HAREKET VEKTÖRLERİ KULLANILARAK NESNE TAKİP ETME

Handenur DEMİRCİ

Yüksek Lisans Tezi
Elektrik ve Elektronik Mühendisliği Anabilim Dalı
Elektronik Bilim Dalı
Yrd. Doç. Dr. Emin Argun ORAL
2014

ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

HAREKET VEKTÖRÜ KULLANILARAK NESNE TAKİP ETME

Handenur DEMİRCİ

ELEKTRİK ve ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
Elektronik Bilim Dalı

ERZURUM
2014

T.C.
ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEZ ONAY FORMU

HAREKET VEKTÖRÜ KULLANILARAK NESNE TAKİP ETME

Yrd. Doç. Dr. Emin Argun ORAL danışmanlığında, Handenur DEMİRCİ tarafından hazırlanan bu çalışma 14/08/2014 tarihinde aşağıdaki jüri tarafından Elektrik ve Elektronik Mühendisliği Anabilim Dalı – Elektronik Bilim Dalı'nda Yüksek Lisans Tezi olarak **oybirliği** ile kabul edilmiştir.

Başkan : Yrd. Doç. Dr. Emin Argun ORAL *İmza* :

Üye : Yrd. Doç. Dr. Bülent ÇAVUŞOĞLU *İmza* :

Üye : Yrd. Doç. Dr. Gökay AKKAYA *İmza* :

Yukarıdaki sonuç;

Enstitü Yönetim Kurulu 21/08/2014 tarih ve 13/1041 nolu kararı ile onaylanmıştır.

Prof. Dr. İhsan EFEOĞLU
Enstitü Müdürü

Not: Bu tezde kullanılan özgün ve başka kaynaklardan yapılan bildirişlerin, çizelge, şekil ve fotoğrafların kaynak olarak kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

ÖZET

Yüksek Lisans Tezi

HAREKET VEKTÖRÜ KULLANILARAK NESNE TAKİP ETME

Handenur DEMİRCİ

Atatürk Üniversitesi
Mühendislik Fakültesi
Elektrik ve Elektronik Mühendisliği Anabilim Dalı
Elektronik Bilim Dalı

Danışman: Yrd. Doç. Dr. Emin Argun ORAL

Nesne takip etme algoritmaları temel olarak nesnenin bulunduğu konumu bütün çerçeve (frame) içinde bularak aynı nesnenin bir sonraki çerçevelerde bulunduğu konumun yeniden bulunması prensibine dayanır. Bu amaçla kullanılan nesneyi bulma ve nesneyi takip etme algoritmaları birbirinden farklı problemlerdir. Bu çalışmada her ikisi de yer almıştır.

Nesne bulma probleminde nesneyi belirgin bir şekilde işaretlemek takip eden adımlarda nesneyi bulmayı kolaylaştıracağı için önemlidir. Bu amaçla Canny kenar/köşe bulma algoritması kullanılmıştır.

Bulunan nesneyi takip etmede ise hareket vektörleri kullanılmıştır. Bu amaçla bir önceki çerçevede bulunan hareket vektörleri bilgisi bir sonraki çerçeveye aktarılarak nesnenin olabileceği alan belirlenmiştir. Nesnenin olabileceği alan içerisinde yapılan spiral tarama ile nesne etkin bir biçimde bulunarak takibi sağlanmıştır. Bu amaçla, tek nesne takibinde hareket vektörleri kullanımı ile nesnenin takibi kolaylıkla gerçekleştirebilir. Ancak çoklu nesne takibinde bulunan hareket vektörlerinin nesnelerle eşleştirilmesi gerekmektedir.

Bu çalışmada tekli ve çoklu nesneler için önce nesne bulma işlemi gerçekleştirilmiş daha sonra da belirlenen hareket vektörleri yardımıyla nesnenin takibi sağlanmıştır.

2014, 73 sayfa

Anahtar Kelimeler: Canny algoritması**,** Hareket Vektörü, Spiral tarama, Çoklu Nesne Takibi, MSE,

ABSTRACT

Master Thesis

OBJECT TRACKING WITH USING MOTION VEKTOR

Handenur DEMİRCİ

Atatürk University
Faculty of Engineering
Department of Electric and Electronic Engineering
Department of Electronic

Supervisor: Asst. Prof. Dr. Emin Argun ORAL

Object tracking algorithms basically rely on detecting the object position on the whole frame and repeating this operation on the following frames. Object tracking and object detection algorithms used for this purpose are different from each other, and they both are used in this study.

In object detection problem, it is important to mark the object clearly so that the object can be easily detected in the following steps. For this purpose Canny edge/corner detection algorithm is used.

Motion vectors are used to track the detected object. For this purpose, the motion vectors estimated in a previous frame are passed to the next frame to define the region in which the object may exist. Then the object can be detected efficiently by performing a spiral scan over this region. For this purpose, the use of motion vectors enables an easy tracking of a single object. But in multi-object tracking, it is important to match the obtained motion vectors with the objects.

In this study, object detection is performed for single and multi-objects, and then object tracking is enabled by the use of obtained motion vectors.

2014, 73 pages

Keywords: Canny algorithm**,** Motion Vector, Spiral Scan, Multiple Object Tracking, MSE

TEŞEKKÜR

Yüksek lisans eğitimim ve bu tez çalışmamda beni her konuda yönlendiren, ilgisini ve desteğini esirgemeyen değerli danışmanım Sayın Yrd. Doç. Dr. Emin Argun ORAL'a en içten teşekkür ve saygılarımı sunarım.

Yüksek Lisans ders eğitimim boyunca burs desteği sağlayan Tübitak Münir Birsel Vakfı'na, çalışmalarım boyunca her türlü bilgisinden faydalandığım değerli hocam Sayın Yrd. Doç. Dr. Bülent ÇAVUŞOĞLU'na, beni bu günlere getiren AİLEME, sevgili hayat arkadaşım Alperen AYDIN'a, yardımlarını esirgemeyen dostlarıma ve iş arkadaşlarıma teşekkür ederim.

Handenur DEMİRCİ
Ağustos, 2014

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler

C	Yapı Matrisi
D	Durum Matrisi
I	Çerçeve (Frame)
K	Kalman Kazancı
M	Ölçme Matrisi
N	Beyaz Gürültü
t	Zamanın bir anı
W	Gauss Gürültüsü
X	Konum Matrisi
x ve y	Piksel Değerleri
Z	Güncel Durum Matrisi
λ_1 ve λ_2	Eigenvalue Değerleri

Kısaltmalar

RGB	Kırmızı, Yeşil, Mavi
HSV	Renk özü, Doygunluk, Değer
Th	Eşik Değer
MV	Hareket Vektörü
DMA	Değişik Hareket Metodu
BMA	Blok Eşleme Algoritması
MPEG	Hareketli Görüntü Uzmanları Birliği
FS	Tam Tarama
SS	Spiral Tarama
MSE	Ortalama Karesel Hata
MAD	Ortalama Mutlak Değer
NTD	Eşik Farklılığı Sayısı
KTC	Kondansatör Gürültüsü

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Nesne takibi, “bilgisayarlı görme (computer vision)” alanı içerisinde önemli bir göreve sahiptir. En basit tanımıyla nesne takibi, video kaydı içerisinde hareket etmekte olan bir nesnenin hareket yörüngesini tahmin etme problemidir. Diğer bir deyişle, bir nesne takip edicinin (object tracker), ardışık video çerçeveleri içerisinde hareket etmekte olan nesneleri algılayarak her bir nesneye eşsiz bir etiket verme işlemidir. (Anonymous 2014)

Hızlı veri işleme özelliğine sahip bilgisayarların artması, yüksek kaliteye sahip ucuz kameraların üretilmesi ve otomatik video analizi için gerekli olan ihtiyacın giderek artması nesne takibi algoritmalarına karşı olan ilgiyi oldukça arttırmıştır.

Nesne takibi ile ilgili literatürde doyurucu sayıda yaklaşım önerilmiştir. Önerilen bu yaklaşımların temel farkları aşağıdaki sorulara verilen yanıtlar yardımıyla ortaya çıkmaktadır.

- Nesne arka plandan nasıl ayrıştırılmalı ve nasıl işaretlenmelidir? Bu işlemler yapılırken nesnenin hangi özelliği kullanılmalıdır?
- Nesne bulunduktan sonra nesnenin bir sonraki çerçevede bulunduğu alan nasıl bulunmalıdır? Bu alan bulunurken zaman ve hız kavramları algoritmayı nasıl etkilemektedir?
- Bulunan hareketler gürültüden nasıl temizlenmelidir?
- Birden fazla nesne takip ediliyorsa nesnelerin hareketleri nasıl eşleştirilmelidir?

Bu sorulara verilen yanıtlar, nesne takip işleminin gerçekleştiği ortama bağlı olmakla birlikte tasarlanacak algoritma yapılarının da belirlenmesini sağlar.

Bu amaçla, nesnelerin ayırt edici özelliği (feature) olarak Canny kenar bulma algoritması nesneleri belirlemede (object detection) etkili bir yöntem olduğu için nesneyi (veya nesneleri) tanımlayıp, arka plandan ayırmak için kullanıldı. Ayrıca mean square error (MSE) temelli bir yaklaşım ile video çerçevelerinin (video frame) bazı bölümlerinin (blok) sıralı ve spiral tarama yöntemleri ile yer değiştirme bilgileri (hareket vektörü-motion vector) hesaplandı. Yaygın olarak kullanılan gürültü temizleme yaklaşımlarının bir arada kullanımı ile geliştirdiğimiz algoritmanın kullanımıyla bu bilgilerin verimliliğini artırıldı. Son olarak özellikle çoklu nesne takibinin söz konusu olduğu durumlarda hangi nesneye ait olduğu bilinmeyen hareket vektörleri, geliştirdiğimiz eşleştirme algoritmasının kullanımı ile her bir nesnenin ayrı ayrı hareket vektörü büyüklüğü tayini ile takip edildi.

2. KURAMSAL TEMELLER

2.1. Nesne Gösterimi

Bir nesneyi tanımlarken ve takip ederken daha iyi bir analiz için nesnenin dikkat çeken veya kolay bulunabilecek özelliklerinden faydalanılmalıdır. Örneğin; denizde bir gemi, akvaryumda bir balık, yolda bir araba, yürüyen insanlar, sudaki baloncuklar gibi belirli bir alandaki nesnelerin her biri farklı özellikleri göz önüne alınarak takip edilmelidir. Nesneler şekillerine, renklerine, dokularına ya da görünüşlerine göre tanımlanabilirler. Bu bölümde öncelikle yaygın olarak kullanılan nesne gösterim yöntemleri incelenmiştir.

2.1.1. Nokta modeli

Bir nesne nokta kullanılarak tanımlanacaksa nesnenin merkezine yada çevresine noktalar yerleştirilerek tanımlanabilir. Bu gösterim genel olarak bir çerçevedeki küçük bölgeleri işaretlemek için kullanılır.

2.1.2. Basit geometrik şekil modeli

Nesneyi işaretlemek için dikdörtgen veya elips gibi basit geometrik şekiller kullanılabilir. Bu gösterim genel olarak şekli değişmeyen ve çoğunlukla tek parçadan oluşan nesneleri işaretlemek için kullanılır.

2.1.3. Silüet veya kontur modeli

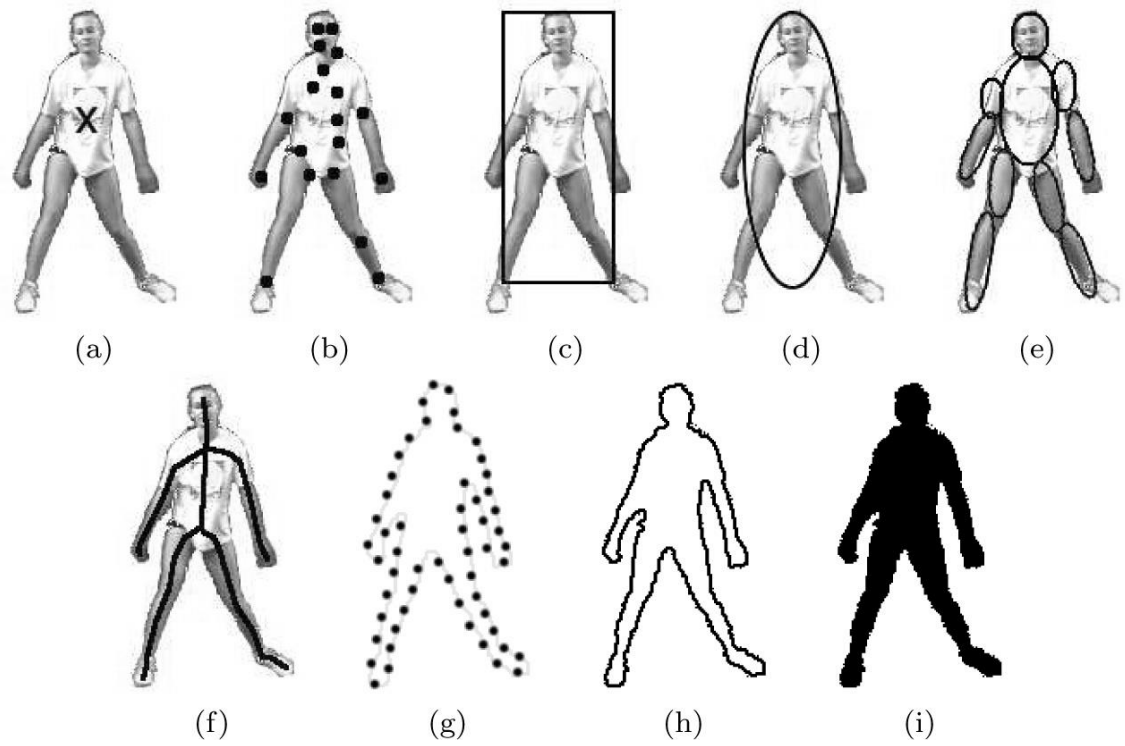
Nesnenin sınırlarını belirtmek için kontur yöntemi kullanılabilir. Bu gösterim genel olarak şekli değişen veya belirli bir şekli olamayan nesneleri işaretlemek için kullanılır.

2.1.4. Parçalı şekil modeli

Bu gösterim birden fazla parçadan oluşan nesnelerin her bir parçasını ayrı ayrı tanımlamak için kullanılır. Örneğin insan vücudunda gövdeyi, bacakları, elleri ayrı ayrı işaretlemek için kullanılır.

2.1.5. İskelet modeli

Nesnenin iskelet modelini belirlemek için kullanılır. Bu gösterim genel olarak daha önceden bilinen nesneleri işaretlemek için kullanılır.



Şekil 2.1. Nesne gösterim yöntemleri (Yılmaz *et al.* 2006)
(a) merkezi nokta gösterimi (b) çoklu nokta gösterimi (c) dikdörtgensel gösterim (d) eliptik gösterim (e) parça tabanlı gösterim (f) iskelet şeklinde gösterim (g,h) kontur modeli gösterimi (i) silüet gösterimi

2.2. Nesne Belirleme Yöntemleri

Nesneleri takip ederken kolay takip edilebilecek ve karışıklığı engelleyecek doğru özelliklerini (feature) seçmek önemli bir rol oynar. Genel olarak nesnenin en çok dikkat çeken özelliği kullanılmalıdır. Örneğin; renk; histogram tabanlı bir gösterim için, kenar-köşe; keskin şekilli bir gösterim için kullanılabilir. Farklı özellikleri birleştirerek tek bir takip algoritması oluşturmak ise en başarılı sonucu verecektir. Bu bölümde nesneyi belirlemek için belli başlı özellikler incelenecektir.

2.2.1. Renk ile nesne belirleme

Nesnenin belirgin rengine temel olarak 2 fiziksel faktör etki eder.

1) Parlaklığın spektral güç dağılımı
2) Nesne yüzeyinin yansıtma özellikleri

İmge işlemede RGB (kırmızı, yeşil, mavi) renk uzayı nesnenin rengini temsil etmek için kullanılır. Fakat RGB uzayı algılanabilen düz bir renk uzayı değildir. RGB uzayındaki renkler arasındaki farklar insanın algıladığı renk farklarına karşılık gelmez. Buna ek olarak RGB boyutları birbiriyle yüksek derecede ilişkilidir. Ayrıca, HSV (Hue, Saturation, Value) neredeyse düzgün bir renk uzayıyken, L*u*v ve L*a*b* , algısal olarak düz renk uzaylarıdır. Fakat bu renk uzayları gürültüye karşı hassastır.

Özet olarak hangi renk uzayının daha etkili olduğu konusunda bir son söz yoktur. Bu sebeple takip için birkaç farklı uzay kullanılmıştır. Bu bölümde RGB uzayını kullanılarak nesnelerin renkleri temsil edilmiştir.

Şekil 2.2. Renk özelliği kullanarak nesne belirleme

Şekil 2.3. Renk özelliği kullanarak nesne belirleme

Şekil 2.4. Renk özelliği kullanarak nesne belirleme

2.2.2. Kenar-köşe ile nesne belirleme

Bir nesnenin kenarları o nesneyi arka plandan ayıran sınırlarını tanımlar. Bir çerçevedeki kenarlar, güçlü yoğunluk zıtlığı olan bölgelerdedir. Bir yoğunluktan diğerine sıçrama noktasıdır. Bir çerçevenin kenarlarını bulmak, çerçevedeki önemli yapısal özellikleri korurken verinin büyük bir kısmını önemli ölçüde azaltır ve gereksiz bilgiyi filtreler. Bu bölümde nesnelerin kenar bulma yöntemleri incelenmiştir.

Kenar bulmanın birçok yolu vardır. Bu metotlar iki gruba ayrılır;

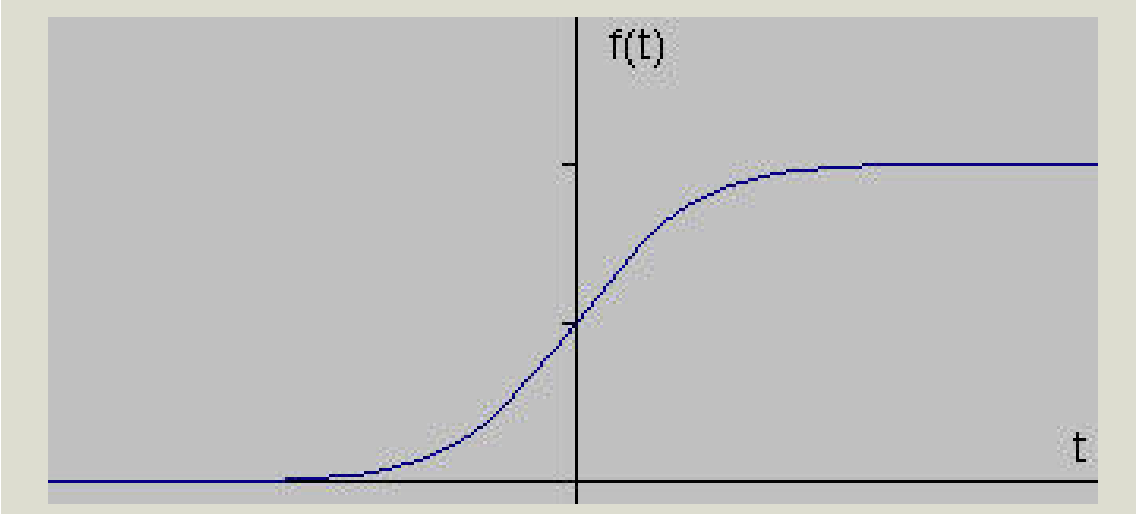
Gradyan (Eğim)
Laplasian metotları

Gradyan metodu; çerçevenin birinci türevindeki maksimum ve minimum değerlere bakma ile kenarları tespit eder.

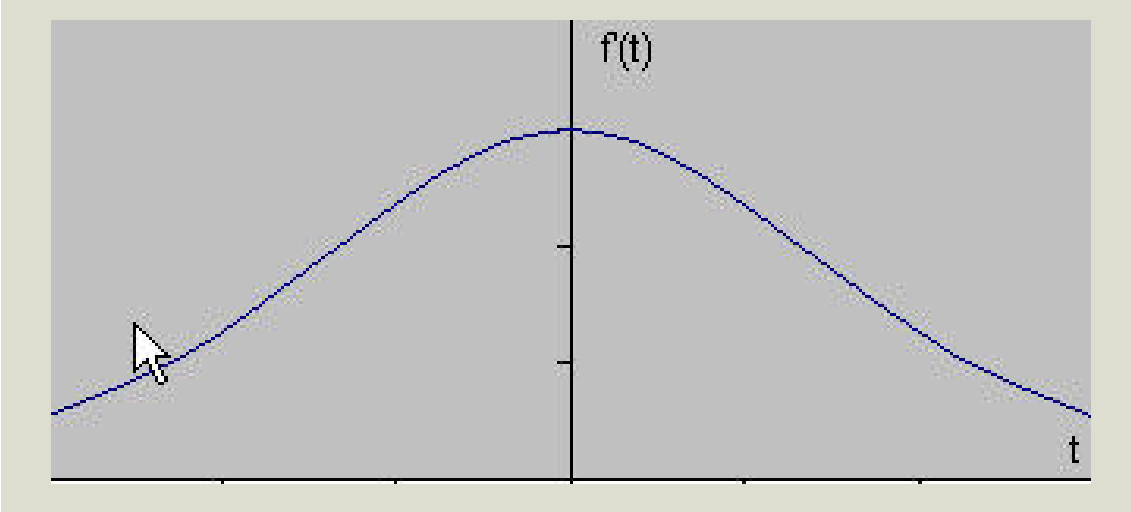
Laplasyan metodu; kenarları bulmak için ikinci türevdeki sıfır geçişlerini arar. Bir kenar, rampa fonksiyonunun bir boyutlu şekline sahiptir.

Şekil 2.5'deki sinyali göz önünde bulunduralım. Yoğunluklar arasındaki sıçrama kenarı verir.

Şekil 2.5'deki işaretin türevi (t'ye göre türevi alınırsa) Şekil 2.6'daki işareti verir



Şekil 2.5. Rampa fonksiyonu (Kazan 2011)



Şekil 2.6. Rampa fonksiyonunun türevi (Kazan 2011)

Türev işlemi orijinal sinyaldeki kenarın merkezindeki maksimum değeri gösterir. Birinci türeve dayanan bu kenar tespiti yöntemi “gradyan filtre” metodunun karakteristik özelliğidir. Canny kenar bulma algoritması Gradyan filtre metodunu kullanan bir yöntemdir.

Pikselin gradyan değeri eşik değerini aşarsa bu piksel nesnenin kenarını gösterir. Kenarları belirten piksellerin renk değerleri kenarın dışındaki piksellerden daha büyüktür.

Laplasyan türev operatörü, çerçevedeki sürekli olmayan grilik seviyelerini belirginleştirecek, grilik seviyesinin çok yavaş değiştiği bölümleri ise azaltacaktır. Dolayısıyla Laplasyan operatörü bir çerçeveye uygulandığında çıkışta kenar çizgileri, sürekli olmayan grimsi noktalar ve siyah arka plana sahip bir çerçeve elde edilir. Laplasyan operatörünün dezavantajı çerçevedeki gürültüye çok duyarlı olmasıdır. Diğer bir dezavantajı ise her bir kenar için iki değer üretmesidir. Gürültünün meydana getireceği bozulmaları engellemek için Laplasyan'dan önce çerçeveye Gauss kerneli uygulanır. Gauss kerneli bir alçak geçiren filtredir. İşlem sonucunda çerçeve bulanıklaşır. Ardından Laplasyan uygulanarak sıfır kesişleri tespit edilir.

Bir fonksiyona ait birinci türev; (Gradyan Metot)

- Çerçevedeki kalın kenarları,

Bir fonksiyona ait ikinci türev; (Laplasyan Metot)

- Çerçevedeki ince kenar, gürültü noktaları gibi ince detayları belirtir.

