
GAZİ ÜNİVERSİTESİ
Gazi
1926

SAYISAL GÖRÜNTÜ İŞLEME İLE ETKİLİ NESNE TAKİP METODLARININ GELİŞTİRİLMESİ VE UYGULANMASI

İsa ŞAHİN

YÜKSEK LİSANS TEZİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAYIS 2014

İsa ŞAHİN tarafından hazırlanan "SAYISAL GÖRÜNTÜ İŞLEME İLE ETKİLİ NESNE TAKİP METODLARININ GELİŞTİRİLMESİ VE UYGULANMASI" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Elektrik Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Doç. Dr. Fırat HARDALAÇ

Elektrik Elektronik Mühendisliği Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ...……

Başkan : Prof. Dr. Etem KÖKLÜKAYA

Elektrik Elektronik Mühendisliği Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ……...

Üye : Yrd. Doç. Dr. Hüseyin POLAT

Bilgisayar Mühendisliği Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ……...

Tez Savunma Tarihi: 23/06/2014

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

…….……

Prof. Dr. Şeref SAĞIROĞLU

Fen Bilimleri Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

İsa ŞAHİN

SAYISAL GÖRÜNTÜ İŞLEME İLE ETKİLİ NESNE TAKİP METODLARININ GELİŞTİRİLMESİ VE UYGULANMASI

(Yüksek Lisans Tezi)

İsa Şahin

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Mayıs 2014

ÖZET

Sayısal görüntü işleme, çok geniş uygulama alanı olan bilgisayarla bütünleşik bir çalışmadır. Sayısal görüntü verileri kullanılarak iyileştirmeler veya daha farklı görüntüler elde edilebilmekte ve nesne tanımlama ve takip işlemleri uygulamanın ihtiyacına göre farklı hızlarda gerçekleştirilebilmektedir. Sayısal görüntü işleme bir dizi işlem basamağından oluşmaktadır. Amaca yönelik bu basamaklar artırılabilir ya da azaltılabilir. Nesne takip uygulamalarında bu işlemler hedef görüntünün yakalanmasıyla başlar ve farklı tekniklerin kullanılmasıyla devam eder. İçerisinde filtreleme, aday görüntünün iyileştirilmesi ve hedef görüntüyle karşılaştırılması aşamalarını uygun matematiksel yöntemlerle barındırmaktadır. Nesne takip teknikleri üzerinde uzun yıllardır yapılan çalışmalarda, farklı birçok yöntem denenmiştir, fakat kullanılan yöntemler, farklı uygulamalar için gerçek ortam resimlerinde istenildiği kadar etkili sonuç vermemiştir. Mükemmel takip için, nesnelerin özelliklerinin ortam özellikleriyle bir arada kullanılmasına dayanan yöntemler üzerine çalışmalar devam etmektedir. Bu çalışmada hedef nesneyi takip için farklı teknikler üzerinde deneyler yapılmış ve bu deney sonuçlarına göre etkili yöntemler seçilmiştir. Geliştirilen yöntemde daha etkili sonuçlar alabilmek amacıyla hedef görüntü ve aday görüntüler üzerinde farklı filtre teknikleri araştırılmış ve uygulanmıştır. Sonuç olarak nesnenin görüntü özellikleri kullanılarak, resimler arasındaki benzerlik hesaplanmakta ve sonrasında eşleşen bölgeler arasında benzerlik katsayısına göre nesneyi bulma işlemi yapılmaktadır.

Bilim Kodu : 905.1.014
Anahtar Kelimeler : Sayısal görüntü işleme, Nesne takip
Sayfa Adedi : 93
Danışman : Doç. Dr. Fırat HARDALAÇ

DEVELOPMENT AND IMPLEMENTATION OF EFFECTIVE OBJECT TRACKING METHODS USING DIGITAL IMAGE PROCESSING

(M. Sc. Thesis)

İsa ŞAHİN

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

June 2014

ABSTRACT

Digital image processing is a study, which is computer integrated with a wide range of application. Using digital image data, the improvements and different images can be obtained and object recognition and tracking operations can be carried out at different speeds according to application needs. Digital image processing consist of a series operational processes. Those processes can be reduced or increased according the purpose. In object tracking application, the processes begin with capture of target image and continue by using different techniques. Tracking techniques consist of some steps of mathematical method like filtering, improving the candidate image and comparing with target image. Throughout the years there have been several different methods used for object recognition. However the methods used are far from providing satisfactory results for real time applications and for real environment images. Studies for ideal object tracking, in relation to their regional or object-oriented features, are currently being carried on.In this study, experiments have been applied on different techniques to track target object and according to the results of these experiments effective methods have been selected. In the methods developed in order to achieve more efficient results, different filtering techniques have been researched and implemented. As a result, using the image properties of object, the similarity between images is calculated and then the process of tracking is applied according to similarity coefficients between matching regions.

Science Code : 905.1.014
Key Words : Digital image processing, Object tracking
Page Number : 93
Supervisor : Assoc. Prof. Dr. Fırat HARDALAÇ

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla ben yönlendiren Hocam Doç. Dr. Fırat HARDALAÇ'a yine kıymetli tecrübelerinden faydalandığım hocam Araş. Gör. Uğurhan KUTBAY'a, ayrıca NOVOS Tıbbi Cihazlar'daki tüm çalışma arkadaşlarıma, manevi destekleriyle beni hiçbir zaman yalnız bırakmayan çok değerli arkadaşım Özlem KAYA'ya teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

RESİMLERİN LİSTESİ

Resim **Sayfa**

1.GİRİŞ

Bir görüntü x ve y koordinatları uzaysal (düzlemsel) olan iki boyutlu bir fonksiyon olarak tanımlanabilir ve herhangi bir koordinat değerinin (x,y) genliğinin bu noktadaki değeri görüntünün gri düzeyi olarak adlandırılır. x,y ve genlik değerleri sonlu olduğunda bu görüntü sayısal görüntüye dönüşmüş olur. Bir sayısal görüntün sonlu sayıdan oluşan elemanlarının her biri belli bir konuma ve bir değere sahiptir. Bu elemanlar resim ve görüntülerin pels ve piksel değerleri olarak adlandırılır. Piksel yaygın olarak, bir sayısal görüntünün öğelerini belirtmek için kullanılan bir terimdir. Daha resmi bir şekilde vizyon olarak adlandırılan bu terim insanın görsel algılamasında çok önemli bir rolü vardır. Ancak insan gözünün görebildiği sınırlar elektromanyetik spektrumun görsel bandı ile sınırlıdır. Bazı görüntüleme cihazları sayesinde gama ışınlarından radyo dalgalarına kadar geniş aralıktaki görüntü elde etmek mümkün olabilmektedir. Böylece alınan bu görüntüler insan gözü için anlamlı hale getirilebilir ve sayısal görüntü işleme çok farklı alanlarda kullanılabilir.

Bu çalışmada öncelikle ikinci bölümde dış dünyadan alınan sayısal görüntünün iyileştirilmesi için gerekli metotlar incelenmiştir. Görüntü gri veya renkli olarak iki şekilde işleme tabi tutulabilir. Yaygın olarak görüntü daha hızlı işlem sağladığı için gri seviyelerde incelenmektedir. Fakat renkli görüntü verileri daha geniş ve doğruluk değeri yüksek sonuçlar ortaya çıkarabilmektedir. Bu yüzden üçüncü bölümde renkli görüntü özellikleri ve görüntü işlemede kullanılan yöntemler üzerinde durulmuştur.

Nesne izle yöntemleri üzerinde geliştirme ve uygulama yapmadan önce dördüncü bölümde yaygın olarak kullanılan izleme yöntemleri detaylı olarak incelenmiştir. Bu bölümden çıkan sonuçlar doğrultusunda avantajlar ve dezavantajlar değerlendirilerek, sonraki bölümlerde yapılacak olan gelişmiş bir nesne izleme yöntemi için veriler değerlendirilmiş olacaktır.

Beşinci bölümde etkin bir nesne izleme yazılımı için örnek çalışmalara yer almaktadır. Görüntü takip uygulamalarında kullanılan işleme teknikleri detaylı bir şekilde incelenmiş ve yazılım çıktılarıyla desteklenmiştir. Bu bölümdeki çalışmada gelişmiş nesne takip metotları üzerinde yoğunlaşılmış ve hataları en aza indirmek için farklı teknikler üzerinde deneyler yapılmıştır. Geliştirilmiş uygulama için kullanılan Mean-Shift algoritması ile

farklı filtrelerle desteklenerek hataların en aza indirilmesi sağlanmıştır. Kullanılan yöntemde nesne takibi için alınan görüntü histogram geri projeksiyon, momentlerin hesaplanması ve görüntü bölütleme gibi aşamalardan geçmektedir ve K-Means filtresiyle doğruluk oranı artırılmaktadır.

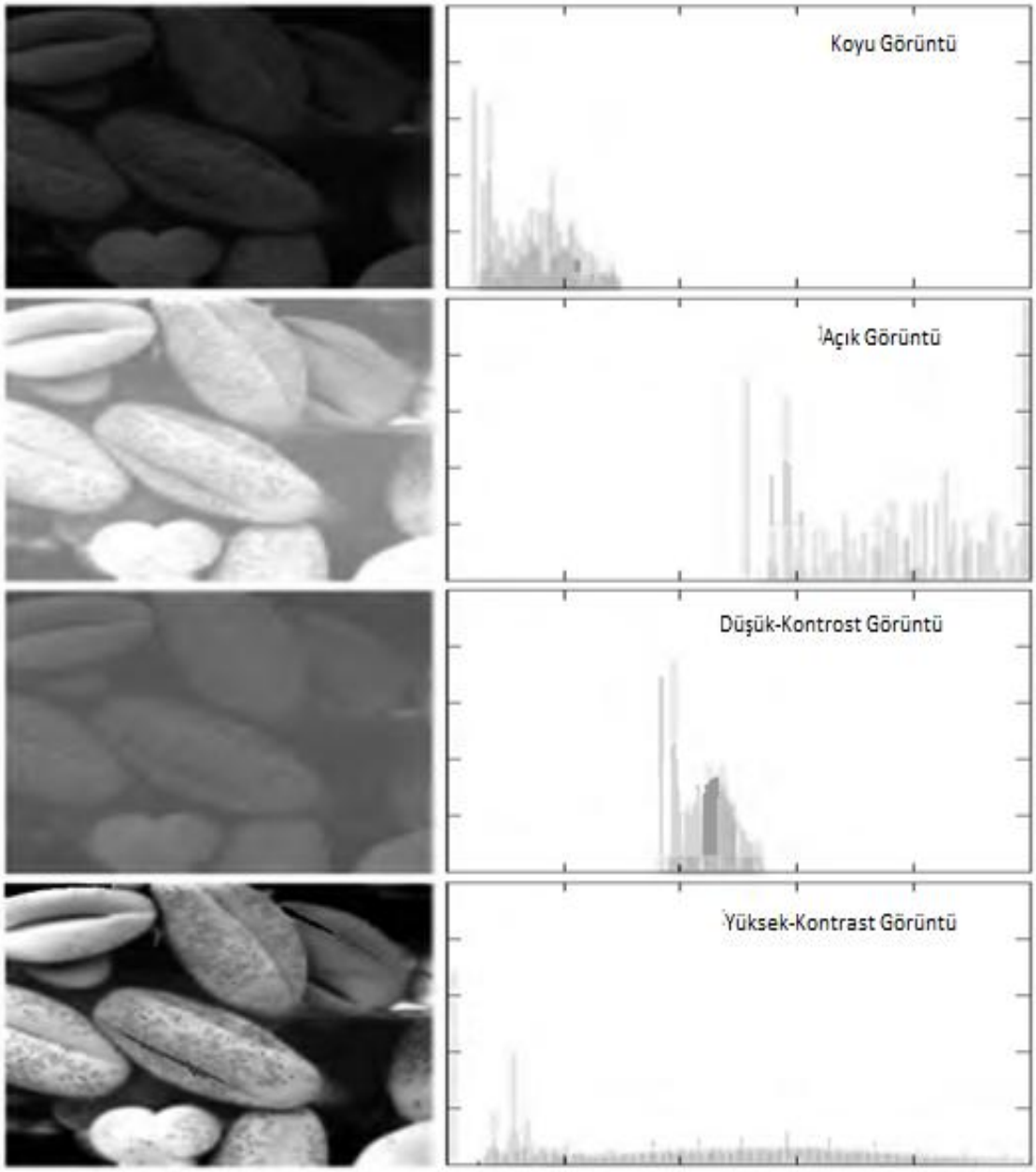
2. FİLTRELEME METODLARI

2.1. Histogram İşleme

rk, Arth gri seviye olduğunda ve nk, rk gri seviyesine sahip olan görüntüdeki piksel sayısı olduğunda; [0,L-1] aralığındaki gri seviyeli sayısal görüntünün histogramı, h(rk)=nk aralıklı fonksiyonu olur. Bir histogramı, her bir değerini görüntüdeki n ile ifade edilen toplam piksel sayısına bölerek standartlaştırmak genel bir uygulamadır. Bu yüzden, standartlaştırılmış bir histogram, k=0,1,…, L-1 için p{rk}=nk jn tarafından elde edilir. Detaya girmeden ifade edecek olursak, p{rk}, rk gri seviyesinin ortaya çıkma ihtimalinin tahminini verir. Standartlaştırılmış bir histogramın bütün bileşenlerinin toplamı 1'e eşit olmaktadır.

Histogramlar, 3 boyutlu (uzaysal) küme işleme teknikleri için temel oluştururlar. Histogram işleme, görüntü geliştirme için etkili bir şekilde kullanılabilir [7]. Kullanışlı görüntü istatistiği sağlamanın yanı sıra, alt başlıklarda, histogramlarda bulunan bilginin aynı zamanda görüntü sıkıştırma ve parçalara ayırma gibi diğer görüntü işleme uygulamalarında da oldukça kullanışlı olduğunu göreceğiz. Histogramları yazılımla hesaplamak kolaydır ve aynı zamanda histogramlar ekonomik donanım uygulamalarına katkıda bulunurlar, böylece gerçek zamanlı görüntü işleme için popüler bir araç haline gelirler. Görüntü geliştirmede histogram işlemenin rolüne bir giriş olarak, dört temel gri seviye özelliklerinde (koyu, açık, düşük zıtlık ve yüksek zıtlık) gösterilen imajın sorgulanan görüntüsü Şekil 2.1 de gösterilmiştir. Figürün sağ tarafı, histogramın bu görüntülerle uygun olduğunu gösterir. Her bir histogram grafiğinin yatay ekseni, rk gri seviye değerlerine uygundur. Eğer değerler standartlaştırılırsa, dikey eksen, h(rk)=nk ya da p{rk}=nkjn değerlerine uygun olur. Bu yüzden, daha önce de ifade edildiği gibi, bu histogram grafikleri h(rk)=nk'ye karşı rk ya da p{rk}=nkjn'ye karşı rk'nin basit grafikleridir. Histogram bileşenlerinin, koyu görüntüde gri ölçeğin düşük (koyu) tarafına yoğunlaşmasına dikkat edilmektedir. Benzer şekilde, parlak görüntünün histogram bileşenleri, gri ölçeğin yüksek tarafına doğru eğilimlidir. Düşük zıtlıklı bir görüntü, dar bir histograma sahiptir ve gri ölçeğin ortasına doğru yoğunlaşacaktır. Monokrom (tek renkli) bir görüntü için, bu durum mat, solmuş gri görünüm anlamına gelir. Son olarak, yüksek zıtlıklı görüntüdeki histogram bileşenlerinin gri ölçeğin geniş bir alanını kapsadığı görülmektedir ve dahası, piksel dağılımının diğerlerinden çok daha yüksek olan birkaç

dikey doğru ile eş dağılımdan çok fazla uzaklaşmadığı görülür. Öngörüsel olarak, pikselleri mümkün olan gri seviyelerin tüm aralığında bulunmaya ve eşit oranda dağıtılmaya meyilli olan bir görüntünün, yüksek zıtlıklı bir görünüme sahip olacağı ve geniş çeşitlikte gri tonlar sergileyeceği sonucunu çıkarmak makuldür. Belirgin etki (ağ etkisi), çok miktarda gri seviye ayrıntısı gösteren ve yüksek dinamik aralığa sahip olan bir görüntü olacaktır. Sadece girdi görüntüsünün histogramında bulunan bilgiye bağlı olarak bu etkiye otomatik olarak ulaşan bir dönüşüm fonksiyonu geliştirmenin mümkün olduğu kısaca gösterilecektir.



Şekil 2.1. Dört temel resim türü: karanlık, aydınlık, düşük kontrastlı, ve bunlara karşılık gelen histogramlar

2.1.1. Histogram eşitleme

Bir süreliğine sürekli fonksiyonları düşünülsün ve geliştirilecek görüntünün gri seviyelerini r değişkeni temsil etsin. Önceki bölümde, r'nin siyahı temsil eden r=0 ve beyazı temsil eden r=1 ile, [0,1] aralığında standartlaştırıldığı varsayıldı. Daha sonra, aralıklı formülleştirme ele alındı ve piksel değerlerinin [0, L-1] aralığında olmasına izin verildi.

Daha önce anlatılan durumlara uyum sağlayan her r için, orijinal görüntüdeki her r piksel değeri için bir s seviyesi oluşturan,

$$s = T(r) \qquad 0 < r < 1 \tag{2.1}$$

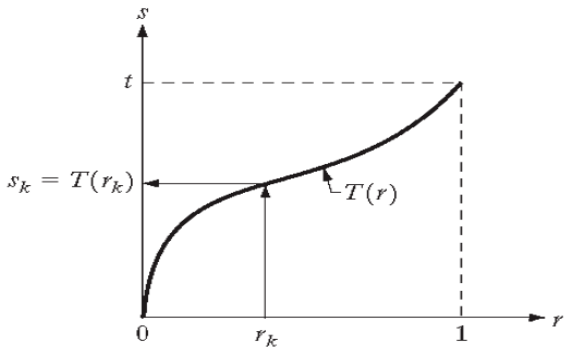
modelinin dönüşümlerine dikkat çekildi. Belirtilecek sebeplerden dolayı, T(r) dönüşüm fonksiyonunun aşağıdaki durumlara uygun olduğu varsayılır.

1. T(r), tek değerlidir ve 0<r<1 aralığında monoton bir şekilde artar ve
2. 0<r<1 için 0<T(r)<1'dir.

1.'daki T(r)'nin tek değerli olma şartı, ters dönüşümün var olacağını garanti etmek için gereklidir ve monotonluk durumu çıktı görüntüsündeki siyahtan beyaza artan sırayı muhafaza eder. Monoton bir şekilde artmayan bir dönüşüm fonksiyonu, yoğunluk aralığının en az bir bölümünde ters olarak sonuçlanabilir. Bu yüzden, çıktı görüntüsünde bazı ters gri seviyeler oluşturur. Bu bazı durumlarda istenilen bir etki olabilirken, bu bölümde çıkarım yapılan sonuç değildir. Son olarak, 2. durumu gri çıktı seviyelerinin, girdi seviyeleriyle aynı aralıkta olacağını garanti eder. Şekil 2.2, bu iki duruma uygun bir dönüşüm fonksiyonunun örneğini verir. s'den r'ye ters dönüşüm ifade edilir.

$$r = T^{-1}(s) \qquad 0<s<1 \tag{2.2}$$

Uygun ters T~1(s)'in tek değerli olmada başarısız olması muhtemeldir.



Şekil 2.2. Tek değerli ve tekdüze artan bir gri-seviye dönüşüm foksiyonu

Bir görüntüdeki gri seviyeler, [0,1] aralığındaki rastgele değişkenler olarak görülebilir. Bir rastgele değişkenin en temel tanımlayıcısı, onun muhtemel yoğunluk fonksiyonudur (MYF). p üzerindeki altsimgeler, pT ve ps'nin farklı fonksiyonlar olduğunu ifade etmek için kullanıldığında, pr(r) ve ps(s), sırasıyla r ve s rastgele değişkenlerinin muhtemel yoğunluk fonksiyonlarını ifade ettiği varsayılırsa, temel ihtimal teorisinin basit bir sonucu, iipr{r}ve T(r)'nin (a) durumuna uygun olmasıdır. Daha sonra dönüştürülmüş değişken s'nin muhtemel yoğunluk fonksiyonu olan ps(s), oldukça basit bir formül kullanılarak elde edilebilir;

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| \tag{2.3}$$

Bu yüzden, dönüştürülmüş değişkeninin muhtemel yoğunluk fonksiyonu s, girdi görüntüsünün MYF gri seviyesi tarafından ve seçilen dönüşüm fonksiyonu tarafından belirlenir. Türevlemenin etkisiz değişkeni w olduğunda, görüntü işlemedeki özel önemin dönüşüm fonksiyonu;

$$s = T(r) = \int_0^r p_r(w)dw \tag{2.4}$$

modelini oluşturur.

