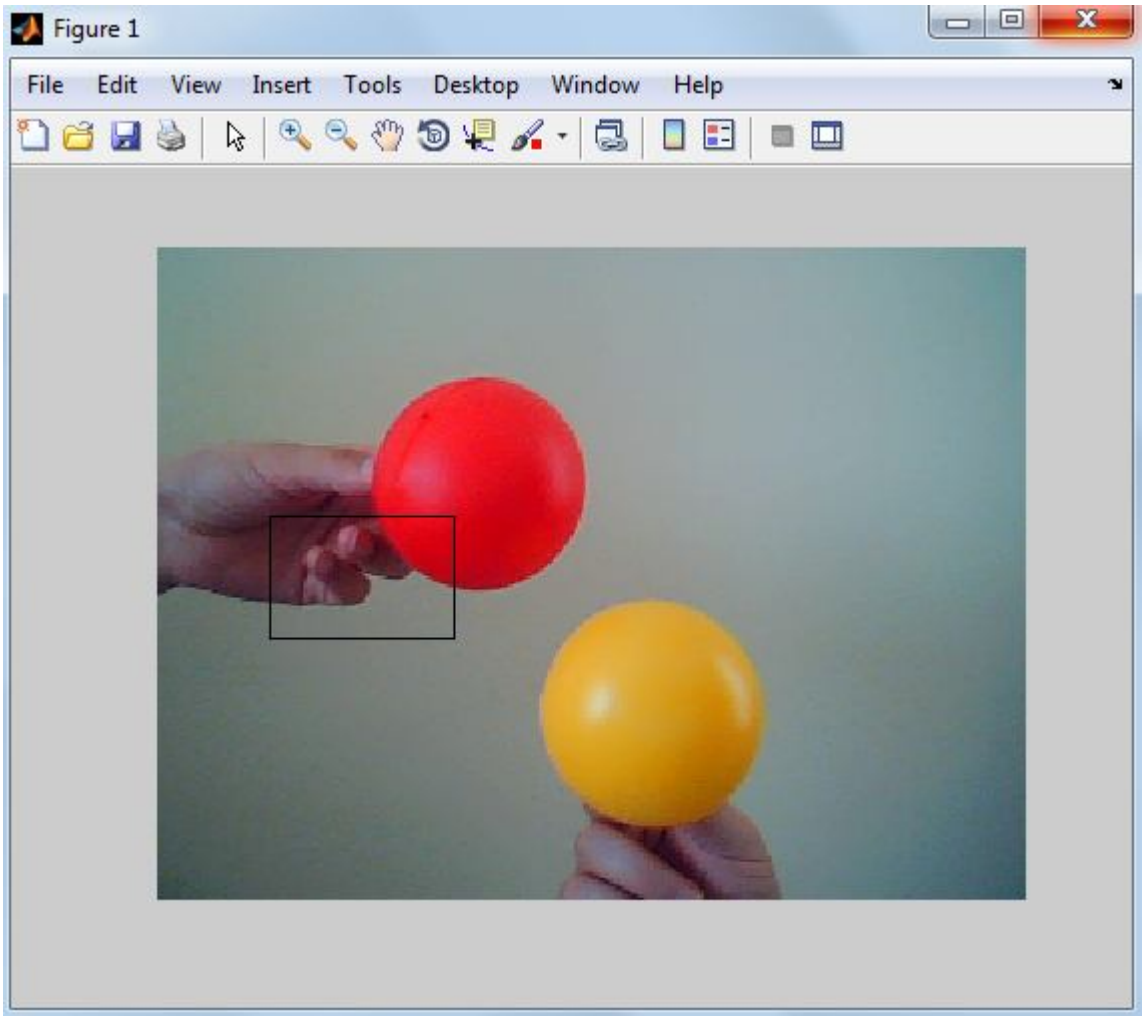
Algoritmanın çalışması ile elde edilen ortalama fps değeri 20 fps olarak elde edildi. Kullanılan yöntemde hareket vektörleri önemli olduğu için hareketin algılanması ile algoritma daha verimli çalışmaktadır. Harektesiz bir ortamda seçili nesneyi takip eden dörtgende merkezde kalmaktadır.

Arkaplan rengine zıt renkteki kırmızı top, Horn ve Schunck algoritması ile takip edildi. Algoritma hızı 20 fps olduğu görüldü. Şekil 3.8'de görüldüğü gibidir.