Görüldüğü gibi ikinci türev birinci türeve göre çerçevedeki ince detayları daha iyi yakalanmasını sağlar. Doğru bir yaklaşımla, bir boyutlu analiz temeline dayanarak iki boyutlu bir çerçevenin türevi hesaplanabilir. Sobel işlemi bir çerçevede 2-boyutlu özel bir eğim (gradyan) ölçümü gerektirir. Bu işlem çerçevede her bir noktada kesin eğim genliğini bulmak için kullanılır.

Sobel kenar algılayıcı; biri x-yönündeki eğimi, diğeri y-yönündeki eğimi hesaplamak üzere 3x3'lük bir konvolüsyon maske çifti kullanır. Konvolüsyon maskesi çerçeve boyutundan daha küçük olur. Mantık olarak bir kerede bir pikseli işleyerek maske çerçevede hareket eder. Çerçevenin üst köşesine yerleştirilen maske, merkezine denk gelen pikselin değerini değiştirerek bir sonraki piksele geçer.

Satır, sütun değerlerini ifade eden değerlerinin değiştirilmesi ile maske hareket ettirilir. 3x3'lük maske ile ilk ve son satır ve ilk ve son sütundaki piksellerin işlenemeyecektir. Eğer maske ilk satırdaki piksele yerleştirilirse, çerçeve sınırlarının dışına taşacaktır. Bu durumda çerçeveye birer satır ve sütun eklenerek bu problem ortadan kaldırılabilir.

2.2.2.a. Canny kenar detektörü

Canny kenar detektörü çok doğru bir şekilde kenarları tespit edebilen ve çok sık kullanılan bir algoritmadır. Bu bölümde algoritmanın nasıl çalıştığı incelenecektir.

Algoritma:

Kenarları bulunacak çerçeve eğer renkli ise öncelikle siyah-beyaz çerçeveye dönüştürülmelidir. Elde edilen siyah-beyaz çerçevedeki gürültünün yok edilmesi için çerçeveye Gauss filtresi uygulanmalıdır. Şekil 2.7'de 5×5 lik bir gauss filtresi verilmiştir.

$$\frac{1}{159} * \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Şekil 2.7. 5×5'lik Gauss filtresi

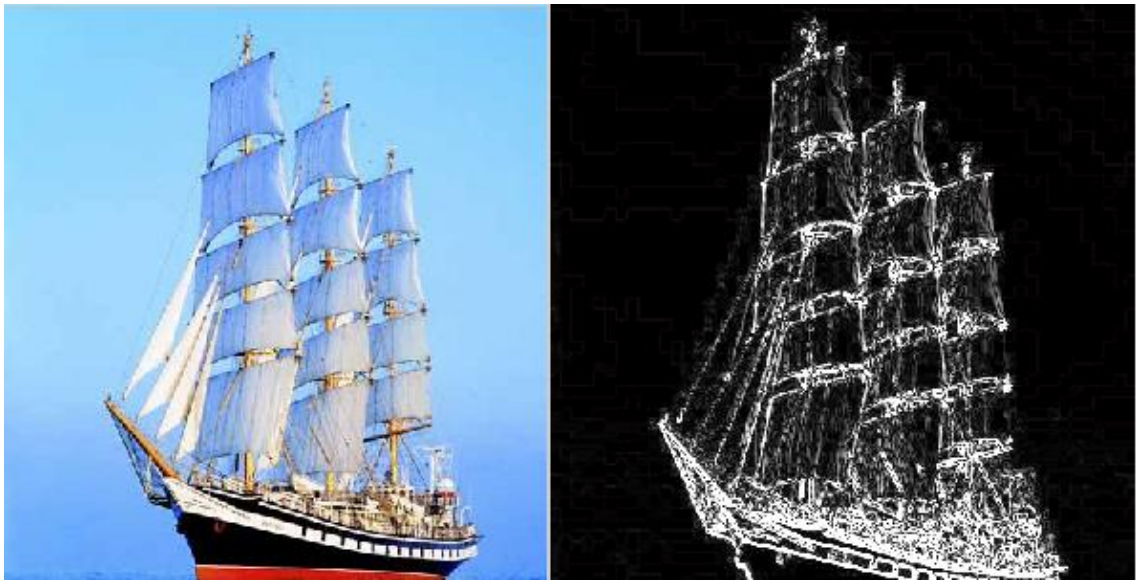
Gauss filtresi uygulanan çerçeveye hem x hem y ekseninde türev alma işlemi yapan Sobel filtresi uygulanmalıdır. Uygulanan bu filtre sonucunda değişimi belirten gradyan hesaplanır.

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

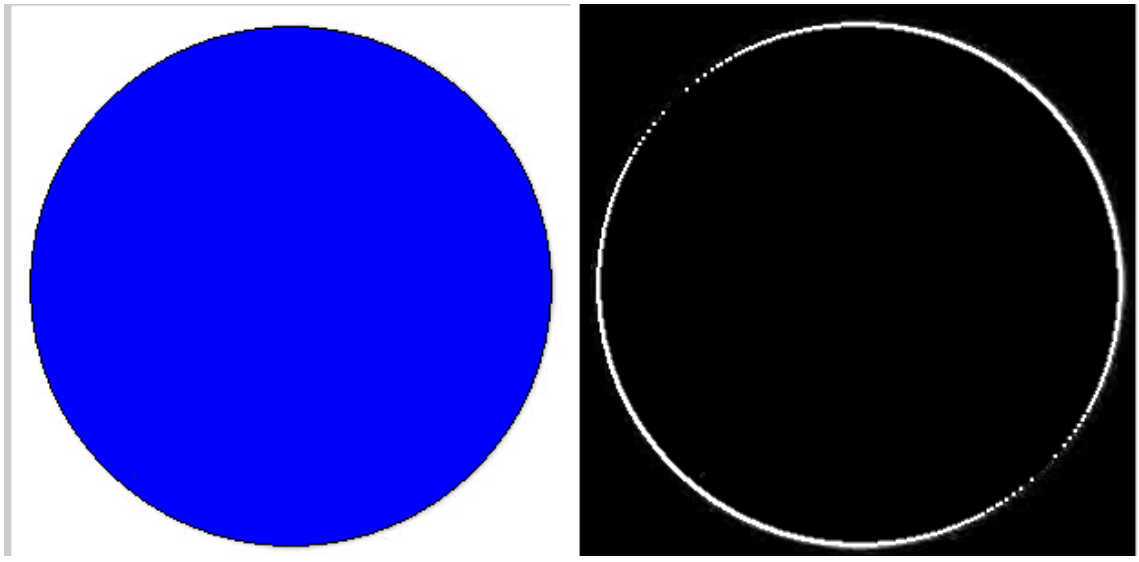
Şekil 2.8. X ve Y yönünde uygulanan Sobel filtreleri

Bulunan gradyan değeri ile eşik değer karşılaştırılarak nesnenin kenarları ve köşeleri bulunacaktır.

Eğer (x,y) pikseli eşik değerden küçükse köşe olarak kabul edilmez.
Eğer (x,y) pikseli eşik değerden büyükse köşe olarak kabul edilir.



Şekil 2.9. Canny kenar detektörü uygulaması



Şekil 2.10. Canny kenar detektörü uygulaması

2.3. Nesne İşaretleme Algoritmaları

İzleme algoritmalarının temel mantığı benzerlik gösterir. Hemen hemen bütün algoritmalar önceki karede tespit edilen nesneyi bir sonraki karede de bulmaya

yöneliktir. İlerleyen karelerde her seferinde nesneyi yeniden aramak zorunda kalmamak için nesneyi hangi özelliğiyle işaretlediğiniz önem kazanmaktadır. Bu bölümde işaretlenen nesneleri takip etmemizi sağlayan algoritmalar incelenecektir.

2.3.1. Nokta ile işaretlenen nesneleri takip etme

Kenarları ya da köşeleri bulunup noktayla işaretlenen nesneleri takip etmek için yaygın kullanılan algoritmalar vardır.

2.3.1.a. KLT işaretleme algoritması

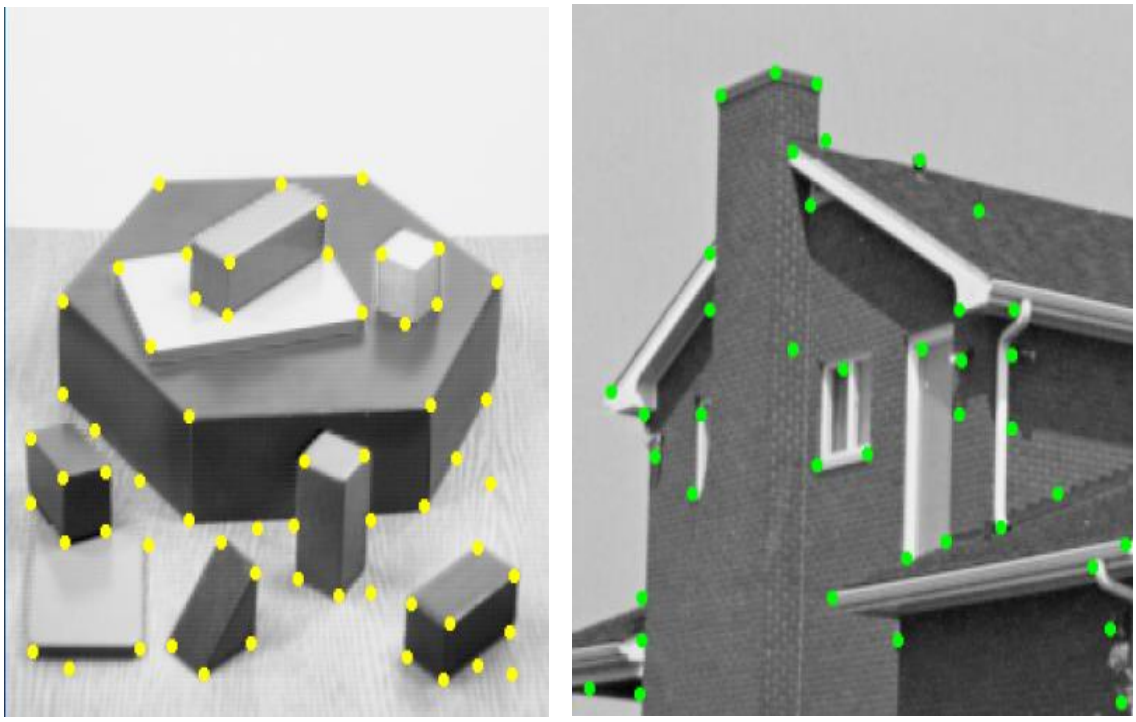
Bu algoritmanın amacı; imgedeki (I) nesnede var olan özellik noktalarına yeni özellik noktaları eklemeye yöneliktir. Bu noktaları yakalamak için takip edilecek noktanın komşuluğundaki pikseller de hesaba katılmalıdır. Bunun için λ_1 ve λ_2 yapı matrisinin (C) eigenvalue değerleri olmak üzere;

$$C = \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

Şekil 2.11. Yapı matrisi

matrisi kullanılır. Buna göre;

Eğer $\lambda_1 > 0$ ve $\lambda_2 = 0$ ise ilgili nokta kenar olarak,
Eğer $\lambda_1 > 0$ ve $\lambda_2 > 0$ ise ilgili nokta köşe olarak belirlenir.



Şekil 2.12. KLT Köşe bulma uygulaması (Dmitrij and Csetverikov)

2.3.2. Arka plan uzaklaştırma/çıkarma ile nesne işaretleme

Arka plan çıkarma iki çerçeve arasındaki hareketi belirlemek için ve eğer her iki çerçevede de varsa arka plan gölgelenmesini çıkarmak için kullanılabilir. Takip edilecek nesnenin olduğu bölge çerçevenin tamamından çıkarılarak arka planın yok edilmesini temel alır. Böylece nesne daha belirgin olur ve hareketleri kolay anlaşılır, değişen pikseller rahat algılanır. Çerçeveler herhangi bir aydınlatma durumuna bakılmaksızın herhangi bir anda çekilmiş olabilir. Eğer nesne arka plandan daha aydınlık bir hale dönüşmüş ise tam tersi yapılır. Çıkarma işlemi, bir çerçevedeki her bir pikselin grilik seviyesinden diğer çerçevenin aynı pikselinin grilik seviyesini çıkarmak anlamına gelir. x>y olmak üzere,

Sonuç=x-y olarak tanımlanır.

Eğer x<y ise sonuç negatif olacaktır. Eğer değerler unsigned karakterleri tutuyorsa en yüksek pozitif değere dönüştürülür. Örneğin;

-1 255 değerini alır

-2 254 değerini alır

Daha iyi bir işlem için sonuç=|x-y| kullanılır.

I (x; y; t) B(x; y; t)

Şekil 2.13. t anında çekilen çerçeve ve arka plan (Tamersoy 2009)

Şekil 2.13'deki t anındaki bir çerçeveden hareketli nesneleri ayırmak için yine t anında Şekil 2.13'deki gibi sadece arka planın çerçevesi alınmalıdır. İki çerçeve arasında uygulanan çıkarma işleminin sonuçları belirlenen eşik değere göre nesnelerin arka plandan ayrılmasını sağlayacaktır.

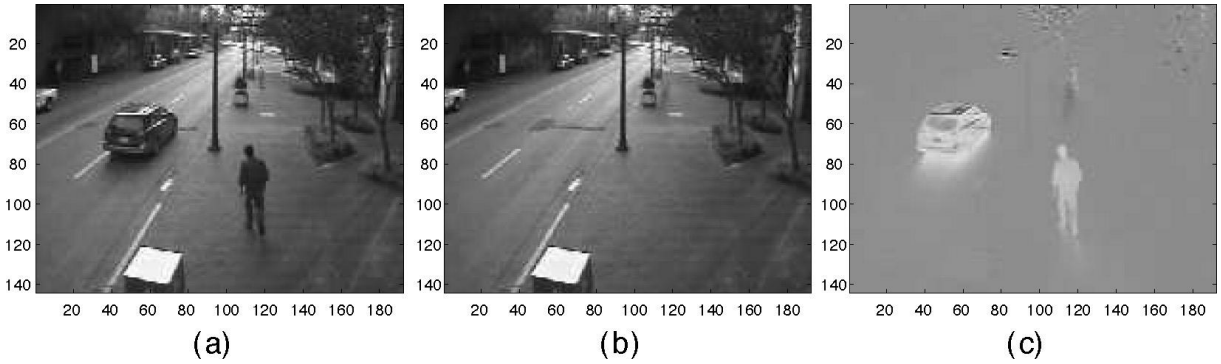
Arka plan bir önceki çerçeve gibi düşünülecek olursa; Th eşik değeri olmak üzere;

$B(x; y; t) = I(x; y; t - 1)$

$|I(x; y; t) - I(x; y; t - 1)| > Th$

Şartını sağlayan noktalar t anında arka plan çıkarılmış imge olarak değerlendirilir.

Şekil 2.14. Arka plan çıkarılmış çerçeve (Tamersoy 2009).



Şekil 2.15. Arka plan çıkarılma adımları (Yılmaz 2006)

2.4. Nesne Takibi

Nesne takibi, "bilgisayarlı görme" alanı içerisinde önemli bir göreve sahiptir. En basit tanımıyla, nesne takibi, video kaydı içerisinde hareket etmekte olan bir nesnenin hareket yörüngesini tahmin etme problemi olarak tanımlanabilir. Diğer bir deyişle, bir nesne takip edicinin (object tracker), ardışık video çerçeveleri içerisinde hareket etmekte olan nesneleri algılayarak her bir nesneye eşsiz bir etiket verme işlemidir. Hızlı veri işleme özelliğine sahip bilgisayarların artması, yüksek kaliteye sahip ucuz kameraların üretilmesi ve otomatik video analizi için gerekli olan ihtiyacın giderek artması, nesne takibi algoritmalarına karşı olan ilgiyi oldukça arttırmıştır.

Nesne takibi ile ilgili literatürde fazla sayıda yaklaşım önerilmiştir. Önerilen bu yaklaşımların temel farkları aşağıdaki sorulara verilen yanıtlar yardımıyla ortaya çıkmaktadır.

- Nesne takibi için hangi nesne sunumu daha uygundur?
- Takip işleminde nesnelere ait hangi özellikler kullanılmalıdır?
- Belirlenen nesne özellikleri nasıl modellenir?

2.4.1. Nesne takip yöntemleri

Nesne takip yöntemleri iki temel görevi içermektedir .

1) Takip edilecek nesnenin güncel imge içerisindeki konumunun tespit edilmesi (object detection, lifting);
2) İmgeler boyunca konumları belirlenen nesneler arasındaki veri bağı ilişkisinin kurulması (data association).

Bu iki görev bağımsız gerçekleştirilebileceği gibi bir arada da çalıştırılabilir. Bağımsız çalıştırıldığında, ilk olarak nesne yakalama algoritmaları yardımıyla ardışık imgelerdeki hareketli nesneler yakalanır. Daha sonra, bu nesneler arasında var olan veri bağı ilişkileri elde edilir. Görevler birlikte çalıştırıldığında ise, güncel nesne bilgilerini hesaplayabilmek için güncel gözlem bilgisi ve öncesel nesne bilgisi birlikte değerlendirilir.

Bahsedilen her iki nesne takip yöntemi de takip edilecek nesneyi şekil ve görünüm modelleri ile ifade eder. Nesne şekli, takip algoritmasının yapabileceği hareketi 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 (affine veya projective 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.

Nesne hareketleri veya görünümleri üzerinde belirli sınırlamalar yapılarak nesne takip problemi basitleştirilebilir. 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 nesne takip işlemi oldukça basitleşir. Buna ilaveten, nesnelerin sayıları, büyüklükleri, görünümleri veya sahip oldukları fiziksel şekillerin önceden belirtilmesiyle de, nesne takip algoritmalarının karmaşıklığı azaltılır.

Nesne takibinde karşılaşılan temel zorluklar aşağıda listelendiği gibidir:

- 3 boyutlu gerçek dünya verilerinin 2 boyutlu çerçeve alanına yansıtılmasıyla meydana gelen bilgi kaybı
- Çerçeve üzerinde meydana gelen gürültüler
- Nesne hareketlerinin karmaşık oluşu
- Nesnelerin ayırt edilebilir fiziksel bir yapıya sahip olamayışları
- Nesne görünümünün bir kısmının veya tamamının engellenmesi (occlusion)
- Karmaşık nesne şekilleri
- Ortamdaki ışık miktarının değişimi
- Gerçek zamanlı uygulamaların gereksinimleri

Şekil 2.16'da nesne takip yöntemleri takip türüne göre sınıflandırılarak gösterilmiştir. Buna göre üç temel nesne takip yöntemi bulunmaktadır:

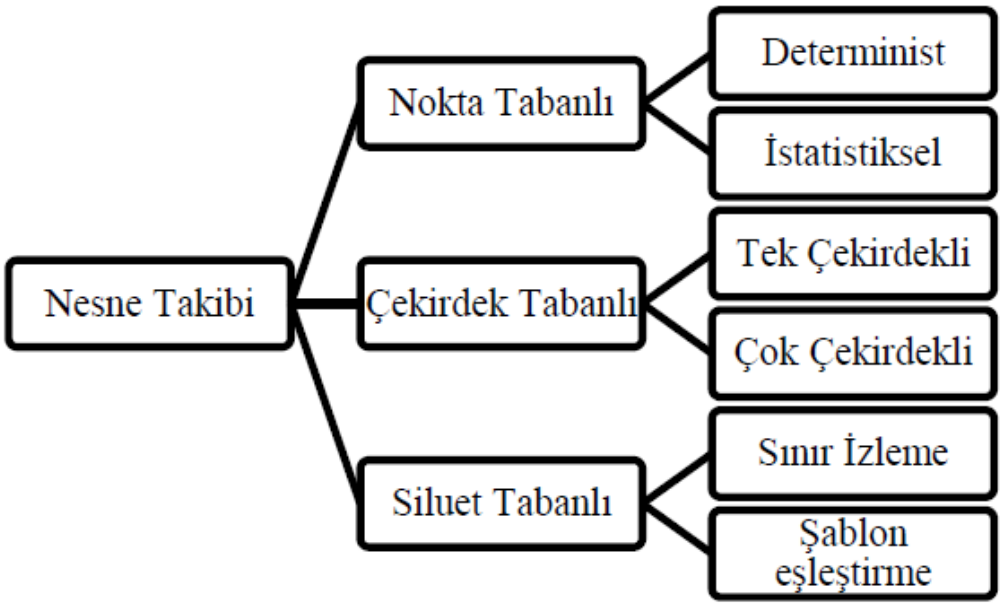
1) nokta tabanlı
2) çekirdek tabanlı

3) siluet tabanlı

Bu yöntemlerle ilgili uygulama alanları da aşağıdaki gibi listelenebilir:

- Hareket tabanlı tanımlama (Örn: Hareket yörüngesi bilinen bir tümörünün akciğer üzerindeki konumun tespiti)
- Gizlice izleme (Örn: Şüpheli aktivitelerin veya istenmeyen hareketlerin tespit edilebilmesi için sabit bir ekran çerçevesinin izlenmesi)
- Video indeksleme (Örn: Çoklu ortam veritabanında bulunan videoların, içerdiği nesneler ve bu nesnelerin hareketleri şeklinde kodlanarak sisteme kaydedilmesi, nesne adları veya hareketleri kullanılarak video kayıtlarına hızlı bir şekilde ulaşılabilmesi).
- İnsan bilgisayar etkileşimi (Örn: İnsan vücut azaları takip edilerek insanın yürüdüğü veya koştuğu sonucunun üretilmesi)
- Trafik izleme (Örn: Trafik akışını düzenlemek için gerçek zamanlı trafik bilgilerinin toplanması.
- Araç yolculuğu (Aracın şoförsüz seyahat edebilmesi veya yol çizgilerini takip edebilmesi).

Bu bölümün alt başlıklarında modern nesne takip yöntemleri hakkında detaylı bilgi verilmektedir.



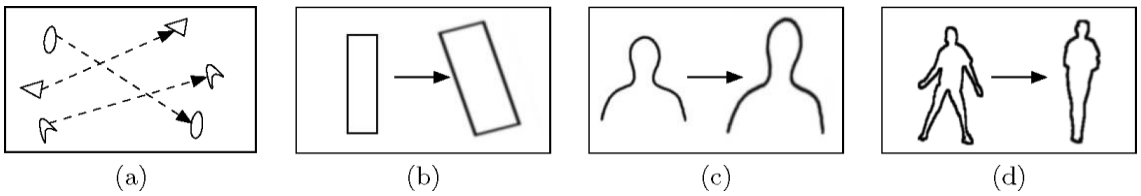
Şekil 2.16. Nesne takip yöntemleri (Yılmaz 2006)

2.4.1.a. 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: Deterministik ve istatistiksel. Aşağıda iki yöntemle ilgili detaylı bilgi verilmektedir.