Eş. 2.4'in sağ tarafı, rastgele değişken r'nin birikimli dağılım fonksiyonu (BDF) olarak tanımlanır. Muhtemel yoğunluk fonksiyonları daima pozitif olduğu ve bir fonksiyonun türevinin, fonksiyonun altındaki alanda olduğunu anımsattığı için; bu dönüşüm

fonksiyonunun tek değerli olduğu ve monoton bir şekilde arttığı anlaşılır ve bu yüzden (a) durumuna uygundur. Benzer şekilde, [0,1] aralığındaki değişkenler için muhtemel yoğunluk fonksiyonunun türevi de [0,1] aralığındadır, bu yüzden (b) durumu da sağlanır.

T(r) dönüşüm fonksiyonu verildiğinde, Eş. 2.4 eşitliğini uygulayarak ps(s)'yi buluruz. En üst seviyesi bakımından belirli bir integralin türevinin, basit şekilde o seviyede değerlendirilen integrantı (tümlevleneni) olduğu temel hesaplamadan (Leibniz'in kuralı) bilinmektedir. Diğer bir deyişle,

$$\begin{aligned} \frac{ds}{dr} &= \frac{dT(r)}{dr} \\ &= \frac{d}{dr}\left[\int_0^r p_r(w)dw\right] \\ &= p_r(r) \end{aligned} \tag{2.5}$$

Eş. 2.5 eşitliğinde bu sonucu dr/ds'nin yerine kullanmak ve bütün muhtemel değerlerin pozitif olduğu sonucu,

$$\begin{aligned} p_s(s) &= p_r(r)\left|\frac{dr}{ds}\right| \\ &= p_r(r)\left|\frac{1}{p_r(r)}\right| \\ &= 1 \qquad 0 \le s \le 1 \end{aligned} \tag{2.6}$$

'i verir.

ps(s) muhtemel yoğunluk fonksiyonu olduğu için, bu durumda [0,1] aralığının dışında 0 olmak zorunda olduğu anlaşılır çünkü s'in tüm değerleri üzerindeki integrali 1'e eşit olmak zorundadır. Eş. 2.6 eşitliğinde verilen ps(s) modelini, eşdağılımlı muhtemel yoğunluk fonksiyonu olarak kabul edilir. Basit şekilde belirtildiği gibi, Eş. 2.6 eşitliğinde verilen dönüşüm fonksiyonunu uygulamanın, eşdağılımlı muhtemel yoğunluk fonksiyonu ile nitelenen rastgele değişken s'yi verdiği gösterildi. Eş. 2.4 eşitliğinde T(r)'nin pr(r)'ye bağlı olduğuna fakat Eş. 2.7 eşitliğinde belirtildiği gibi sonuçta oluşan ps(s)'nin daima eş dağılımlı ve pr(r) modelinden bağımsız olduğuna dikkat edilmelidir.

Aralıklı değerler için, muhtemel yoğunluk fonksiyonları ve integrallerin yerine olasılıklar ve toplamlar ele alınır. Bir görüntüde rk gri seviyesinin meydana gelme ihtimali,

$$p_r(r_k) = \frac{n_k}{n} \qquad k = 0,1,2,....,L-1 \tag{2.7}$$

tarafından yaklaşık olarak tahmin edilir. Bu bölümün başında da belirtildiği gibi, görüntüdeki toplam piksel sayısı n'dir, rk gri seviyesine sahip piksellerin sayısı nk'dir ve görüntüdeki muhtemel gri seviyelerin toplam sayısı L'dir. Eş. 2.6 eşitliğinde verilen dönüşüm fonksiyonunun aralıklı versiyonu,

$$\begin{aligned} s_k &= T(r_k) = \sum_{j=0}^{k} p_r(r_j \\ &)\sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0,1,2,...,L-1 \\ &= 1 \qquad 0 \le s \le 1 \end{aligned} \tag{2.8}$$

'dir.

Böylece, işlenmiş çıktı görüntüsü, Eş. 2.8 eşitliği aracılığıyla girdi görüntüsündeki rk seviyeli her pikseli çıktı görüntüsünde sk seviyeli uygun piksel ile eşlemleyerek elde edilir. Daha önce belirtildiği gibi, pr(rk)'ye karşı rk çizimi bir histogram olarak adlandırılır. Eş. 2.8 eşitliğinde verilen dönüşüm (eşlemleme), histogram eşitleme ya da histogram doğrusallaştırma olarak adlandırılır. Eş. 2.9 eşitliğindeki dönüşümün, bu bölümde daha önce bahsedilen (a) ve (b) durumlarına uygun olduğunu göstermek zor değildir [33].

Aralıksız emsalinin aksine, bu aralıklı dönüşümün, eşdağılımlı bir histogram olan eşdağılımlı muhtemel yoğunluk fonksiyonunun aralıklı karşılığını vermeyeceği kanıtlanamaz. Fakat görüleceği gibi, Eş. 2.9 eşitliğinin kullanımı, histogram-eşitlenmiş görüntü seviyelerinin gri ölçeğin daha dolu aralığına dağılması için girdi görüntüsünün histogram dağılımının genel eğilimine sahiptir.

Bu bölümde daha önce, tüm gri ölçeği kapsayan gri seviye değerlerine sahip olmanın pek çok avantajı tartışıldı. Bu eğilime sahip gri seviye oluşturmaya ek olarak, az önce elde edilmiş metot, tamamen "otomatik" olan bir avantaja daha sahiptir. Diğer bir deyişle,

verilen bir görüntü, histogram eşitleme işlemi, daha fazla parametre ayrıntısına ihtiyaç duymadan verilen görüntüden direkt olarak elde edilen bilgiye dayanan Eş. 2.8 eşitliğini uygulamayı kapsar. Aynı zamanda tekniği uygulamak için gerekli olan hesaplamaların kolaylığına da dikkat edilir.

s'den r'ye ters dönüşüm,

$$r_k = T^{-1}(s_k) \qquad k = 0,1,2,...,L-1 \tag{2.9}$$

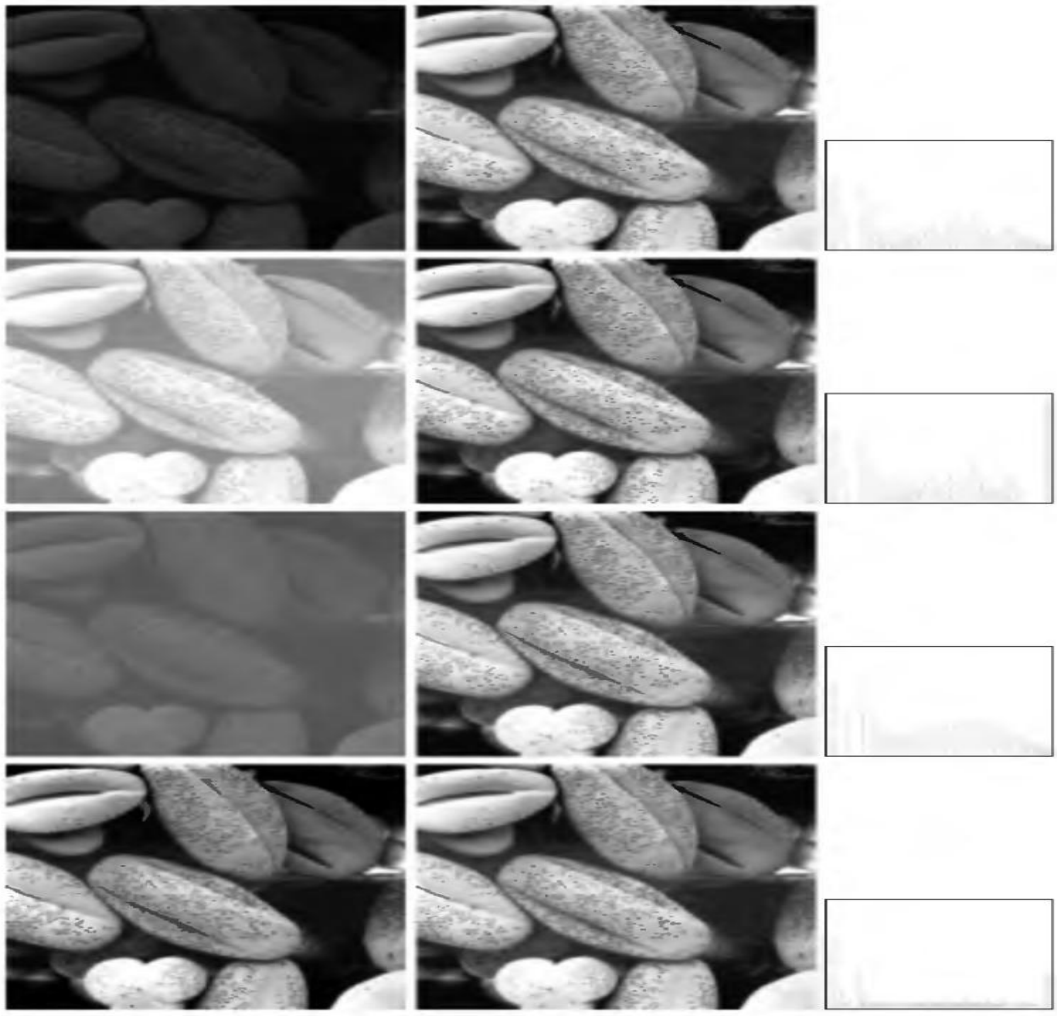
'den elde edilir.

Eş. 2.9 eşitliğindeki ters dönüşümün, sadece ve sadece girdi görüntüsündeki rk, k=0,1,2,…,L-1 seviyelerinden hiçbiri eksik olmazsa bu bölümde daha önce ifade edilen (a) ve (b) durumlarına uygun olduğu gösterilebilir. Histogram eşitlemede ters dönüşüm kullanılmamasına rağmen, gelecek bölümde geliştirilen histogram-eşleme şemasında merkezi bir rol oynamaktadır. Aynı zamanda sonraki bölümlerde, histogram işleme tekniklerinin nasıl uygulandığına dair ayrıntılar tartışılacaktır.

Şekil 2.3 (a), 4 görüntüyü göstermektedir ve Şekil 2.3 (b), bu görüntülerin her birinde histogram eşitleme uygulamasının sonuçlarını göstermektedir. İlk üç sonuç (yukarıdan aşağıya doğru) kayda değer gelişme göstermiştir. Beklendiği gibi, histogram eşitleme, dördüncü resimde kayda değer görsel bir farlılık oluşturmamıştır. Çünkü bu görüntünün histogramı, gri ölçeğin tüm görüntüsüne (izgesine) yayılmıştır. Şekil 2.3 (b)'deki görüntüyü oluşturmak için kullanılan dönüşüm fonksiyonları figür 2.3'te gösterilmiştir. Bu fonksiyonlar, Eş. 2.9 eşitliği kullanılarak orijinal görüntülerin histogramlarından elde edildi (Şekil 2.3 (b)). 4 nolu dönüşümün, dördüncü girdi görüntüsündeki gri seviyelerin neredeyse eşit oranda dağıtıldığı belirtilmiş ve temel düz bir şekle sahip olduğu gösterilmiştir. Belirtildiği gibi, bu durumdaki histogram eşitlemenin, imajın görüntüsü üzerinde göz ardı edilebilir bir etkiye sahip olması beklenmektedir.

Eşitlenmiş görüntülerin histogramları Şekil 2.3 (c)'de gösterilmektedir. Bütün bu histogramlar farklı iken, histogram-eşitlenmiş görüntülerin kendileri görsel olarak çok benzerdir. Bu beklenmeyen bir durum değildir çünkü sol kolondaki görüntüler arasındaki fark içerik değil, zıtlıktır. Başka bir deyişle, görüntüler aynı içeriğe sahip olduğu için,

histogram-eşitleme işleminden oluşan zıtlıktaki artış, ortaya çıkan görsel olarak farksız görüntülerdeki herhangi bir gri seviye farklılığını açıklamak için yeterlidir. Sol kolonda görüntülerin önemli zıtlık farklılıkları verildiğinden, bu örnek, uyarlanabilir bir geliştirme aracı olarak histogram eşitlemenin gücünü gösterir.



Şekil 2.3. a) Görüntüler b) Histogram eşitleme sonuçları c) İlgili histogramlar

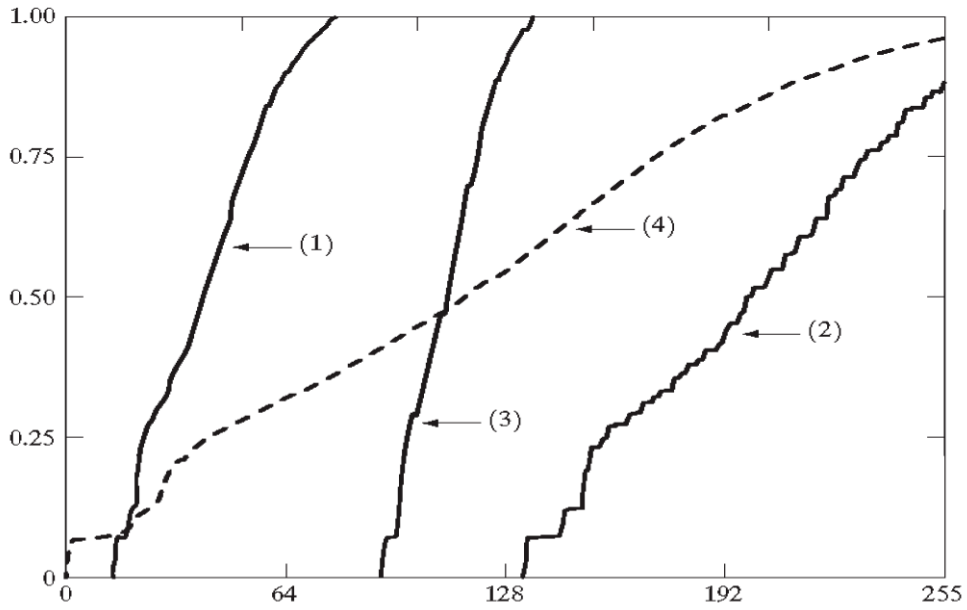
2.1.2. Histogram eşleme (tanımlama)

Önceki bölümde ifade edildiği gibi, histogram eşitleme, eşdağılımlı histograma sahip bir cıktı görüntüsü üretmeye çalışan bir dönüşüm fonksiyonunu otomatik olarak gösterir. Otomatik geliştirme arzu edildiğinde, bu iyi bir yaklaşımdır çünkü bu tekniğin sonuçları tahmin edilebilirdir ve metodu uygulaması kolaydır. Bu bölümde, eşdağılımlı bir histogram üzerine bir geliştirme temellendirmeye çalışmanın en iyi yöntem olmadığı uygulamaların varlığı gösterilmektedir. Özellikle, işlenmiş görüntüye sahip olmasını

istediğimiz histogramın şeklini belirleyebilmek bazen yararlı olmaktadır. Verilen histograma sahip işlenmiş bir görüntüyü oluşturmak için kullanılan metot, histogram eşleme ya da histogram tanımlama olarak adlandırılır.

Metodun gelişimi

Bir anlığına aralıksız gri seviyeler olan r ve z düşünüldüğünde (aralıksız rastgele değerler olarak değerlendirilir), ve pr(r) ve pz(z), kendi uygun aralıksız muhtemel yoğunluk fonksiyonlarını belirlesinler.



Şekil 2.4. Dönüşüm fonksiyonları

Burada r ve z sırasıyla girdi ve çıktı (işlenmiş) görüntülerinin gri seviyelerini ifade etmektedir. pz(z), çıktı görüntüsünün sahip olması istenilen belirlenmiş muhtemel yoğunluk fonksiyonuyken, verilen girdi görüntüsünden pr(r)'yi tahmin edilebilir.

İntegralleme işlemindeki etkisiz değişken w olduğunda, s;

$$s = T(r) = \int_0^r p_r(w)dw \tag{2.10}$$

özelliği ile rastgele bir değişken olsun. Biz bu ifadeyi, Eş. 2.9 eşitliğinde verilen histogram eşitlemenin aralıksız versiyonu olarak tanımlanır. İntegralleme işlemindeki etkisiz değişken f olduğunda;

$$G(z) = \int_0^z p_z(t)dt = s \tag{2.11}$$

özelliği ile rastgele bir z değişkeninin tanımlanacağı varsayılsın. Bu iki eşitlikten, G{z}=T(r) olduğu ve bu yüzdende z'nin;

$$z = G^{-1}(s) = G^{-1}\left[T(r)\right] \tag{2.12}$$

durumunu sağlamak zorunda olduğu anlaşılır.

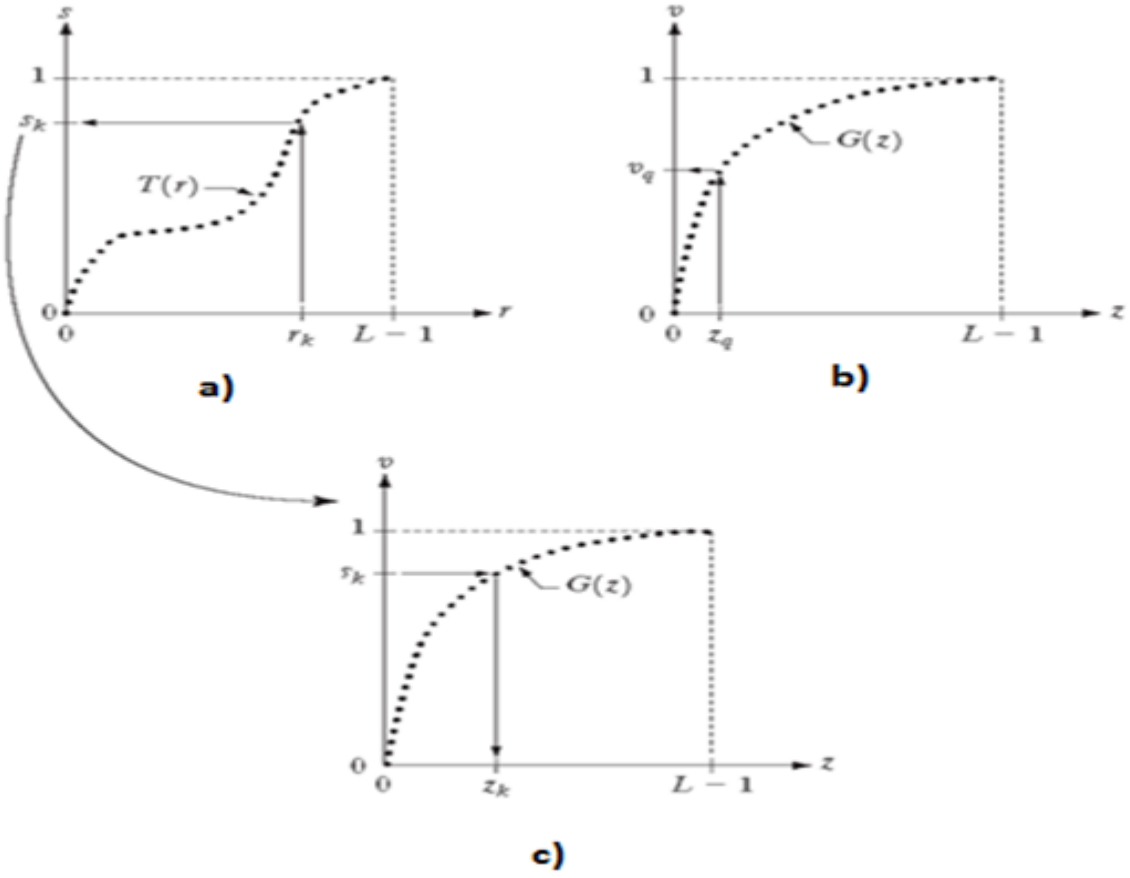
Uygulama

Uygulama aşağıdaki gibi başlar;

1. Her gri seviye takımı; {//}, {&)}, ve {z;}, j=0,1,2,…, L-1, Lx1 hacminin tek boyutlu bir dizilişidir.
2. r'den s'ye kadar olan ve s'den z'ye kadar olan bütün eşlemler, verilen bir piksel değeri ve bu dizilişler arasındaki basit çizelgelerdir.
3. Bu dizilişlerin her bir elemanı, örneğin, sk, iki adet önemli bilgi içermektedir: k altsimgesi, dizilişteki elemanların yerlerini belirler ve s, bu yerlerin değerlerini belirler.
4. Sadece tamsayı piksel değerleri ile uğraşmamız gerekmektedir. Örneğin, 8-bitlik bir görüntü durumunda, L=256'dır ve az önce bahsedilen her bir dizilişin elemanları, 0 ve 255 arasındaki tamsayılardır. Bu durum, daha önce histogram işleme tekniklerinin geliştirilmesini kolaylaştırmak için kullandığımız standartlaştırılmış [0,1] aralığının yerine [0,L-1] aralığındaki gri seviye değerleri ile çalışılacağı anlamına gelir.