Şekil 3.8. Horn ve Schunck algoritması ile kırmızı topun takibi

Optik akış yöntemi ile hareketli nesnenin takibinde en çok rastlanan sorun nesnenin kapatılmasıdır yani önüne herhangibir nesne gelmesi görüş alanından takip edilen nesnenin kaybolmasıdır. Horn ve Schunck algoritması ile takip edilen nesne başka nesne ile kapatılsa veya görüş alanından kaybolup yeniden görünse de algoritma başarı ile çalışmakta ve takip işlemi sürmektedir. Horn ve Schunck MATLAB uygulaması sonucunda, tekrarlama sayısı ve eşik değeri seçiminin nesnelerin doğru hız tahmininde önemli rol oynadığı görülmüştür. Genel yumuşatma bileşenlerini kullanan Horn ve Schunck yöntemi iki boyutlu verilerde daha iyi çalışmaktadır. Şekil 3.9 ile görüş alanından kaybolan ve sarı top ile kapatılan kırmızı topun yeniden görüş alanına girdiğinde takibinin sürdüğü görülebilir.



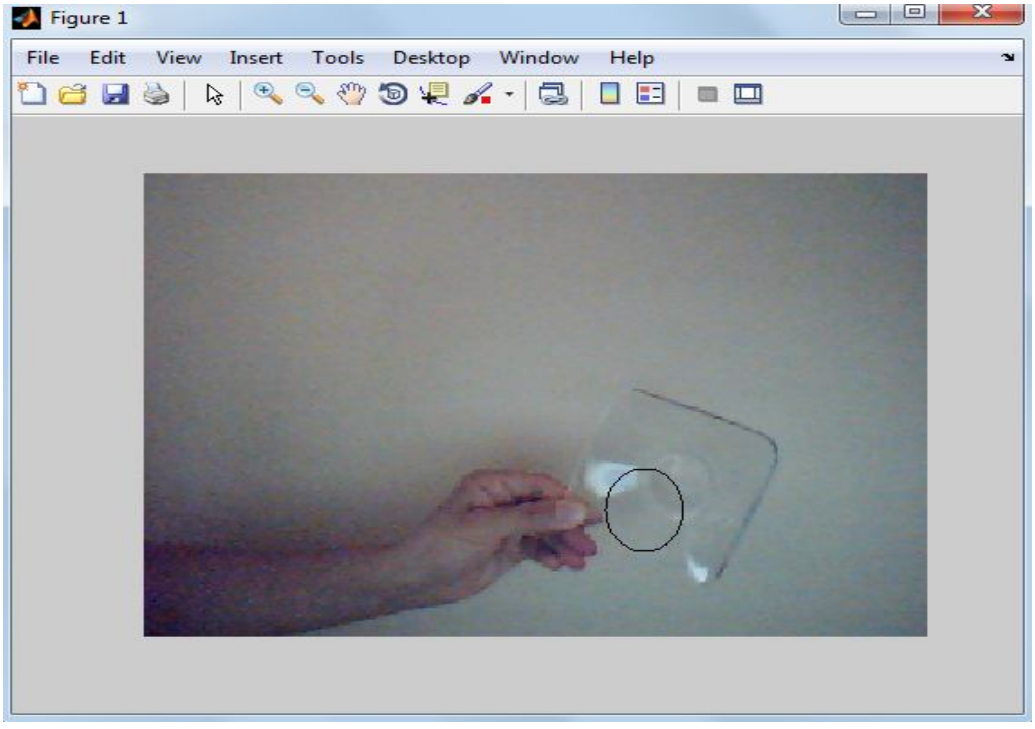
Şekil 3.9. Görüş alanından kaybaolan kırmızı topun yeniden görünmesi ile takibi

4. TARTIŞMA VE SONUÇ

Günümüz verileri, çok çeşitli ve devasal boyutlarda veri tabanlarına sahiptirler. Bu çeşitli veriler bahsettiğimiz gibi Multimedya verileri oluşturmaktadır. Video en önemli multimedya verisi olarak içeriği oldukça zengin verilerden oluşmaktadır.

Bu tezde, video görüntülerinde hareketli nesneyi takip edebilmek için, renk yoğunluk dağılımlarına göre(histogram) ortalama değer kayması yöntemi ve hız yön vektörlerine göre de optik akış yöntemi kullanılmıştır, gerçek zamanda nesne takibi yapılmıştır. Algoritmalar ile ilgili istatistiksel bilgiler, denklemler ve görsel çıktılar sunulmuştur.