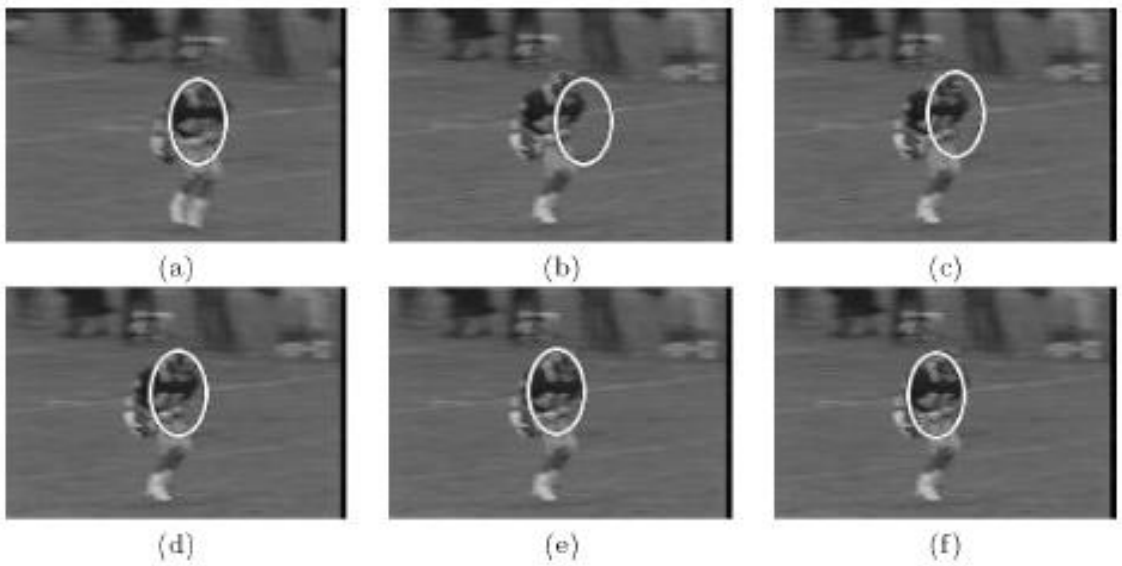


Şekil 2.17. Nesne takip yöntemleri (Yılmaz 2006)
(a) birden fazla nokta eşleştirmesi, (b) dikdörtgensel bir bölgenin değişimi,(c,d) kontür değişimi

2.4.1.b. Kernel tabanlı nesne takibi

Kernel yaklaşımı nesnenin görünüşünü ve şeklini temel alır. Örneğin nesnenin histogram ilişkisine bakarak eliptik yada dikdörtgensel bir şablon olabilir.

Kernel izleme metodu tipik olarak bir çerçeveden bir sonrakine temel bir nesne bölgesi tarafından temsil edilen nesnenin hareketini işleyerek gerçekleştirilir. Nesne hareketi genellikle parametrik hareket formu veya sonraki çerçevelerde hesaplanan yoğun akış alanı şeklindedir. Bu algoritmalar ile kullanılan görünümü temsil, izlenen nesnelerin sayısı ve nesne hareketini hesaplama bakımından farklıdır.

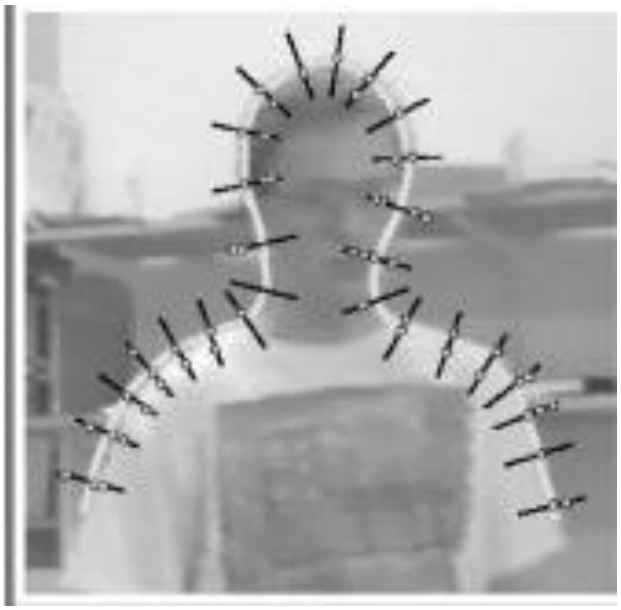


Şekil 2.18. Kernel takip algoritması uygulaması (Yılmaz 2006)

2.4.1.c. Silüet tabanlı nesne 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. 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.



Şekil 2.19. Silüet takip algoritması uygulaması (Yılmaz 2006)

2.5. Nokta Takibi

2.5.1. Deterministik yöntemler

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

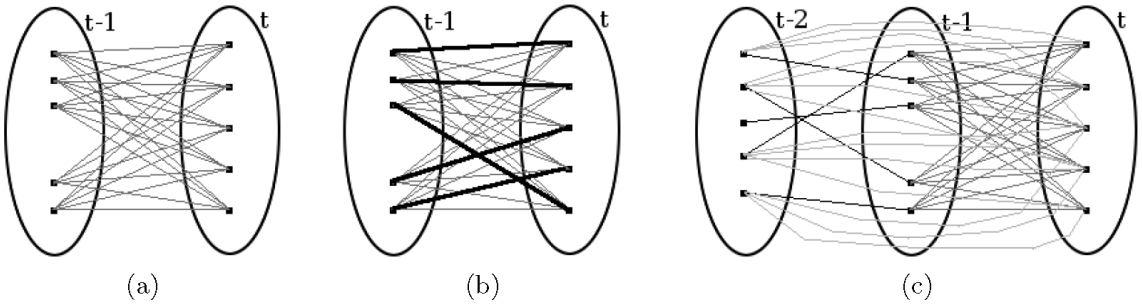
- Yakınlık: Bir imgeden diğer bir imgeye geçerken nesne pozisyonlarının önemli ölçüde değişmeyeceğini ima eder.

- 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.

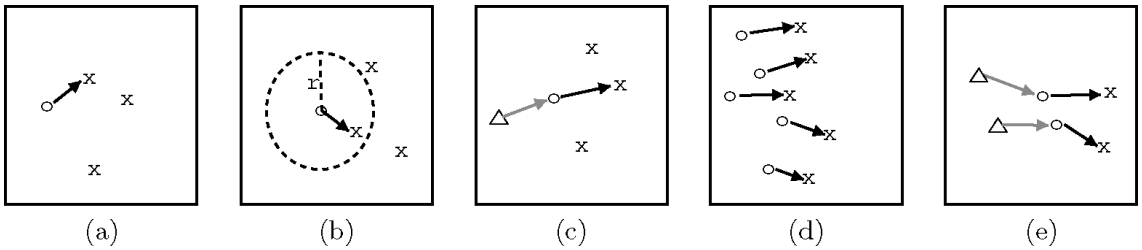
- Küçük hız değişimi (yumuşak hareket): Nesnenin hız yönünün önemli bir ölçüde değişmeyeceğini ima eder.

- Genel hareket: Küçük bir komşuluktaki nesnelerin hızlarının benzer olması için sınırlama uygulanır. Bu sınırlama, birden fazla nokta ile temsil edilen 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.



Şekil 2.20. Deterministik nokta takibi uygulaması (Yılmaz 2006)



Şekil 2.21. Deterministik nokta takibi uygulaması (Yılmaz 2006)

2.5.2. 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.

Ölçümler genellikle imgelerdeki nesne pozisyonlarını içerir. Bu bölümde istatistiksel yöntemlerin imge içerisindeki nesnelere ait durum tahminini nasıl formülleştirdikleri ve hangi çözümleri sundukları detaylı bir şekilde incelenecektir. Buna göre problemin tanımı için Kalman filtreleme yöntemi ve parçacık filtreleme yöntemi kullanılır.

İmge üzerinde hareket eden bir nesnenin istatistiksel olarak durum tahmin problemi şu şekilde formüle edilebilir (Xue vd., 2008). Takip edilecek nesnenin durum bilgisi (örneğin pozisyon) X^t:t=1,2,.. şeklinde bir dizi olarak tanımlanır. Zaman boyunca durum üzerindeki dinamik değişim denklem (1) ile ifade edilir:

$$X^t = f(X^{t-1}) + W^t \tag{2.1}$$

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

$$Z^t = \mathrm{h}\,(X^t) + N^t \tag{2.2}$$

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 diyişle, sonrasal olasılık yoğunluk fonksiyonunu (posterior probability density function) ($X^t \mid Z^{1,\dots,t}$) 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 fonksiyonu (prior probability density function) p($X^t \mid Z^{1,\dots,t-1}$) hesaplanır. Daha sonra sonrasal yoğunluk fonksiyonunun hesaplanabilmesi için güncel ölçümün maksimum olabilirlik fonksiyonu ($Z^T \mid X^t$) kullanılır. Buna göre sonrasal yoğunluk fonksiyonu denklem (3)'de gösterildiği gibi hesaplanır.

$$\mathrm{p}\,(X^t \mid Z^{1,\dots,t}) = \frac{p\,(X^t \mid Z^{1,\dots,t-1})\,p(Y^t \mid X^t)}{p\,(Y^t \mid Z^{1,\dots,t-1})} \tag{2.3}$$

Denklem (2.3)'deki p ($Y^t \mid Z^{1,\dots,t-1}$) 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 X^1 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.

2.5.2.a. Kalman filtreleme yöntemi

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

$$\bar{X}^t = \mathrm{D}X^{t-1} + \mathrm{W} \tag{2.4}$$

$$\bar{\Sigma}^t = \mathrm{D}\Sigma^{t-1}D^T + Q^t \tag{2.5}$$

$\bar{X}^t$ ve $\bar{\Sigma}^t$, 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 Z^t nesne durumunu güncellemek için kullanır.

$$K^t = \bar{\Sigma}^t M^t [\mathrm{M}\bar{\Sigma}^t M^t + R^t]^{-1} \tag{2.6}$$

$$X^t = \bar{X}^t + K^t [Z^t - \mathrm{M}\bar{X}^t] = X^t = \bar{X}^t + K^t * \mathrm{v} \tag{2.7}$$

$$\bar{\Sigma}^t = \bar{\Sigma}^t - K^t \mathrm{M}\bar{\Sigma}^t \tag{2.8}$$

Yukarıdaki denklemlerde v yenilik olarak adlandırılır ve M ölçme matrisidir. K, durum modellerinin yayılımı için kullanılan kalman kazancıdır. Dikkat edilmesi gereken nokta; 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 fitler) olarak bilinir.

$$\begin{pmatrix} x^t \\ y^t \\ \dot{x}^t \\ \dot{y}^t \end{pmatrix} = \begin{pmatrix} 1 & & 1 & \\ & 1 & & 1 \\ & & 1 & \\ & & & 1 \end{pmatrix} \begin{pmatrix} x^{t-1} \\ y^{t-1} \\ \dot{x}^{t-1} \\ \dot{y}^{t-1} \end{pmatrix} + \mathbf{w}^t$$

Şekil 2.22. Kalman filtre durum modeli eşitliği (Smith 2009)

Şekil 2.23. Kalman filtre uygulaması (Smith 2009)

2.5.2.b. 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. Bu dezavantaj parçacık filtreleme yöntemiyle çözülebilir. Parçacık filtrede, t anındaki şartsal durum yoğunluğu π ağırlığına sahip N tane parçacık içeren örnek küme ile ifade edilir.

Her bir parçacık , $\{S_t^{(n)}: n = 1, \dots, N\}$ ve her bir parçacığın ağırlığı ise $\pi_t^{(n)}$ (örnekleme olasılığı) olarak gösterilir. Ağırlıklar parçacığın önemi veya onun gözlenme frekansı olarak tanımlanabilir. Her bir $(S_t^{(n)}, \pi_t^{(n)})$'nin hesaplama maliyetinin azaltılması için birikmiş (katlanmış) bir ağırlık $c^{(n)}$ de aynı zamanda hafızaya yüklenir ($c^{(N)}$ =1). t anındaki yeni örnekler t-1 anındaki ,$(s_{t-1} = \{(s_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)})$:n=1,…,N} 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.

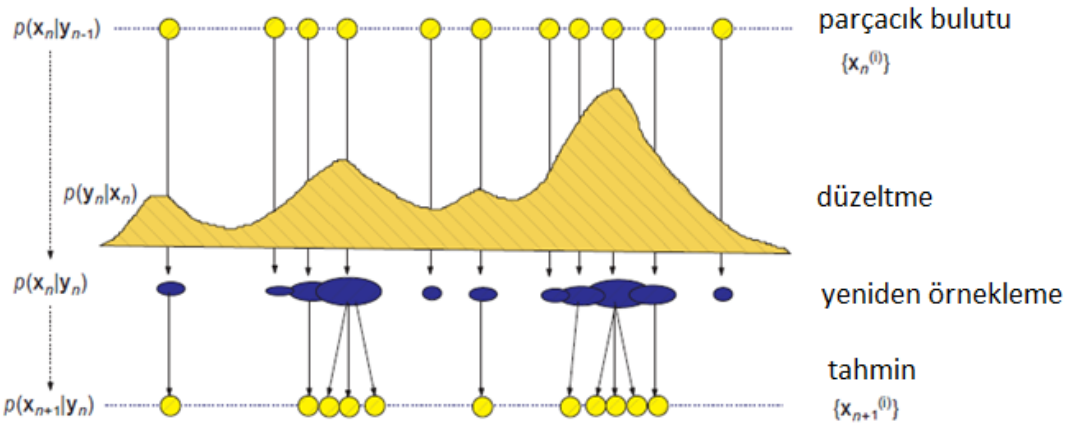
1) Seçme adımı: s_{t-1}'den N tane rastgele $\hat{s}_t^{(n)}$ örnek seçilir. Seçme işlemi yapılırken r ϵ [0,1] olmak kaydıyla rastgele bir değişken üretilir ve $c_{t-1}^{(j)} > r$ ve $\hat{s}_t^{(n)} = s_{t-1}^{(j)}$ şartlarını sağlayan en küçük j indeksine sahip örnekler seçilir.

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

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

Elde edilen yeni S_t örnekleri $\varepsilon_t = \Sigma_{n=1}^{N} \pi_t^{(n)} f\left(s_t^{(n)}, W\right)$ eşitliğinde kullanılarak yeni nesne pozisyonları hesaplanabilir. Parçacık filtre tabanlı nesne takip edicilerin başlangıç değerleri, örnekleme dizisini kullanan sistemlerin eğitimiyle veya ilk ölçümler ($s_0^{(n)} \sim X_0$) kullanılarak gerçekleşir.

Sistem ilk ölçümler kullanılarak başlatılırsa, her bir örneğin ağırlık değeri $\pi_0^{(n)}$ =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.

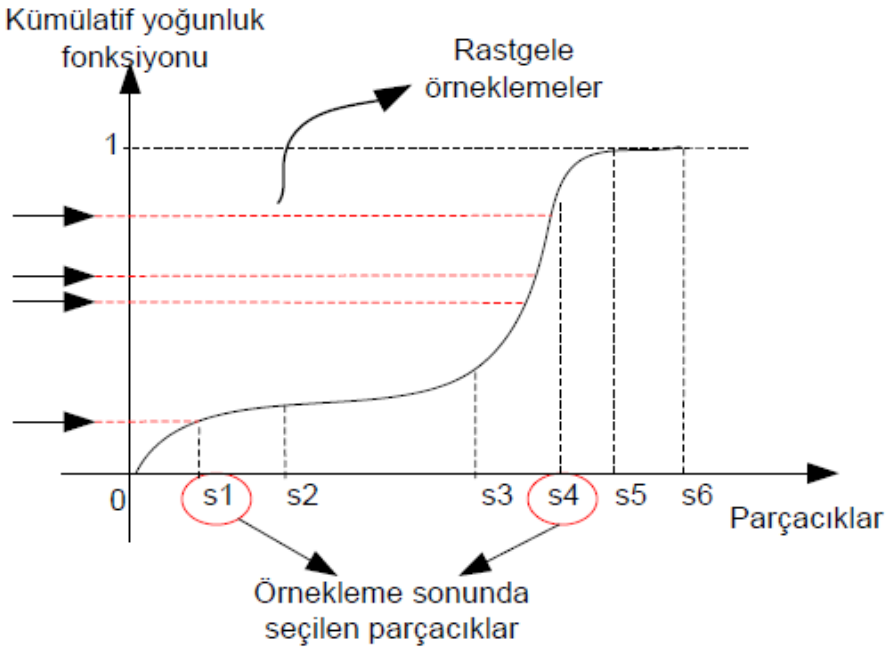


Şekil 2.24. Parçacık filtre uygulaması (Thrun and Kosecka 2007)

2.5.2.c. Çekirdek tabanlı nesne takibi

Çekirdek tabanlı nesne izleyiciler, parametrik olmayan tahmin ediciler gurubu içerisinde yer almaktadırlar. Parametrik olmayan sistemlerde sabit bir fonksiyon yapısı söz konusu

değildir ve bir tahmin gerçekleştirileceği zaman dağılıma ait tüm veri değerleri göz önünde bulundurulmalıdır. Bunun yanında parametrik sistemlerde sabit bir fonksiyon yapısı ve sabit parametre değerleri bulunmaktadır.



Şekil 2.25. Kümülatif yoğunluk fonksiyonu (Akbulut 2005)

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.

Olasılık yoğunluk dağılımı, takip edilecek nesnenin her pikselinin diğer pikseller üzerindeki etkisini ifade eder. 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 cazipliğini azaltır ve bu sınırlamaların olmadığı çekirdek yoğunluk fonksiyonlarının kullanılmasına neden olur.

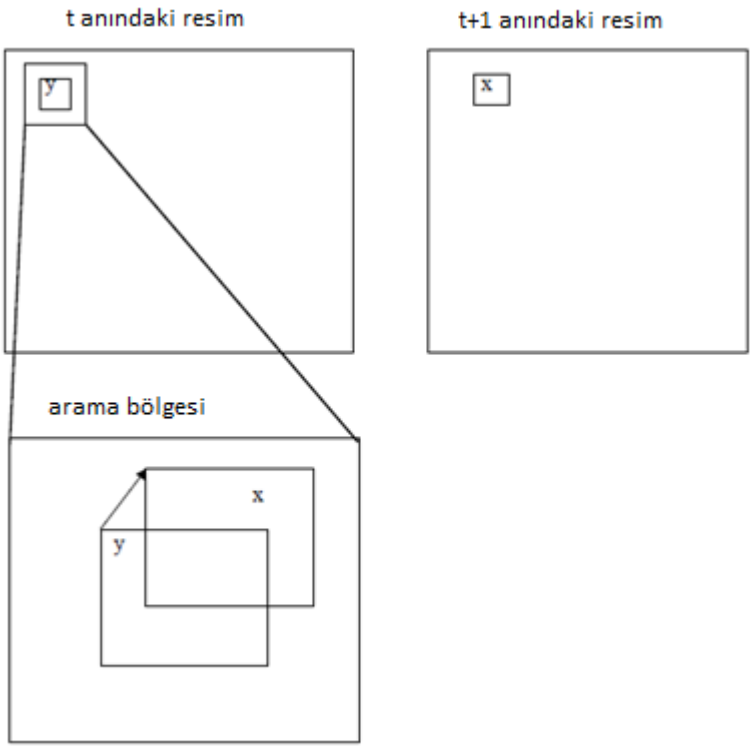
Histogramdaki her bir parçanın başlangıç ve bitiş noktalarına olan bağımlılığını kaldırmak için çekirdek yoğunluk tahmin edicileri her bir veri noktasını merkez kabul ederek veriyi belirlenen çekirdek fonksiyonundan geçirir ve her bir veri noktası için bir yoğunluk değeri elde edilir. Böylelikle çekirdek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

2.6. Hareket Vektörü Bulma

2.6.1. Hareket vektörü tanımlama

MPEG kendi yüksek sıkıştırma oranını hareket hesaplaması ve dengelemesini kullanarak başarır. Mpeg çerçeveden çerçeveye geçişlerde, çerçevede çok az değişiklik olması durumunu avantaj olarak kullanır. Bu sebeple makro bloklar çerçeveler arasında karşılaştırılır, tüm makro blokların kodlanması yerine iki makro blok arasındaki fark kodlanır ve iletilir. Şekil 2.26’da ileri hareket dengelemesi anlatılmıştır.

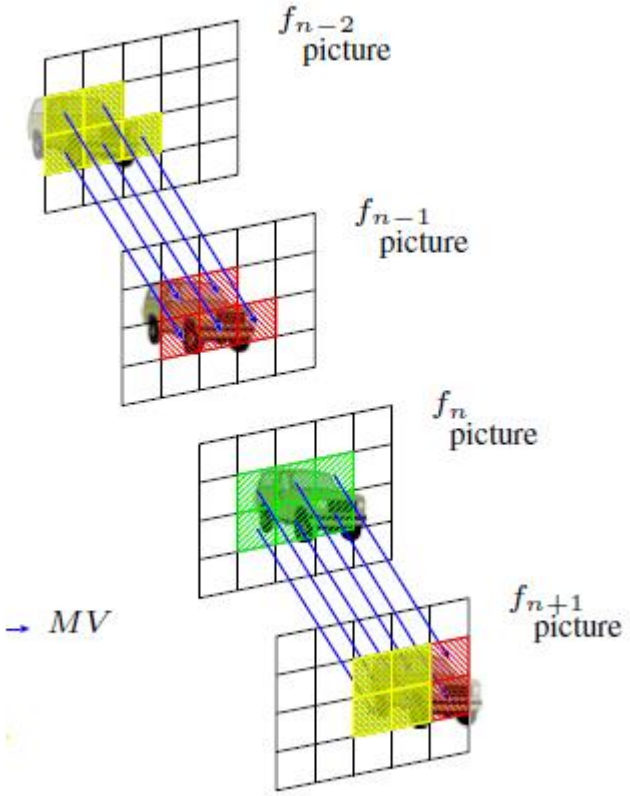
Makro blok x kodlamak istediğimiz makro blok, makro blok y ise referans çerçeveeki karşılığıdır. x için en iyi eşleşmeyi bulabilmek için y etrafında bir arama yapılır bu arama belirli bir alan içinde sınırlı kalır bu arama alanı dışında iyi bir eşleşme bulunsa bile kullanılmaz. Bu iki makro blok arasındaki yer değiştirme x ile ilişkilendirilmiş hareket vektörünü verir.



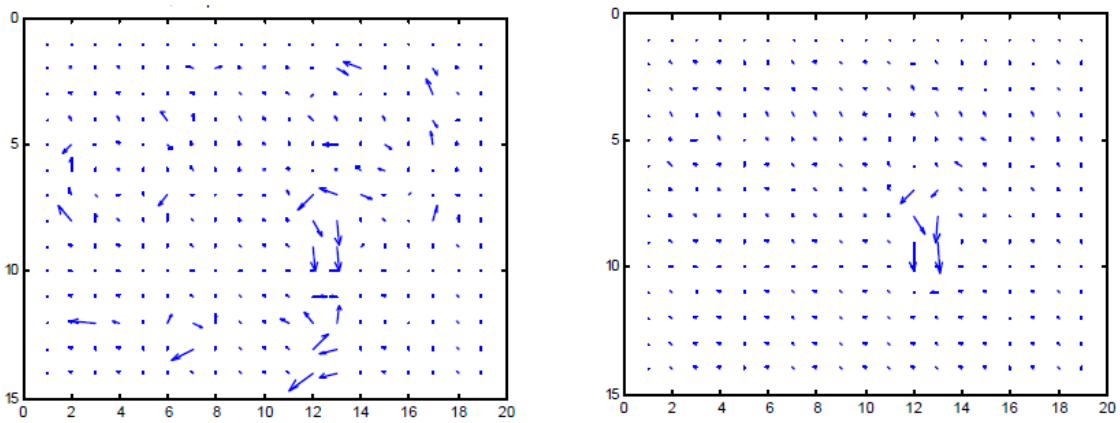
Şekil 2.26. Arama bölgesi belirleme (Gilvarry 1999)

2.6.2. Hareket vektörü takip etme

Blok eşleştirme algoritması tüm ortak hareket vektörlerini bulabilen en güvenilir algoritmadır. Hareketi bulduğumuz alanı mikro yada makro blok içine alıp bir sonraki çerçevede de aynı bloğu bulmaya çalışılır. Sonraki çerçevede bulunan alan ile önceki çerçevede belirlenen alan arasında hareket vektörü oluşturulur. Blok eşleştirme algoritması en yaygın algoritma olmasına rağmen, pratik uygulamalarda bazı problemlere sahiptir. Sorunların çoğu gürültü varlığında ortaya çıkar. Çerçeve arka plan gürültüsü veya kamera gürültüsüne sahipse; blok eşleştirme algoritması sahte hareket vektörleri üretebilir. Bunun için arama bölgesinin büyüklüğü doğru seçilmeli veya başka iyileştirme yaklaşımları kullanılmalıdır.