Histogram eşlemenin gerçekte nasıl uygulanabildiğini görmek için, Şekil 2.5(a) ve Şekil 2.5(c) arasında gösterilen bağlantıyı bir anlığına göz ardı ederek, Şekil 2.5(a) dikkate alınır. Şekil 2.5(a), verilen görüntüden elde edilen s=T(r) varsayıma dayanan aralıklı dönüşüm fonksiyonunu gösterir. Görüntüdeki ilk gri seviye olan rl, ile eşlenir; ikinci gri

seviye olan r2, s2 ile eşlenir, k'inci seviye olan rk, sk ile eşlenir; ve bu şekilde devam eder (buradaki önemli nokta bu değerler arasındaki düzenli uyumdur). Dizilişteki her Sj değeri Eş. 2.12 eşitliğini kullanarak önceden hesaplanır; bu yüzden eşleme işlemi, s'nin uygun değerlerini belirlemek için dizilişteki dizin olarak pikselin gerçek değerini kullanır. Bu işlem özellikle kolaydır çünkü tam sayılar ele alınmaktadır. Örneğin, 127 değer ile 8-bitlik bir piksel için s eşlemi, muhtemel 256 pozisyondan dizilişte 128. pozisyonda bulunur (sj, 0'dan başladığımızı hatırlayın). Burada durup önceki bölümde tanımlanan metot ile girdi görüntüsünün her piksel değeri eşlemlenirse, çıktı Eş. 2.12 eşitliğine göre histogram-eşitlenmiş bir görüntü olacaktır.



Şekil 2.5. a) T(r) aracılığıyla rk'dan sk'ya eşlenmiş grafik yorumu b) G(z) aracılığıyla zq ve vq'nun eşlemesi c) sk'dan zk'ya ters eşleme

Histogram eşleme işlemini uygulamak için, bir adım öne gidilmek zorundadır. Şekil 2.5 (b), Eş. 2.12 eşitliği kullanılarak verilen pz(z) histogramından elde edilen varsayıma dayanan bir G dönüşüm fonksiyonudur. Herhangi bir zq için, bu dönüşüm fonksiyonu uygun bir vq değeri verir. Bu eşlem, Şekil 2.5 (b)'deki oklar tarafından gösterilmektedir.

Diğer taraftan, verilen her vq değeri için, G'den uygun olan zq değeri bulunur. Figüre göre, grafik olarak bunların hepsi, okların yönünün vq eşlemine uyum sağlayan zq'ya çevrileceği anlamına gelir. Fakat Eş. 2.12 eşitliğindeki tanımdan, uyum sağlayan alt simgeler için v=s olduğu anlaşılır, bu yüzden, sk=T(rk) eşleminden daha önce hesaplanan her sk değeriyle uyum sağlayan zk'yi bulmak için tam olarak bu işlem kullanılabilir. Bu düşünce, Şekil 2.5 (c)'de gösterilmektedir.

Gerçekte z'lere sahip olunmadığı için (bu değerleri bulma işleminin, kesinlikle histogram eşlemenin amacı olduğu hatırlatılmaktadır.), s'den z'yi bulmak için bir tür tekrarlayan tabloya başvurulmak zorundadır. Tam sayıların ele alması, bunu kolay bir işlem yapmaktadır. Öncelikli olarak, vk=sk olduğu için, Eş. 2.11 eşitliğinden aranılan z'lerin G(zjO=sk> ya da (G{zk)~st)=O eşitliğini sağlamak zorunda olduğu bilinmektedir. Böylece, sk ile uyum sağlayan zk değerini bulmak için yapılması gereken şey, bu eşitliğin k=0,1,2,…, L-1'i sağlaması için z değerlerini yinelemektir. Bu durum, z üzerinde yineleme yapılacağı için G'nin tersini bulmak zorunda olunmaması dışında Eş. 2.13 eşitliğindeki durum ile aynıdır. Tam sayılar ele alındığında, (G(zk)-sk)=0 eşitliğini sağlamak için en yakın alınabilecek değer, [0,L-1] aralığındaki en küçük tamsayı z olduğunda her bir k değeri için zk=z olsun, şöyle ki;

$$\left(G(\hat{z}) - s_k\right) \geq 0 \qquad k = 0,1,2,...,L-1 \tag{2.13}$$

2.2. Düzeltilmiş Uzaysal Filtreler

Düzeltme filtreleri bulanıklık ve pürüz giderme için kullanılır. Bulanıklaştırma, nesne çıkarmadan önce görüntüden küçük ayrıntıların çıkarılması ile doğrular ve kıvrımlar halindeki boşlukları bağlama gibi ön işleme basamaklarında kullanılır. Pürüz giderme, doğrusal bir filtreyle bulanıklaştırılarak yapılabileceği gibi aynı zamanda doğrusal olmayan filtrelemeyle de yapılabilir.

2.2.1. Düzeltilmiş doğrusal filtreler

Düzelten doğrusal uzaysal bir filtrenin çıktısı, filtre maskesinin yakınında bulunan piksellerin ortalamasıdır. Bu filtreler bazen ortalama filtreleri olarak adlandırılır. Bölüm

4’te açıklanan sebeplerden dolayı, aynı zamanda alçak iletimli filtreler olarak da atıfta bulunulur.

Düzeltme filtrelerinin arkasındaki fikir basittir. Görüntüdeki her bir piksel değerini, filtre maskesi tarafından belirlenen civardaki gri seviyelerin ortalaması ile yer değiştirilirse, bu işlem gri seviyelerde azaltılmış keskin geçişli bir görüntüyle sonuçlanır. Rastgele pürüzler tipik olarak gri seviyelerdeki keskin geçişlerden oluştuğu için, düzeltme işleminin en açık uygulaması pürüz gidermedir. Fakat (bir görüntünün neredeyse her zaman istenen özellikleri olan) ayrıtlar da, gri seviyelerde keskin geçişler tarafından nitelenir; bu yüzden ortalama filtreler, ayrıtları bulanıklaştıran istenmeyen bir etkiye sahiptir. Bu işlem tipinin diğer bir uygulaması, yetersiz sayıda gri seviye kullanmaktan oluşan yanlış kenarların düzeltilmesini içerir. Ortalama filtrelerin başlıca kullanımı, bir görüntüdeki “alakasız” ayrıntının atılmasıdır. “Alakasız” kelimesiyle, filtre maskesinin büyüklüğüne göre küçük olan piksel alanları kastedilmektedir. Son uygulama, bu bölümde gösterilecektir.

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\qquad \frac{1}{16}\times\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$$

Şekil 2.6. İki adet 3x3 düzeltilmiş(ortalama) filtre maskesi (Her maskenin önündeki sabit çarpanlar katsayı değerlerinin toplamına eşittir)

Şekil 2.6, iki adet 3x3 düzeltme filtresini göstermektedir. Birinci filtrenin kullanımı, maske altındaki piksellerin standart ortalamasını verir. Bu durum en iyi Eş. 2.14 eşitliğindeki maskenin katsayılarını yer değiştirerek görülebilir: maske tarafından tanımlanan 3x3’lük civardaki piksellerin gri seviyelerinin ortalaması;

$$R=\frac{1}{9}\sum_{i=1}^{9}z_i \tag{2.14}$$

‘dir.

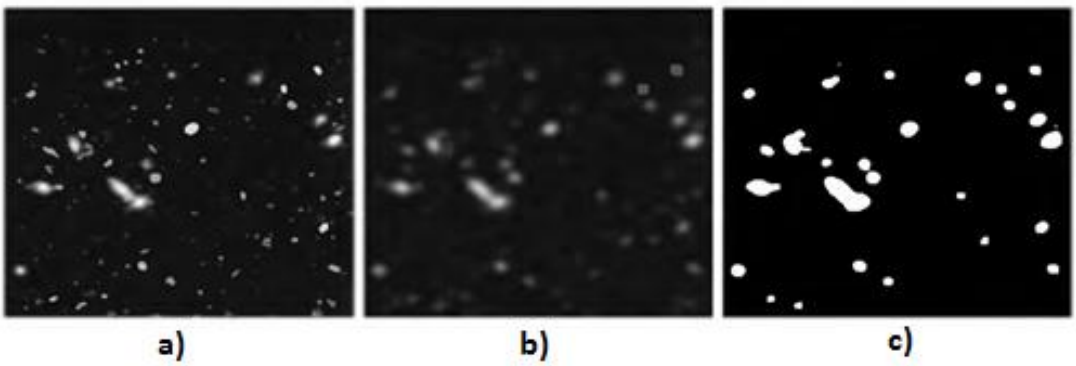
1/9 olması yerine, filtrenin katsayılarının hepsinin 1 olduğu görülmektedir. Buradaki düşünce, 1 değerli katsayılara sahip olmanın sayısal olarak daha etkili olmasıdır. Filtreleme işleminin sonunda bütün görüntü 9’a bölünür. m x n maskesi, 1 fmn’ye eşit olan standartlaştırılmış bir sabite sahip olacaktır. Bütün katsayıları eşit olan uzaysal ortalama filtresi bazen filtre kutusu olarak adlandırılır.

Şekil 2.6’da gösterilen ikinci maske, biraz daha ilginçtir. Bu maske, piksellerin farklı katsayılar ile çarpıldığını ve böylece bazı piksellere diğerleri pahasına daha fazla önem verildiğini belirtmekte kullanılan terim olan ağırlıklı ortalama olarak adlandırılır. Şekil 2.6’de gösterilen maskede, ortalamanın hesaplanmasında maskenin merkezindeki piksele daha fazla önem verilerek diğerlerinden daha yüksek bir değerle çarpılır. Diğer pikseller, maskenin merkezine olan uzaklıklarının fonksiyonu olarak ters ağırlıklıdırlar. Köşegen terimleri, (V2 faktörü ile) dikey sınırlarından çok daha uzaktır ve bu yüzden, merkez pikselin yakın sınırından daha az ağırlıklıdır. Merkez noktasını en yükseğe ağırlıklandırarak orijine olan mesafesi artan bir fonksiyon olarak katsayıların değerlerini azaltmanın arkasındaki temel strateji, düzeltme işlemindeki bulanıklığı azaltmaktır. Aynı genel amacı sağlaması için başka bir ağırlık seçilebilirdi. Fakat Şekil 2.6’nin maskesindeki bütün katsayıların toplamı 16’ya eşittir ve 2’nin tamsayı kuvvetine sahip olduğu için bilgisayar uygulamasında önemli bir özelliktir. Uygulamada, Resim 2.1’deki ya da benzer düzenlemelerdeki maskelerden birini kullanarak düzeltilen görüntüler arasındaki farkı görmek genelde zordur, çünkü bir görüntüde herhangi bir bölgede yayılan bu maskelerin alanı çok küçüktür. Filtre büyüklüğünün bir fonksiyonu olarak düzeltme işleminin etkileri, orijinal bir görüntüyü ve sırasıyla 35 pikselin ve n=3, 5, 9, 15 boyutlarının ortalama kare filtrelerini kullanarak elde edilen uygun düzeltilmiş sonuçlarını gösteren Resim 2.1’da gösterilmiştir. Bu sonuçların temel özelliği aşağıdaki gibidir: n=3 için, görüntünün tamamında hafif bir bulanıklık görürüz fakat beklendiği gibi, filtre maskesiyle yaklaşık olarak aynı boyutta olan ayrıntılar kayda değer şekilde daha fazla etkilenmiştir. Örneğin; 3x3 ve 5x5 kareler, küçük “a” harfi ve ince tanecikli pürüz, görüntünün geri kalanıyla karşılaştırıldığında belirgin bir bulanıklık gösterir. Pürüzün daha az vurgulanması olumlu bir sonuçtur. Gri dairelerin ve özelliklerin (harflerin) sivri (tırtıklı) sınırlarının gayet iyi bir şekilde düzeltildiği görülmektedir.

n=5 için sonuç benzerdir, bulanıklıkta küçük bir artış vardır. n=9 için, kayda değer şekilde daha fazla bulanıklık görülür ve siyah dairenin %20'si, gri içerik seviyesi sınırdaki piksellere yakın nesnelerdeki bulanıklığın harmanlama etkisini gösteren daha önceki üç görüntüdeki arka plan kadar farklı değildir. Pürüzlü dikdörtgenlerin belirgin düzeltilişine dikkat edin. n=15 ve 35 için sonuçlar, görüntüdeki nesnelerin boyutlarına bağlı olarak uç sınırdadır. Aşırı bulanıklığın bu tipi, genellikle görüntüden küçük nesneleri yok etmek için kullanılır. Örneğin, üç küçük kare, dairelerden iki tanesi ve pürüzlü dikdörtgen alanlarının çoğu, Resim 2.1 (f)'deki görüntünün arka planı ile harmanlanmıştır. Aynı zamanda bu figürde siyah sınırın vurgulandığı görülmektedir. Bu durum, 0'lı (siyah) orijinal görüntünün sınırını tamponlayıp sonrada tamponlanan alanın kırpılmasının sonucudur. Siyahların bazıları, bütün filtrelenmiş görüntülerle harmanlanmıştır, fakat daha büyük filtrelerle düzeltilen görüntüler için tamamen uygusuz olmuştur.

Önceki bölümlerde tartışıldı üzere, uzaysal ortalamanın önemli bir uygulaması, ilgili nesnenin kesintisiz bir temsilini elde etmek amacıyla bir görüntüyü bulanıklaştırmaktır, yani küçük nesnelerin yoğunluğu arka plan ile harmanlanır bulunması kolay olan büyük nesneler olurlar. Maskenin boyutu, arka plan ile harmanlanacak nesnenin boyutunu belirler. Örnek olarak, dünya çevresindeki bir yörüngede Hubble teleskopundan elde edilen bir görüntü Resim 2.2 (a)'de gösterilmiştir. Resim 2.2 (b) bu görüntüye, 15x15'lik ortalama maskenin uygulanmasının sonuçlarını göstermektedir. Pek çok nesnenin ya arka plan ile harmanlandığını ya da yoğunluklarının kayda değer şekilde azaldığı görülmektedir. Yoğunluklarını temel alarak nesneleri yok etmek için bunun gibi bir eşikleme işlemi takip etmek normaldir. Bulanık görüntüdeki en yüksek yoğunluk değeri olan %25'e eşit eşik değeri Resim 2.2 (c)'de gösterilmiştir. Bu sonuçları orijinal görüntüyle karşılaştırılırsa, o görüntüdeki en büyük ve en parlak olarak düşünülen şeyin makul bir temsili olduğu görülebilir.

Resim 2.1 a) Asıl görüntü, 500x500 b) – f) kare ortalama filtre maskesi ile elde edilmiş düzeltilmiş filtre sonuçları



Resim 2.2 a) Hubble Uzay Teleskobundan alınmış görüntü b) 15x15 ortalama maske ile işlenmiş görüntü c) b'nin eşikleme sonucu elde edilmiş görüntü

2.2.2. Sıralı istatistik filtresi

Sıralı istatistik filtreleri, filtre tarafından çevrelenen alanda bulanan görüntüdeki pikselleri sıralamaya ve daha sonra merkez piksel değerini sıralama sonuçlarından elde edilen değerle değiştirmeye bağlı bir karşılığı olan doğrusal olmayan uzaysal filtrelerdir. Bu kategorideki en iyi bilinen örnek, isminin de ifade ettiği gibi bir piksel değerini o pikselin civarındaki gri seviyelerin ortancası ile yer değiştiren ortanca filtresidir (pikselin orijinal değeri, ortancanın hesaplanmasında kullanılmaktadır). Ortanca filtreleri oldukça popülerdir çünkü belli başlı bazı rastgele pürüzler için, benzer boyuttaki doğrusal düzeltme filtrelerinden kayda değer şekilde daha az bulanıklıkla mükemmel pürüz azaltma imkânı sağlamaktadır. Ortanca filtreleri, özellikle darbe pürüzü durumunda etkilidir, aynı zamanda bir görüntü üzerinde birleştirilmiş siyah beyaz görünümünden dolayı tuz ve biber olarak adlandırılır.

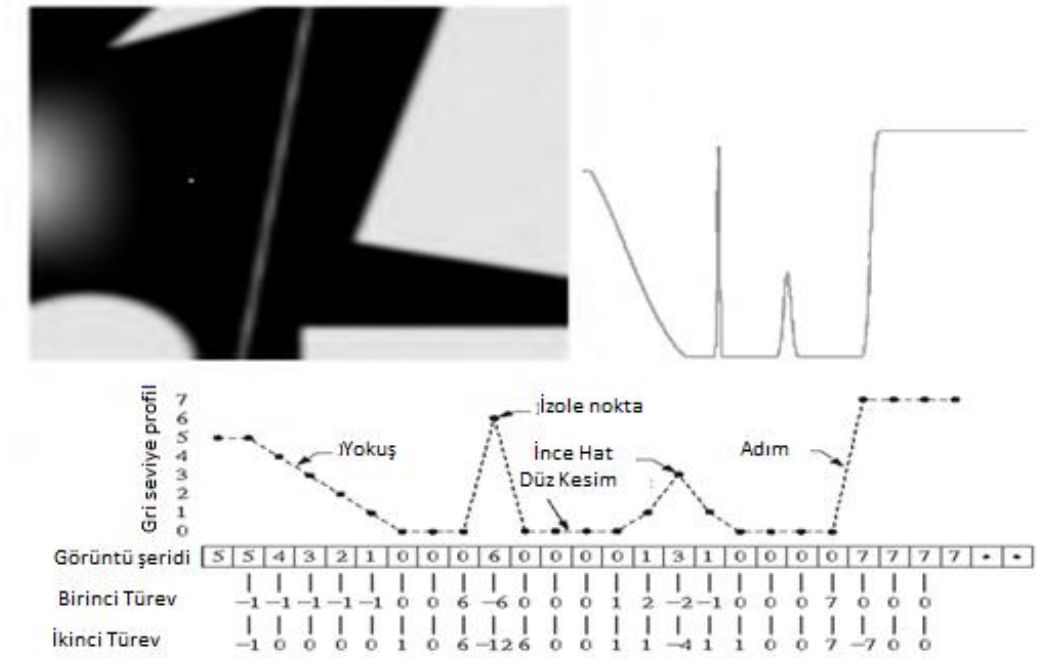
Bir grup değerin ortancası olan f için, gruptaki değerlerin yarısı f'den küçük ya da f'ye eşittir ve diğer yarısı f'den büyük ya da f'ye eşittir. Görüntüde bir noktaya ortanca filtresi uygulamak için, ilk olarak söz konusu piksel değerlerini ve sınırları sıralanır ve ortancalarına karar verilir ve bu değeri o piksele tahsis edilir. Örneğin, 3x3'lük bir alanda ortanca beşinci en büyük değerdir, 5x5'lik bir alanda ortanca on üçüncü en büyük değerdir ve bu şekilde devam eder. Bir alandaki birkaç değer aynı olduğunda, bütün eşit değerler gruplanır. Örneğin, 3x3'lük bir alanın A(0,20,20,20,15,20,20,25,100) değerlerine sahip olduğu varsayılsın. Bu değerler, 20'nin ortanca olduğu A(0,15,20,20,20,20,20,25,100) olarak sıralanır. Bu yüzden, ortanca filtrelerinin temel fonksiyonu, farklı gri seviyelerdeki noktalara yakınındakiler gibi olması için baskı yapmasıdır. Aslında, yakınındakilere göre açık ya da koyu olan ayrılmış piksel kümeleri ve alanı n2f2'den az olanlar (bir buçuk filtre alanı), bir nxn medyan filtresi tarafından yok edilir. Bu durumda yok etmek, alanın ortanca yoğunluğuna baskı yapmak anlamına gelir. Daha büyük kümeler kayda değer şekilde daha az etkilenir.