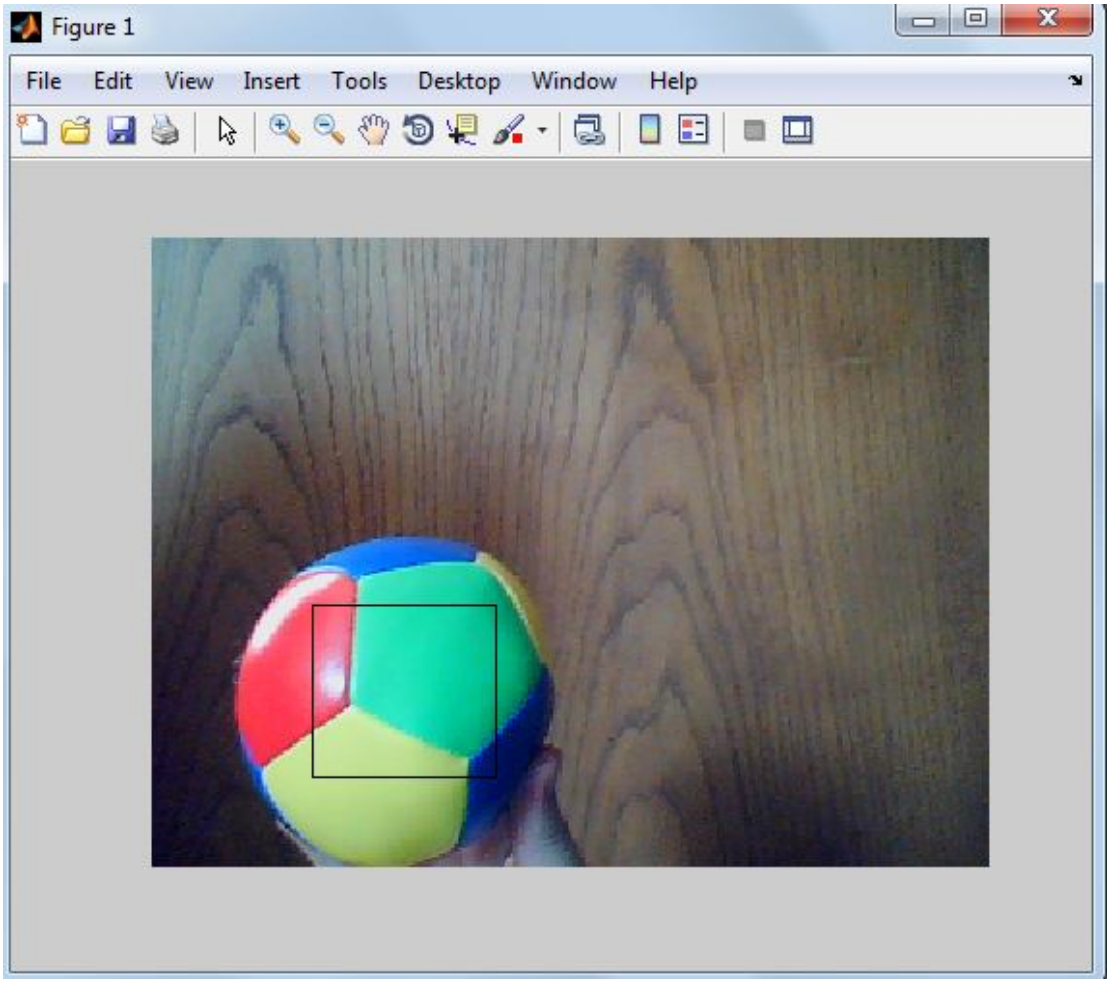
Webcamdan alınan görüntülerde insan yüzü, oyun topu ve hareket halindeki farklı nesneler, bu iki algoritma ile takip edilmiştir. Nesneler arkaplan ile aynı olduğunda nesne takibi klasik ortalama değer kayması metoduyla başarılı şekilde izlenemediği için algoritmada Epannechnikov kerneli kullanıldı. Böylece hareketli nesne takibi başarı ile sağlanmıştır. Webcamdan alınan görüntüdeki hareketli nesne arka plan rengine benzer seçildi ve takip algoritması çalıştırıldı. Algoritmanın çalışma hızını gösteren grafik ve nesnenin takibi Şekil 4.1'de gösterilmektedir.



Şekil 4.1. Arka plan rengine benzer nesnenin ortalama değer kayması ile izlenmesi

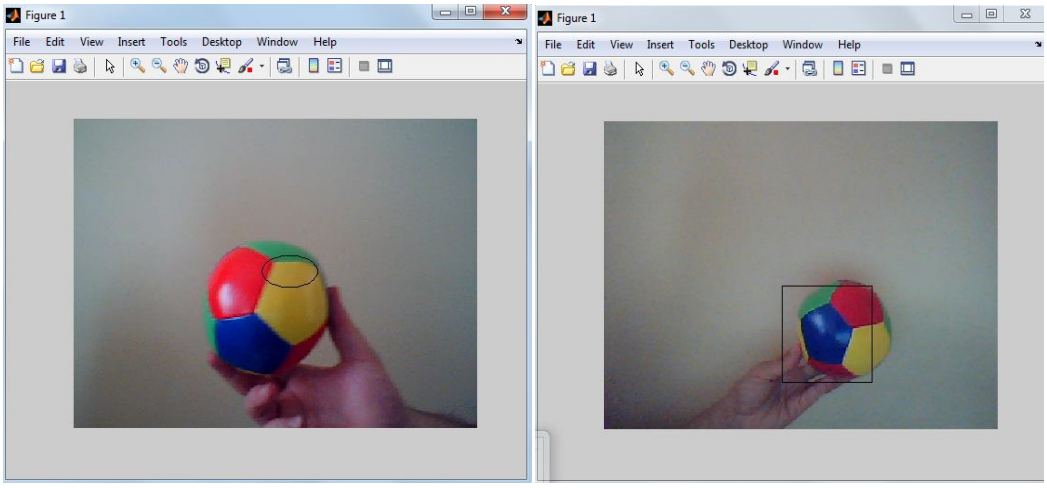
Farklı renklerde farklı boyutlarda çeşitli nesneler takip edildi ve algoritmaların hızında herhangibir değişikliğe rastlanmadı.