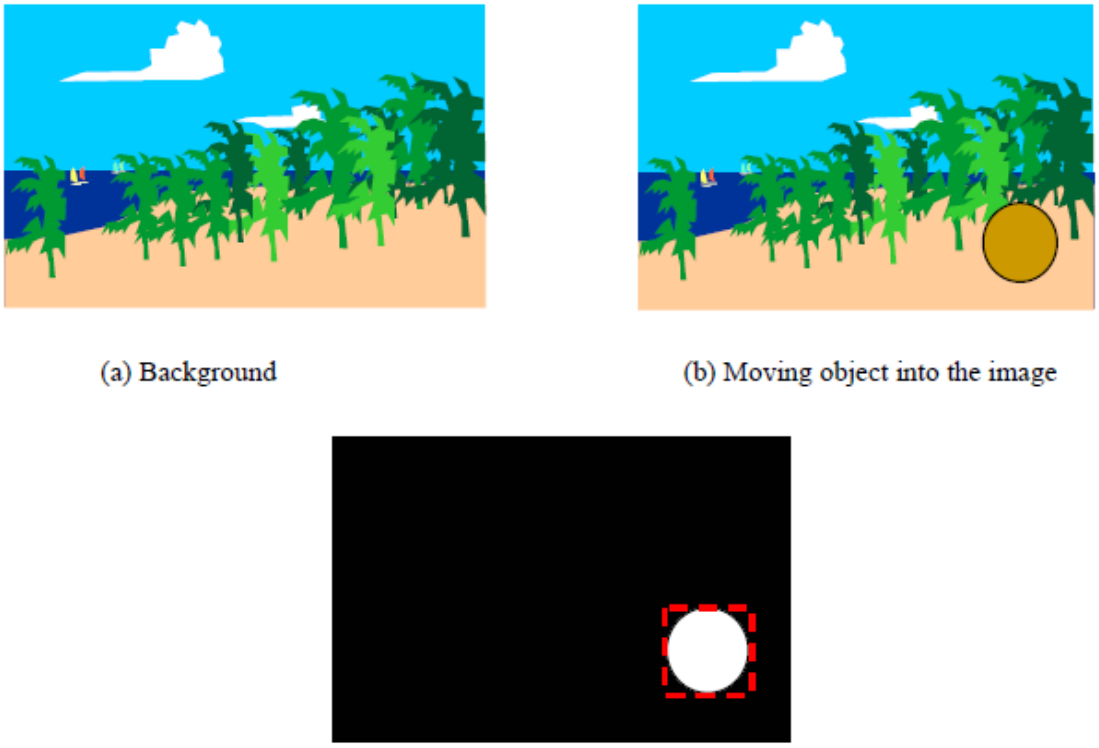


Şekil 2.27. Hareket vektörleri gösterimi (Collins 2006)



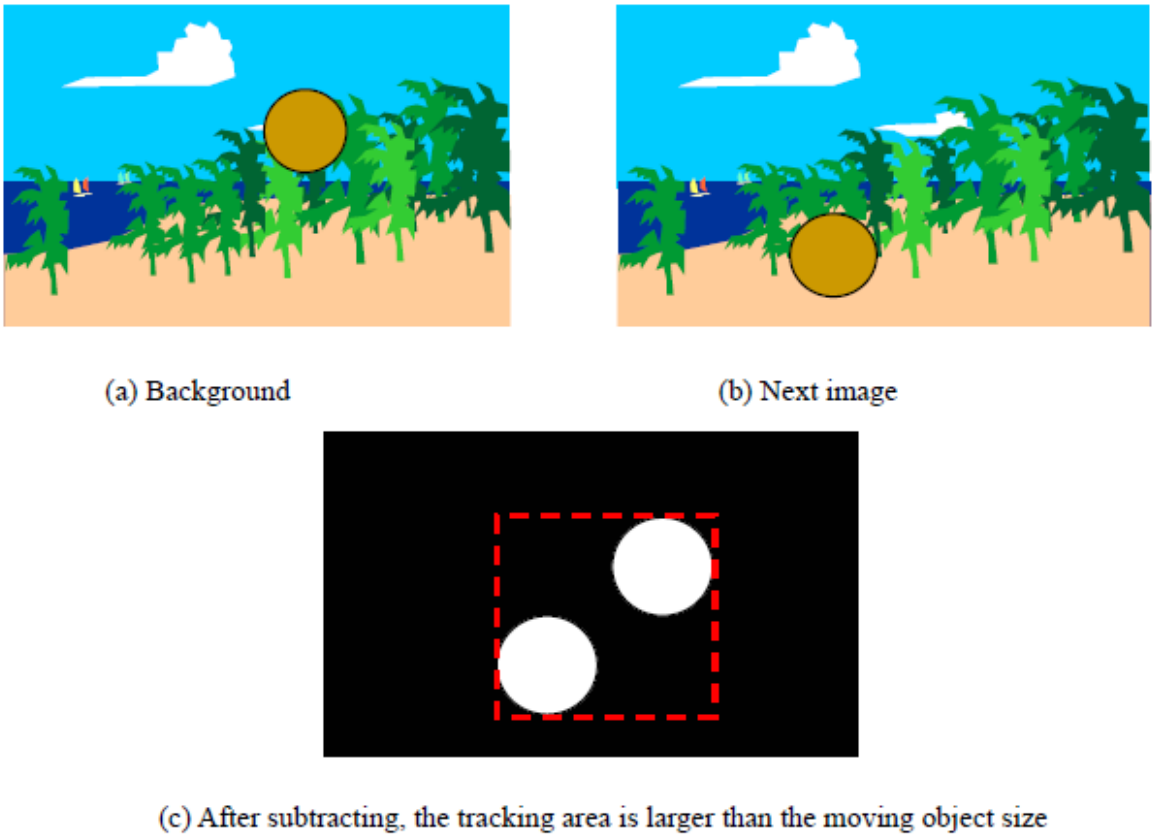
Şekil 2.28. Hareket vektörleri (Sebe 2002)

Geleneksel olarak değişik hareket metodu (DMA) basit olduğu için hareket eden nesneleri takip etme için kullanılır. Bu algoritmayı kullanmaya başladığımızda hareket etmeyen nesnelerin bulunduğu bir arka plan çerçevesi yakalanır. Daha sonra hareket eden bir nesne gözleme alanına girdiğinde ikinci çerçeve yakalanır. İkinci çerçeveden birinci çerçeve çıkartılarak iki çerçeve arasındaki fark bulunur. Ve hareket nesnenin pozisyonu elde edilir. Birbirini takip eden çerçeveler arasındaki mutlaka farkların toplamının hesap edilmesi gerekir. Ufak değişiklikleri filtrelemek için eşik değer kullanılarak hareket eden nesne daha doğru bir şekilde takip edilebilir.



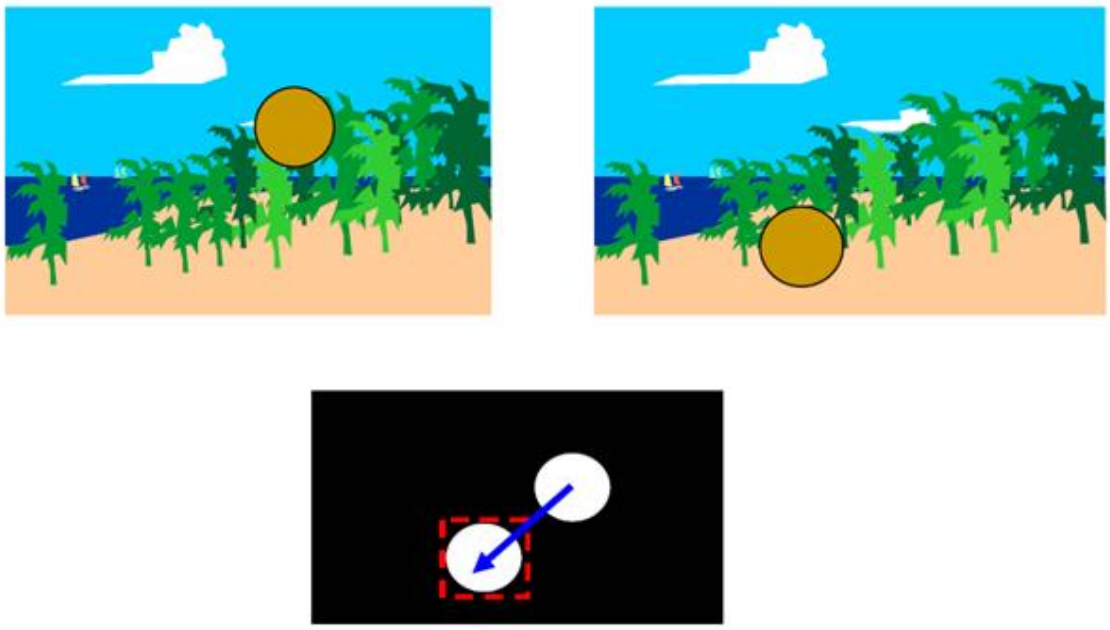
Şekil 2.29. Hareketli nesneyi belirleme (Tang *et al.* 2003)

Ama hareket eden nesne birbirini takip eden her iki çerçevede de bulunursa hareket eden nesnenin takip alanı fazladan hesaplanabilir. Bu dezavantajın üstesinden gelebilmek için blok eşleştirme algoritması (BMA) kullanılır. Bu algoritmada izleme alanının boyutunu ayarlamak için hareket hesaplamasından faydalanılır.



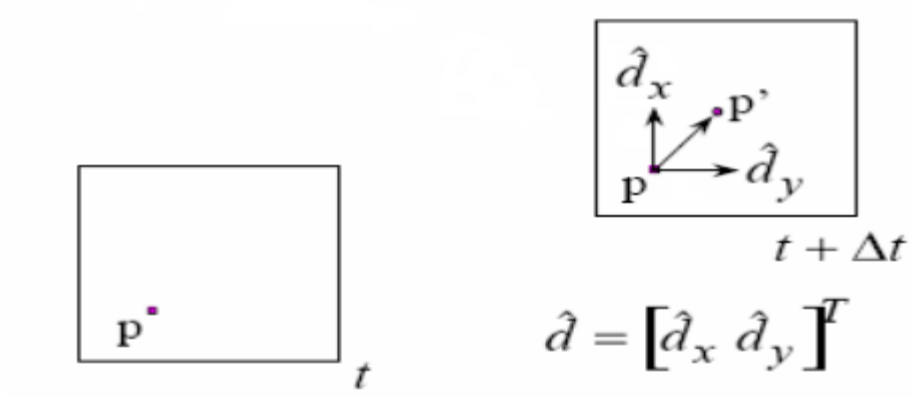
Şekil 2.30. İki çerçeve arasındaki farkların bulunması **(**Tang *et al*. 2003)

Blok eşleştirme algoritmasının (BMA) temel fikri hali hazırdaki çerçeveyi eşit boyutta ufak bloklara bölerek her bir blok için şimdiki bloğa en fazla benzeyen ve bir önceki çerçevenin arama alanında bulunan ilgili bloğu bulmaktır. Bu sebeple bir önceki çerçevedeki en iyi eşleşmiş blok şimdiki bloğun hareket kaynağı olarak seçilir. İki blok arasındaki fark hareket vektörünü verecektir. Takip alanındaki blokların tüm hareket vektörleri bulunduğunda en fazla tekrar eden hareket vektörü izleme alanı boyutunun düzeltilmesi için seçilir.



Şekil 2.31. Hareketli nesnenin vektörünün bulunması (Tang 2003)

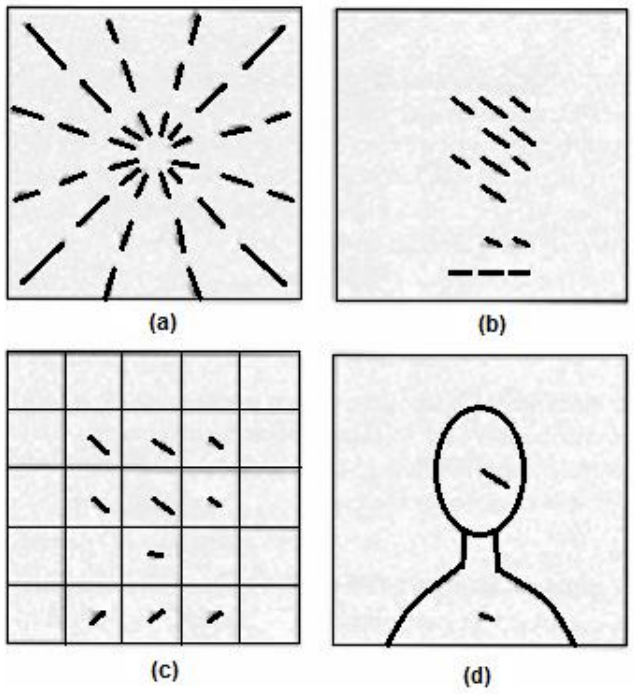
Bilindiği gibi videolar çerçevelerden oluşmaktadır. Ardışık çerçeveler arası ise global olarak bir yer değişim söz konusudur. Bu yer değişim hareket vektörünü oluşturmaktadır. Hareket vektörü başka bir ifadeyle t anındaki çerçevenin (t+1) anında genel olarak ne kadar yer değiştirdiğidir. Hareket vektörü kestirimi MPEG-1. MPEG-2. MPEG-4. H.261. H.262 gibi video kodlama sistemlerinde yaygın olarak kullanılmaktadır. Şekil 2.32'de bir hareket vektörü gösterilmiştir, t anındaki p noktası t+Δt anında p' noktasına taşınmıştır. Dolayısıyla p' noktası ile p noktası arasında bir hareket vektörü oluşmaktadır. Bu hareket vektörünün x-eksenindeki bileşeni d_x vektörü, y-eksenindeki bileşeni ise d_y vektörüdür. d_x ve d_y bileşenleri ise d vektörünü yani hareket vektörünü oluşturmaktadır.



Şekil 2.32. Hareket vektörünün gösterilmesi (Akbulut 2005)

Hareket gösterimi için çeşitli yöntemler vardır:

a) Genel bazlı (global based)
b) Piksel bazlı (pixel based)
c) Blok bazlı (block based)
d) Bölge bazlı (region based)



Şekil 2.33. Hareket gösterimi için çeşitli yöntemler (Akbulut 2005)

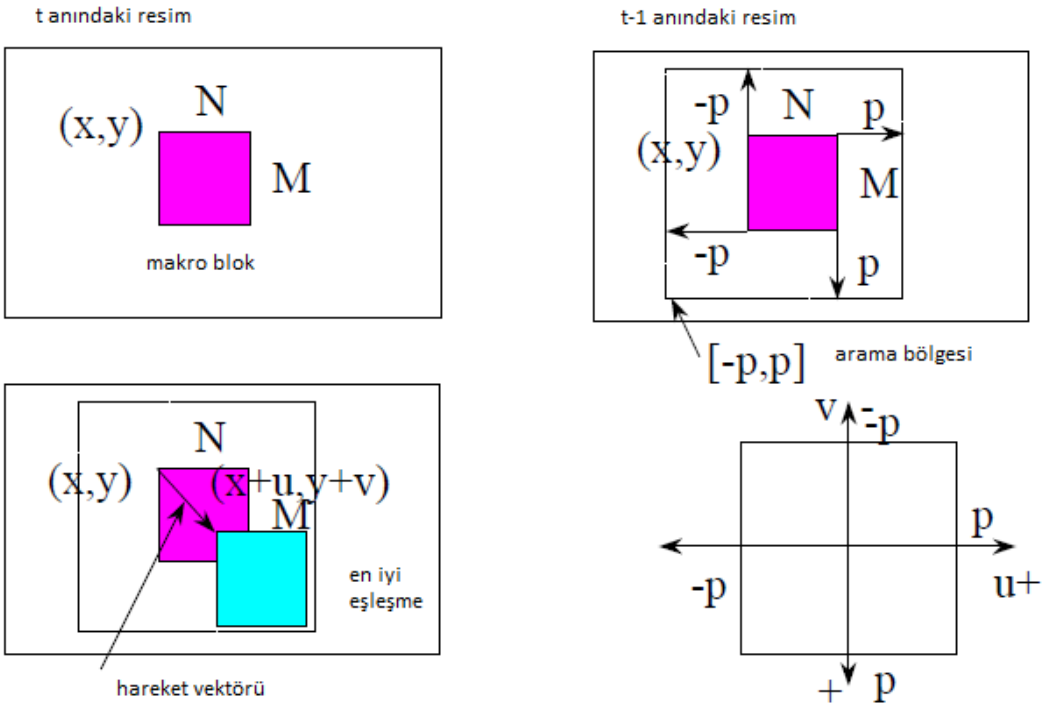
Video stabilizasyonunda çerçevelerin genel hareketi önemli olduğu için bu çalışmada genel yani global bazlı hareket kestirimi kullanılacaktır. Global bazlı hareket kestiriminde blok bazlı hareket kestiriminde olduğu gibi çerçeveler bloklara ayrılmaz. Tek bir blok kullanılır. Blok büyüklüğü hareket tahmini için önemli olmaktadır.

Hareket kestirimi için çeşitli yöntemler geliştirilmiştir. Bu yöntemler şunlardır:

- Tam arama (Full search)
- Logaritmik arama (2-D logarithmic search)
- Üç adımlı arama (three-step search)
- Baklava biçimli arama (diamond search)
- Çapraz arama (cross search)
- Spiral arama (spiral search)

Bu arama yöntemleri, arama prosedürleri (search procedures) veya arama stratejileri (search strategy) olarak da bilinmektedir. Hareket kestirimi yönteminde en iyi sonucu veren tam aramadır (FS). Fakat bu arama yönteminin hesap yükünün çok fazla olması beraberinde yeni algoritmaların bulunmasını sağlamıştır. Tam arama yöntemi dışındaki diğer arama yöntemleri hızlı algoritmalardır. Hesapsal yükleri azdır. Bu yöntemlerden bulunan hareket vektörleri tam arama yönteminde bulunan hareket vektörlerine yakın değerlerdir.

Arama stratejilerinden herhangi birini kullanırken hareket vektörlerini bulmada blok eşleme kriteri (block matching criteria) önemli olmaktadır.



Şekil 2.34. Blok eşleme algoritması (Geckle 2000)

2.6.3. Blok eşleme kriterleri

Hareket vektörlerini bulmada blok eşleme kriterleri kullanılmaktadır. Farklı arama stratejileri olduğu gibi farklı blok eşleme kriterleri vardır. Kriter olarak iki ölçüt kullanılır. Bunlar, maksimum çapraz-korelasyon (cross-correlation) ve minimum hatadır.

(Minimum error). Minimum hata kriterlerinin isimleri ve eşitlikleri sırasıyla şunlardır:

• Ortalama karesel hata (mean square error)

$$\mathrm{MSE}(d_1, d_2)= \frac{1}{(N_1 * N_2)} \Sigma_{n1}\Sigma_{n2}[\mathrm{s}(n_1, n_2, k)\text{-}\mathrm{s}(n_1 + d_1, n_2 + d_2, k + 1)]^2$$

• Ortalama mutlak değer (mean absolute difference)

$$\mathrm{MAD}(d_1, d_2)= \frac{1}{(N_1 * N_2)} \Sigma_{n1}\Sigma_{n2}|\mathrm{s}(n_1, n_2, k)\text{-}\mathrm{s}(n_1 + d_1, n_2 + d_2, k + 1)|$$

• Eşik farklılığı sayısı (number of thresholded differences)

$$\mathrm{NTD}(d_1, d_2)= \frac{1}{(N_1 * N_2)} \Sigma_{n1}\Sigma_{n2}\ \mathrm{N}(\mathrm{s}(n_1, n_2, k),\mathrm{s}(n_1 + d_1, n_2 + d_2, k + 1))$$

$$\mathrm{N}(\alpha,\beta)=1 \leftrightarrow | \alpha\text{-}\beta | > T_0$$

$$\mathrm{N}(\alpha,\beta)=0 \leftrightarrow | \alpha\text{-}\beta | \leq T_0$$

$T0$: eşik değeri

Bu bağıntılarda;

(d_1, d_2) hareket vektörü
k, imgenin çerçeve numarasını
(n_1, n_2) , imgenin k.cı çerçevesinin koordinatlarını

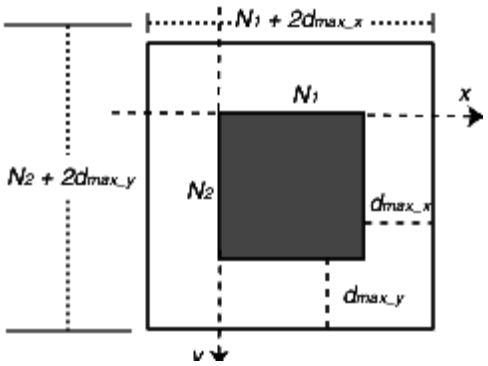
N_1,N_2 ise çerçeveden alınacak bloğun büyüklüğünü belirtmektedir.

Blok eşleme kriterlerinde en iyi sonucu veren maksimum çapraz-korelasyonudur. Fakat pratikte gerçekleştirmek zordur. Çünkü hesapsal yükü diğerlerine göre çok fazladır. Video kodlama sistemlerinde ve stabilizasyon işlemlerinde genellikle minimum ortalama mutlak değer (MAD) kullanılmaktadır. Bunun nedeni MAD kriterinin diğer blok eşleme kriterlerine göre daha az hesapsal yük içermesidir.

2.6.3.a. Tam arama

Hareket vektörü kestiriminde en iyi sonucu veren algoritmadır. Muazzam bir hesap yükü gerektirir. Çünkü bu algoritma geniş kapsamlı ve ayrıntılı bir aramadır. Tam arama global-bazlı veya block-bazlı olarak gerçekleştirilebilir. Ancak yavaşlığı önemli bir dezavantaj olarak karşımıza çıkmaktadır.

Tam aramada ardışık iki çerçeve karşılaştırılır. k'cı çerçeveden seçilecek büyük bir blok (k+1)'ci çerçevede aranır. Uygun bir blok eşleme kriteri seçilerek, genellikle MAD seçilir en düşük hatayı veren blok bulunur. Bu iki blok arasındaki yer değişim vektörü bize hareket vektörünü vermektedir. Şekil 2.35 incelenecek olursa, aranacak blok N_1, N_2 ile belirtilen bloktur. $d_{\max\_x}$ ve $d_{\max\_y}$ ifadeleri ise aranacak bloğun sınırlarını belirtmektedir. Yani arama alanını göstermektedir.



Şekil 2.35. Arama bölgesinin belirlenmesi (Akbulut 2005)

Çoğu uygulamalarda arama alanı (search area) önceden belirlenmiş bir tamsayı ile sınırlandırılır. Bu sınırlama (2.9) ve (2.10) eşitlikleri ile verilmektedir.

$$-M_1 \leq d\max\_x \leq M_1 \tag{2.9}$$

$$-M_2 \leq d\max\_y \leq M_2 \tag{2.10}$$

Tam arama yönteminin hesapsal yükünün fazla olmasına örnek olarak bloklar arası ne kadar karşılaştırma olduğuna bakmak yeterlidir. (2.11) nolu eşitlikte karşılaştırmanın ne kadar olacağını belirten matematiksel formül verilmiştir.