Ortanca filtresi, görüntü işlemedeki en yararlı sıralı istatistik filtresi olmasına rağmen, kesinlikle tek değildir. Ortanca, sıralanmış sayı grubunun 50. yüzdelik dilimini temsil eder fakat okuyucu, temel istatistiklerden sıralamanın pek çok ihtimale uygun olduğunu hatırlayacaktır. Örneğin, 100. yüzdelik dilimi kullanmak, görüntüdeki en parlak noktayı bulmada kullanışlı olan maksimum filtre ile sonuçlanır. 3x3'lük maksimum filtrenin

karşılığı, R=max{zft\k=1,2,…,9} tarafından verilir. 0. yüzdelik dilim filtresi, bu durumun tam zıttı amaç için kullanılan minimum filtredir.

2.3. Keskinleştirilmiş Uzaysal Filtreler

Keskinleştirme işleminin temel amacı, ya yanlışlıkla ya da özel bir görüntü elde etme metodunun doğal bir etkisi olarak bulanıklaşmış ayrıntıyı geliştirmek ya da görüntüdeki ince ayrıntıyı vurgulamaktır. Görüntü keskinleştirme işlemleri değişiklik göstermektedir ve elektronik basım ve tıbbi görüntülemeden endüstriyel denetleme ve askeri sistemlerdeki özerk kılavuzluğa kadar sıralanan uygulamaları kapsar. Uzaysal bölgedeki görüntü bulandırmanın, alandaki piksel ortalaması tarafından yapılabileceği görülmüştür. Ortalama alma, integralleme işlemine paralel olduğu için, keskinleştirmenin uzaysal türevleme tarafından yapılabileceği sonucuna varmak mantıklıdır. Bu bölümdeki tartışma sayısal türevleme ile keskinleştirme işlemlerinin tanımlanması ve uygulanmasının çeşitli yollarını ele alır. Temel olarak, türev işleminin yanıt kesinliği, işlemin uygulandığı noktadaki imajın kesiklik derecesine orantılıdır. Bu yüzden, görüntü türevleme; ayrıtları, (pürüz gibi) diğer kesiklikleri ve gri seviye değerleri yavaşça değişen deempha-boyutlu alanları geliştirir. Bu iki tanımın, daha önce ifade edilen birinci ve ikinci sıranın türevlerine ilişkin durumlarını sağladığı kolayca doğrulanabilir. Bunu görmek için ve görüntü işleme durumlarındaki birinci ve ikinci sıra türevleri arasındaki temel benzerlikler ve farklılıkları vurgulamak için, Şekil 2.7'da gösterilen örnek incelenebilir. Şekil 2.7 a), çeşitli katı nesneler, bir doğru ve tek bir pürüz noktası içeren basit bir görüntüyü göstermektedir. Şekil 2.7 b), pürüz noktasını kapsayarak merkez boyunca görüntünün yatay gri seviye profilini (tarama çizgisi) göstermektedir. Bu profil, bu figüre ilişkin örnek vermek için kullanılacak tek boyutlu bir fonksiyondur. Şekil 2.7 c), bir pürüz noktası, bir doğru ve daha sonra bir nesnenin ayrıtıyla karşılaştığında birinci ve ikinci sıra türevlerinin nasıl hareket ettiğini analiz etmeyi mümkün kılan yeterli sayılar ile profilin sadeleştirilmiş durumunu göstermektedir. Basitleştirilmiş grafiğimizde eğimdeki geçiş dört piksele yayılır, pürüz noktası tek bir pikseldir, doğru üç piksel kalınlığındadır ve gri seviye basamaklarındaki geçiş bitişik pikseller arasında meydana gelir. Gri seviyelerin sayısı sekiz seviyeye kadar sadeleştirilmiştir.



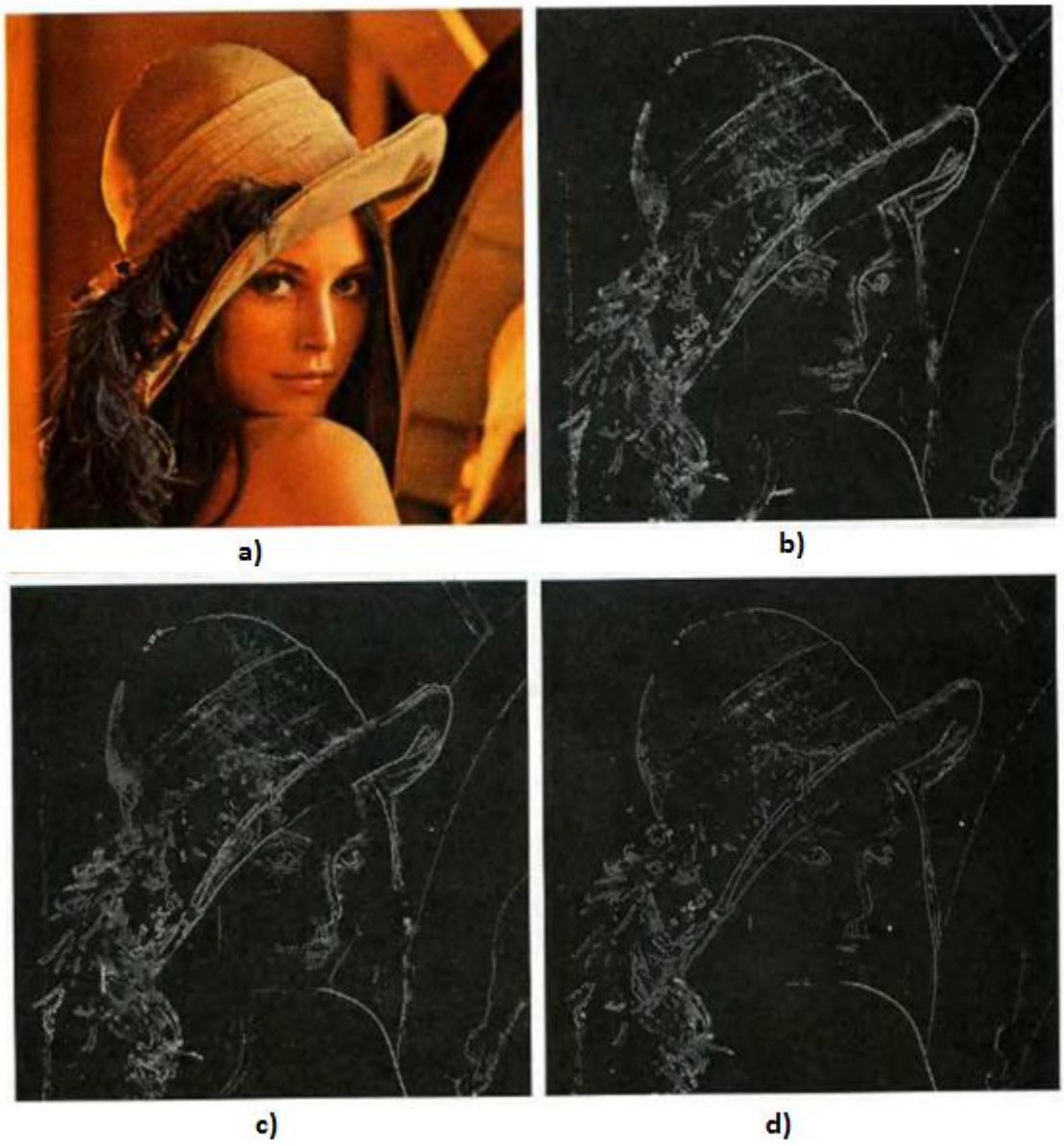
Şekil 2.7. a) Basit bir görüntü b) İzole gürültü noktasını da içeren, 1-D yatay gri-seviye profil c) Basitleşmiş profil yorumu

2.4. Renkli Ayrıt Saptama

Ayrıt saptama, görüntüyü parçalara ayırmak için önemli bir araçtır. Bu bölümde, renkli doğrusal uzayda ayrıtların direkt olarak hesaplanmasına karşı tek bir görüntü tabanındaki ayrıtları hesaplama konusuyla ilgilenilecek. Eğim işlemlerinde ayrıt saptama uygulanmıştır. Maalesef, tartışılan eğim vektörel büyüklükler için tanımlanmamıştır. Bu yüzden, bireysel şekillerde eğim hesaplamanın ve sonra sonuçları renkli bir görüntü oluşturmak için kullanmak hatalarla sonuçlanacaktır.

Resim 2.3 b), tartışılan vektör metodu kullanılarak elde edilen Resim 2.3 a)'daki görüntünün eğimidir. Resim 2.3 c), her bir (x,y) koordinatındaki üç görüntü bileşeninin uygun değerlerini ekleyip bir eğim görüntü bileşeni oluşturarak ve her bir RGB bileşen görüntüsünün eğimini hesaplayarak elde edilen görüntüyü göstermektedir. Görüntü eğim vektörünün ayrıt ayrıntıları, Resim 2.3 c)'deki tek düzlem görüntü eğrisindeki ayrıntıdan daha eksiksizdir. Resim 2.3 d)'deki görüntü, her (x,y) noktasındaki iki eğim görüntüsü arasındaki farkı göstermektedir. Her iki yaklaşımında makul sonuçlar verdiğine dikkat etmek önemlidir. Resim 2.4 b)'deki ekstra detayın fazladan hesaplama yüküne değip

değmediği, (tek düzlemlerin eğimini standartlaştırmak için kullanılan Sobel işlemlerinin uygulanmasına karşılık olarak) sadece verilen problemin gereksinimleri tarafından belirlenebilir. Resim 2.4, eklenip ölçeklendiğinde Resim 2.4 c)'yi elde etmek için kullanılan üç görüntü eğim bileşenini göstermektedir.



Resim 2.3. a) RGB görüntü b) RGB renk vektör uzayında hesaplanmış gradyan c) Görüntü bazlı hesaplanmış gradyanlar d) b ve c arasındaki fark işlemi görüntüsü

Resim 2.4. a) Resim 2.3'deki görüntünün bileşen gradyan görüntüsü b) Kırmızı bileşen c) Yeşil bileşen d) Mavi bileşen

2.5. Görüntü Ortalama Filtresi

17(x,y) pürüzünün, /(x,y) orijinal görüntüsüne eklenmesiyle elde edilen g(x,y) pürüzlü bir görüntü düşünülürse; yani, her bir (x,y) koordinat çiftinde pürüzü ilintisiz (korelasyonsuz) ve ortalama 0 değerine sahip olduğunda;

$$g(x,y) = f(x,y) + \eta(x,y) \tag{2.15}$$

'dir.

Sıradaki işlemin amacı, bir grup {gi(x,y)} pürüzlü görüntü ekleyerek pürüz içeriğini azaltmaktır. Daha sonra;

$$E\{\bar{g}(x,y)\} = f(x,y) \tag{2.16}$$

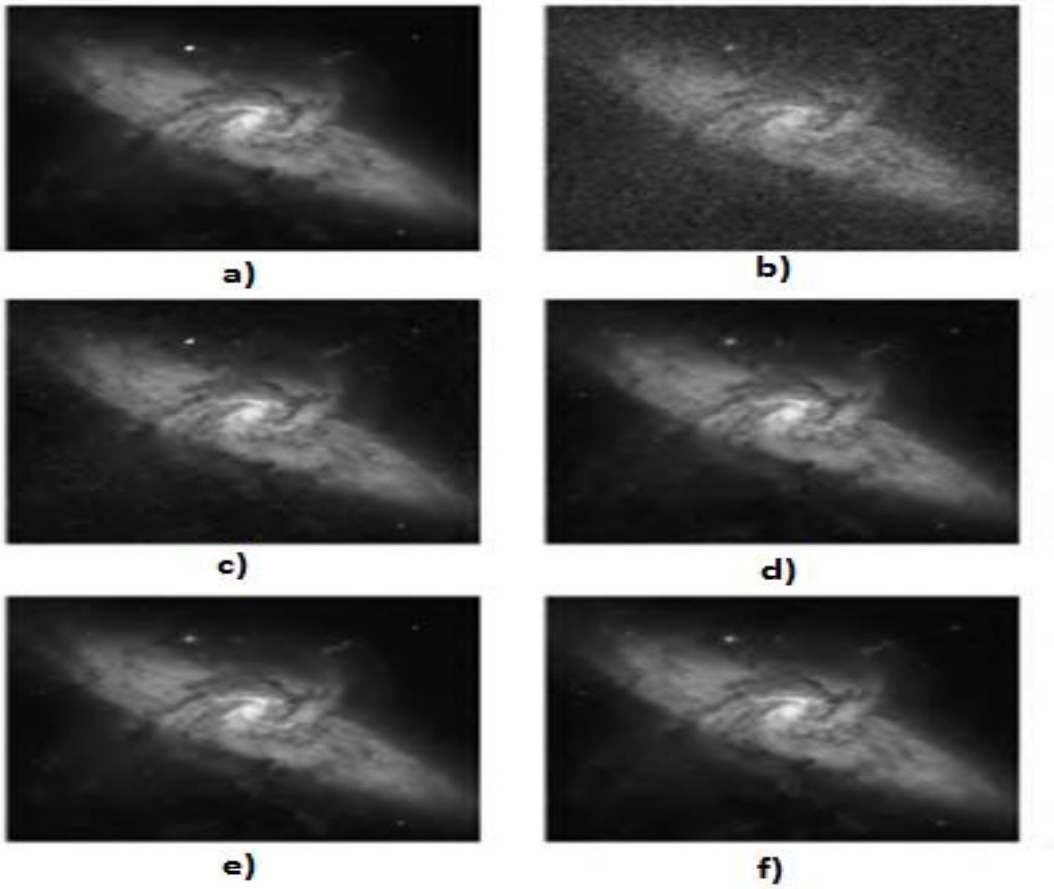
ve

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)} \tag{2.17}$$

olduğu anlaşılır.

Ortalama görüntünün önemli bir uygulaması, analizde neredeyse faydasız olan tek görüntüleri açıklamak için sık sık pürüz algılayıcısına neden olarak çok düşük ışık seviyelerinin rutin olduğu astronomi alanındadır. Resim 2.5 a), geniş görüş alanına sahip gezegen kamerası ile NASA'nın Hubble Uzay Teleskopu tarafından çekilen NGC 3314 isimli galaksi çiftinin bir görüntüsünü göstermektedir. NGC 3314, Hydra takımyıldızının güney yarımküresi yönünde dünyadan yaklaşık 140 milyon ışık yılı uzaklıkta bulunmaktadır. Öndeki galaksinin merkezinin yakınlarında bir rüzgârgülü şekli oluşturan parlak yıldızlar, yıldızlar arası gaz ve tozdan oluşmuştur. Resim 2.5 b) aynı görüntüyü göstermektedir ama 64 gri seviyelik standart sapma ve 0 ortalama ile korelasyonsuz Gauss pürüzü ile bozulmuştur. Bu görüntü, bütün pratik kullanımlar için faydasızdır. Resim 2.5 c), (f) boyunca, sırasıyla 8, 16, 64 ve 128 görüntülerinin ortalama sonuçlarını göstermektedir. K=128 ile elde edilen sonuçların görsel görünümdeki orijinale makul şekilde yakın olduğu görülmektedir.

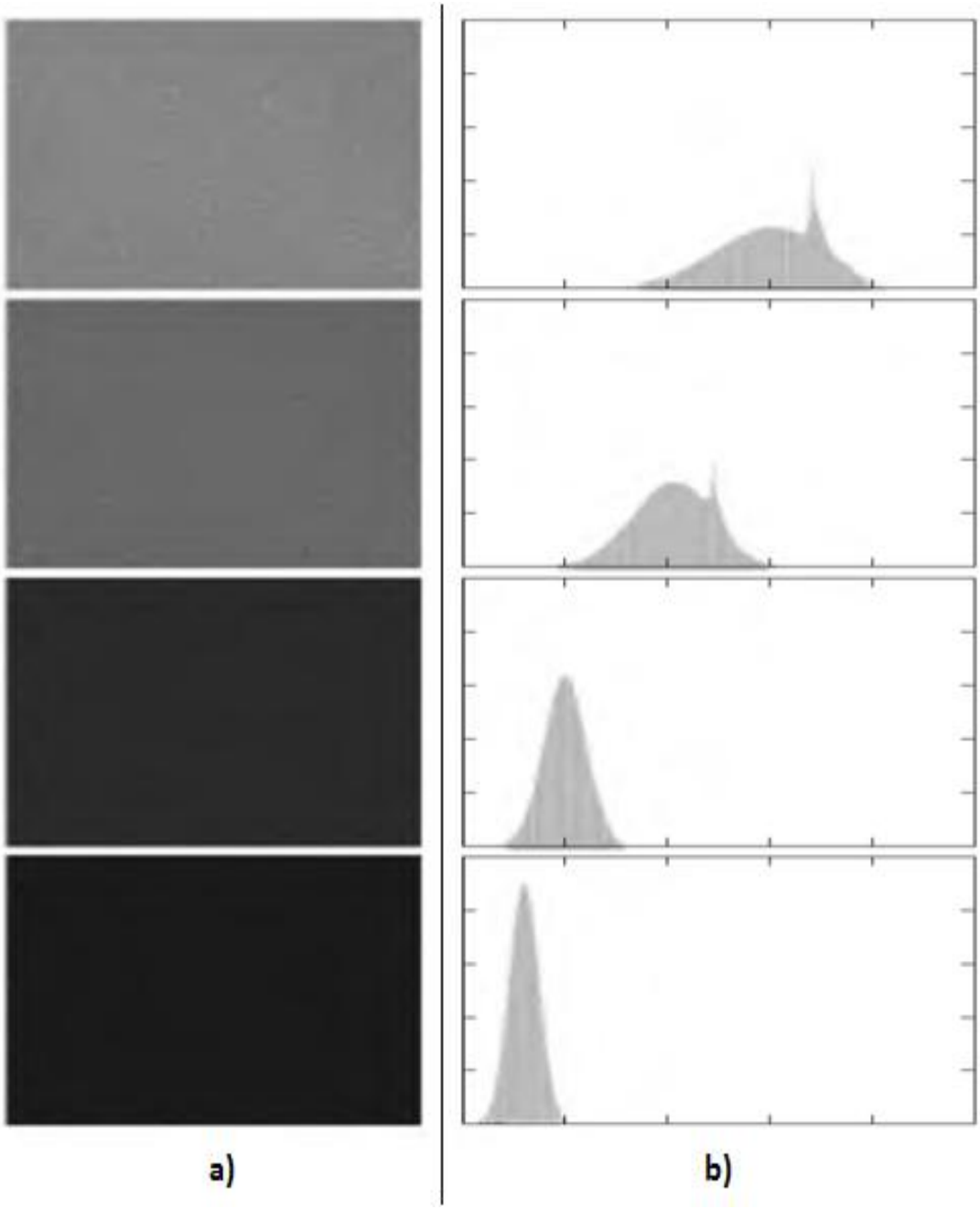
Görsel görünümdeki pürüz azalmasının, artan K'nin bir fonksiyonu olarak nasıl meydana gelmesi konusunda Resim 2.5'den daha iyi bir değerlendirme elde edilebilir. Bu figür, (f) boyunca orijinal [Resim 2.5 a)] ve Resim 2.5 c)'deki her bir ortalama görüntü arasındaki görüntülerin farkını göstermektedir. Görüntü farklılığına uygun histogramlar da bu resimde gösterilmektedir. Her zamanki gibi, histogramlardaki dikey ölçek piksel sayısını temsil eder ve [0,2.6 X 104] aralığındadır. Yatay ölçek gri seviyeyi temsil eder ve [0,255] aralığındadır. Histogramlarda, K arttıkça görüntü farklılığının ve standart sapmanın azaldığı görülmektedir. K arttıkça daha koyu olan Şekil 2.8'ün sol kolonundaki görüntü farklılığının artan ortalamasının etkisi de görülebilir.