Kullanılan diğer yöntem olan optik akış yöntemi için Horn ve Schunck algoritması kullanıldı. Video akışında hareketli nesne takibinde Optik akış algoritması kullanıldığında nesne görüş alanından çıktığında ve tekrar göründüğünde başarılı bir izleme yapılamamaktaydı. Takip edilen nesne görüş alanından çıkıp tekrar göründüğünde veya nesne başka bir nesne ile örtüldüğünde Horn ve Schunck algoritması ile başarılı bir takip yapılabilmektedir. Horn ve Schunck algoritması hareket algılama tabanlı çalıştığından dolayı nesneyi takip etmek için hareket arar. Çok hızlı hareketlerde bile güvenilir takip yapılabilir. Şekil 4.2'de webcamdan alınan görüntüde hareketli nesne görüş alanından çıkıp yeniden göründüğünde takip işlemi devam etmektedir.



Şekil 4.2. Görüş alanından çıkan nesnenin Horn ve Schunck yöntemi ile takibi

Ayrıca takip için seçilen seçim çerçevesi ne kadar büyük olursa o kadar iyi takip işlemi her iki algoritma içinde iyi olmaktadır. Şekil 4.3'de hem ortalama değer kayması(a) hemde Horn ve Schunck algoritması(b) ile seçim çerçevesi büyük seçildiğinde takip işlemi gösterilmektedir.



(a) (b)

Şekil 4.3. (a) ve (b) algoritmaları ile nesnenin takibi

Yapılan çalışmanın en büyük avantajı gerçek zamanda çalışıyor olabilmesidir. Gerçek zamanda çalışan birçok sistemde nesne takibi bu yöntemler kullanılarak yapılabilir. Farklı renklerdeki farklı ortamlardaki ve farklı şekillerdeki her nesne bu algoritmalar ile birbirine yakın hızlarda izlenebilir.

5. ÖNERİLER

Tez çalışmasında iki farklı nesne izleme yöntemi kullanılmıştır. Ortalama Değer Kayması yöntemi, nesnenin renk dağılımına dayalı bir yöntem izlediği için, takip edilen nesne ve arkaplan renginin benzerliği durumlarında izleme tam başarı sağlayamadığı için renk dağılımına ek olarak hareket dağılımı bilgisi de göz önünde bulundurularak daha güvenilir nesne takibi yapılmıştır. Kullanılan yöntem, nesne ile bulunduğu ortamın aynı renk yoğunluğuna sahip olduğu durumlarda, farklı şekil ve renkteki nesneleri aynı hızda izleyerek başarılı bir sonuç vereceğinden birçok alanda nesne takibi güvenle yapılabilir. Nitekim bu yönde birçok çalışma yapılmıştır. Akılı güvenlik sistemlerinde en iyi doğrulukta insan tanıma işlevi bu alanda yapılan çalışmalardan yalnızca birisidir.

Çalışmada kullanılan ikinci uygulama Optik Akış yönteminin genel fark teknikleridir. Fark tekniklerinde karşılaşılan ana sorunlar arasında süreksizlik yer almaktadır. Fark teknikleri süreksiz nesneleri (örneğin, hareketi sırasında başka bir nesne tarafından örtülen nesneler) istenildiği gibi elde edemez ve bu nedenle hız tahmini algoritmaları tek başına yetersiz kalır. Bu tür sorunlar için nesne tanıma gibi ek algoritmalar kullanılması gerekmektedir. Görüş alanı içinde birden çok nesne takibi yapılacaksa eğer bu nesneler görüş alanı içerisinde aynı derinlikte hareket etmemeleri nedeniyle hız tahmini algoritmasına ek olarak derinlik tahmini de göz önüne alınırsa daha iyi sonuçlar elde edilecektir. Ayrıca hız tahmini algoritmalarında piksellerin parlaklık genliği temel alındığından gürültü etkisi yumuşatma yapılarak giderilmelidir fakat bu çözümde de küçük hareketlerin algılanmasını engelleyecektir.