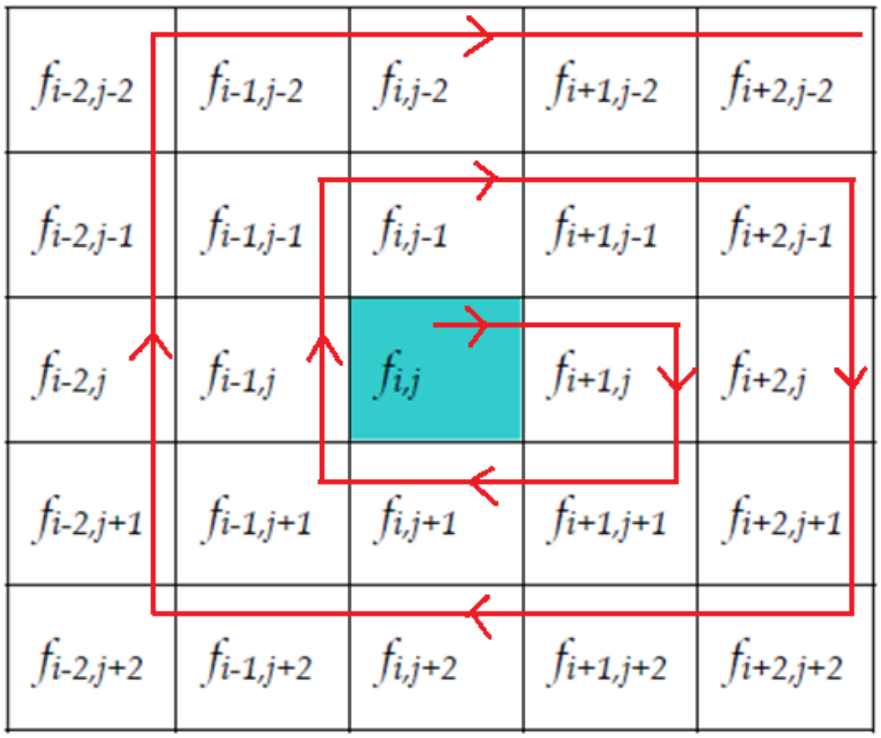
$$(2M_1 +1)*(2M_2 +1) \tag{2.11}$$

M_1=16 , M_2=16 için denklemin sonucu 1089 çıkmaktadır. Yani 1089 farklı nokta için blok karşılaştırılması yapılacaktır. Görüldüğü gibi karşılaştırma değeri çok yüksektir. RGB uzayındaki (Red, Green, Blue) çerçeveler için bir piksel değeri 3 ayrı renkle ifade edildiği için saniyede gereken karşılaştırma sayısı eşitlik (2.12)'de verilmiştir.

$$3N_1N_2F(2M_1 +1)^2 \tag{2.12}$$

F=saniyede gösterilen çerçeve sayısıdır.

2.6.3.b. Spiral tarama



Şekil 2.36. Spiral arama algoritması

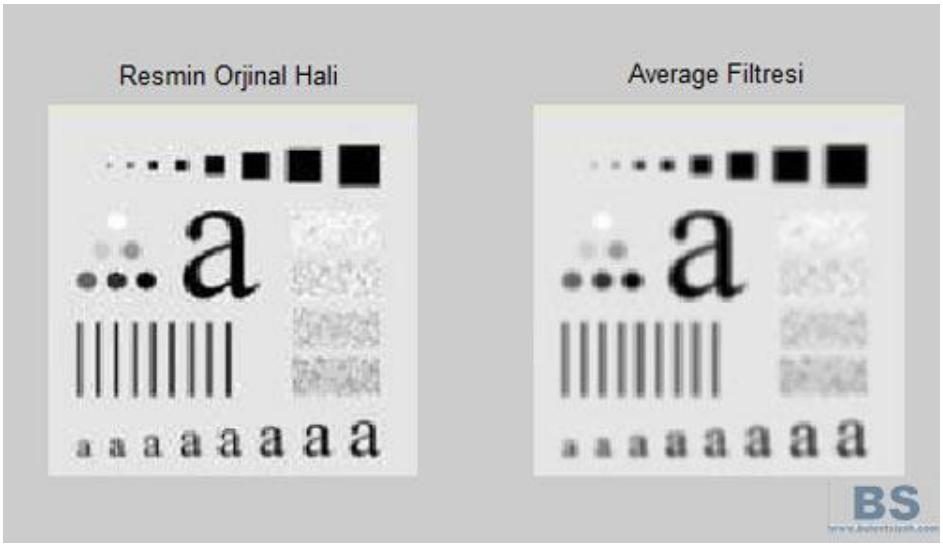
Şekilde görüldüğü gibi spiral tarama resmin merkezindeki blokla eşleştirme aramaya başlar. Sonra spiral şeklinde bir sonraki bloğu eşleştirme yapmak için kullanır. Spiral taramada zaten hareketin en olası pikselinden taramaya başlandığı için zaman ve hız açısından daha iyi sonuçlar elde edilmektedir.

2.7. Filtreler

Gürültü dış kaynaktan gelerek sinyalin bozulmasını sağlayan etkilerdir. Çerçevenin elde edilmesi sırasında gürültü karışabileceği gibi çerçevenin iletilmesi sırasında da gürültü karışabilir. Bilinen en yaygın gürültü çeşidi gaussian gürültüsüdür. Standart olan gauss gürültüsü, KTC gürültüsü denen kondansatörlerin reset gürültüsünden gelen sesleri içeren termal gürültü olarak bilinir. Gauss gürültüsü sinyalde rastgele dalgalanmaların sebep olduğu ve zamanın tümüne yayılan beyaz gürültünün idealize bir formudur.

2.7.1. Ortalama filtresi

Çerçevedeki her piksel yerine komşuları ile beraber ortalaması alınarak yeniden hesaplanır. Çerçevedeki gri düzeyler arasında keskin geçişler azalır; daha yumuşak geçişler söz konusudur. Çerçeve üzerindeki kenarlarda bulanıklaşmaya (blur) yol açarlar.



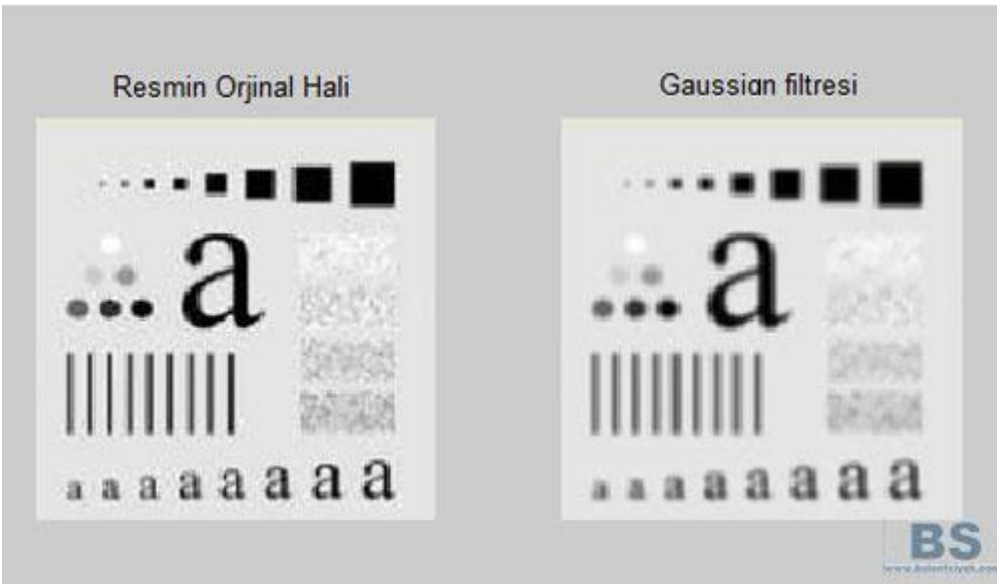
Şekil 2.37. Ortalama filtresi uygulaması (Siyah 2013)

2.7.2. Gaussian filtresi

Ortalama filtrenin Gaussian dağılımını kullanarak biraz daha değiştirilmiş hali Gaussian filtresi olarak bilinir. Gaussian filtreleme aynı zamanda bir fourier dönüşümüdür. Gauss filtre ile sonsuz bir transfer fonksiyonuna karşılık mekansal alanda sonlu bir pencerede (tarama penceresi) filtreleme yapılabilmektedir. Bu da filtrelemenin temel problemini daha kolay çözülebilir bir hale getirilebilir.

Gauss Filtresinin Avantajları: Filtreleme önce yatay ardından çıkan sonuçla düşey eksende gerçekleştirilebilir. Bu filtrede alçak geçiren filtre ile resmin konvolüsyonu yapılır. Bu konvolüsyon her pikselin değerini komşuluğundaki piksellerin değerlerine

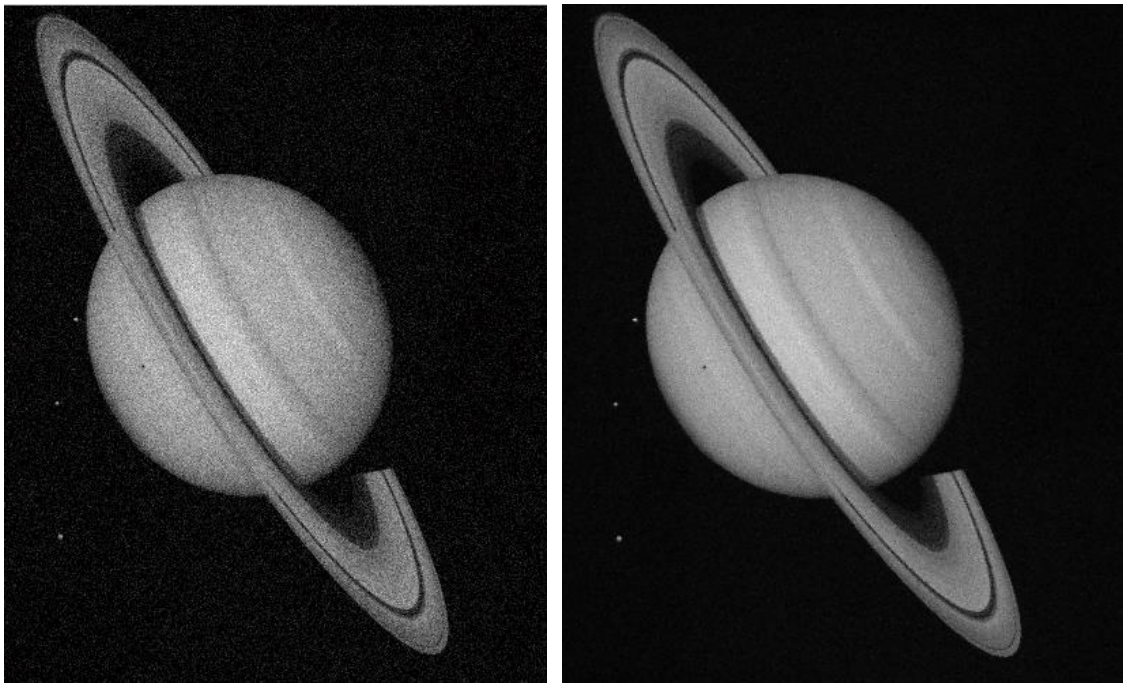
uyumlu hale getirir. Genelde bu değer belirli sayıdaki piksellerin ağırlıklı ortalama değeri olur. Bu durum çerçevede blur oluşturur. Yani çevresindeki piksellere göre çok düşük yada çok yüksek değerli pikseller ortalama değere yaklaşır.



Şekil 2.38. Gaussian filtresi uygulaması (Siyah 2013)

2.7.3. Wiener (Adaptive) filtresi

Wiener filtrenin özelliği resmin varyansına uyum sağlamaktır. Varyans büyükse wiener daha az smoothing sağlar, varyans küçükse wiener daha çok smoothing sağlar. Bu yaklaşım gauss filtreden daha iyi sonuç verir. Adaptive filtre lineer filtreye göre kenar ve yüksek frekanslı kısımları göstermede daha seçicidir.



Şekil 2.39. Wiener filtre uygulaması (Siyah 2013)

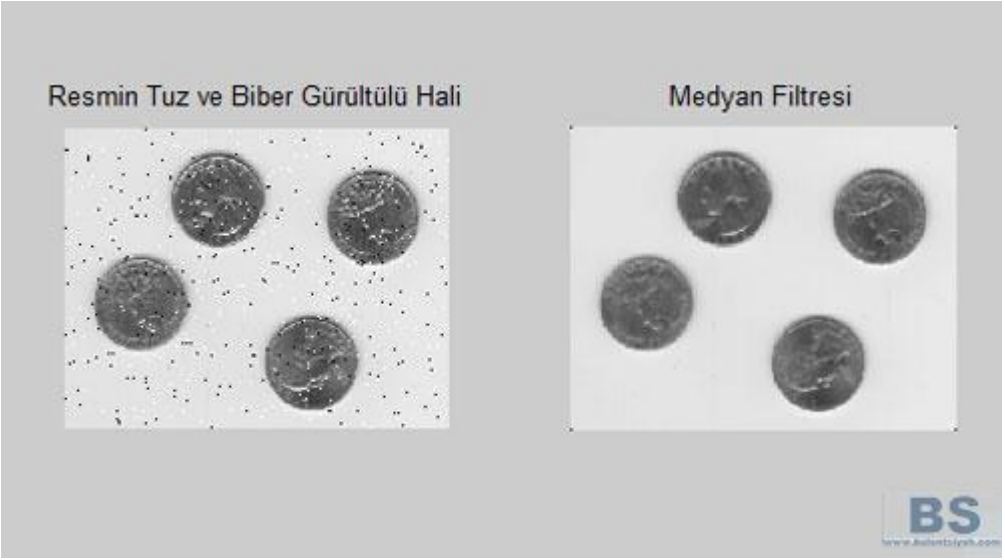
2.7.4. Median filtresi

Bu filtreleme yönteminde, orjinal sıralanmış piksel komşularının arasındaki ortanca değer ile değiştirilir. Bunun ağırlıklı ortalama filtrelerinden farkı şudur: Ağırlıklı ortalama filtrelerinde, komşuların ağırlıklı ortalaması alınır, hesaplanan bu değer orijinal piksel ile yeniden ortalanarak sonuç bulunur. Ortanca filtresinde ise, komşuluk değerleri önce sıraya konulur, sonra ortadaki değer alınır. Bu değer doğrudan sonuç kabul edilir. Ortanca değeri net elde edebilmek için genellikle tek sayıda komşu seçilir. Eğer hesaplamada çift sayıda komşu kullanılırsa, bu durumda ortada kalan iki pikselin aritmetik ortalaması kullanılır.

Ortanca filtre; Uzaysal çözünürlüğü bozmadan, kopuk (bağımsız) nokta veya çizgi gürültülerini temizlemek için kullanılır.

Medyan filtreler çerçeveden tuz-biber gürültüsünün kaldırılmasında da iyidir ve aynı zamanda çerçevedeki nesnelerin kenarlarının blurlaşmasına sebep olurlar.

Şekil 2.40. Median filtre uygulaması (Kavitha and Mythili 2011)



Şekil 2.41. Median filtre uygulaması (Siyah 2013)

2.7.5. Histogram filtresi

Belirli bir alandaki pikselleri o alanın merkezindeki piksel değerine eşitlemek için kullanılır.

5	5	5	5	5	5	5
5	6	6	6	6	6	5
5	6	7	7	7	6	5
5	6	7	7	7	6	5
5	6	7	7	7	6	5
5	6	6	6	6	6	5
5	5	5	5	5	5	5

Şekil 2.42. Histogram filtre uygulanmadan önce

7	7	7	7	7	7	7
7	7	7	7	7	7	7
7	7	7	7	7	7	7
7	7	7	7	7	7	7
7	7	7	7	7	7	7
7	7	7	7	7	7	7
7	7	7	7	7	7	7

Şekil 2.43. Histogram filtre uygulandıktan sonra

Histogram denkleştirme çerçeve işleme yazılımlarının en önemli parçalarından biridir. Bu işlem zıtlığı düzeltir ve histogram denkleştirmenin hedefi tek biçimli bir histogram elde etmektir. Bu teknik bütün çerçeve için veya çerçevenin bir parçası için kullanılabilir.

Histogram denkleştirme histogramı düzleştirmez. Yoğunluk dağılımını tekrar dağıtır. Eğer bir çerçevenin histogramında pek çok tepeler ve vadiler varsa, denkleştirme sonucu yine tepeler ve vadiler olacaktır, fakat tepeler ve vadiler kaydırılır. Bu yüzden histogram denkleştirmeyi tanımlamak için düzleştirmeden ziyade yayma daha iyi bir terim olacaktır.

Histogram denkleştirme bir nokta işlemi olduğu için yeni yoğunluklar çerçeveye eklenecektir. Var olan değerler yeni değerlere adreslenir, fakat sonuç çerçevesindeki yoğunlukların sayısı orijinal sayısından daha az veya eşit olacaktır.

İşlemler

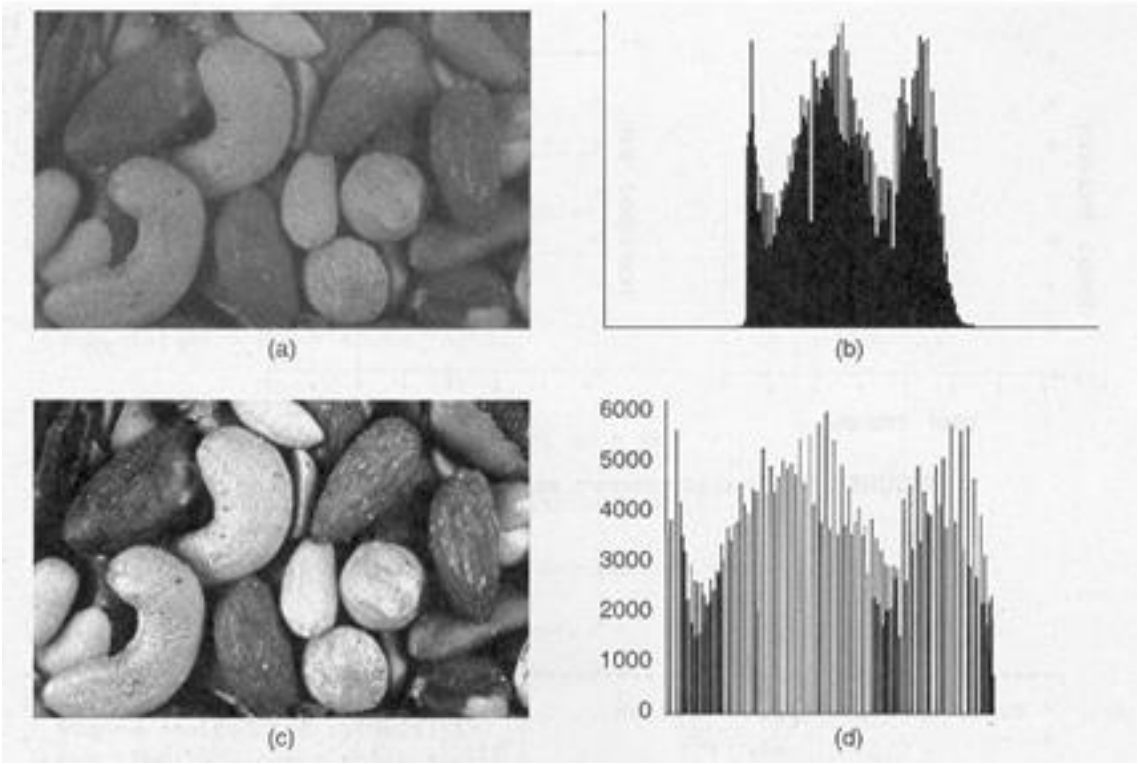
1. Histogramı hesapla
2. Normalleştirilmiş histogramı hesapla
3. Giriş çerçevesini çıkış çerçevesine dönüştür.

İlk adım çerçevedeki her piksel değerinin sayılması ile hesaplanabilir. Sıfır dizisi ile başlayabilirsiniz. 8-bitlik pikseller için dizinin boyutu 256'dır (0-255). Çerçeveyi ayrıştır ve her bir dizi elemanını uyan piksel değerine göre arttır.

İkinci adım histogram değerlerinin toplamının başka bir dizide saklanmasını gerektirir. Bu dizide I. eleman 0'dan I. elemana kadar olan elemanların toplamını içerecektir.

255. Eleman 0,1,…254,255 histogram elemanlarının toplamını içerecektir. Daha sonra bu dizi her bir elemanı (maksimum piksel değeri)/(piksel sayısı) ile çarpılarak normalleştirilir. 8-bitlik 512x512'lik bir çerçeve için bu sabit 255/262144 olacaktır

Histogram denkleştirme koyu bölgelerde detayları olan çerçevelerde daha iyi çalışır. Bazı kişiler diğer işlemleri uygulamadan önce çerçeveye histogram denkleştirme uygular. Bu, iyi bir uygulama değildir çünkü kaliteli çerçevelerin kalitesi bozulabilir.



Şekil 2.44. Histogram filtre uygulaması (Kazan 2011)

3. MATERYAL ve YÖNTEM

Bu bölüm tek nesne ve çoklu nesnelerin takibi olarak iki alt başlık altında incelenecektir. Her iki yaklaşımda da nesnelerin belirlenmesi için hareket vektörlerinden faydalanıldığından bunların belirlenmesi ile ilgili kısım benzerlik göstermektedir ve Tek Nesne Takibi başlığı altında açıklanmıştır. Ancak çoklu nesnelerin takibinde belirlenen hareket vektörleri uygun şekilde düzenlenerek; kaç tane nesne ve ne şekilde hareket ettikleri bilgisinin ayrıca oluşturulması gerektiğinden, bu durum Çoklu Nesne Takibi başlığı altında ayrıca sunulmuştur.

3.1. Tek Nesnenin Takibi

Bu başlık altında gerçekleştirilen nesne takibi bu nesneye ait hareket vektörü belirlenerek sağlanmıştır. Dolayısıyla amaç bir video içerisinde takip eden çerçevelerin her birinde ilgilenilen nesneye ait hareket vektörlerinin sağlıklı olarak belirlenmesi temeline dayanmaktadır. Bu amaçla kullanılan yaklaşım çerçevenin gerekli bölümünün 8x8 piksel ebatlarında alt bloklara ayrıştırılıp, her birine ait hesaplanan hareket vektörü bilgilerinin bir araya getirilerek, tüm nesnenin sahip olduğu hareket vektörünün dolayısıyla da nesnenin gerçekleştirdiği yer değiştirme bilgisinin hesaplanmasının belirlenmesine yöneliktir.

Bu amaçla kullanılan algoritmamız belirli özel durumlar için farklılık gösterse de aşağıda verilen ana bileşenlerden oluşmaktadır.

3.1.1. Hareket vektörü bilgisi ile nesnenin yer değiştirme bilgisini belirleme algoritması:

Bu algoritma biri önceki biri de yeni olmak üzere iki farklı çerçeve üzerinde çalışır.

- Adım 1: Yeni çerçevede nesne arama penceresi belirlenir

- Adım 2: Nesne arama penceresi alt bloklara ayrılır.
- Adım 3: Her bir alt bloğun diğer çerçevelerde de tespit edilmesi için kullanılacak ilgili özniteliği (feature) belirlenir.
- Adım 4: Önceki çerçevenin belirli bir alt bloğuna ait öznitelik matrisi yeni çerçevenin arama penceresi içerisindeki her bir bloğa ait öznitelik matrisleri ile karşılaştırılarak ilgili hareket vektörü belirlenir.
- Adım 5: Bu şekilde her bir bloğa ait belirlenen tüm hareket vektörleri bir nesneye ait çok sayıda gözlemlenmiş hareket vektörleri olarak değerlendirilip, tüm bloklara ait hareket vektörleri aşağıda açıklandığı şekilde bir araya getirilerek, nesnenin hareket vektörü belirlenir.
- Adım 6: Yeni çerçeve ve bunu takip eden çerçeveler sırasıyla eski ve yeni çerçeveler olarak güncellenirler ve algoritma Adım1'den itibaren tüm video çerçeveleri için tekrarlanır.