Resim 2.5. a) Galaxy Pair NGC 3314 görüntüsü b) Gaussian gürültüsüyle bozulmuş görüntü c) – f) K=8,16,64 ve 128 ile ortalanmış sonuçlar

Ekleme işlemi, aralıksız integralin aralıklı formülüdür. Gökbilimsel gözlemlerde, az önce tanımlanan metoda eşit bir işlem, uzun bir süre boyunca aynı görüntüyü gözlemleyerek pürüz azaltma için benzer bir sensor ya da CCD'nin integralleme imkânını kullanmaktır. Fakat ağ etkisi, az önce tartışılan yönteme paraleldir. Sensörü soğutmak (ya da sadeleştirmek) pürüz seviyesini azaltır [5].

Görüntü çıkarma işleminde, iş 8bitlik bir görüntü üzerinde sonuçları göstermeye gelince iki ya da daha fazla 8 bitlik görüntü eklemek özel ilgi gerektirir. K toplamlarındaki değerlerde, 8 bitlik görüntüler 0'dan 255xK'ye kadar sıralanabilir. Bu durumda tekrar 8 bite ölçeklemek, basit şekilde sonuçları K'ye bölmeyi içerir. Doğal olarak, doğruluğun bir kısmı bu işlemde kaybolur fakat eğer görüntü 8 bit ile sınırlı olmak zorundaysa bu durum kaçınılmazdır.



Şekil 2.8. a) Yukarıdan aşağıya: Resim 2.5 deki görüntülerdeki fark değerleri b) Histogram değerleri

3. RENKLİ GÖRÜNTÜ İŞLEME

3.1. Renk Temelleri

İnsan beyni tarafından izlenen renk algılama ve yorumlama işlemi hala tamamen anlaşılamayan psikososyolojik bir olgu olmasına rağmen rengin fiziksel doğası deneysel ve teorik sonuçlar ile desteklenen formal (biçimsel) bir temelde ifade edilebilir.

1666'da Sör Isaac Newton, bir demet güneş ışığı cam prizmadan geçtiğinde ortaya çıkan güneş ışığının beyaz olmadığını mordan kırmızıya sıralanan renklerin kesintisiz izgesinden oluştuğunu keşfetti. Şekil 3.1'in gösterdiği gibi renk izgesi altı geniş bölgeye ayrılabilir: mor, mavi, yeşil, sarı, turuncu ve kırmızı. Dolgu renginde bakıldığında (Şekil 3.2), izgedeki hiçbir renk ansızın bitmez fakat her bir renk yumuşak biçimde bir sonrakine döner.

Temel olarak, insanların ve bazı hayvanların nesnelerdeki algıladıkları renkler, nesneden yansıyan ışığın doğası ile belirlenir. Şekil 3.2'de gösterildiği gibi, görünebilir ışık elektromanyetik izgede nispeten dar bir frekans şeridinden oluşur. Görünen tüm dalga boylarında dengeli olan ışığı yansıtan bir nesne gözlemciye beyaz olarak görünür. Fakat sınırlı görülebilir bir izge aralığında yansıtmayı tercih eden bir nesne bazı renk gölgeleri ortaya koyar. Örneğin, yeşil nesneler diğer dalga boylarındaki enerjiyi emerken 500 ile 570 nm aralığındaki dalga boyları ile ışığı yansıtır. Işık tanımlaması renk biliminin merkezidir. Eğer ışık akromatik(renksiz) ise, onun tek katkısı yoğunluğu ya da miktarıdır. Renksiz ışık gözlemcilerin siyah beyaz televizyon setinde gördükleri şeydir ve şimdiye kadar görüntü işleme tartışmamızın dolaylı bir bileşenidir. Şu ana kadar sayısız kere kullanılan gri seviye terimi, siyahtan griye ve son olarak da beyaza doğru sıralanan yoğunluğun sayısal ölçüsünü ima etmektedir.

Şekil 3.1. Prizmadan geçen pasif beyaz ışık sayesinde oluşan renk spektrumu.

Şekil 3.2. Elektromanyetik spektrumun görünür aralığında oluşan dalgaboyları

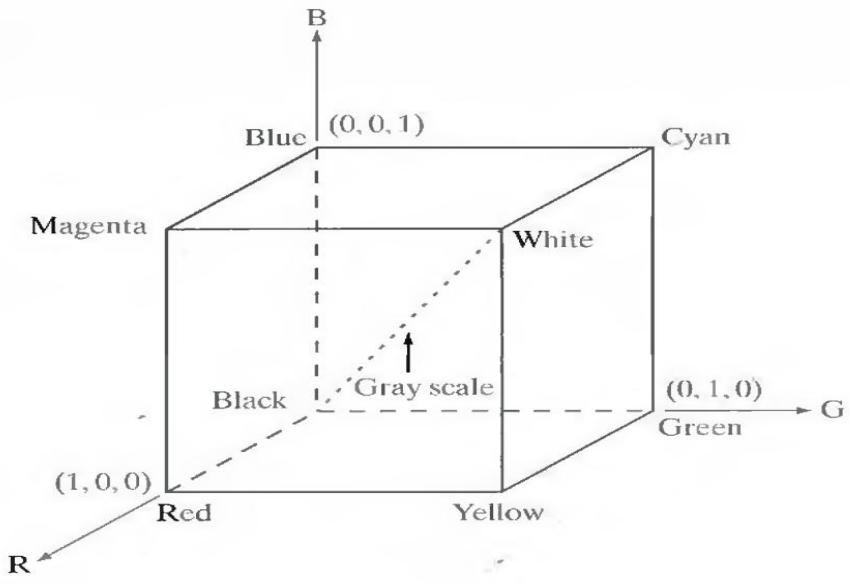
3.2. Renk Modelleri

3.2.1. RGB renk modeli

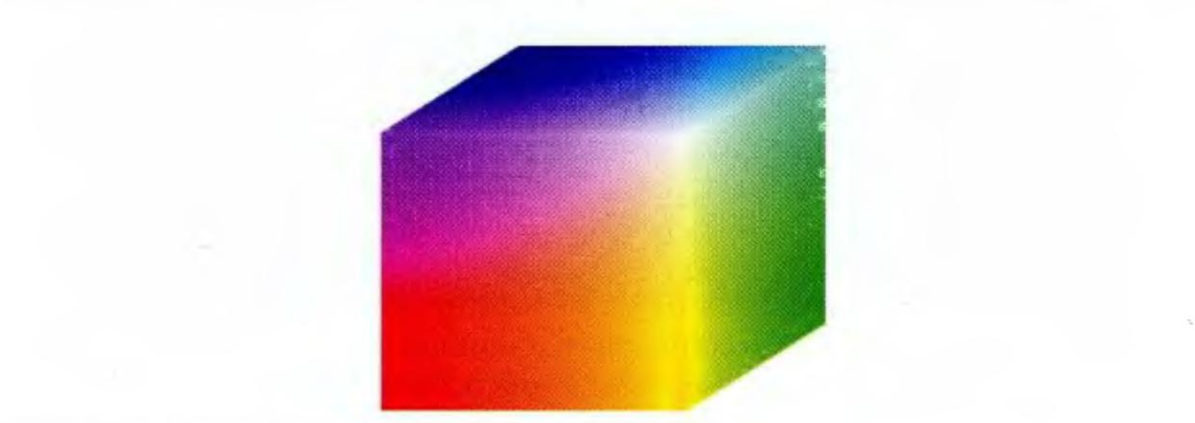
RGB modelinde her renk, kırmızı yeşil ve mavinin başlıca görüngesel bileşenlerinde görülmektedir. Bu model, Kartezyen koordinat sistemini temel almaktadır. Etkinin renk boşluğu, üç köşedeki RGB değerlerinin bulunduğu Şekil 3.3'de gösterilen küptür; orijinde siyah, orijinden en uzak köşede beyaz ve diğer üç köşede siyan(açık mavi) magenta(mor-pembe) ve sarıdır. Bu modelde gri ölçek(RGB değerlerine eşit olan noktalar), bu iki noktaya katılan doğru boyunca siyahtan beyaza doğru uzamaktadır. Bu modeldeki farklı renkler küpün içindeki ya da üzerindeki noktalardır ve orijinden uzanan vektörler ile tanınır. Kolaylık için, varsayım, figür 3.4'te gösterilen küpün birim küp olması için tüm renk değerlerini standartlaştırılmasıdır. Yani, R,G ve B'nin bütün değerlerinin [0,1] aralığında olduğu varsayılmaktadır.

RGB renk modelinde gösterilen görüntüler üç görüntü bileşeninden oluşur, her bir temel renk için bir tanedir. Bir RGB monitöre konulduğunda, bu üç görüntü bileşik bir görüntü

oluşturmak için ışıltılı bir ekran üzerinde birleşir. RGB boşluğundaki her bir pikseli temsil etmek için kullanılan bitlerin sayısı, piksel derinliği olarak adlandırılır. Kırmızı, yeşil ve mavi görüntülerinin her birinin 8 bitlik bir görüntü olduğu RGB görüntüsünü düşünülebilir. Bu şartlar altında her bir RGB renk pikselinin [yani, (R,G,B) değer üçlüsü] 24 bitlik bir derinliğe sahip olduğu söylenebilir (görüntü düzlemi X düzlem başına bit sayısı). Dolgu rengi görüntü terimi, 24 bitlik bir RGB renkli görüntüsünü belirlemek için sık sık kullanılır. 24 bitlik bir RGB görüntüsündeki renklerin toplam sayısı B8K=16,777,216'dır. Şekil 3.4, Şekil 3.3'deki grafikle uyum sağlayan 24 bitlik RGB renkli küpünü göstermektedir.



Şekil 3.3. RGB renk küpü şeması (Ana köşegen boyunca uzanan noktalar gri değerlerdir.)



Şekil 3.4. RGB 24-bit renk küpü

3.2.2. CMY ve CMYK renk modelleri

Bölüm 3.2.1'de belirtildiği gibi, siyan, magenta ve sarı, ışığın ikincil renkleridir ya da alternatif olarak pigmentlerin temel renkleridir. Örneğin; siyan pigmenti ile kaplanmış bir yüzey beyaz ışık ile aydınlatıldığında, yüzeyden hiç kırmızı ışık yansımaz. Yani siyan; kırmızı, yeşil ve mavi ışıktan eşit miktarda oluşan beyaz ışıktan yansıyan kırmızı ışığı çıkarır. Renkli yazıcılar ve kopyalayıcılar gibi kâğıt üzerine renkli pigmentler bırakan cihazların çoğu CMY bilgi girişi ya da dâhili olarak bir RGB'den CMY iletişimi gerçekleştirmeyi gerektirir. Bu iletişim, bütün renk değerlerinin [0,1] aralığında standartlaştırıldığı varsayım sağlandığında basit bir işlem kullanılarak gerçekleştirilebilir. Eş. 3.1 eşitliği, kırmızı ışık içermeyen saf siyan ile kaplı bir yüzeyden yansıyan ışığı gösterir (yani, eşitlikteki C=1-R).

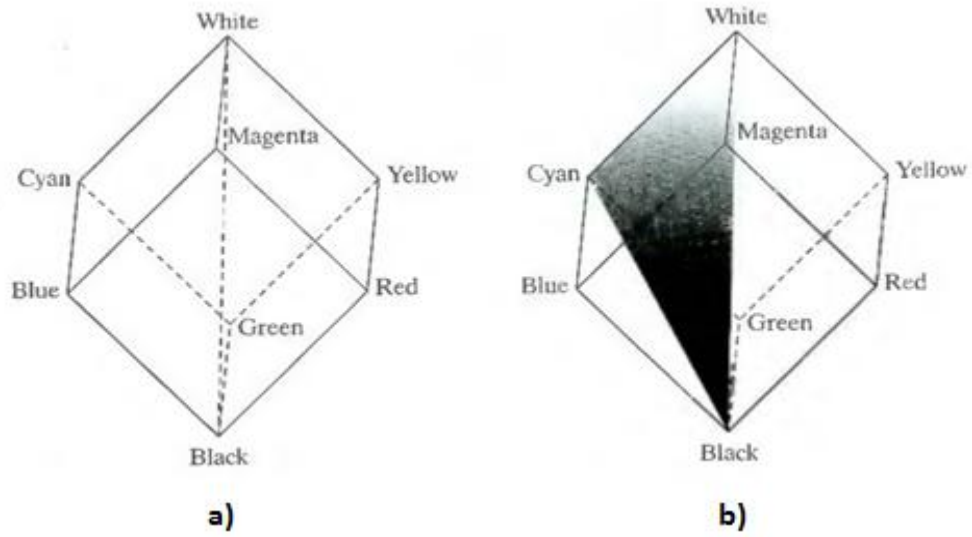
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.1}$$

Benzer şekilde, saf magenta yeşil rengi yansıtmaz ve saf sarı mavi rengi yansıtmaz. Eş. 3.1 eşitliği aynı zamanda, RGB değerlerinin, bireysel CMY değerlerini 1'den çıkararak elde edilen bir grup CMY değerinden kolay bir şekilde elde edilebileceğini açığa çıkarmaktadır. Daha önce belirtildiği gibi, görüntü işlemede bu renk modeli basılı kopya çıktısından kaynaklanan bağlantıda kullanılır, bu yüzden genellikle CMY'den RGB ters işlemi birazcık pratik ilgiden yoksundur.

Figür 3.4'e göre, eşit miktarda siyan, magenta ve sarı pigment öncülleri siyah ışığı oluşturur. Uygulamada, yazdırma işlemi için bu renkleri birleştirmek bulanık bir siyah oluşturur. Bu yüzden, (yazdırma işlemindeki baskın renk olan) gerçek siyahı oluşturmak için, CMYK renk modelini meydana getiren dördüncü bir renk olan siyah eklenir. Bu yüzden, basımcılar "dört renkli yazma işleminden bahsederken, CMY renk modelinin üç rengi ve birde siyah rengi kastetmektedirler.

3.2.3. HSI renk modeli

Daha önce gördüğümüz gibi RGB ve CMY modellerinde renk oluşturmak ve bir modelden diğerine geçiş yapmak kolay bir işlemdir. Daha önce dikkat çekildiği gibi bu renk sistemleri donanım uygulamaları için ideal olarak uygundur. Üstelik RGB sistemi, kırmızı, yeşil ve mavi öncülleri güçlü bir şekilde algılayan insan gözü ile iyi bir şekilde eşlenir. Maalesef RGB, CMY ve diğer benzer renk modelleri, insan yorumlaması için pratiklik açısından renkleri tanımlamada uygun değildir. Örneğin; bir kişi, rengini oluşturan her bir öncülün yüzdesini vererek bir otomobilin rengini kastedemez. Dahası, biz renkli görüntülerin, tek bir görüntü oluşturmak için birleştirilen üç temel görüntünün birleşimi olduğu düşünülmez. İnsanlar renkli bir nesne gördüklerinde, onu renk tonu, canlılığı ve parlaklığı ile tanımlarlar. Parlaklık, pratik olması mümkün olmayan öznel bir tanımlayıcıdır. Yoğunluğun renksizlik kavramını içine alır ve renk algısını tanımlamadaki anahtar faktörlerden birisidir. Yoğunluğun (gri seviye), tek renkli görüntülerin en yararlı tanımlayıcısı olduğu bilinmektedir. Bu miktar kesinlikle ölçülebilir ve kolay bir şekilde yorumlanabilir. HSI (renk tonu, canlılık, yoğunluk) olarak adlandırılan takdim edilmek üzere olunan model renkli bir görüntüdeki yoğunluk bileşenini renk taşıyan bilgiden (renk tonu ve canlılık) ayırır. Sonuç olarak HSI renk modeli, bu algoritmaların geliştiricisi ve kullanıcısı olan insanlara doğal ve içgüdüsel gelen renk tanımlamalarına dayalı görüntü işleme algoritmaları geliştirmek için ideal bir araçtır. RGB'nin, (renkli bir kamera tarafından yakalanan bir görüntü ya da bir monitör ekranındaki görüntü görünümü gibi) görüntü renk üretimi için ideal olduğunu fakat renk tanımlaması için kullanımının çok daha sınırlı olduğu söylenerek özetlenebilir. Takip eden materyaller, bunu yapmak için etkili bir yol sağlar. Şekil 3.5 a)'da gösterildiği gibi renkli küpü A,1,1) beyaz noktası ile @,0,0) siyah noktası üzerine direkt olarak koyarsak, bu durum daha açık bir hale gelir. Şekil 3.5 ile olan bağlantısında belirtildiği gibi, yoğunluk (gri ölçek), bu iki eksene katılan doğru üzerindedir. Şekil 3.5'te gösterilen düzenlemede, siyah ve beyaz eksenlere katılan doğru (yoğunluk eksenleri) dikeydir. Bu yüzden, Şekil 3.5'teki her bir renk noktasının yoğunluk bileşenini belirlenmek istenirse, renk noktasını içeren ve yoğunluk eksenlerine dik bir düzleme geçilir. Düzlemlerin yoğunluk eksenleri ile kesişimi, [0,1] aralığındaki yoğunluk değerli bir noktayı verir. Aynı zamanda bir rengin canlılığının (saflığının), yoğunluk eksenlerinin uzaklık fonksiyonu gibi arttığı belirtildi. Aslında yoğunluk eksenleri üzerindeki noktaların canlılığı (doygunluğu), bu eksen üzerindeki bütün noktaların gri olmasından elde edilen delillerde olduğu gibi sıfırdır.