İlerki çalışmalarda her iki yöntemi birlikte hibrit olarak kullanarak yeni bir nesne takip yöntemi önerilebilir. Bu sayede her iki yöntemin avantajlı yanları sadece kullanılabilir. Gerçek zamanda çalışabilen hem hızlı hem renk dağılımına uyan hemde hareketi çok iyi algılayabilen yeni algoritmalar hazırlanabilir.

KAYNAKLAR

[1] http://en.wikipedia.org/wiki/Video_tracking , Video izleme, 9 Kasım 2009.

[2] Sheskin, D.J., 2003. Handbook of Parametric and Nonparametric Statistical Procedures, Chapman & Hall/CRC.

[3] Ateş S. ve Demir E., 2009. Uzaktan Algılamada Çözünürlüğe Bağlı Veri Kazanımı Potansiyeli, TMMOB Harita ve Kadastro Mühendisleri Odası 12. Türkiye Harita Bilimsel ve Teknik Kurultayı , 2.

[4] Ertürk S. ve Urhan O., 2004-2005. İmge İşleme, Kocaeli Üniversitesi Mühendislik Fakültesi Elektronik ve Haberleşme Mühendisliği Bölümü ders notları.

[5] Pitas I., 2000. Digital Image Processing Algorithms and Applications, John Wiley&Sons, Inc., New York, Aristotale University of Thessalariki.

[6] Altuntaş, C. ve Çorumoğlu, Ö., 2002. Uzaktan Algılama Görüntülerinde Digital Görüntü İşleme ve RSImage Yazılımı, Selçuk Üniversitesi Jeodezi ve Fotogrametri Mühendisliği Öğretiminde 30. Yıl Sempozyumu, Konya.

[7] Xiuping J., Richards, J.A., 1999. Remote Sensing Digital Image Analysis.

[8] Sunar, F.,2000. İstanbul Teknik Üniversitesi Yayınlanmamış Ders Notları.

[9] Erdas Corp., 1991. ERDAS Field Guide, Second Edition, Version 7.5., England.

[10] Maillet, S.M., Sharaiha, Y. M., 1999. Binary Digital Image Processing, 3nd edition., Portland.

[11] Mather, P.M., 1996. Computer Processing of Remotely-Sensed Images, England.

[12] Mangalhaes, J., Ruger, S., Mining Multimedia Slient Concepts for Incremental Information Extraction, Imperial Colege London Department of Computing South Kensington Campus London, UK.

[13] Gonzales R.C. , Woods R.E., 2002. Digital image processing, Prentice Hall.

[14] Rosin P.L., Ellis T., 1995. Image Difference Threshold Strategies and Shadow Detection, in Proeeding.British Machine Vision Conference.

[15] Ekinci M., Nabiyev V. V., 2001. Trafik Görüntü Dizisinde Araçların Sınıflandırılması ve İzlenilmesi, 9.Sinyal İşleme ve Uygulamaları Kurultayı.

[16] Gutshess D., Trajkovic M., Cohen-Sola E., Lyons D., Jain A. K., 2001. A Background Model Initialization Algorithm for Video Surveillance, IEEE International Conference on Computer Vision.

[17] Collins, R. T., Lipton, A. J., Fujiyoshi, H. and Kanade, T., 2001. Algorithms for Cooperative Multisensor Surveillance, Proceeding of IEEE, 10,89.

[18] Haritaoğlu, I., Harwood, D. and Davis, L. S., 2000. W4: Real-Time Surveillance of People and Their Activities, IEEE Tranactions on Pattern Analysis and Machine Intelligence, 8,10.

[19] Toyama, K., Krumn, J., Brumit, B. and Meyers, B., 1999. Wallflower: Principles and Practice of Background Maintenance, 7th IEE International. Conference on Computer Vision, November.

[20] Talu, M.F., Türkoğlu, İ. ve Cebeci, M., 2010. Ortalama Kayma Algoritması Kullanılarak Gerçek Zamanlı Çekirdek Tabanlı Nesne Takibi, IEEE 18.Sinyal işleme ve iletişim uygulamalari kurultayi, Diyarbakir.

[21] Grimson W., Stauffer C., Romano R. and Lee L., 1998. Using Adaptive Tracking to Classify and Monitor Activities in a Site, in Proceeding of IEEE Conerence. on Computer Vision and Reconition.

[22] Vass, J., Palaniappan, K. and Ahuang, X., 1998. Automatic Spatio-Temporal Video Sequence Segmentation, in Proeeding IEEE International Conference on Image Processing.

[23] Bettencourt, K. and Somers, D. C., 2007. Effects of task difficulty on multiple object tracking performance, Journal of Vison, 7(9),898.

[24] Yilmaz, A., Javed, O., and Shah, M., 2006. Object Tracking: A Survey, ACM Comput. Surv., 38(4),13.

[25] Doucet, A., Johansen, A.M., 2008. A tutorial on particle filtering and smoothing: fifteen years later. Technical report, Department of Statistics, University of British Columbia.