3.1.2. Tek nesne algoritması adımları açıklamaları

Bu bölümde ilgili hareket vektörü belirleme algoritması yürütülürken gerçekleştirilen işlem adımlarının açıklamaları verilmiştir.

Yeni çerçevede nesne arama penceresinin belirlenmesi: Bu amaçla ya çerçevenin tamamı ya da çerçevede ilgili nesnenin bulunma ihtimali olan tüm bölgesi kullanılmaktadır. İkinci durumdaki arama penceresi, en son belirlenmiş hareket vektörü etkisiyle iki çerçeve arasında geçen süre içerisinde oluşacak yer değiştirme büyüklüğü kadar bir önceki çerçeveye ait arama penceresi her doğrultuda genişletilerek elde edilir. Bu çalışmada bu arama penceresi genişletme büyüklüğü için en kötü hal üst değeri belirlenmiş ve tüm çalışmada bu sabit değer kullanılmıştır.

Nesne arama penceresinin alt bölümlere ayrılması: Arama penceresi 8x8 piksel ölçülerinde alt bloklara ayrılmış ve her biri yatay ve düşeydeki sırası kullanılarak endekslenerek işaretlemiştir.

Her bir alt bloğun ayırt edilmesinde kullanılacak ilgili özniteliğinin (feature) belirlenmesi: Bu çalışmada öznitelik olarak ilgili alt bölüme ait kenar (edge) bilgileri esas alınmıştır. Bu amaçla da yaygın olarak kullanılan Canny edge detector algoritması uygulanarak her bir alt bloğa ait kenar harita bilgisi elde edilmiş, bunlar alt blok endekslerine göre bir araya getirilerek ilgili çerçevenin ayırt edici öznitelik matrisi olarak kullanılmışlardır.

Alt blokların taranması: Önceki çerçevenin belirli bir alt bloğuna ait öznitelik matrisi yeni çerçevenin arama penceresi içerisindeki her bir bloğa ait öznitelik matrisleri ile karşılaştırılarak ilgili hareket vektörü belirlenir. Karşılaştırma işlemi iki büyüklük arasındaki farkın (hatanın) ortalama gücü büyüklüğü olan MSE (mean square error) kriterine göre değerlendirilir. Önceki çerçevenin belirli bir alt bloğuna ait minimum MSE değerini veren yeni çerçeve alt bloğu, ilgili bloğun yeni çerçeveye ait bloğun koordinatlarına hareket ettiği yaklaşımı ile bu hareket vektörü ilgili yer değiştirme vektörü olacak şekilde belirlenir. Karşılaştırma işleminde yeni çerçeve alt bloklarının taranma sırası kullanılan algoritmanın hızı dolayısıyla da performansı açısında önem kazanmaktadır. Buna göre, tarama işlemini arama penceresi içerisinde

- Tüm alt bloklar başlangıçtan sona sırayla
- Önceki çerçeve ilgili alt bloğu koordinatları bilgisi kullanılarak adaptif bir yaklaşımla

olmak üzere iki farklı biçimde gerçekleştirilebilir. Adaptif yaklaşımla kullanılan algoritma performansının ne şekilde geliştirdiği araştırma bulguları ve sonuçlar kısmının çizelge 4.1 bölümünde karşılaştırılmalı olarak sunulmuştur.

Hareket vektörünün belirlenmesi: Belirlenen tüm hareket vektörleri bir nesneye ait çok sayıda gözlemlenmiş hareket vektörleri olarak değerlendirilip, bir araya getirilerek o nesnenin hareket vektörü belirlenir. Bu işlem yapılırken elde edilen bir nesneye ait farklı hareket vektörü büyüklükleri çok sayıda gürültülü gözlem olarak değerlendirilir. Bu gürültünün kaynağı; gerçekten çerçevelerde oluşabilecek gürültü veya farklı

koordinatlarda yer alan ancak birbirinin aynı olan alt blokların karıştırılmasından kaynaklanan gürültü olabilmektedir. Dolayısıyla elde edilen bu ölçüm büyüklüklerinin doğasında yer alan bu gürültünün uzaklaştırılması gerekir. Bu amaçla elde edilen hareket vektörlerinin histogramında yer alan aykırı yani ortalamanın çok üzerinde veya altında olan değerler hatalı (gürültülü) olarak kabul edilip ayıklandıktan sonra, doğru kabul edilen kalan değerlerin ortalaması alınarak iki çerçeve arasında ki tek nesneye ait hareket vektörü belirlenmiş olur.

Güncelleme: Yeni çerçeve ve bunu takip eden çerçeveler sırasıyla eski ve yeni çerçeveler olarak güncellenirler ve algoritma Adım1'den itibaren tüm video çerçeveleri için tekrarlanır. Yeni çerçeve arama penceresinin resmin tamamı olmadığı durumlarda, bu arama penceresi bir önceki arama penceresi koordinatları elde edilen son hareket vektörünün belirlediği kadar ötelenerek güncellenir ve algoritma sürdürülür.

3.2. Çoklu Nesne Takibi

Bu bölümde birden fazla nesne takibi, yine bu nesnelere ait hareket vektörleri belirlenerek gerçekleştirilmiştir. Burada kullanılan yaklaşım; çerçevenin nesnelerin bulunduğu kabul edilen arama penceresinin 8x8 piksel ebatlarında alt bloklara ayrıştırılarak, nesnelerinin her birine ait hesaplanan hareket vektörlerinin x ve y eksenlerinde birbiriyle uygun şekilde eşleştirilip, her bir nesneye ait düzlemdeki hareket vektörlerini bulma temeline dayanmaktadır.

3.2.1. Hareket vektörü bilgisi ile nesnenin yer değiştirme bilgisini belirleme algoritması

Bu algoritma iki farklı çerçeve üzerinde çalışır. Bunlardan biri önceki diğeri de yeni çerçeve olarak adlandırılır.

- Adım 1: Arama penceresi yeni çerçevenin tamamı olarak seçilir.
- Adım 2: Nesne arama penceresi 8x8 alt bloklara ayrılır.

- Adım 3: Önceki çerçevenin belirli bir alt bloğuna ait öznitelik matrisi yeni çerçevenin arama penceresi içerisindeki her bir bloğa ait öznitelik matrisleri ile karşılaştırılarak, tüm hareket vektörleri belirlenir.
- Adım 4: Belirlenen hareket vektörlerinin birbirlerine olan uzaklıkları hem x hem de y eksenlerinde değerlendirilip, birbirlerine en yakın olanlar eşleştirilir ve ilgili nesnenin düzlemdeki hareket vektörü belirlenir.
- Adım 5: Yeni çerçeve ve bunu takip eden çerçeveler sırasıyla eski ve yeni çerçeveler olarak tüm nesneler için güncellenirler ve algoritma Adım1'den itibaren tüm video çerçeveleri için tekrarlanır.

3.2.2. Çoklu nesne algoritması adımları

Bu bölümde ilgili hareket vektörü belirleme algoritması yürütülürken gerçekleştirilen işlem adımlarının açıklamaları verilmiştir.

Nesne arama penceresinin alt bölümlere ayrılması: Arama penceresi 8x8 piksel ölçülerinde alt bloklara ayrılmış ve her biri yatay ve düşeydeki sırası ölçüsünde endekslenerek işaretlemiştir.

Alt blokların taranması: Önceki çerçevenin belirli bir alt bloğuna ait öznitelik matrisi yeni çerçevenin arama penceresi içerisindeki her bir bloğa ait öznitelik matrisleri ile karşılaştırılarak ilgili hareket vektörü belirlenir. Bu işlem bölüm 3.1.2'de verilen alt blokların taranması ile aynı olup daha fazla detaya girilmemiştir.

Hareket vektörünün belirlenmesi (gürültü ayıklama): Çerçevede MSE (mean square error) kriterine göre gerçekleştirilen alt blokların taranması adımı sonrasında x ve y eksenlerinin her ikisi içinde birer hareket vektörü matrisi elde edilir. Bu hareket vektörü matrislerinin boyutu her bir 8x8 lik alt bloğa bir tane gelecek şekilde çerçevenin büyüklüğüne bağlıdır. Mesela 512x512 lik bir çerçeve için bu matris 64x64 ebatlarında iki ayrı matris olacaktır. Çalışmamızda bunların indekslenmesi de alt blokların indekslenmesi gibi yapılmıştır. Bu iki adet hareket vektörü matrisleri incelendiğinde

çerçeve içerisindeki nesnenin hareket alanı içerisindeki parlaklık değişimi, kamera çekim gürültüsü, arka plan gürültüsü, ortam gürültüsü vb. etkenlerden dolayı her bir nesneye ait blokların hareket vektörü değerleri bir takım gürültülü gözlemler şeklinde tespit edilmektedir.

Yani bu matristeki her bir nesneye ait bloklar, tespit edilmek istenen gerçek hareket vektörü büyüklüğü ve bozucu etkilerin neticesinde oluşan gürültü bileşenlerinden oluşmaktadır.

Çalışmamızda bu gürültü bileşenlerinin uzaklaştırılmasında sırasıyla median ve histogram filtreleme yöntemlerinin yer aldığı bir algoritma kullanılmıştır. Buna göre; önce hareket vektörü matrisinin farklı boyutlardaki pencerelerine median filtreleme işlemi uygulanır. Bunun sonrasında elde edilen median filtrelenmiş hareket vektörü matrislerinde her bir nesneye ait blokların hareket vektörü büyüklükleri; merkez blokta nesneye ait gerçek hareket vektörü değerleri, sınır bloklarda ise bu değerden giderek azalan hareket vektörleri değerlerinin yer aldığı büyüklükler şeklinde sıralanmaktadır. Bu etkiyi gidermek amacıyla merkezde ağırlıklı miktarda bulunan gerçek hareket vektörü değerlerini belirlemek için histogram filtreleme adımı kullanılmıştır. Bu yaklaşımla ilgili nesneye ait tüm blokların hareket vektörü değerleri, histogram filtrede baskın olarak bulunan büyüklükler ile yer değiştirilmiştir. Sonuç bölümünde farklı boyutlardaki median ve histogram filtrelerin algoritma sonuçları verilmiştir.

Hareket vektörlerinin eşleştirilmesi: Bulunan hareket vektörlerinin hangi nesneye ait olduğunu belirlemek algoritmanın önemli bir noktasıdır. Bir nesne her iki eksende de hareket edebileceği gibi sadece x ekseninde veya sadece y ekseninde de hareket edebilir. Bu durumda nesnenin hareket etmediği eksendeki hareket vektörü büyüklüğü sıfır olacaktır. Aynı zamanda hareketli nesneye ait olmayan (arka plan bilgisi) bloklara ait hareket vektörleri de sıfır olmalıdır. Dolayısıyla sıfır büyüklüğündeki hareket vektörünün ne tür bir bloğa ait (arka plan bloğu veya ilgili eksende hareket olmayan blok) olduğunun anlaşılması kullanılan algoritmada gürültü ayıklamadan sonraki en önemli adımı oluşturmaktadır.

Bu amaçla kullanılan algoritma aşağıdaki şekilde çalışmaktadır:

- x eksenindeki ve y eksenindeki nesnelere ait hareket vektörü matrisleri ayrı ayrı vektör haline getirilir.
- Vektörlerdeki sıfır değerine sahip elemanlar yukarıda açıklanan sebepten dolayı vektörden çıkarılır.
- Vektörün histogramı hesaplanır.
- Histogramdaki değerleri büyükten küçüğe doğru sıralanır.
- En büyük elemandan başlanarak bu değerlerin birikmiş toplamı %90'a giren histogram büyüklükleri tutulup, kalanlar atılır.
- Tutulan histogram büyüklüğü değerlerinin hareket vektörü matrisindeki ait oldukları blokların indekslerinin ortalaması alınarak ilgili nesneye ait hareket vektörünün orijinini gösteren indeks bulunur. Bu indeksten hareketle videonun ilgili çerçevesindeki nesneye ait orijinin piksel bazındaki koordinatları hesaplanır. Bu işlem hem x hem de y eksenleri için ayrı ayrı tekrarlanır.

x ekseninde bulunan orijinlerle y ekseninde bulunan orijinler arasındaki kartezyen çarpımlarının oluşturduğu ikili koordinat çiftleri arasındaki öklid uzaklıkları hesaplanır. Bunlardan önceden belirlenmiş bir eşik değerin altında olanlar aynı nesneye ait farklı eksenlerdeki hareket vektörleri olarak eşleştirilirken, eşik değerinin üzerindekiler ise tek bir eksende hareket eden nesnelerin hareket vektörleri olarak tanımlanır ve buna uygun olarak işaretlenir.

Güncelleme: Yeni çerçeve ve takip eden çerçeveler sırasıyla her bir nesne için eski ve yeni çerçeveler olarak güncellenirler ve algoritma Adım1'den itibaren tüm video çerçeveleri için tekrarlanır. Arama penceresinin çerçevenin tamamı olmadığı durumlarda, bu arama penceresi bir önceki arama penceresi koordinatları elde edilen hareket vektörünün belirlediği oranda kaydırılarak güncellenerek algoritma sürdürülür.

4. ARAŞTIRMA BULGULARI ve TARTIŞMA

Bu bölümde yapılan farklı çalışmalar verilmiş, elde edilen sonuçlar değerlendirilmiştir.

4.1. Tamamıyla Gerçek Olmayan Video Çerçevelerinde Elde Edilmiş Algoritma Sonuçları

Bu bölümde verilen sonuçların elde edildiği video çerçevelerinin ya arka planı ya da nesneleri gerçek olmayıp MATLAB ortamında simülasyon olarak oluşturulmuştur. Yapılan çalışmalar dört ana başlık altında incelenebilir. Bunlar;

1. Sentetik olarak hazırlanmış, homojen /siyah zemin üzerinde beyaz nesnenin kenar bulma algoritması ile takibi.

2. Sentetik olarak hazırlanmış, homojen /siyah zemin üzerinde beyaz nesnenin hareket vektörleri kullanılarak takibi.

3. Gerçek olarak kaydedilmiş video çerçevesinde, homojen /siyah zemin üzerinde sarı nesnenin (tenis topu) kenar bulma algoritması ile takibi.

4. Gerçek olarak kaydedilmiş video çerçevesinde, homojen /siyah zemin üzerinde sarı nesnenin (tenis topu) hareket vektörleri kullanılarak takibi.

5. Gerçek olarak kaydedilmiş video çerçevesinde, heterojen zemin üzerinde birden fazla beyaz nesnenin hareket vektörleri kullanılarak takibi.

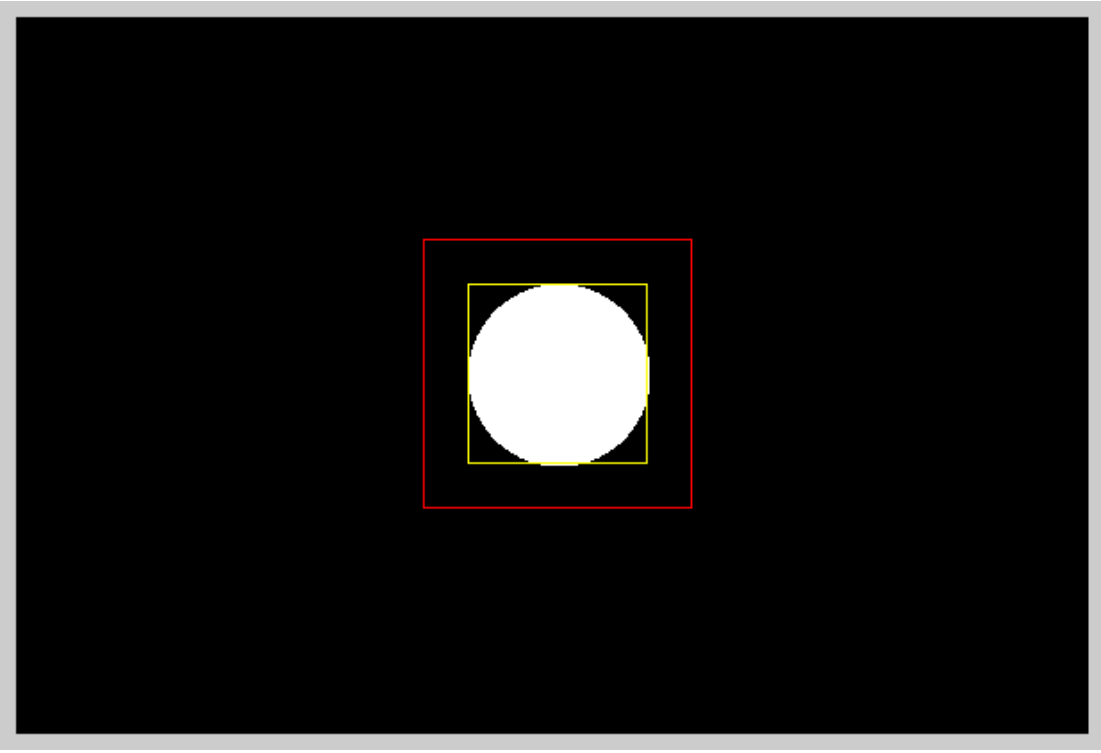
Bu durumda kullanılan tarama yaklaşımı önemli bir rol oynamaktadır. Algoritma sonucunda elde edilen sonuçlar spiral taramanın normal taramaya göre daha hızlı cevap veren bir algoritma olduğudur. Çünkü spiral taramada işlem eşleştirme yapılacak bloğun bir sonraki çerçevede olma ihtimalinin yüksek olacağı alandan başlanılarak yürütülür.

Normal taramada ise işlem resmin sol üst köşedeki ilk bloğundan başlanır ve resmin tamamı taranır. Bu tarama işleminin dezavantajı aranan bloğun bulunduğu bölgeye gelmeden önce gereksiz birden fazla blok taranmasıdır.

Tamamıyla gerçek olmayan video çerçevelerine uygulanan her bir çalışmanın detayı ve elde edilen sonuçları aşağıda sunulmuştur.

Algoritma 1: Homojen arka plan üzerinde hareket eden sentetik olarak hazırlanmış nesneyi her çerçevede kenar bulma algoritması yardımıyla tespit edip, takip etme. Burada sarı dikdörtgen nesnenin kenarlarını kullanarak oluşturulur. Kırmızı dikdörtgen ise 2. çerçevede resmin tamamının değil de bu alan için kenar algoritması uygulanması için yol gösterir.

Algoritma-1 sonuçları:



Şekil 4.1. Homojen arka plan üzerinde hareket eden sentetik olarak hazırlanmış nesnenin “Canny” algoritmasıyla bulunması

Algoritma 2: Homojen arka plan üzerinde hareket eden sentetik olarak hazırlanmış nesneyi her çerçevede hareket vektörleri yardımıyla tespit edip, takip etme. Burada amaç nesneye ait hareket vektörlerini bulup bir sonraki çerçevede nesneyi arayacağımız bölgeyi doğru belirlemektir.

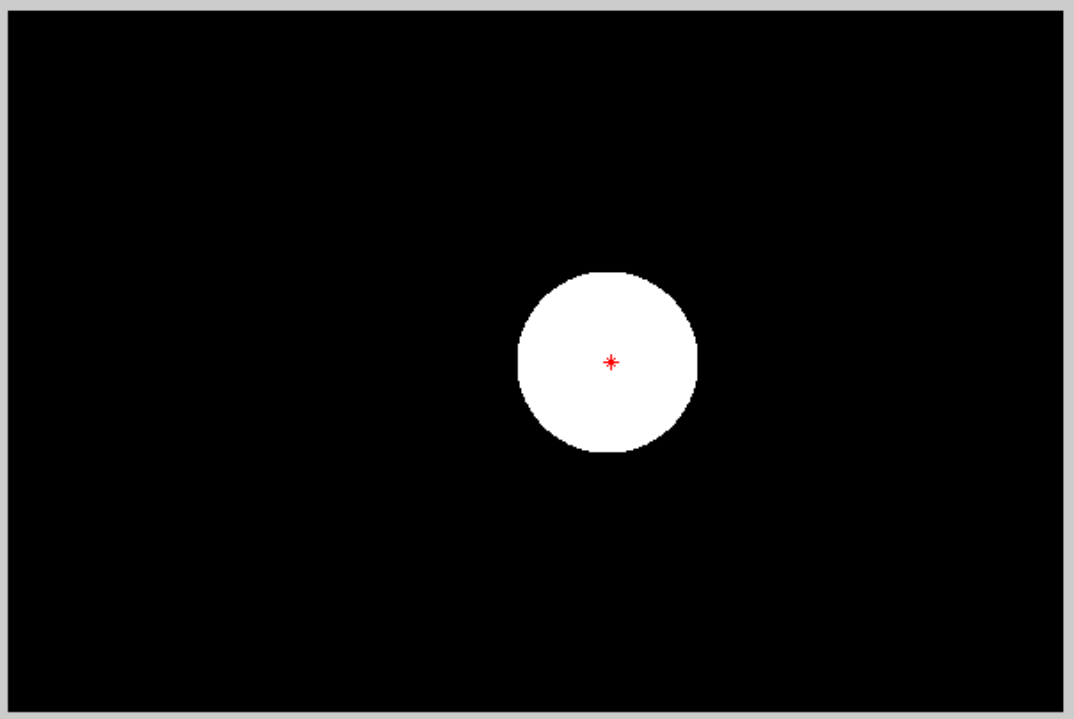
Algoritma-2 sonuçları:

X ekseni hareket vektörleri =

0 0 0 0 0 0 0

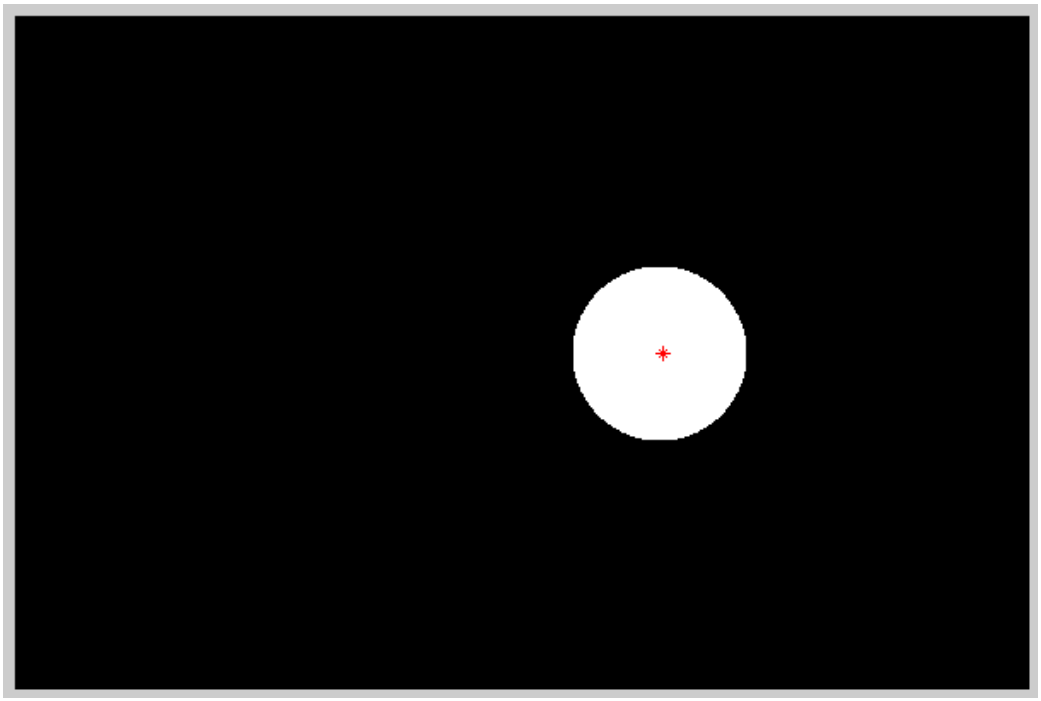
Y ekseni hareket vektörleri =

0 4 8 12 16 20 24

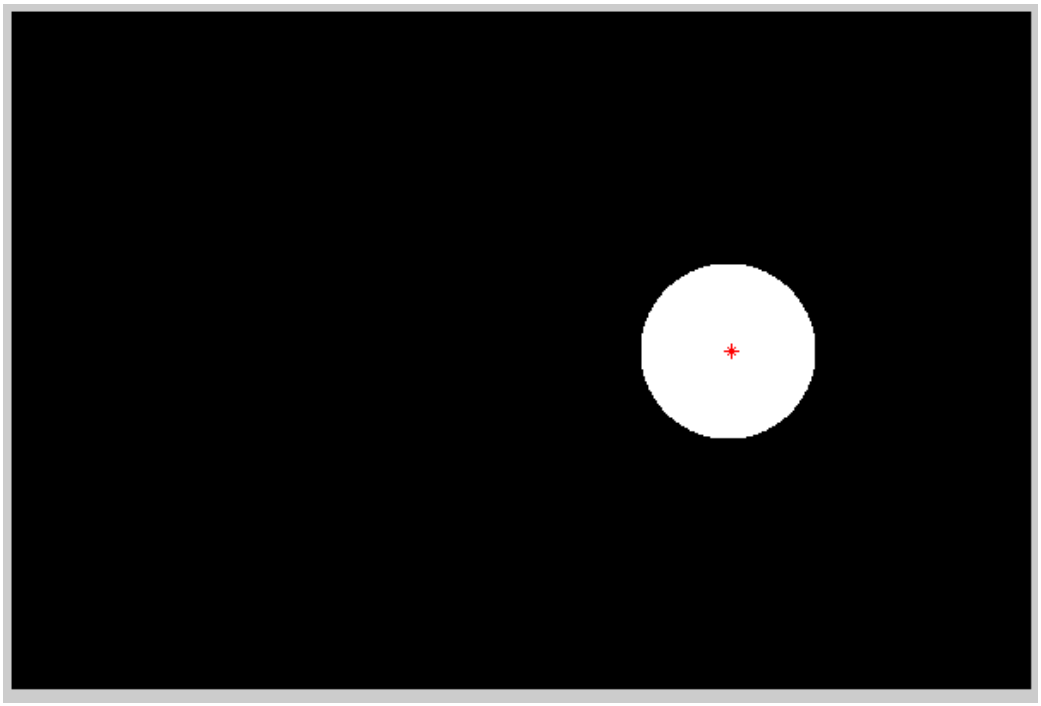


(a)

Şekil 4.2. (devam)



(b)



(c)

Şekil 4.2. Homojen arka plan üzerinde hareket eden nesnenin hareket vektörleriyle bulunması
(a) 10. Çerçeve, (b) 20. Çerçeve, (c) 30. Çerçeve

Şekil 4.2’de görüldüğü gibi uygulanan algoritma ile homojen arka plan üzerinde hareket eden beyaz nesne, her çerçevede hareket vektörleri kullanılarak, merkezi hesaplanıp kırmızı yıldız ile işaretlenmiştir.