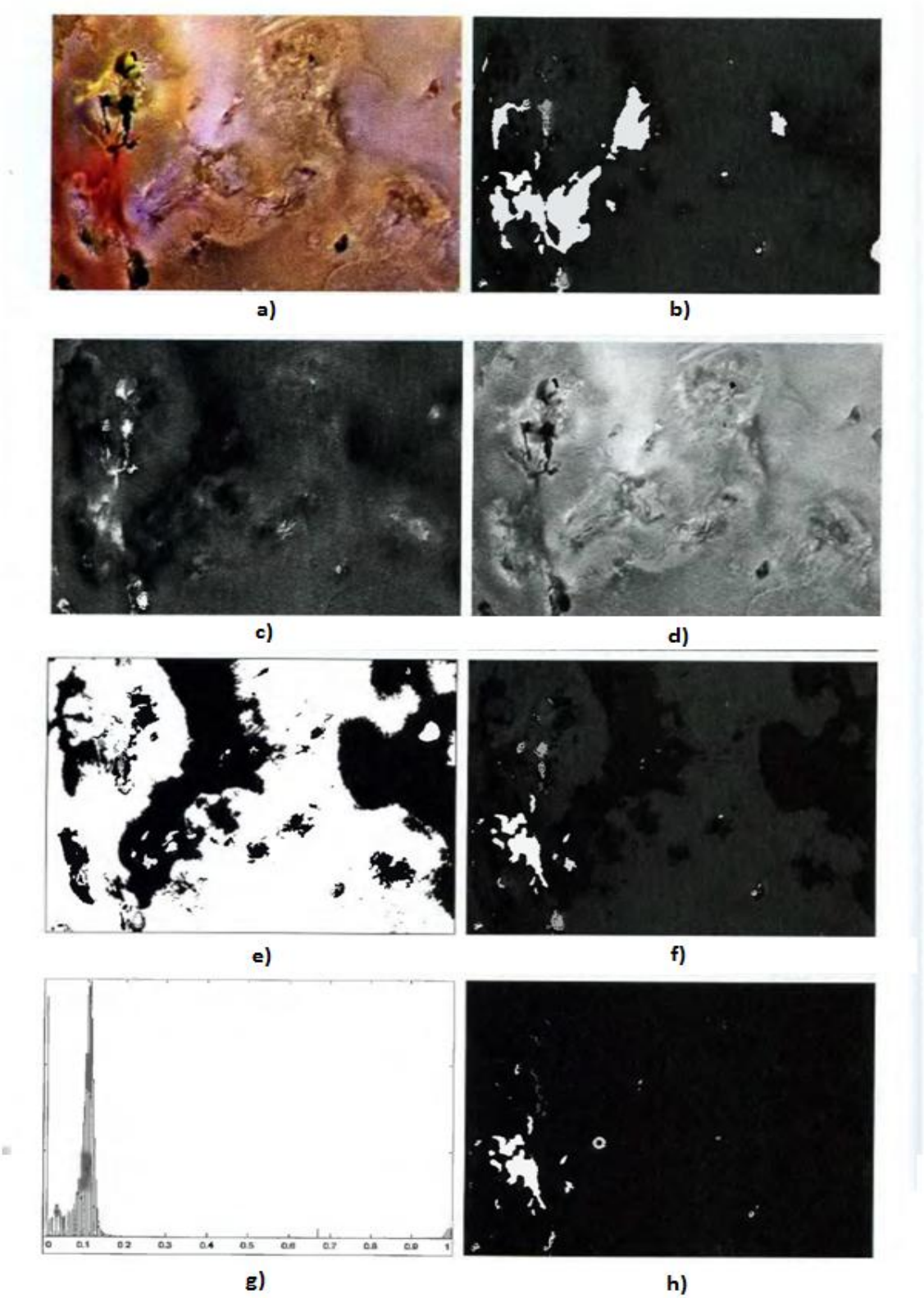


Şekil 3.5. RGB ve HSI renk modelleri arasındaki kavramsal ilişki

3.3. Renk Bölütlemesi

3.3.1. HSI renk alanındaki bölütleme

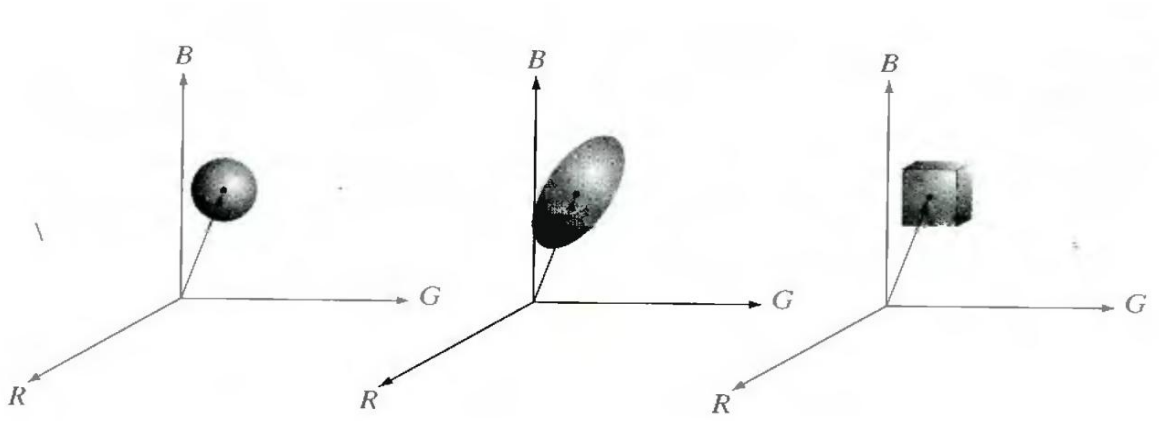
Renk temelli bir görüntü bölütlenmek istenirse ve dahası bu işlem bireysel düzlemler üzerinde yürütülmek istenirse, ilk olarak HSI alanının düşünülmesi doğaldır çünkü renk, görüntünün renk tonunda rahatlıkla temsil edilebilir. Tipik olarak canlılık, görüntünün renk tonundaki etkinin bölgelerine ayırmak için bir maskeleme görüntüsü olarak kullanılır. Görüntü yoğunluğu, renkli görüntülerin bölütlenmesinde daha az sıklıkla kullanılır çünkü hiçbir renk bilgisi taşımaz. Aşağıdaki örnek, bölütlendirme işleminin HSI sistemlerinde nasıl gerçekleştiğinin bir örneğidir. Sahte renk metotları tarafından gerçekleştirilmesine rağmen bu görüntüdeki bölütlendirme, genellikle alan kaybı olmayan dolgu rengi olarak işlenebilir (bölütlendirilebilir). (d) boyunca Şekil 3.6 b) figürleri onun IISI bileşen görüntüleridir. Şekil 3.6 a) ve (b)'deki renklerin, kırmızının mavi-magenta tarafı üzerinde olduğunu belirten nispeten yüksek renk tonu değerlerine sahip alanlarının karşılaştırılması olduğu görülmektedir. (Şekil 3.6'ya bakınız). Şekil 3.6 e), canlılık görüntüsündeki maksimum %10'a eşit bir sınır ile canlılık görüntüsünü sınırlandırarak ortaya çıkmış ikili bir maskeyi göstermektedir. Sınırdan yüksek her piksel değeri 1'e (beyaz) ayarlanır. Diğerlerinin tamamı 0'a (siyaha) ayarlanır. Şekil 3.6 f) görüntü renk tonlu maskenin ürünüdür ve Şekil 3.6 g) ürün görüntüsünün histogramıdır (gri ölçeğin [0.1] aralığında olduğu belirtilmiştir).



Şekil 3.6. HSI uzayında görüntü bölütleme a) Asıl görüntü b) Hue c) Doyum d) Yoğunluk e) İkili doygunluk maskesi (siyah=0) f) b ve e'nin ürünü g) f'nin histogramı h) a'nın kırmızı bileşenlerinin bölütlemesi

3.3.2. RGB doğrusal uzayında bölütleme

Bu bölümde bahsedildiği gibi HSI alanında çalışmak daha içgüdüsel olsa da, bölütleme, RGB renk vektörleri kullanılarak elde edilen daha iyi sonuçların bulunduğu bir alandır. Yaklaşım kolaydır. Amacın, bir RGB görüntüsünde belirlenmiş renk aralığının nesnelerini bölütlemek olduğu varsayılsın. Etki renklerini temsil etmek için verilen bir grup örnek renk noktasından bölütlenmek istenilen ortalama renk tahmini elde edilebilir. Bu ortalama renk, RGB vektörü tarafından belirlensin. Bölütlemenin amacı, belirlenmiş aralıkta ya da aralığın dışında bir renge sahip olan görüntüdeki her bir RGB pikselini sınıflandırmaktır. Bu karşılaştırmayı gerçekleştirmek için benzerlik ölçüsüne sahip olmak gereklidir. En basit ölçülerden birisi Öklid uzaklığıdır. z, RGB alanında rastgele seçilmiş bir nokta belirlesin. Eğer aralarındaki uzaklık belirlenmiş eşikten, ZH, az ise z'nin a'ya benzediği söylenebilir. z ve a arasındaki Öklid uzaklığı, D(z,a)=IIz-aII tarafından verilir.



Şekil 3.7. RGB vektör bölütlemesi için veri bölgelerini kapsayan üç ayrı yaklaşım

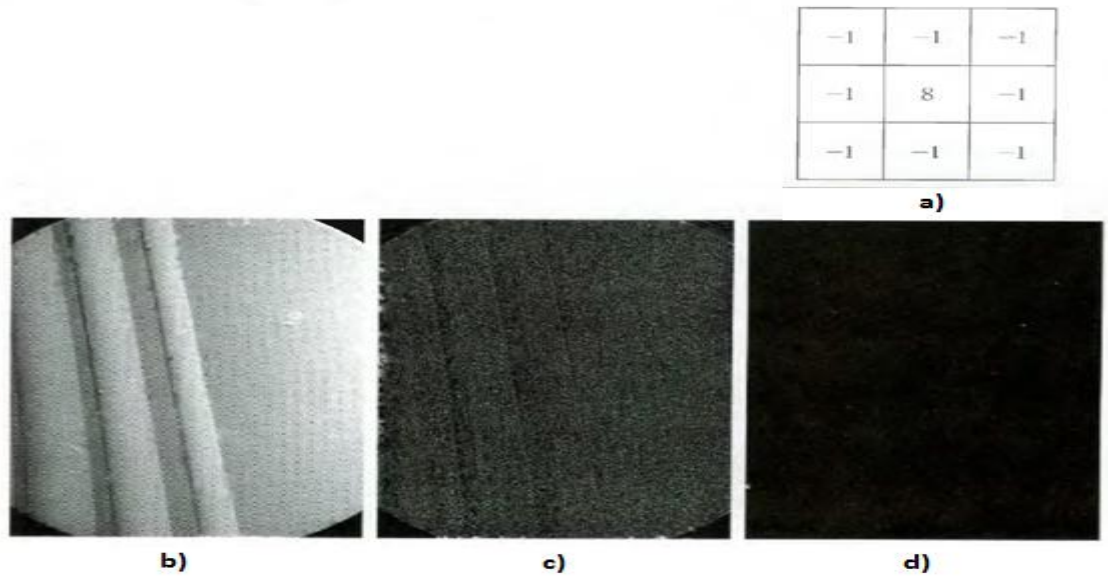
4. NESNE İZLEME METOTLARI

4.1. Nokta İzleme

Bir görüntüdeki ayrılmış noktaların belirlenmesi kural olarak kolaydır. Şekil 4.1 a)'da gösterilen maskeyi kullanarak, T'nin negatif olmayan bir eşik olduğu ve R'nin Eş. 4.1 eşitliği tarafından verildiği yerde;

$$R \geq T \tag{4.1}$$

olursa, bir noktanın, maskenin merkezlenmiş olduğu bölgede belirlendiği söylenebilir. Temel olarak bu formül, merkez noktası ve onun civarı arasındaki ağır basan farklılıkları ölçmektedir. Buradaki düşünce ayrılmış bir noktanın, (gri seviyesi arka planından kayda değer şekilde farklı olan ve türdeş ya da neredeyse türdeş olan alanda bulunan bir nokta), etrafındakilerden oldukça farklı olacağıdır ve bu yüzden bu tür bir maske ile kolayca belirlenebilir [3]. Şekil 4.1 a)'da ki maskenin, laplas işlemi ile bağlantılı olarak Şekil 4.1 d)'de gösterilen maskeyle aynı olduğu görülmektedir. Fakat buradaki vurgu, tam olarak noktaların belirlenmesi üzerindedir. Yani, etkinin dikkate alınan farklılıkları, (7 ile belirlendiği gibi) dikkate alınan ayrılmış nokta olmak için yeteri kadar büyük olanlardır. Maske karşılığının sabit gri seviye alanlarında sıfır olacağı belirtilerek maske katsayılarının toplamının sıfır olduğuna dikkat edilmelidir.



Şekil 4.1. a) Nokta tespiti maskesi b) Gözenekli bir türbin kanadının X-ray görüntüsü c) Nokta tespit sonucu d) Eş.4.1 kullanılarak elde edilmiş sonuç

Görüntünün sağ üst çeyreğindeki boşluklu jet motor tribün ağzının X-ray görüntüsünü gösteren Şekil 4.1 b)'nin yardımı ile bir görüntüden ayrılmış noktaların bölütlenmesi örneklerle açıklanmıştır. Boşluk içine gömülü tek siyah bir piksel vardır. Şekil 4.1 c), X-ray görüntüsüne nokta bulucu maskenin uygulanmasının sonucudur ve Şekil 4.1'de, Şekil 4.1 c)'deki en yüksek mutlak piksel değerinin %90'ına eşit olan 7 ile Eş. 4.1 eşitliğini kullanmanın sonucunu göstermektedir. Tek piksel bu görüntüde açık bir şekilde görünür (piksel, yazdırma işleminden sonra görülebilmesi için manuel olarak büyütülür). Bu tip belirleme işlemi oldukça özelleştirilmiştir çünkü belirleyici maske alanında türdeş bir arka plana sahip olan tek piksel süreksizliklerine (kesikliklerine) bağlıdır. Bu durum sağlanmadığında, bu bölümde tartışılan diğer metotlar gri seviye süreksizliklerini belirlemek için daha elverişlidir.

4.2. İhtimale Dayalı Bir Yaklaşımı Kullanarak İzleme

Pek çok izleme algoritması, hem hareket eden nesneleri belirlemek için arka plan modelini hem de belirlenen nesnelerin hareket modellerini analiz etmek için nesne izlemeyi kullanır. Bu durumda, beklenti maksimizasyonu (en üst düzeye çıkarma) (BM), arka planı modellemek ve hareket eden nesneleri belirlemek için kullanılır. İzleme işlemi, renkli histogram nesnelerini temel almaktadır. Beklenti maksimizasyonu kullanılarak arka plana ait olan bir piksel değerinin olasılığı hesaplanabilir. Bu esnada, nesneye ait olan pikselin olasılığını hesaplamakta kullanılan nesne izleme işlemi için nesnenin renkli histogramı kullanılır [12].

Sık sık kullanılan uygulamaların zor şartlarından dolayı problem hala sorun teşkil eder. Değişen aydınlatma, dalgalanan ağaçlar, su, görüntü değişiklikleri ve gölgeler, problemi sorunlu hale getiren örnek sorunlardır. Ve aynı zamanda izleme işlemindeki nesneler düşünülebilir değildir. İnsanlar kalıplaşmış bedenler değildirler ve onların hareketlerini pek çok durumda tahmin etmek zordur. İnsanlar yön değiştirir, hızlanır, durur, etkileşim kurar, bir araya gelir ve birbirlerinden uzaklaşırlar. Birbirlerini engelledikleri işlemde görüntünün başka bir yerde görünmesine izin verilmelidir. Bu durum, mükemmel algoritmanın henüz bulunamamasının ve yakında da bulunamayacağının sebebidir.

4.2.1. İhtimale dayalı integralleme

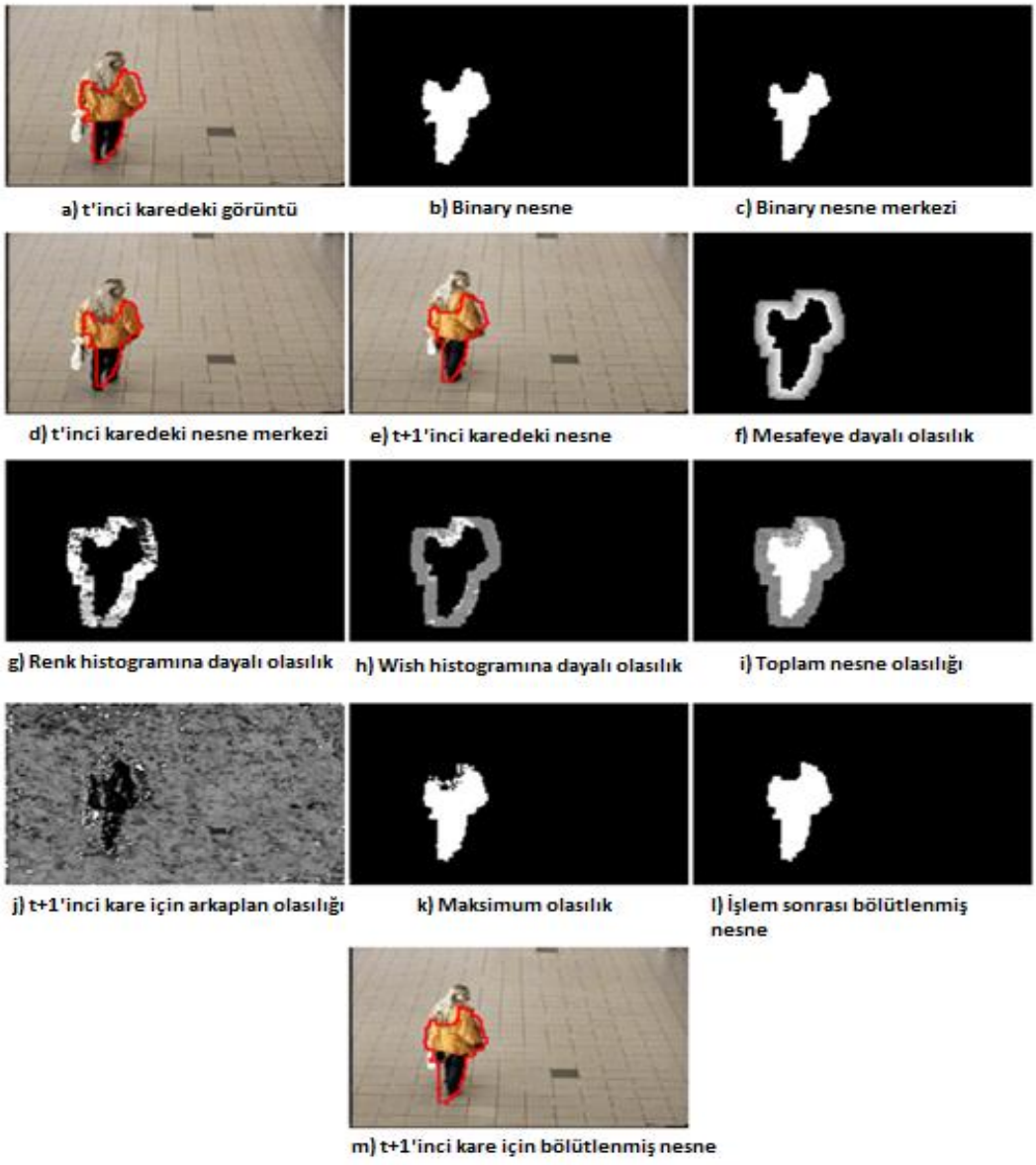
Arka plan modelleme işleminin bir sonucu olarak her bir piksel için, pikselin arka plan olma ihtimaline dair bir tahmin elde edilebilir. Nesne olan piksel için bir ihtimale sahip olunursa, arka plan ve herhangi bir nesne arasındaki piksel sınıflandırmasında hiçbir eşiğe gerek duyulmayacaktır. Pikseller bölüme en yüksek ihtimalle tahsis edilir. İzlenmek istenilen nesnelerin fiziğine bakarak, iki çerçeve arasındaki nesnelerin sadece belli bir uzaklığa kadar hareket edebildiğini ve şekillerinin sadece birazcık düzelebildiği görülebilir. Bu yüzden nesnenin merkezi (merkez sözcüğü ile sınır civarında belirli bir kalınlık görünüşü hariç tüm nesne kastedilmektedir) nesnenin bozulumu ile fazla değişmeyecektir ve nesnelerin sınırlı hızından dolayı, kapanma miktarında önemli bir değişikliğe sahip olmayacaktır. Bu yüzden nesnenin merkez pozisyonunu histogram eşleme işlemi ile belirlemek nispeten güvenilirdir. Yeni konumda nesnenin merkezinde bulunan bütün piksellerin nesneye ait oldukları neredeyse kesindir, bu yüzden onların ihtimali eşlem hatasına bağlı olarak bire yakındır.

4.2.2. İntegralleme

Bu bölümde bazı uygulama ayrıntıları tanımlanacaktır. İlk olarak sahte konumlar verilecektir sonra uygulamanın bazı kısımları daha ayrıntılı bir şekilde açıklanacaktır. Her bir çerçeve için uygulanmış algoritmanın işlem basamakları aşağıdaki gibidir[10];

1. Eski nesnelerin tamamı ile başlayın (Resim 4.1 a) ve b)'ye bakınız ve merkezlerini hesaplayınız (Resim 4.1 c)'ye bakınız).
2. Yeni görüntüyü yorumlayınız.
3. Tahmin edilen bölgeleri elde etmek için Kalman tahminini kullanınız ve bütün nesneler için ROI'yı araştırınız.
4. Bütün nesnelerin merkez bölgesini bulmak için histogram eşleme yapınız (Resim 4.1 e)'ye bakınız).
5. Her bir piksel için ihtimali hesaplayınız (nesne merkezinin kenarından belirli bir uzaklıkta bulunan piksel için sadece bu nesnelerin ihtimalini hesaplayınız).
6. İhtimallere bağlı olarak nesnelerin merkezi dışında kalan tüm pikselleri sınıflandırınız (Resim 4.1 k)'ye bakınız).

7. Arka plan olarak sınıflandırılan bu pikseller için, muhtemel yeni nesneler belirlemek için ihtimallerini sınırlayınız (eşikleyiniz). Ön plan piksellerini gruplandırınız ve yeni nesneler oluşturunuz.
8. Çok küçük, ayrık vesaire olan nesneleri bulmak için bütün nesneler üzerinde işlem sonrasını uygulayınız (Resim 4.1 l)'ye bakınız).
9. Tüm arka plan pikselleri için arka plan modelini yenileyiniz.
10. Tüm nesneler için ağırlık merkezini hesaplayınız ve Kalman modelini, nesneyi, nesne merkezi renkli histogramlarını yenileyiniz (Resim 4.1 m)'ye bakınız).