[26] Dong, Y. and DeSouza, G.N., 2009. A new hierarchical particle filtering for markerless human motion capture, Computational Intelligence for Visual Intelligence, IEEE Workshop on, pp14 – 21.

[27] Xue J., Zheng N., Geng J. and Zhong X., 2008. Tracking multiple visual targets via particle-based belief propagation, IEEE Trans Syst Man Cybern B Cybern. 38(1),196-209.

[28] Leven, W. F. and Lanterman, A.D., 2009. Unscented Kalman Filters for Multiple Target Tracking With Symmetric Measurement Equations, Proceedings of SPIE, the International Society for Optical Engineering, 5810,56-67.

[29] Smal, K., Draegestein, N. Galjart, W. Niessen, E. Meijering, 2008. Particle Filtering for Multiple Object Tracking in Dynamic Fluorescence Microscopy Images: Application to Microtubule Growth Analysis, IEEE Transactions on Medical Imaging, 27(6),789-804.

[30] Martinez, W. L., Martinez, A. R., 2002. Computational Statistics Handbook with MATLAB, Chapman & Hall/CRC. Chapter 8.

[31]http://school.maths.uwa.edu.au/~duongt/seminars/intro2kde. 03/02/2008.WEB, Ushakov, 2001.

[32] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/ AV04 05/MISHRA/kde.html, 03/02/2008.

[33] Raykar, V. C., Duraiswami, R. 2006. Fast optimal bandwidth selection for kernel density estimation, Proceedings of the sixth SIAM International Conference on Data Mining, pp. 524-528.

[34] Munder, S., Schnorr, C., Gavrila, D.M., 2008. Pedestrian Detection and Tracking Using a Mixture of View-Based Shape–Texture Models, IEEE Transactions on Intelligent Transportation Systems, 9(2), 333 – 343.

[35] Cordea, M. D., Petriu, E. M., Petriu, D. C., 2008. Three-Dimensional Head Tracking and Facial Expression Recovery Using an Anthropometric Muscle-Based Active Appearance Model, IEEE Transactions on Instrumentation and Measurement, 57(8), 1578 – 1588.

[36] Bertalmio, M., Sapiro, G., and Randall, G., 2000. Morphing active contours. IEEE Trans. Patt. Analy. Mach. Intell. 22, 7, 733–737.

[37] Lee, S., Kang, J., Shin, J., Paik, J., 2007. Hierarchical active shape model with motion prediction for real-time tracking of non-rigid objects.

[38] Talu, M. F., 2010. Nesne Takip Yöntemlerinin Sınıflandırılması, İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, yıl:9, sayı:18, s.45-63.

[39] Özüntürk, E., 2007. Optik Akış ile Hareket Tespiti, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[40] Horn, B.K.P. and Schunck, B.G. , 1981. Determining Optical Flow, Artificial Intelligence, 17, 185-204.

[41] Barron, J.L. and Thacker, N.A., 2004. Tutorial: Computing 2D and 3D Optical Flow, Tina Memo No.012.

[42] Lucas, B.D. and Kanade, T., 1981. An Iterative Image Registration Technique with an Application to Stereo Vision, DARPA Image Understanding Workshop, pp121-130.

[43] Lucas, B.D., 1984. Generalized Image Matching by the Method of Differences, doctoral dissertation.

[44] http://en.wikipedia.org/wiki/File:Opticfloweg.png, Optik Akış. 07/10/2011.

[45] Fukunaga, K. and Hostetler, L., 1975. Density Function,with Applications in Pattern Recognition, IEEE Trans. Info. Theory 21:32-40.

[46] Cheng, Y., 1995. Mean Shift, Mode Seeking and Clustering, IEEE Trans. PAMI, 17:790-799.

[47] Comaniciu, D. and Meer, P., 1997. Robust Analysis of Feature Spaces: Color Image Segmentation, IEEE Conf. Comp. Vis. and Pattern Recogn., 750-755.

[48] Comaniciu, D. and Ramesh, V., 2000. Robust Detection and Tracking of Human Faces with an Active Camera ,Proc.3 rd IEEE Intrnl. Workshop. Visual Surveillance, pp.11–18.

[49] Özden,M., 2005. Ortalama Kayma Algoritmasının Geliştirilerek Görüntü Dizilerinde Hareketli Nesne Takibi ve Görüntü Kesimleme Amaçlı Kullanılması, Yüksek Lisans Tezi, Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü, Kırıkkale.

EKLER

Nesne izleme programı bir ana bölüm ve dokuz fonksiyondan meydana gelmektedir.