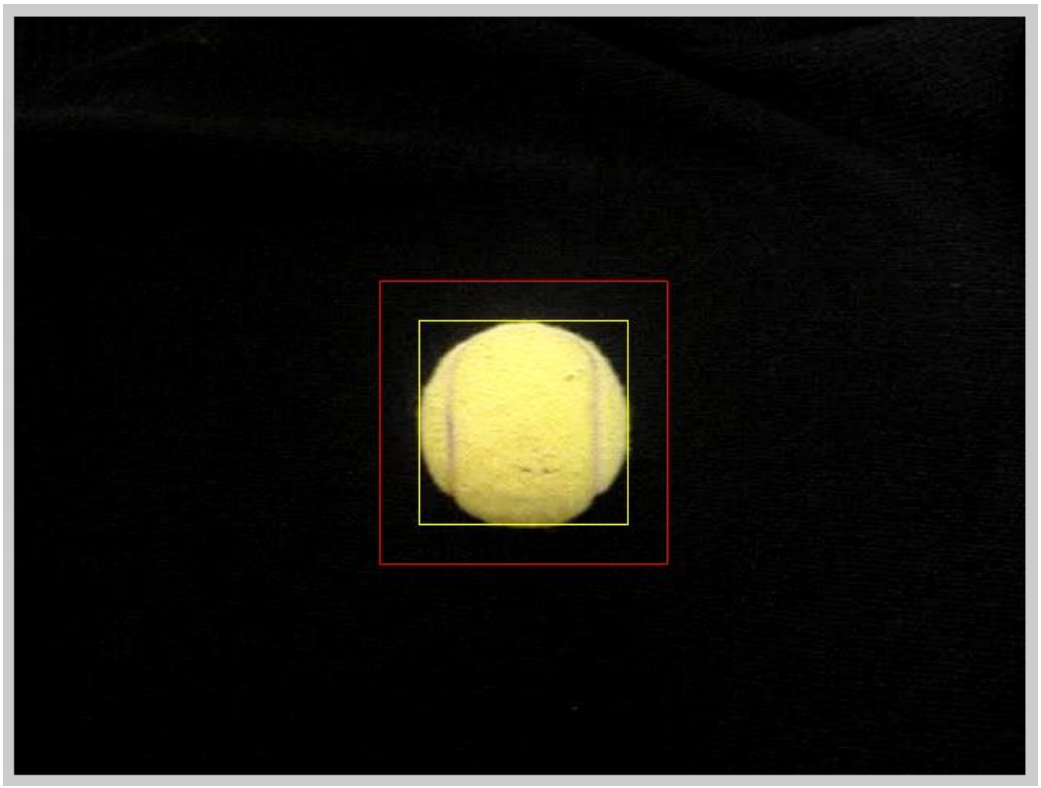
(a)

(b)

Şekil 4.3. (a) hareketli nesnenin x eksenindeki hareket vektörleri matrisi, (b) hareketli nesnenin y eksenindeki hareket vektörleri matrisi

Algoritma 3: Homojen arka plan üzerinde hareket eden gerçek olarak kaydedilmiş video çerçevesinde nesneyi her çerçevede kenar bulma algoritması yardımıyla tespit edip takip etme. Burada sarı dikdörtgen nesnenin kenarlarını kullanarak oluşturulur. Kırmızı dikdörtgen ise 2. çerçevede çerçevenin tamamının değil de bu alan için kenar algoritması uygulanması için yol gösterir.

Algoritma-3 sonuçları:



Şekil 4.4. Homojen arka plan üzerinde hareket eden gerçek olarak kaydedilmiş video çerçevesinde nesnenin "Canny" algoritmasıyla bulunması

Algoritma 4: Homojen arka plan üzerinde hareket eden gerçek olarak kaydedilmiş video çerçevesinde nesneyi her çerçevede hareket vektörleri yardımıyla tespit edip takip etme. Burada amaç nesneye ait hareket vektörlerini bulup bir sonraki çerçevede nesneyi arayacağımız bölgeyi doğru belirlemektir.

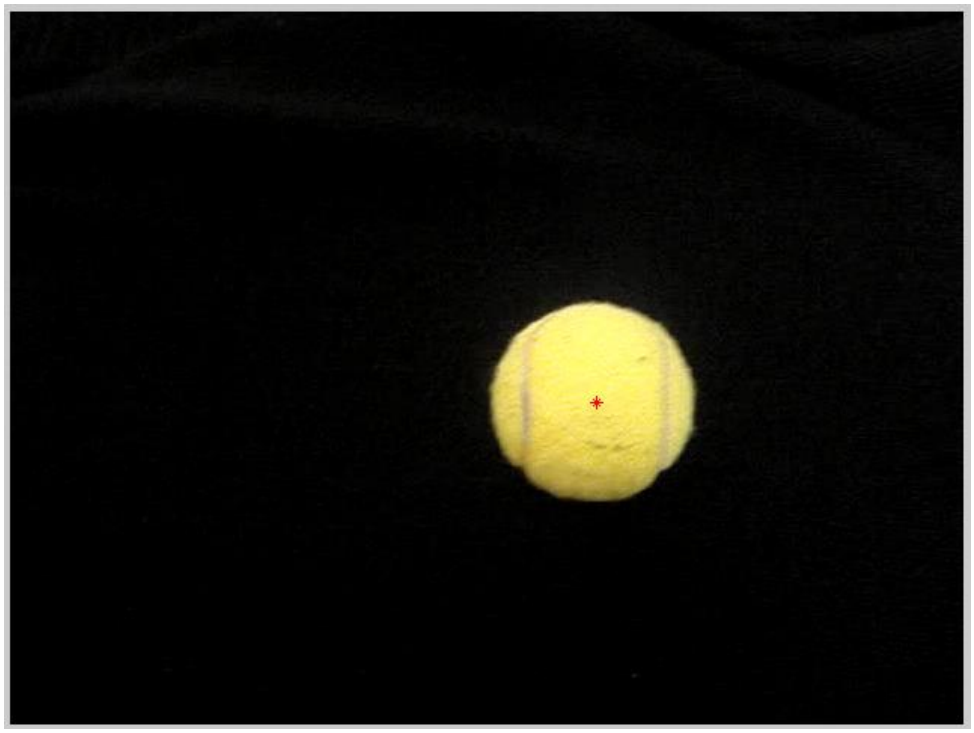
Algoritma-4 sonuçları:

X ekseni hareket vektörleri=

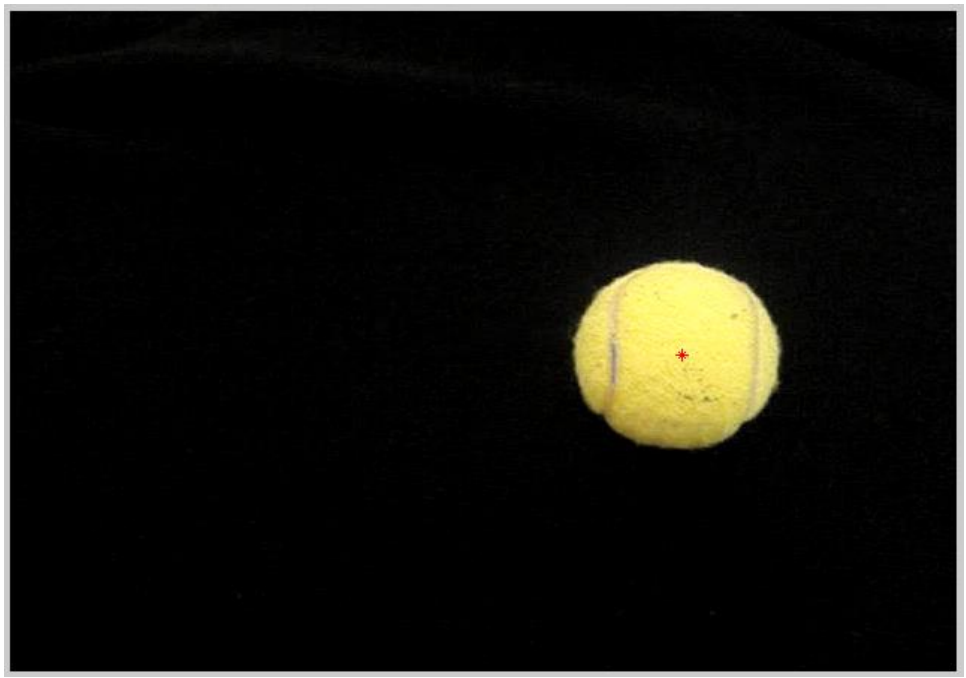
0 1 2 4 5 6 11

Y ekseni hareket vektörleri=

0 4 5 16 25 26 53

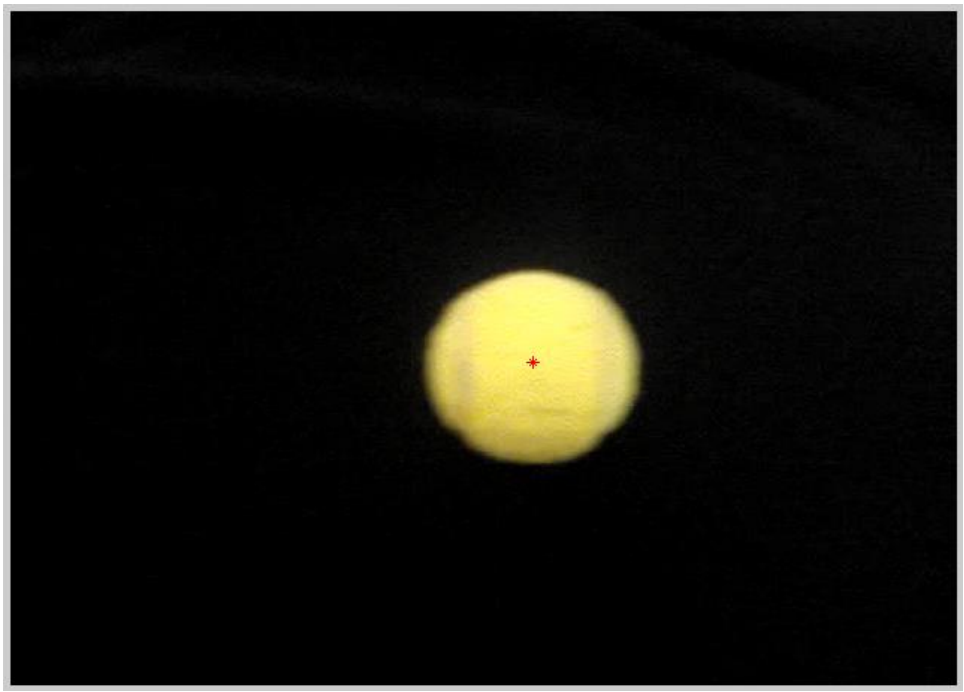


(a)



(b)

Şekil 4.5. (devam)



(c)

Şekil 4.5. Homojen arka plan üzerinde hareket eden gerçek olarak kaydedilmiş video çerçevesinde nesnenin hareket vektörleriyle bulunması
(a) 10. Çerçeve, (b) 20. Çerçeve, (c) 30. Çerçeve

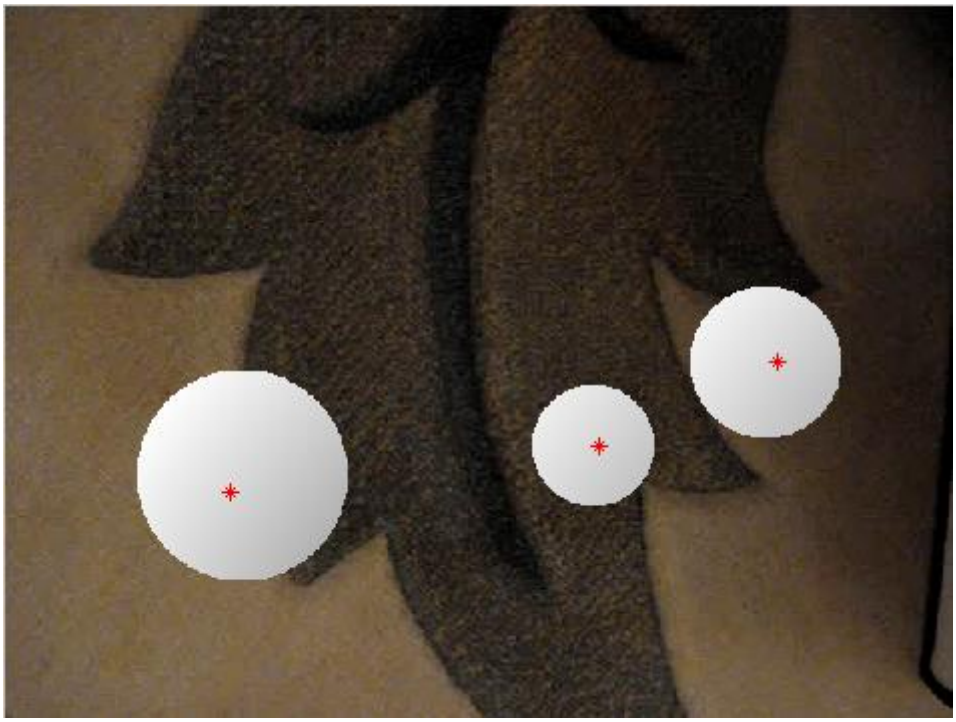
Şekil 4.5'de görüldüğü gibi uygulanan algoritma ile homojen arka plan üzerinde hareket eden gerçek olarak kaydedilmiş sarı nesne (tenis topu), her çerçevede hareket vektörleri kullanılarak nesnenin merkezi hesaplanıp kırmızı yıldız ile işaretlenmiştir.

Algoritma 5: Heterojen arka plan üzerinde hareket eden sentetik video çerçevesinde birden fazla nesneyi her çerçevede hareket vektörleri yardımıyla tespit edip takip etme. Burada nesnelere ait hareket vektörleri eşleştirmesi önemlidir.

Algoritma-5 sonuçları: Aşağıda kaydedilmiş bir geri plan görüntüsü üzerine oluşturulmuş 3 adet farklı büyüklükte ve farklı doğrultularda hareket eden toplar şeklinde hareketli nesnelerin yer aldığı bir video görüntüsü üzerinde elde edilen sonuçlar verilmiştir. Hesaplanan tüm hareket vektörleri kullanılan algoritma ile muhtemel nesneler ile eşleştirilip bu bilgiden hareketle hesaplanan nesnelere ait merkez

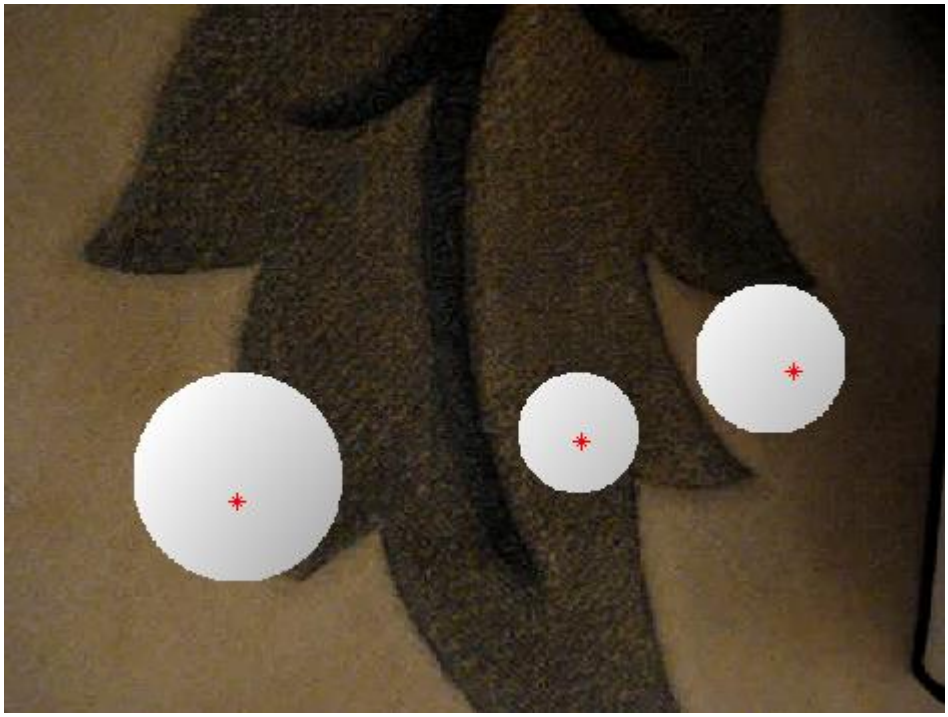
noktalar görüntü çerçeveleri üzerinde işaretlenip, birbirini takip eden 5 çerçeve için Şekil 4.6'da verilmiştir. Verilen sonuçlardan da görüleceği üzere, algoritma ile elde edilen sonuçlar görsel olarak da tatminkâr sonuçlar vermektedir.

Kullanılan algoritmaların çalışma süreleri açısından performans karşılaştırması Çizelge 4.1'de verilmiştir. Burada görüldüğü üzere arka planın homojen olduğu durumlarda nesnenin sentetik veya gerçek video nesnesi olma durumlarına göre algoritma hızı değişmektedir. Öte yandan, homojen arka plan üzerinde gerçek video nesnesinde nesne gürültüsünden ve video çekim gürültüsünden kaynaklanan gürültüler olmaktadır. Kullanılan algoritmalarda yer alan gürültü temizleme aşamaları algoritmanın işlem yükünü artırıp, işlem süresinin daha uzun olmasına sebebiyet vermektedir.