Resim 4.1. Algoritma ayrıntıları

Video izlemenin önemli iki basamağı bu uygulamayla gösterilmiştir. Nesne tanımlama ve izleme ihtimale dayalı bir algoritma kullanılarak birleştirilebilir. Bu yaklaşım, daha önceden belirlenmiş nesneleri tekrar bulmak için hiçbir eşik gerektirmeyen bir avantaja sahiptir. Bu durum aynı zamanda daha iyi nesne bölütlenmesi ve (kısmi) kapanmanın daha kolay belirlenmesini sağlar. Nesnenin merkezi örnek eşleme kullanılarak izlenir. Nesne sınırına yakın olan pikseller, nesne piksellerine ilgili pozisyonları değiştirmek için izin verirken nesne sabitindeki renk bilgisini tutmak için çabalayan bir algoritmaya dayalı ihtimali kullanan her bir çerçevede belirlenir. Deneyler, insanlar gibi kalıplaşmış olmayan bedenleri izlerken bu durumun bir avantaj olduğunu göstermektedir.

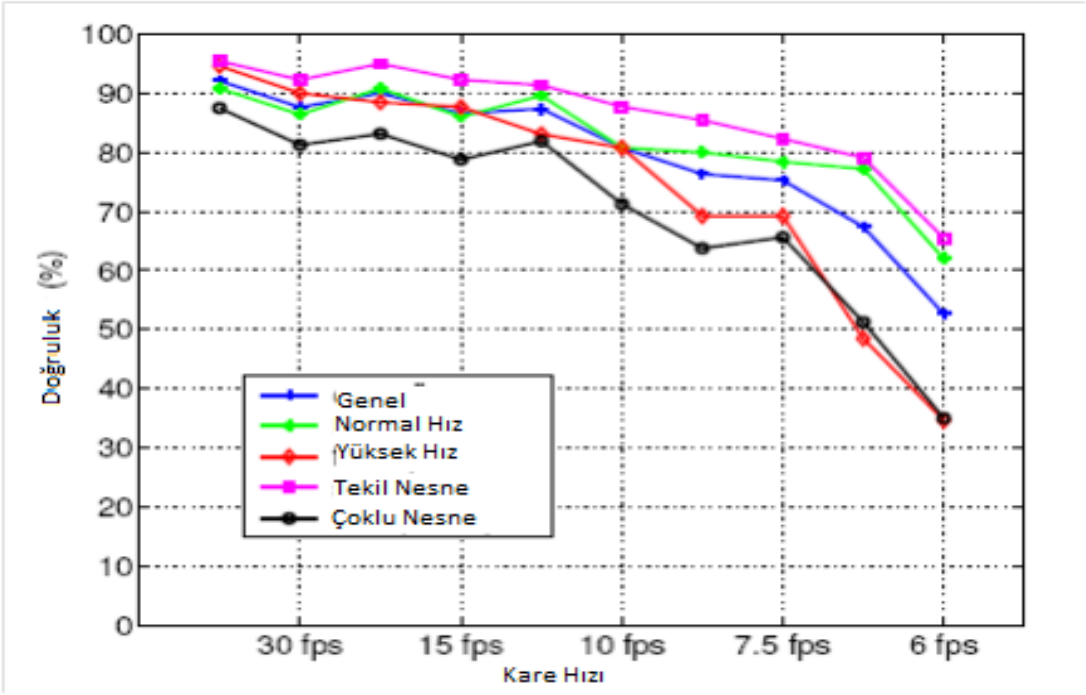
4.3. Çok Merkezli İzleme

Nesnelerin hızlı harekete sahip olduğu düşük çerçeve oranlı videolar için bir nesne izleme algoritması üzerinde çalışmalar yapılmıştır. Geleneksel ortalama rotasyon izleme işlemi geniş olan bir nesnenin tekrar yerleştirilmesi durumunda başarısız olur ve ardışık çerçeveler arasındaki bölgeleri örtüşmez. Tez çalışmasında bu probleme, yüksek hareket alanlarında ortalanmış çoklu merkezleri kullanarak bir çözüm sağlanması amaçlanmıştır. Ek olarak, yenilenen güncelleme mekanizmasında arka plan ve örnek benzerlikleri olarak adlandırılan iki ihtimal terimini birleştirerek ortalama rotasyonun yakınsama özelliklerini geliştirmek için uygulamalar yapılmıştır. Ön deneyler önerilen metodun geçerliliğini kanıtlar.

İzleme işlemi iki temel göreve sahiptir; hareket eden yeni nesneyi belirlemek ve aynı çerçevedeki daha önceki nesnelerin bölgelerini bulmak. Sabit kamera ayarları için yeni nesneleri belirleme, arka plan çıkarılarak yapılabilir, yani sabit görüntünün bir referans modeli ile çerçeveyi karşılaştırılarak işlem uygulanır.

Tek merkezli ortalama rotasyon izleme işleminin gerçekleştirilmesinde düşük çerçeve oran dizisini kullanmanın etkilerini anlamak için, test sonuçlarının belirtilmesinde kesin referans (hareket eden nesnenin sınırları ve yörüngeleri) belirttik ve ortalama rotasyon temelli izleme metodunun doğruluğu değerlendirilebilir. Test sonuçları, hem iç hem de dış görüntüleri, kısmi ya da tam kapatmaları (engellemeleri), yayalar araçlar bisikletler gibi değişik nesne çeşitlerini tanımlamaktadır. Değerlendirme sonuçları Şekil 4.2'te sunulmuştur. Çerçeve oranı küçük değerler aldıkça, izleme metodunun başarı derecesinin

düştüğü gözlenmiştir [14]. Performanstaki düşüş, hızlı hareket eden nesneleri içeren sonuçlarda daha açıktır.



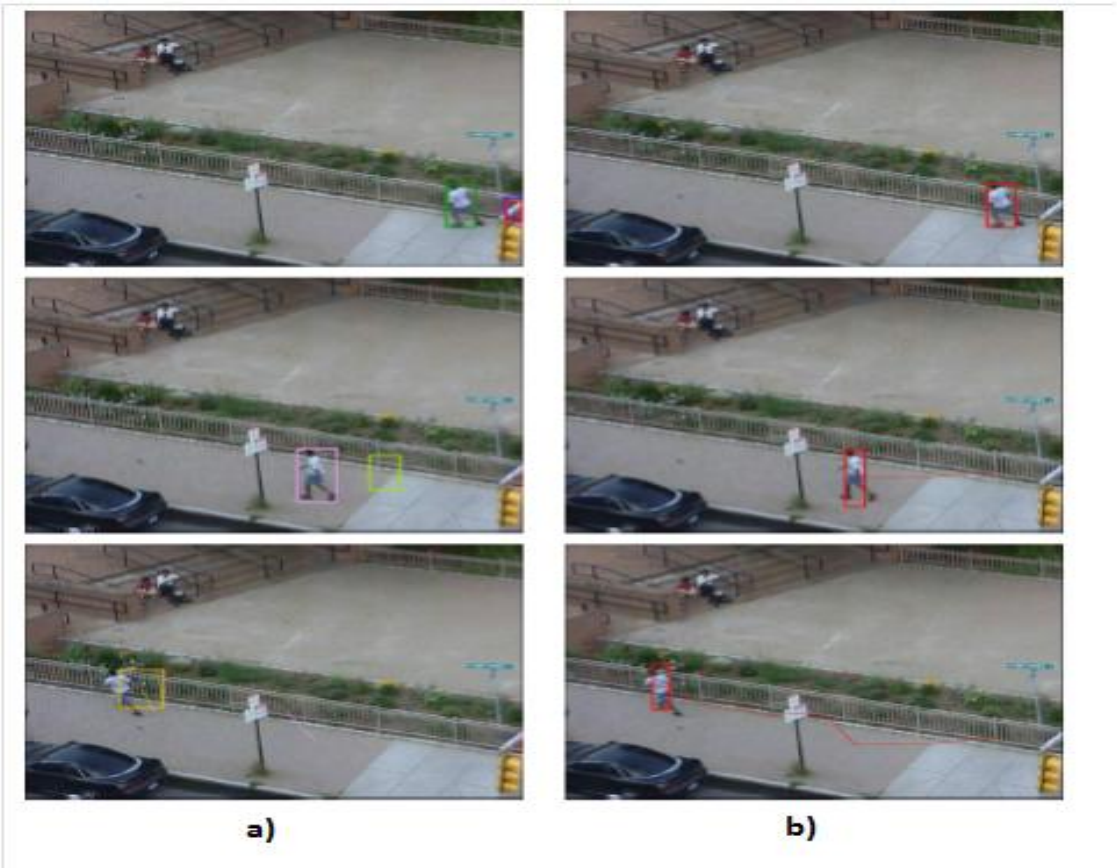
Şekil 4.2. Kare işleme hızı ve nesne hızı arasındaki ilişki

4.3.1. Çok merkezli ortalama rotasyon

Bir Bayes (Bayesian) yenileme mekanizması kullanılarak çoklu katmanlar tarafından oluşturulan istatistiksel bir arka plan modeli değerlendirilebilir ve çerçevedeki ön plan piksellerinin belirlenmesi için çerçeve muhtemel arka plan modelleriyle karşılaştırılır. Arka plan üretimi, aydınlatma miktarının değişimine göre bilinen katsayılara uyum sağlamaya imkân tanımaktadır. Bir uzaklık eşlemi elde etmek için, renk pikselleri ve uyum sağlayan modeller arasındaki uzaklığı hesaplanır. Ön plan maskesi, piksel renk değişimi kullanıp uzaklık eşlemini sınırlayarak hesaplanır, böylece eşik değeri her bir piksele uyum sağlar ve zaman içinde değişir. Belirlenen iki nesneye dikkat ederiz. Yeterli sayıda çerçeve ile izlenmeyen nesneler, muhtemel nesneler olarak işaretlenir. Nesne oluşturmak için bağlı bileşenler kullanılır. Her bir nesnenin bölgesini tahmin ettikten sonra, onlar bağlı bileşenlerle eşleştirilir. Belirli bir sayıda veya çerçevede bağlı bileşenlerin hiç biriyle eşleşmezse nesne silinir. Nesnelerin hiç biriyle eşleşmeyen bağlı bileşenlere göre yeni nesneler oluşturulur. Kullanılan nesneler, (Şekil 4.3'te gösterildiği gibi) çoklu ortalama

rotasyon merkezleri ile izlenir onların şekilleri, uzaklık eşlemi kullanan bir iç/dış bölümlendirici tarafından ayarlanır.

Şekil 4.3. Standart Mean-Shift ve çok merkezli Mean-Shift arasındaki farklar



Resim 4.2. Nesne takip sonuçları a) Tek merkezli b) Çok Merkezli

4.4. Arka plan Çıkarma İşlemi ile İzleme

Bir video dizisinden hareket eden nesneleri belirlemek, pek çok bilgisayar görüntülü uygulamada temel ve kritik bir işlemdir. Genel olan yaklaşım, arka plan modelinden kayda değer şekilde farklılaşan video çerçeve bölümünde hareket eden nesneleri tanımlayan arka plan çıkarma işlemini uygulamaktır. İyi bir arka plan çıkarma algoritması geliştirme işleminde pek çok sorun vardır. İlk olarak, aydınlatmadaki değişikliklere dayanıklı olmalıdır. İkinci olarak, hareket eden yapraklar, yağmur, kar ve diğer hareket eden nesnelerden oluşan gölgeler gibi sabit olmayan arka plan nesnelerini belirlemekten kaçınmalıdır. Son olarak, dâhili arka plan modeli, araçların durması ve hareket etmesi gibi arka planda gerçekleşen değişikliklere hızlı bir şekilde tepki vermelidir [4].

4.5. Etkili Görünüm Filtreleriyle İzleme

İzlenen nesne, referans pencere ve aday pencere arasındaki benzerlik ölçüsünü yükselterek tespit edilebilir. Maksimum değer, ya belirleyici bir yol ya da olasılıksal bir yol ile uygulanabilir.

Belirleyici metotlar, bu bölge ile hedef pencere arasındaki benzerlik ölçüsünü yükselten bir bölgeyi tekrar tekrar araştırarak her bir çerçevedeki izlenmiş nesneyi saptar. Bu metotlar, sayısal olarak etkilidir. Fakat metotlar, yerel maksimumu yakınsayabilir. Onlar arka plan çeldiricilerine, dağınıklığa, kapatmaya, hızlı hareket eden nesnelere duyarlıdırlar. Bu problemler, durum uzayındaki çoklu hipotezleri muhafaza eden olasılıksal metotlar ile giderilebilir ve bu şekilde yerel maksimuma daha fazla dayanıklılık kazandırır. Çeşitli olasılıksal metotlar arasında, partikül filtreleri (PF) çok başarılıdır. Partikül filtreleri basittir, dayanıklıdır, etkilidir ve pek çok sorun çıkaran işlemde başarı elde etmiştir. Partikül filtreleri eş zamanlı olarak, çoklu hipotezleri izler ve bir grup rastgele örnek partikül ile durum uzayında sonra gelen muhtemel yoğunluk fonksiyonunu tekrar tekrar tahmin eder. Hem görünüm modeli hem de benzerlik ölçüsü, partikül filtrelerinin performansı için çok önemlidir. Bu partiküller, benzerlik ölçüsüne göre (yani, ihtimal fonksiyon gözlemi) ağırlıklandırılır ve partikül örnekleme işlemi aynı zamanda benzerlik ölçüsüne bağlıdır.

Aslında, benzerlik ölçüsünün etkililiği ve dayanıklılığı, hem belirleyici hem de olasılıksal metotların performanslarını önemli ölçüde etkilemektedir. Bu tez, partikül filtrelerini geliştirme bağlamındaki benzerlik ölçüsü ya da etkili bir gözlem modeli geliştirmeye odaklanır. Bu tezin temel katkıları aşağıdaki gibi özetlenebilir;

1. Etkili bir SMOG görünüm modeli ve SMOG-temelli benzerlik ölçüsü.
2. SMOG parametrelerini etkili bir şekilde hesaplanması için yeni bir teknik.
3. Yeni bir şekil benzerlik ölçüsü.
4. Eksiksiz bir SMOG izleme algoritması.

Diğer birkaç popüler metot ile yapılan deneyler ve karşılaştırmalar, önerilen metodun, hızlı hareketleri, dağınıklılığı, ölçeklemeyi, değişen görünümü içeren sorunlu durumların varlığında nesneleri izleme işleminde ümit vaat eden bir performans sağladığını göstermektedir.

5. ETKİLİ NESNE TAKİP METODUNUN UYGULANMASI

5.1. Görüntü Moment Sentroidleri

Görüntü merkez çıkarımı görüntü işlemede yaygın olarak kullanılır [23, 24, 29]. Alan [29], sınır [24] ve başlangıç moment merkezi tekniği önerilen yöntemde nesnenin konumunu elde etmek için kullanılmıştır.

Görüntü momentleri tekniği ilk olarak Hu tarafından örüntü tanıma uygulamalarında kullanıldı [30]. Bir görüntünün moment görüntü piksel yoğunluğu olarak hesaplanabiliyordu. Tüm RGB kanalların piksel renk değerlerini kullanan momentler şöyle tanımlanabilir;

$$M_{ij} = \int_S x^i y^j I(x,y)dxdy \tag{5.1}$$

Bir sayılsa görüntünün momentlerini hesaplamak için sıfırıncı ve birinci momentler, Eş. 5.2, Eş. 5.3 ve Eş. 5.4'te gösterildiği gibi ayrık zamanda hesaplanmalıdır [28];

$$M_{00}(zeroth\ order\ moment) = \sum_x \sum_y x^0 y^0 I(x,y) = \sum_x \sum_y I(x,y) \tag{5.2}$$

$$M_{10}(first\ order\ moment\ x) = \sum_x \sum_y x^1 y^0 I(x,y) = \sum_x \sum_y xI(x,y) \tag{5.3}$$

$$M_{01}(first\ order\ moment\ y) = \sum_x \sum_y x^0 y^0 I(x,y) = \sum_x \sum_y yI(x,y) \tag{5.4}$$

Denklemde $I(x,y)$ piksel yoğunluğunu, x ve y görüntünün koordinatını göstermektedir. Alan olarak tanımlanan birinci dereceden momentler şöyle temsil edilir;

$$\bar{x} = \frac{M_{10}}{M_{00}}, \bar{y} = \frac{M_{01}}{M_{00}} \tag{5.5}$$

Bir görüntünün tüm resim içindeki ağırlığı moment merkezleri kullanılarak bulunabilir.

5.2. Histogram Geri Projeksiyon

Histogram geri projeksiyon hedefe ait olan 'Region Object of Interest' (ROI)'nin olasılığını hesaplamak için kullanılan bir yöntemdir [32]. Aynı zamanda hedefin yönünü, görüntüdeki dağılımın uzunluk ve genişlik olasılığını hesaplamak için kullanılır [33].

Nesnenin konumuna karar verilmesi için, histogram geri projeksiyon yöntemi arkaplandan ayrılan nesnenin bölgesine uygulanabilir. [34]. Histogram geri projeksiyon modelinde, hedef ve görüntü, renklendirilmiş histogram olarak temsil edilir [35].

Hedefin açısı (θ) şu şekilde tanımlanabilir;

$$\theta = \frac{\arctan\left[\dfrac{2\dfrac{M_{11}}{M_{00}} - x_c y_c}{\left[\dfrac{M_{20}}{M_{00}} - x_c^2\right] - \left[\dfrac{M_{02}}{M_{00}} - y_c^2\right]}\right]}{2} \tag{5.6}$$

Denklemde M görüntünün mometini, x_c ve y_c ise piksel numaralarını belirtmektedir. Hedefin uzunluk ve genişlik olasılığı şu şekilde gösterilebilir;

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \tag{5.6}$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}} \tag{5.7}$$

Denklemde l hedefin uzunluk olasılığını, w ise hedefin genişlik olasılığını göstermektedir. a, b ve c şu şekilde gösterilir;

$$a = \frac{M_{20}}{M_{00}} - x_c^2 \;;\; b = \frac{M_{11}}{M_{00}} - x_c y_c ; c = \frac{M_{02}}{M_{00}} - y_c^2 \tag{5.8}$$

5.3. K-Means Bölütleme

Bu bölümde K-Means kümeleme algoritması, önerilen yöntem için asistan filtre tekniğini tanımlayan denklemler ile açıklanmaktadır. En yaygın yöntem pixel renklerini "k" gruplara bölütlemek ve görüntü özelliği olarak k kümeler içindeki her rengin tekrarlanma sayısını bulmak için renk özelliklerini kullanmaktır [33,34]. Bir hedef görüntü, gri görüntü için 2^8, renkli görüntü için ise 2^{24} renk derinliği içeren "k" renklere sahiptir. Mean-shift takip işleminden önce uygulanan K-Means algoritması ile daha etkili bir takip için ön hazırlık çalışması yapılmış olur.

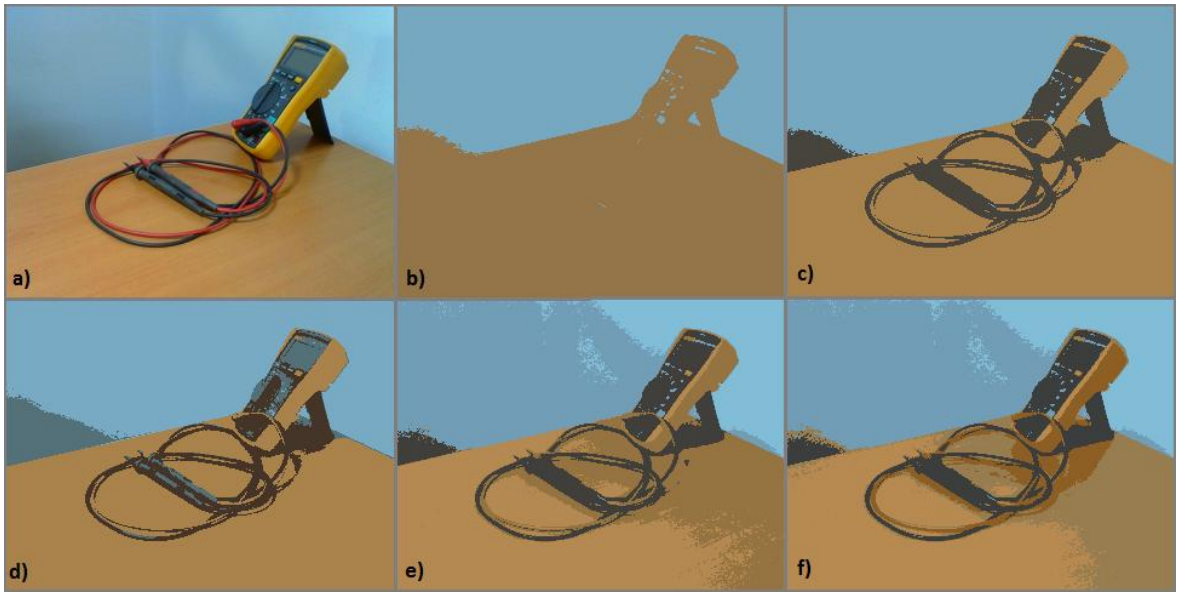
Merkez vektör (K kümeler) olarak belirtilen tüm kümeler için [35];

$$c_k = (c_{k,1}, c_{k,2}, ..., c_{k,n}) \tag{5.9}$$

Pikselleri, en yakın küme merkez vektörüne uygulamadan önce, aşağıdaki gibi her pikselin öklid uzunluğunu alıyoruz;

$$d(p_j^i, c_k) = \sqrt{(p_{j,1}^i - c_{k,1})^2 + (p_{j,1}^i - c_{k,2})^2 + ... + (p_{j,n}^i - c_{k,n})^2} \tag{5.10}$$

Denklemde *M,* her resimdeki piksellerin sayısını belirtir. Piksellerin gri değerlerinin ortalamasını bulmak için, her piksel en yakın merkezinin kümesine tahsis edilebilir. Daha sonra yeni ve eski küme merkezlerinin benzerliğini kontrol edebiliriz. Orijinal merkez ve hesaplanan merkez benzer olana kadar iterasyon devam eder. Benzer olduklarında, döngü durur. Resim 5.1'de, bazı alternatif yakınsama örnekleri gösteriliyor.