Main.m

```
clear all,clc, close all

vidobj = videoinput('winvideo', '1','YUY2_320x240');
set(vidobj,'ReturnedColorSpace','rgb');
set(vidobj,'TriggerRepeat',Inf);
triggerconfig(vidobj, 'manual');

m=5;
flag=false;
start(vidobj)

izlemeMetodu=input('İzleme Metodunu Giriniz: ( 1- Mean shift 2-Optical Flow
)\n');

ax=image('DeleteFcn','stop(vidobj)');
sz=get(vidobj,'VideoResolution');
set(ax,'CData',zeros(sz)');
axis off, axis image, axis ij
ln = line; text_handle=text('Position',[sz(1) 1]);
FPS=zeros(150,1);
eskiResim=zeros(get(vidobj,'VideoResolution'))';
for j=1:150
    tic
    simdikiResim = getsnapvuruş(vidobj);
    set(ax,'CData',simdikiResim)
    drawnow
    if strcmp(get(findobj('Type','figure'),'SelectionType'),'alt')

        rect=getrect;
        rect=ceil(rect);
        template = rgb2gray(imcrop(simdikiResim,rect));

        elps=rect(4:-1:3)/2;
        elpsmerkez=rect(2:-1:1)+elps;
        I_bined =
uint8(reshape(griformAl(simdikiResim(:),m),size(simdikiResim,1),[]));
        [SatirSutun, yeniyer] = elipsciz(elpsmerkez, elps, size(simdikiResim));
        secim = secimAl(simdikiResim, elpsmerkez, elps, m);
        index = 2;
        elpsmerkez1 = elpsmerkez;
        elpsonce = elps;
        elpssimdi = elps;


        merkez(1,1)=rect(1)+rect(3)/2;
        merkez(1,2)=rect(2)+rect(4)/2;
        obje = double(template);
        obje2=simdikiResim;
        flag=true;
    end
    if flag
        guncelResim = simdikiResim;
        if izlemeMetodu==1
            ImgSize = size(guncelResim);
            [elpsmerkez1] = meanShift(elpsmerkez1, elpssimdi, secim, guncelResim,
m);
            index = index + 1;
```

```
            [SatirSutun, yeniyer] = elipsciz(elpsmerkez1, elpssimdi,
size(guncelResim));
            set(ln,'XData',SatirSutun(:,2),'Ydata',SatirSutun(:,1))
        elseif izlemeMetodu==2
            resimler(:,:,1)=double(rgb2gray(eskiResim));
resimler(:,:,2)=double(rgb2gray(guncelResim));
            [Vx,Vy] = optikakis(resimler,100,1);
            th=0.1; c=sqrt(Vx.^2+Vy.^2)>th;
            [iy ix]=find(c);
            merkez=[mean(ix(:)) mean(iy(:))];
            p1=merkez-rect(3:4)/2; p2=p1+[rect(3) 0]; p3=p1+[0 rect(4)];
p4=p1+rect(3:4);
            set(ln,'XData',[p1(1) p2(1) p4(1) p3(1) p1(1)],...
                   'YData',[p1(2) p2(2) p4(2) p3(2) p1(2)])
        end
        drawnow
    end
    eskiResim=simdikiResim;
    tm=toc;
    FPS(j)=ceil(1/tm);
end
algoritma{1}='MeanShift';
algoritma{2}='Optic Flow';
disp('Ortalama FPS=')
ortalamaFPS=mean(FPS)
```

meanshift.m

```
function [y1] = meanShift(y0, elpssimdi, secim, guncelResim, m)
ImgSize = size(guncelResim);
resimvar = 0;
if (resimvar == 1)
    f = figure;
end
devam = true;
iterasyonsayisi = 0;
while (devam == true && iterasyonsayisi < 20)
    [y0_SatirSutun, y0_yer] = elipsAl(y0, elpssimdi, ImgSize);
    [p_y0, griform] = histogramAl(elpssimdi, y0, y0_SatirSutun,
guncelResim(y0_yer), m);
    if (resimvar == 1)
        plot(q,'r'); hold on;
        plot(p_y0,'b'); hold off;
        pause
    end
    [w] = agirliklar(secim, p_y0, griform);
    sum_w = sum(w);
    y1 = [sum(y0_SatirSutun(:,1).*w)/sum_w sum(y0_SatirSutun(:,2).*w) / sum_w];
    [y1_SatirSutun, y1_yer] = elipsAl(round(y1), elpssimdi, ImgSize);
    y1_yer=y1_yer(y1_yer>0);
    y1_yer=y1_yer(y1_yer<=numel(guncelResim));
    [p_y1, griform] = histogramAl(elpssimdi, y1, y1_SatirSutun,
guncelResim(y1_yer), m);
    if (sum(sqrt(p_y1.*secim)) > sum(sqrt(p_y0.*secim)))
        if (max(abs(y0-y1)) > 0.5)
            y0 = round(y1);
            devam = true;
        else
            y1 = round(y1);
            devam = false;
        end
    else
       y1 = y0;
       devam = false;
    end
    iterasyonsayisi = iterasyonsayisi + 1;
```

```
end
y1 = round(y1);
if (resimvar == 1)
    close(f)
end
```