(a)

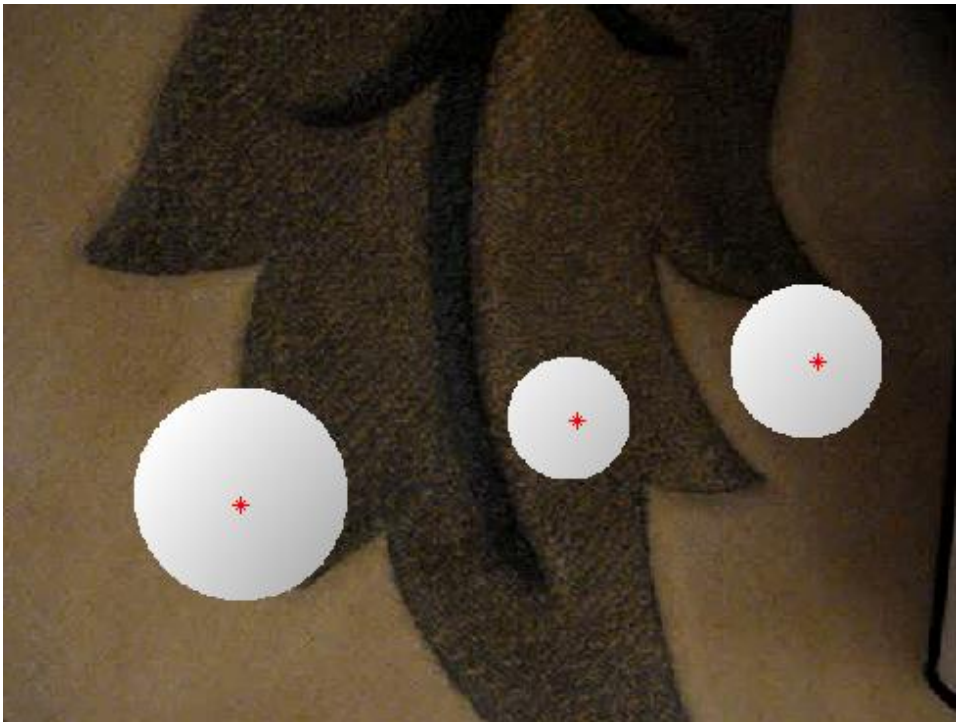
Şekil 4.6. (devam)



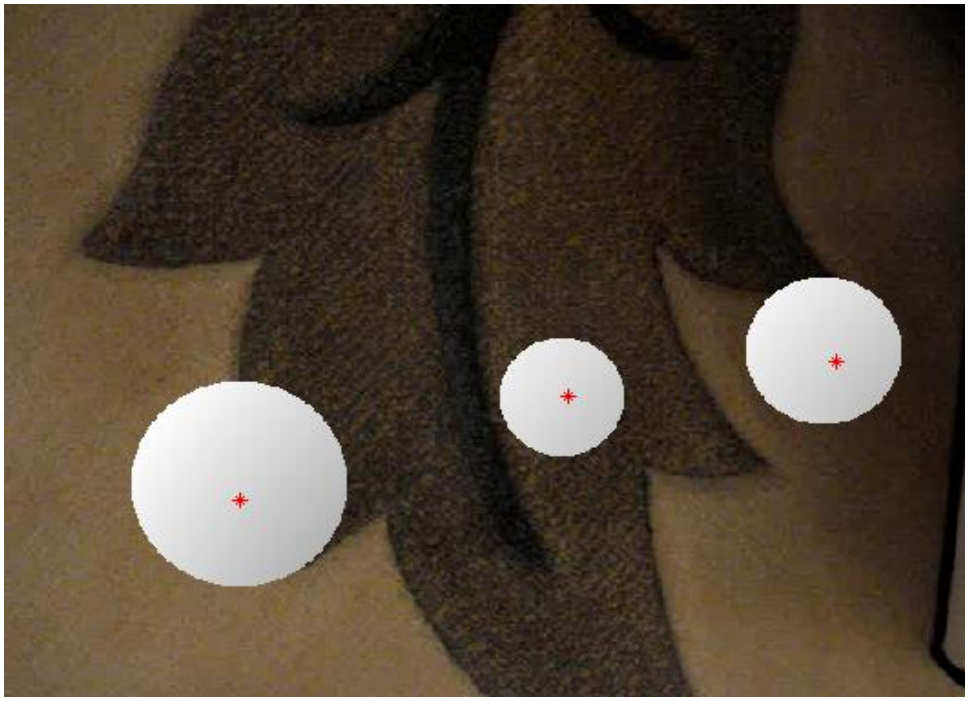
(b)

(c)

Şekil 4.6. (devam)

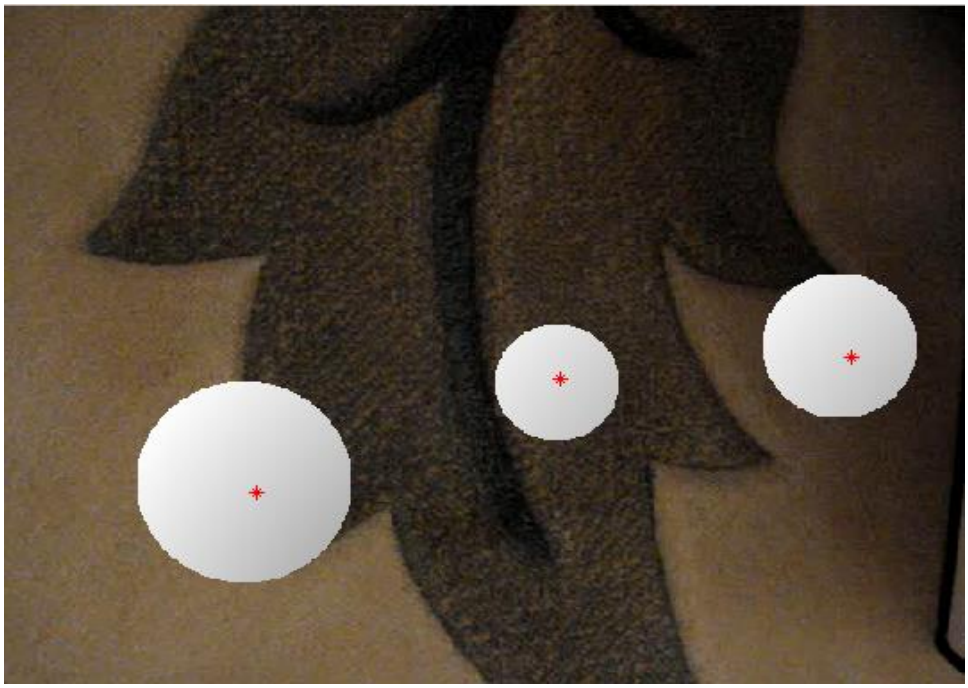


(d)



(e)

Şekil 4.6. (devam)



(f)

Şekil 4.6. Heterojen arka plan üzerinde sentetik olarak oluşturulmuş birden fazla nesnenin yer aldığı birbirini takip eden beş video çerçevesinde nesne takibi
(a) 24. Çerçeve, (b) 25. Çerçeve, (c) 26. Çerçeve, (d) 27. Çerçeve, (e) 28. Çerçeve, (f) 29. Çerçeve.

Şekil 4.6'da görüldüğü gibi uygulanan algoritma ile heterojen arka plan üzerinde hareket eden birden fazla beyaz nesne, her çerçevede hareket vektörleri kullanılarak nesnelerin merkezi hesaplanıp kırmızı yıldız ile işaretlenmiştir

Çizelge 4.1. Normal ve spiral tarama algoritmaları için algoritma çalışma süreleri

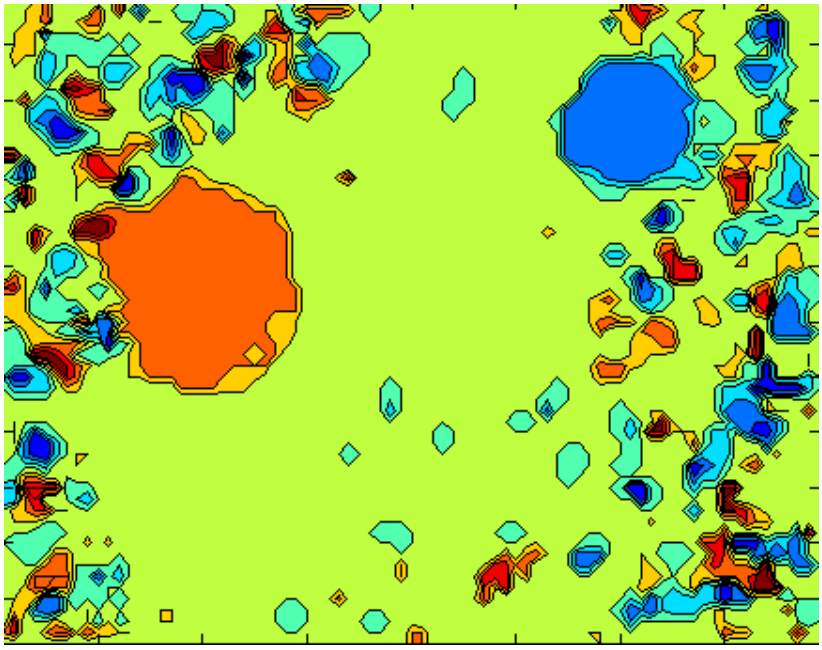
	Homojen arka plan sentetik nesne	Homojen arka plan gerçek video nesnesi	Heterojen arka plan sentetik nesnesi
normal tarama	53 sn	70 sn	189 sn
spiral tarama	3.5 sn	5.75 sn	7 sn

4.2. Gürültü Giderme

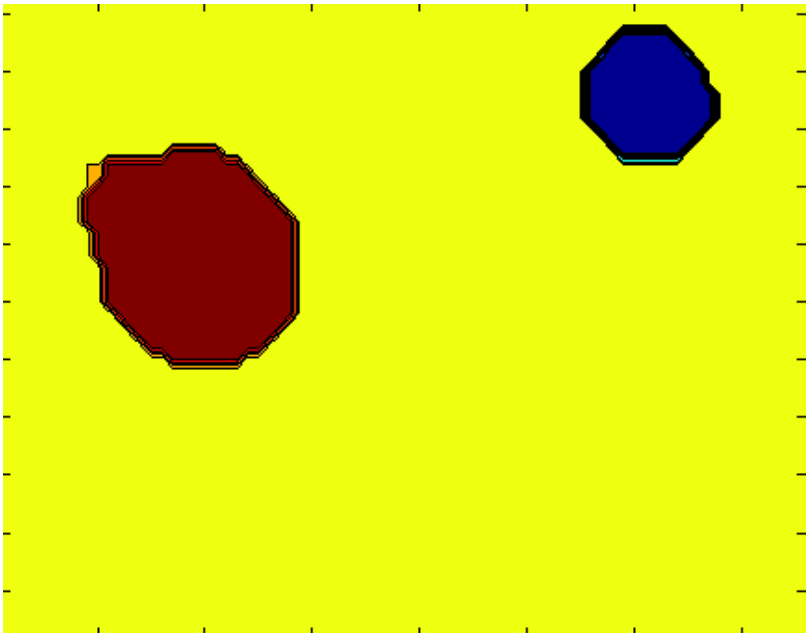
Algoritmalarda gürültü ayıklama adımı önce median sonrasında histogram filtrenin bir arada kullanımı ile gerçekleştirilmiştir. Bu yaklaşımın hareket vektörleri matrisleri üzerinde yer alan gürültüyü gidermedeki performansıyla ilgili sonuçlar bu bölümde sunulmuştur. Literatürde bu amaçla kullanılan median filtrenin performansının algoritmalarımızda yer alan histogram filtre ile artırılabildiği Çizelge 4.2 ve Çizelge 4.3'ün karşılaştırılmasından görülmektedir. Çizelge 4.2'de elde edilen hareket vektörleri matrisindeki gürültünün ayıklanması için sadece farklı ebatlarda iki boyutlu median filtrenin kullanılması durumunda elde edilen ve karesel hataların ortalaması şeklinde hesaplanmış hata değerleri verilmiştir. Burada sırasıyla 3x3, 5x5, 7x7, 9x9 ve 11x11 komşuluğunda median filtre uygulanarak hareket vektörleri matrislerinde bulunan gürültüler ayıklanmıştır. Öte yandan Çizelge 4.3'te aynı sonuçlar her bir median filtreleme işleminin sonrasında yine sırasıyla 3x3, 5x5, 7x7, 9x9 ve 11x11 komşuluklarında gerçekleştirilen histogram filtreleme için verilmiştir.

Örnek olarak 7x7 komşuluğunda yapılan median filtrelemedeki hata değeri olan 1.5960 olan büyüklüğü, sonrasında kullanılan histogram filtreleme ile 1.2564 değerine düşmüştür. Her iki çizelgedeki bütün verilen karşılaştırmasından her bir durumda histogram filtrenin kullanımıyla yapılan hata değerlerinin azaldığı açıkça görülmektedir.

Bu da kullanılan bu gürültü giderme yaklaşımının işe yaradığını göstermektedir. Bu durum görsel olarak Şekil 4.7, Şekil 4.8'de verilmiştir. Burada Şekil 4.7'de görülen x eksenine ait median filtreleme sonucunda elde edilen hareket vektörleri matrisindeki gürültü değerleri, Çizelge 4.3'den elde edilen en optimum filtreleme boyutu olan 7x7 komşuluğunda median ve histogram filtrelemenin bir arada uygulanması durumunda temizlenmiş ve sonuçlar Şekil 4.8'de sunulmuştur. Yani hareket vektörü matrisinde nesnelere (2 adet top) ait olmayan gürültü bileşenleri etkin bir biçimde ayıklanmıştır.



Şekil 4.7. Sadece 3x3 boyutunda median filtre uygulanması sonucunda x eksenine ait hesaplanan hareket vektörü büyüklükleri matrisinin MATLAB contour komutu ile eş yükselti eğrileri şeklindeki grafiği



Şekil 4.8. 7x7 boyutunda median ve histogram filtrelerin bir arada uygulanması sonucunda x eksenine ait hesaplanan hareket vektörü büyüklükleri matrisinin MATLAB contour komutu ile eş yükselti eğrileri şeklindeki grafiği

Öte yandan sunulan sonuçların bütünlüğü açısından, sadece histogram filtreleme sonucunda elde edilen benzer hata büyüklükleri ise Çizelge 4.4'te sunulmuştur. Beklendiği üzere burada ki değerler her iki durumdan da büyük olup histogram filtrenin tek başına en iyi yaklaşım olmadığını göstermektedir.

Sadece Çizelge 4.3 ün incelenmesinde görüleceği üzere, histogram filtrenin boyutu sabit kaldığı durumlarda median filtrenin boyutu artarken hata oranı azalmaktadır. Median filtrenin boyutunun sabit kaldığı durumlarda ise histogram filtrenin artması belirli bir değere kadar (7x7) hatayı azaltırken, bu değeri aşıldığında hatayı artırmaktadır. Çizelgelerde görüldüğü gibi en az hata histogram filtre boyutunun küçük, median filtre boyutunu yüksek olduğu durumlarda elde edilmiştir.

Çizelge 4.2. Sadece median filtre ile gürültü ayıklamada elde edilen hareket vektörü matrislerindeki ortalama hata büyüklükleri

median filtre boyutu	**x ekseni**	**y ekseni**	**toplam**
3x3	4,3924	3,9908	8,3832
5x5	1,1688	1,2242	2,393
7x7	0,6831	0,9129	1,596
9x9	0,6672	0,9809	1,6481
11x11	0,7443	0,9719	1,7162

Çizelge 4.3. Hem median hem de histogram filtrelerin bir arada kullanımı ile gürültü ayıklamada elde edilen hareket vektörü matrislerindeki ortalama hata büyüklükleri

HİSTOGRAM FİLTRE BOYUTU

MEDIAN FİLTRE BOYUTU		**3x3**	**5x5**	**7x7**	**9x9**	**11x11**
3x3	**x**	2,905	1,2939	0,6229	0,6284	0,6887
	y	2,6178	13.249	0,9239	0,8979	0,9039
	toplam	5,5228	2,6188	1,5468	1,5263	1,5926
5x5	**x**	0,8411	0,6154	0,5289	0,5392	0,5626
	y	0,9514	0,8596	0,7105	0,6681	0,7217
	toplam	1,7925	1,475	1,2394	1,2073	1,2843
7x7	**x**	0,5179	0,5316	0,4984	0,4945	0,5275
	y	0,7436	0,7562	0,758	0,7535	0,7588
	toplam	1,2615	1,2878	1,2564	1,248	1,2863
9x9	**x**	0,508	0,5336	0,5146	0,5566	0,5045
	y	0,5966	0,7056	0,731	0,7726	0,8408
	toplam	1,1046	1,2392	1,2456	1,3292	1,3453
11x11	**x**	0,4666	0,5445	0,5764	0,5842	0,6396
	y	0,5727	0,7186	0,7773	0,9091	0,982
	toplam	1,0393	1,2631	1,3537	1,4933	1,6216

Çizelge 4.4. Farklı boyutlardaki sadece histogram filtrelerin uygulandığı program sonucu karşılaştırmaları

histogram filtre boyutu	**x ekseni**	**y ekseni**	**toplam**
3x3	14,3961	12,8165	27,2126
5x5	7,0084	5,6737	12,6822
7x7	3,5233	3,0141	6,5374
9x9	1,7038	1,8385	3,5423
11x11	1,1001	1,96	3,0601

4.3. Tamamıyla Gerçek Video Çerçevelerinde Elde Edilmiş Algoritma Sonuçları

Bu bölümde verilen sonuçlarda video çerçevelerinde hem arka planı hem de hareketli olan nesne gerçektir. Burada yer alan arka plan karışık renkli zemin şeklindeki kilim, hareketli nesne ise sarı renkteki tenis topundan oluşmakta olup, hareket vektörleri kullanılarak bunu takibi yapılmıştır. Video çerçevelerinde arka planın heterojen ve nesnenin gerçek nesne olması sebebiyle hem arka plan gürültüsü hem de nesne ve video çekim gürültüleri algoritmayı etkilemektedir. Önceki bölümlerde açıklanan gürültü giderme yaklaşımlarıyla bu genel durumda da gürültü bileşenlerinin etkin bir biçimde giderilebildiği Şekil 4.9 ve Şekil 4.10'da görsel olarak verilmiştir. Bu şekiller ile verilen sonuçlar sırasıyla x ve y eksenleri için ilgili video çerçevelerinde elde edilen hareket vektörleri matrislerini ve birbirini takip eden beş video çerçevesinde nesne takibi için elde edilen ve nesne merkezinin kırmızı yıldız ile işaretlendiği sonuçları yansıtmaktadır.

```
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  -3  -3  -3  -3  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  -3  -3  -3  -3  -3  -3  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  -3  -3  -3  -3  -3  -3  -3  -3  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  -3  -3  -3  -3  -3  -3  -3  -3  -3  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  -3  -3  -3  -3  -3  -3  -3  -3  -3  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  -3  -3  -3  -3  -3  -3  -3  -3  -3  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  -3  -3  -3  -3  -3  -3  -3  -3  -3  -3  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  -3  -3  -3  -3  -3  -3  -3  -3  -3  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  -3  -3  -3  -3  -3  -3  -3  -3  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  -3  -3  -3  -3  -3  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Şekil 4.9. 11x11 boyutunda median ve histogram filtrelerin bir arada uygulanması sonucundaki x eksenine ait hareket vektörleri matrisi

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 6 6 6 0 0 0 0 0 0
0 0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 6 6 6 6 0 0 0 0 0
0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 6 6 6 6 0 0 0 0 0
0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 6 6 6 6 0 0 0 0 0
0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 6 6 6 6 0 0 0 0 0
0 0 0 0 0 0 0 4 4 4 4 6 6 4 4 4 6 6 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Şekil 4.10. 11x11 boyutunda median ve histogram filtrelerin bir arada uygulanması sonucundaki y eksenine ait hareket vektörleri matrisi

(a)

(b)

(c)

(d)

(e)

Şekil 4.11. Heterojen arka plan üzerinde gerçek nesnenin yer aldığı birbirini takip eden beş video çerçevesinde nesne takibi. Nesne merkezi kırmızı yıldız ile işaretlenmiştir.
(a) 10. Çerçeve, (b) 11. Çerçeve, (c) 12. Çerçeve, (d) 13. Çerçeve, (e) 14. Çerçeve

5. SONUÇ

Bu çalışmada, nesne takibinde etkili bir yöntem geliştirilmeye çalışıldı. Çalışma, tek bir nesne takibi ve birden fazla nesne takibi olmak üzere iki aşamada gerçekleştirildi. Nesneyi arka plandan ayırmak ve belirgin bir şekilde işaretlemek için Canny kenar bulma algoritması kullanıldı. Nesnenin hangi yöne ve nasıl gittiğini bulmak için MSE kriteri kullanıldı. MSE kriterinde sıralı tarama yöntemine göre daha hızlı ve daha doğru sonuçlar elde etmemizi sağlayan spiral tarama yöntemi geliştirildi. Spiral tarama sonucunda elde edilen hareket vektörleri median ve histogram filtreleri yardımıyla gürültülerden ve bozucu etkilerden temizlendi. Sonrasında nesnenin hareket vektörü belirlendi. Birden fazla nesnenin takip edildiği durumlarda hareket vektörleri nesnelerin merkez koordinatlarına göre birbiriyle eşleştirildi. Bu yeni yaklaşımda hareket vektörü kullanarak bir veya birden fazla nesne takibinde hızlı ve doğru sonuçlar elde edildi.

Videolarda arka plandan veya nesnenin video çekim yada hareket gürültüsünden kaynaklanan bozucu etkileri ortadan kaldırmak için etkili bir yöntem olan median filtre kullanıldı. Ve çerçevede gözle görülür şekilde başarılı bir düzeltme sağlandı. Median filtrenin doğası geriye kalan küçük miktardaki bozucu etkiler ise histogram filtre kullanılarak videodan temizlendi.

MSE kriteri uygulanılırken farklı tarama yöntemleri kullanıldı. Normal tarama yönteminde bir sonraki çerçevede aranacak blok resmin tamamında ilk bloktan başlanarak aranmaya başlandı. Bu yöntemin uzun sürede sonuçlanması sonucunda adaptif bir yaklaşım olan spiral tarama geliştirildi. Bu yöntemde bir sonraki çerçevede aranacak blok resmin tamamında değil, bloğun olması en muhtemel arama bölgesinden başlanarak tarandı. Ve blok eşleştirmesinin kısa sürede başarıyla gerçekleştiği sonucuna varıldı.

Birden fazla nesne takibinde, her bir nesne için uygulanan algoritmalar sonucunda elde edilen hareket vektörlerinin nesnelerle eşleştirilmesi, vektörlerin eksenlerdeki merkez

noktalarının koordinatlarının birbirlerine olan uzaklık durumlarına bakılarak yapılmıştır. Bu eşleştirmeler küçük hata oranlarıyla başarıyla sonuçlanmıştır.

Birden fazla nesne takibinde, tek bir nesneyi takip etmeye göre daha fazla algoritmaya ve daha fazla süreye ihtiyaç duyulmuştur. Ancak MSE, spiral tarama ve hareket vektörleri eşleme yöntemlerinin bu ihtiyaçları başarıyla karşıladığı sonucuna varılmıştır.

KAYNAKLAR

Akbulut,O., 2005. Blok Hareket Kestirimi. Kocaeli Üniversitesi, Türkiye.

Anonymous, 2014a. http://en.wikipedia.org/wiki/Video_tracking.[10,10,2009,WEB].

Collins, R., 2006. Mean Shift Tracking. Department of Computer Science and Engineering, 40 s, USA.

Csetvericov, D., 2000. Basic Algorithms For Digital Image Analysis. Institute Of Informatics Universty, 24 s, Budapest.

Geckle,W., 2000. Block Matching Algorithms For Motion Estimation. Lisans Tezi. IEEE Transactions, volume 7 issue 2, 429-433 s.

Gilvarry,J., 1999. Extraction of Motion Vectors from an MPEG Stream. School of Electronic Engineering Dublin City University, Ireland.

Gyaourova, A., Kamath,C., Cheung,S., 2003. Block Matching For Object Tracking. Lawrence Livermore National Laboratory, California.

Hu, J., Juan, C., Wang, J., 2008. A Spatial Color Mean Shift Object Tracking Algorithm. Pattern Recognition Latters, volume 29 issue 16, 2165-2173 s.

Ilic, S., 2012. Mean Shift Tracking. Techischen Universitat München, Germany.

Kazan,S., 2011. Analysis of the Moment Based Image Normalization Effects. 2010 IEEE 18. Signal Processing and Communications Apllications Conference, Diyarbakır.

Kim,Y., 2007. Obejcet Tracking in a Video Sequence. Lisans Tezi, Stanford Universty, USA.

Liu, B. Zaccarin, A., 1993. New Fast Algorithms For The Estimation of Block Motion Vectors. IEEE Transactions, volume 3 issue 2, 148-157 s.

Mythili, C., Kavitha,V., 2011. The Research Bulletin of Jordan ACM. Efficient Technique For Color Image Noise Reduction, part 2. Universty Collage of Engineering Nagercoil, India.

Mythili, C., Kavitha,V., 2011. The Research Bulletin of Jordan ACM. Efficient Technique For Color Image Noise Reduction, part 2. Universty Collage of Engineering Nagercoil, India.

Nedovic, V., 2003. Tracking Moving Video Objects Using Mean Shift Algorithm. Universty Of Amsterdam, Holland.

Oppenheim, A., Verghese,G., 2010. Signals Systems and Inference class notes. MIT , USA.

Sebe, İ., 2002. Object Tracking Using Multiple Constraints. Y.Lisans Tezi, Stanford Universty, USA.

Siyah,B., 2013. Görüntü Filtreleme Uygulamaları ve Amaçları. www.bulentsiyah.com

Smith, K., 2009. 2D Tracking. Selected Topics in Computer Vision. Light Microscopy and Screening Center, Zurich.

Tamersoy, B., 2009. Background Subtraction. The Universty of Texas, 28 s, Austin.

Tang, H., Wu, T., Lin,Y., 2003. Real Time Object Image Tracking Based On Block Matching Algorithm. Universty of Wisconsin, 18 s, USA.

Thrun, S., Kosecka, J., 2007. Tracking Motion Stanford CS223B Computer Vision, USA.

Wang, Y., Chen,F., Guo,H., 2012. Kernel Based Target Tracking With Spatial Histogram. Acta Automatica Sinica, volume 38 issue 3, 430-435 s.

Yılmaz, A., Javed, O., Sahah, M., 2006. A Survey Of Object Tracking. Ohio State Univesty, 45 s,USA.

ÖZGEÇMİŞ

Handenur DEMİRCİ 1987 yılında Erzurum'da doğdu. İlk, orta ve lise öğrenimini Erzurum'da tamamladı. 2005-2009 yılları arasında Atatürk Üniversitesi Elektrik&Elektronik Mühendisliği ana bilim dalında lisans derecesini aldı. Mezuniyetinin ardından 2009 yılında Yüksek lisans eğitimine başladı. 2011 yılında TRT Genel Müdürlüğü'nde mühendis olarak göreve başladı ve halen bu görevi sürdürmektedir.