optikakis.m

```
function [Vx,Vy] = optikakis(images,alpha,iterasyon)
[yukseklik,genislik,framesayisi]=size(images);

Vx = zeros(yukseklik,genislik);
Vy = zeros(yukseklik,genislik);

for k = 1:framesayisi-1
  Ix = zeros(yukseklik-1,genislik-1,framesayisi-1);
  Iy = zeros(yukseklik-1,genislik-1,framesayisi-1);
  It = zeros(yukseklik-1,genislik-1,framesayisi-1);

  for x = 2:genislik-1
    for y = 2:yukseklik-1
      Ix(y,x,k) = (images(y+1,x+1,k  )-images(y+1,x,k  )+...
                   images(y  ,x+1,k  )-images(y  ,x,k  )+...
                   images(y+1,x+1,k+1)-images(y+1,x,k+1)+...
                   images(y  ,x+1,k+1)-images(y  ,x,k+1))/4;

      Iy(y,x,k) = (images(y,x  ,k  )-images(y+1,x  ,k  )+...
                   images(y,x+1,k  )-images(y+1,x+1,k  )+...
                   images(y,x  ,k+1)-images(y+1,x  ,k+1)+...
                   images(y,x+1,k+1)-images(y+1,x+1,k+1))/4;

      It(y,x,k) = (images(y+1,x  ,k+1)-images(y+1,x  ,k)+...
                   images(y  ,x  ,k+1)-images(y  ,x  ,k)+...
                   images(y+1,x+1,k+1)-images(y+1,x+1,k)+...
                   images(y  ,x+1,k+1)-images(y  ,x+1,k))/4;
    end
  end

  for nn = 1:iterasyon
    for x = 2:genislik-1
      for y = 2:yukseklik-1
        Vxbar = (Vx(y-1,x)+Vx(y,x+1)+Vx(y+1,x)+Vx(y,x-1))/6+...
          (Vx(y-1,x-1)+Vx(y-1,x+1)+Vx(y+1,x+1)+Vx(y+1,x-1))/12;

        Vybar = (Vy(y-1,x)+Vy(y,x+1)+Vy(y+1,x)+Vy(y,x-1))/6+...
          (Vy(y-1,x-1)+Vy(y-1,x+1)+Vy(y+1,x+1)+Vy(y+1,x-1))/12;

        temp = (Ix(y,x,k)*Vxbar+Iy(y,x,k)*Vybar+It(y,x,k))/(alpha^2 + Ix(y,x,k)^2
+ Iy(y,x,k)^2);

        Vx(y,x) = Vxbar-Ix(y,x,k)*temp;
        Vy(y,x) = Vybar-Iy(y,x,k)*temp;
      end
    end
  end

end
```

.....

agirliklar.m

elipAl.m

elipsciz.m

griformAl.m

histogramAl.m

kernelAl.m

secimAl.m

Kullanılan diğer fonksiyon dasyalarıdır. Kodlar uzun olduğu için Ekler bölümünde sadece 3 ana fonksiyon gösterilmiş diğerleri ise CD formatında enstitüye sunulmuştur.

ÖZGEÇMİŞ

01/01/1983 Elazığ doğumluyum. İlköğrenimimi 1989-1997 yılları arasında Elazığ Atatürk İlköğretim Okulunda tamamladım. Orta öğrenimime Elazığ Yabancı Dil ağırlıklı Balakgazi Lisesinde 1997-2001 yılları arasında devam edip 2001 yılında mezun oldum. 2002 yılında Elazığ Fırat Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümünde lisans eğitimime başlayıp 2006 yılında tamamladım. 2007 Bahar Döneminde Elazığ Fırat Üniversitesi Fen Bilimleri Enstitüsü Yazılım Ana Bilim Dalında yüksek lisans eğitimime başladım ve devam etmekteyim.

17/10/2007 tarihinde Ankara Tapu Kadastro Genel Müdürlüğü Strateji Geliştirme Daire Başkanlığı Yönetim Bilgi Sistemleri BİM şubesinde sözleşmeli programcı olarak göreve başladım. Daha sonra sınavını kazandığım Maliye Bakanlığı Strateji Geliştirme Daire Başkanlığı Yönetim Bilgi Sistemleri BİM şubesinde sözleşmeli programcı olarak görevime devam ettim. 19/12/2009 tarihinde Gümrük Müsteşarlığı Muhabere ve Elektronik Daire Başkanlığında sözleşmeli çözümleyici olarak çalıştım. Son olarak 15/07/2010 tarihinde PTT Genel Müdürlüğü Teknik İşler ve Otomasyon Daire Başkanlığına Bilgisayar Mühendisi olarak atandım ve çalışmaktayım.