Resim 5.1. K-Means yaklaşım örnekleri a) Anlık görüntü b) 2 küme x 4 yineleme c) 3 küme x 8 yineleme d) 4 küme x 18 yineleme e) 5 küme x 8 yineleme f) 6 küme x 8 yineleme

5.4. Bhattacharyya Katsayısı

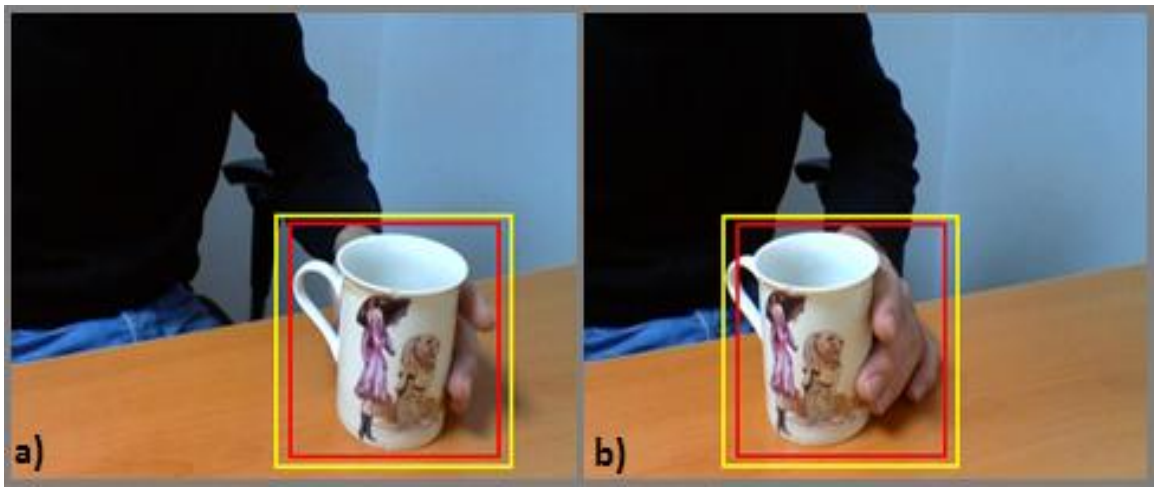
Takip uygulamalarında, iki resim parçası arasındaki benzerliği bulmamız gerekir. Renk diyagramları, genelde resimlerin özelliklerini nitelendirmek için kullanılır.Bir resimdeki nesne, resim histogramında gösterilir. Tanımayı uygulamak için, önceden seçilmiş takip edilen nesnenin kaynak histogramına ve bir aday histograma ihtiyacımız vardır. Kaynak ve aday histogramları eleştirmek için Bhattacharyya Coefficient, Earth Movers Distance, Chi Squared, Euclidean Distance gibi bazı metotlar kullanılabilir. Bu çalışmada, takip kusurlarını azaltmak için, sapma ve hatalarda daha fazla avantaj sağladığından mean-shift tekniğini kullanan Bhattacharyya katsayısı metodunu öneriyoruz [39].

Bhattacharyya katsayısı, “bin” özdeş kutularıyla normalize edilmiş histogramlarla ilgilenir ve aynı zamanda hedef histogram ve aday histogram arasındaki normalize mesafe anlamına gelir [40,41]. Bhattacharyya katsayısının tahmini hesaplanması iki histogramla açıklanmıştır [42]:

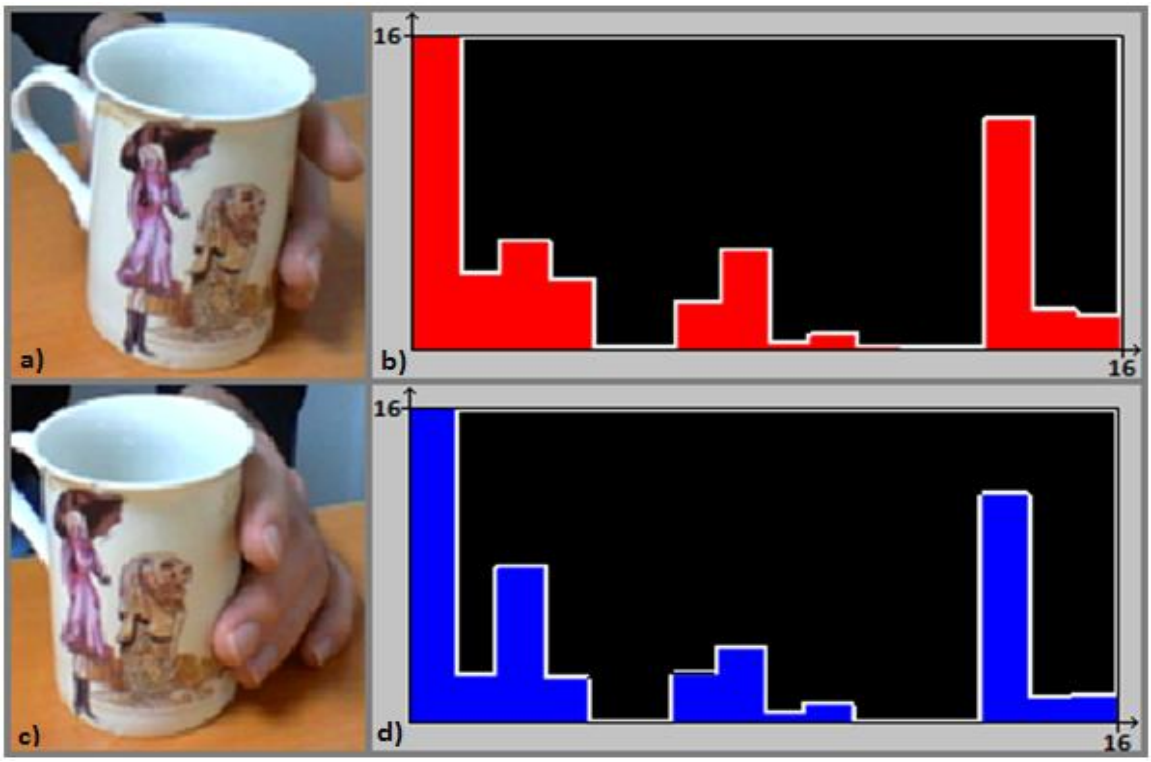
$$C_{Bha}\left(p_c^i, q\right) = \sum_{u=1}^{m} \sqrt{p_{cu}^i . q_u} \tag{5.11}$$

Denklemde m, histogramın boyutudur ve "bin" için indekstir. p ve q, hedef ve adayın histogram fonksiyonlarıdır. Algoritmamızda, kararlılık ve kesinliği artırmak için m asıl değer olmak zorundadır. Değer, metodun optimum performansı için kayda değerdir.

Standart takip metodunu kullanmak yerine, dikdörtgen nesne tekniğini kullanıyoruz.Takip edilmek istenen hedef bölgeyi kapsayan ilk bölge seçilir. Hedef lokalizasyonla, ilgilenen görüntüdeki nesnenin konumsal koordinatlarını azaltabiliriz. Böylece gerçek zamanlı bir tarama işlemine daha da yaklaşılmış olur [43]. Resim 5.2'de, hedef ve aday histogram çıkarımı için uygulanan görüntüler gösterilmiştir. Birinci ve ikinci resim için ilk kareler Resim 5.2'de tanımlanmıştır. m=16 boyutunu kullanıyoruz ve o değer için histogram sonuçları Resim 5.3'de gösteriliyor.



Resim 5.2. a) Hedef görüntü b) Aday için anlık görüntü



Resim 5.3. Hedef ve aday nesne için histogram sonuçları a) Hedef nesne b) Aday nesne c) Hedef için 4x2x2 bin d) Aday histogramı için 4x2x2 bin

Pozisyon parametresini içeren Bhattacharyya katsayısının matematiksel fonksiyonu beta, aşağıdaki fonksiyonda gösteriliyor [28]:

$$J_{Bha}(\beta) = \sum \sqrt{{}_{\gamma ti}\beta + {}_{\omega ti}(1-\beta) q_{ti}} + \sum \sqrt{{}_{\gamma bi}\beta + {}_{\omega bi}(1-\beta) q_{bi}} \tag{5.12}$$

Eğer katsayı 1 ise, hedef ve aday nesne mükemmel bir şekilde benzerdir.Hatasız bir takip için m matematik fonksiyonu 1'e yakın olmalıdır. Çizelge 5.1'de, kullanılan yöntem için Resim 5.3 a) ve Resim 5.3 c)'deki histogramları verilen objelerin bin değerleri çıkarılmış ve Bhattacharyya katsayıları hesaplanmıştır.

Çizelge 5.1. Resim 5.3 b) ve Resim d)'de belirtilmiş hedef ve aday histogramlar için Bhattacharyya katsayısı

Dizi	Hedef Frekans p_u	Aday Frekans q_u	$\sqrt{p_u q_u}$
1	0,04962519	0,06994858	0,05891699
2	0,2113193	0,2009226	0,2060554
3	0,1813343	0,186706	0,1840006
4	0,02848576	0,03334846	0,03082136
5	0	0	0
6	0	0	0
7	0,06394303	0,07108288	0,0674185
8	0,04955022	0,06284029	0,05580099
9	0	0,001890502	0
10	0,03178411	0,02540835	0,02841798
11	0,01334333	0,009830611	0,01145308
12	0	0,0001512402	0
13	0	0	0
14	0,03418291	0,03357532	0,03387775
15	0,1535982	0,1391409	0,146191
16	0,1828336	0,1651543	0,1737692
Bhattacharyya Katsayısı			0,9967228

Çizelge 5.1, sadece hedef ve adayı içeren giriş alanını kullanan karşılaştırma örneği çıktılarını göstermektedir. Tezde önerilen yöntem ilk kare için ön tanımlı nesne lokalizasyonunu içermektedir. Ön tanımlı nesne lokalizasyon alanı, nesneyi daha hızlı takip etmek için kullanılmıştır. Resim 5.2'de görüldüğü gibi yeşil kare ön tanımlı tarama alanını, kırmızı kare histogram alanını göstermektedir.

5.5. Mean-Shift Takip Yöntemi

Bu kısımda hedef nesneyi biçimlendirmek için, önce nesnenin niteliklerinin bilgisini alıyoruz.

Nitelik olarak, modelimizde renkler ve normalize edilmiş histogramlar kullanılıyor. Yani video zincirinde, nesne, IR(Initial Region) olarak adlandırılan dikdörtgenin yerleştirilmesiyle seçiliyor. İlk karedeki hedef için normalize edilmiş renk histogramları

kullanarak, aynı lokasyonda IR (Initial Region) ile lokalize edilen sonraki kareler için aday histogramlar oluşturulur.

Önceki adımlarda iyi lokalizasyon yapmış olmak,iyi takiple neticelenir. Bu yazıda, Mean-shift algoritmasıyla birleştirilmiş nesne lokalizasyonunu önerilmektedir. Mean-shift, sonraki notasyonlarla anahat belirleyen, parametrik olmayan, tekrarlı bir teknik olduğundan düzeltilmiş takip sağlar. Bu özellikler çoğu makine öğrenme uygulamalarında uygulanır (45,46).

Mean-shift algoritması, yeni bir takip ağırlık merkezi bulmak için Mean-shift vektörlerini sırayla tekrarlayarak yeniden kurar. Hedefin referans renk histogramı:

$$\hat{q} = \{\hat{q}_u\}_{u=1,\ldots,m} \sum_{u=1}^{m} (\hat{q}_u = 1) \tag{5.13}$$

Hedef ve aday histogramların arasındaki uzaklığı almak için ölçü (metrik) [47]:

$$d_{\hat{q}}(y) = \sqrt{1 - \hat{\rho}[\hat{p}(y), \hat{q}]} \tag{5.14}$$

$\rho[p(y), q]$ hedef ve aday histogramların Bhattacharyya katsayısıdır ve aşağıdaki denklemle gösterilir:

$$\hat{\rho}(y) = \hat{\rho}[\hat{p}(y), \hat{q}] = \sum_{u=1}^{m} \left(\sqrt{\hat{\rho}_u(y)\hat{q}_u}\right) \tag{5.15}$$

Dikdörtgen hedef alanı, video sırasında ilk çerçevede belirtilir. Renk modeli, bu ilk dikdörtgen alanla hesaplanır. Bu önerilen metot, takip için hedefin dikdörtgen alanının merkezini kullanır. Takip adımları uygulandıktan sonra, aranan ölçek alanı, uyarlanabilir Mean-shift takipçisi tarafından değiştirilir. Ölçek değiştirilerek ve arama pozisyonunu güncellenerek, adayın renk histogramı hedef histograma yaklaştırılmış olur.

Hedef model $q=\{q_u\}_{u=1,\ldots,m}$ ve aday model $p=\{p_u\}_{u=1,\ldots,m}$ için renk ağırlıklı histogram (p^c, q^c) ve arkaplan ağırlıklı histogram (p^b, q^b) şöyle gösterilir [30]:

$$I_u^c(x) = C^c \sum_{i=0}^{n} k[r_i(x)]\delta[b(x_i) - u]$$
$$I_u^b(x) = C^b(x) v_u \sum_{i=0}^{n} k[r_i(x)]\delta[b(x_i) - u] \quad (5.16)$$

RGB resminde $b(x_i)$, x_i için histogram dizisinin seri numarasını temsil eder. Deneylerde 4x4x4 kullanılmaktadır. $C_c = 1/\sum_{i=1}^{n} k[r_i(x)]$ "normalize" edilmiş bir katsayıdır ve δ delta fonksiyonudur. Formul kk'de v_u , 0 ile 1 arasında minimum değer alan arkaplan ağırlık vektörüdür [47,48].

Aday lokasyonu *x*'in optimal güncelleme problemi, $\rho(q,p)$ benzerlik fonksiyonu tarafından şu denklem ile çözülebilir:

$$\Delta x = \arg\max\{\rho[q, p(x_0 + \Delta x)]\} \quad (5.17)$$

Denklemde x_0 ilk noktayı gösterir ve $p(x_0)$ 'i kullanan maksimizasyon denklemi söyle belirtilir [46,47]:

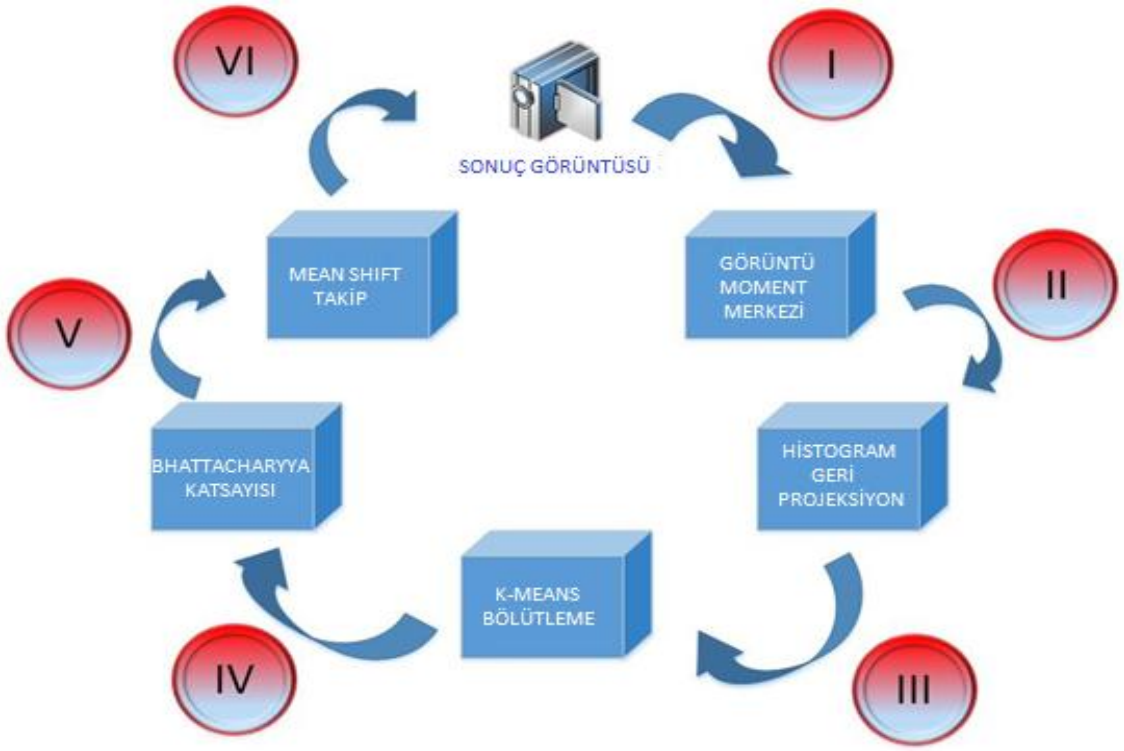
$$x = \arg\max \sum_{u=0}^{m} p_u(x) \sqrt{\frac{q_u}{p_u(x_0)}} \quad (5.18)$$

6. SONUÇ VE ÖNERİLER

Bu çalışmada klasik nesne takip uygulamalarından farklı olarak, histogram eşitleme, moment merkez hesaplama, bölütleme, filtreleme gibi bazı görüntü işleme basamaklarını bir arada içeren iyileştirme işlemleri üzerinde çalışılmıştır. Nesne takip yöntemi; gözetleme [15], sürücü yardım sistemleri [16], akıllı bina sistemleri [17], video özetleme [18,19] ve medical sistemler [20-22] gibi birçok yapay görme uygulamasında [23-25] kullanıldığından dolayı görüntü işleme basamaklarında kullanılacak önceden belirlenmiş işlem parametreleri çok değişkenlik gösterebilmektedir. Bu yüzden tez çalışması üzerinde durulan yöntemde ana takip işlem basamağına geçmeden önce uygulanan filtreler ve geliştirilmiş K-Means algoritması ile daha etkili bir takip için ön hazırlık çalışması yapılmış olur. Böylece değişkenlik gösteren işlem parametrelerinin farklı uygulamalar sonuca olumsuz etkisi azaltılmış olur.

Tez çalışmasında önerilen takip yöntemi Şekil 6.1 de gösterilmiştir. İşlem basamakları şu şekilde özetlenebilir:

1. Histogram eşitleme
2. Görüntü moment merkezi çıkarımı (I)
3. Histogram geri projeksiyon (II)
4. K-Means bölütleme (III)
5. Bhattacharyya katsayısı çıkarımı (IV)
6. Mean Shift algoritması (V)
7. Sonuç görüntüsü çıkarımı (VI)



Şekil 6.1. Önerilen yöntem için akış diyagramı

Görüntü moment merkezi yöntemi moment merkezlerini analitik olarak bulur. [23, 24]. Çalışmada önerilen yöntem için sıfırıncı, birinci ve ikinci dereceden momentler kullanılır.

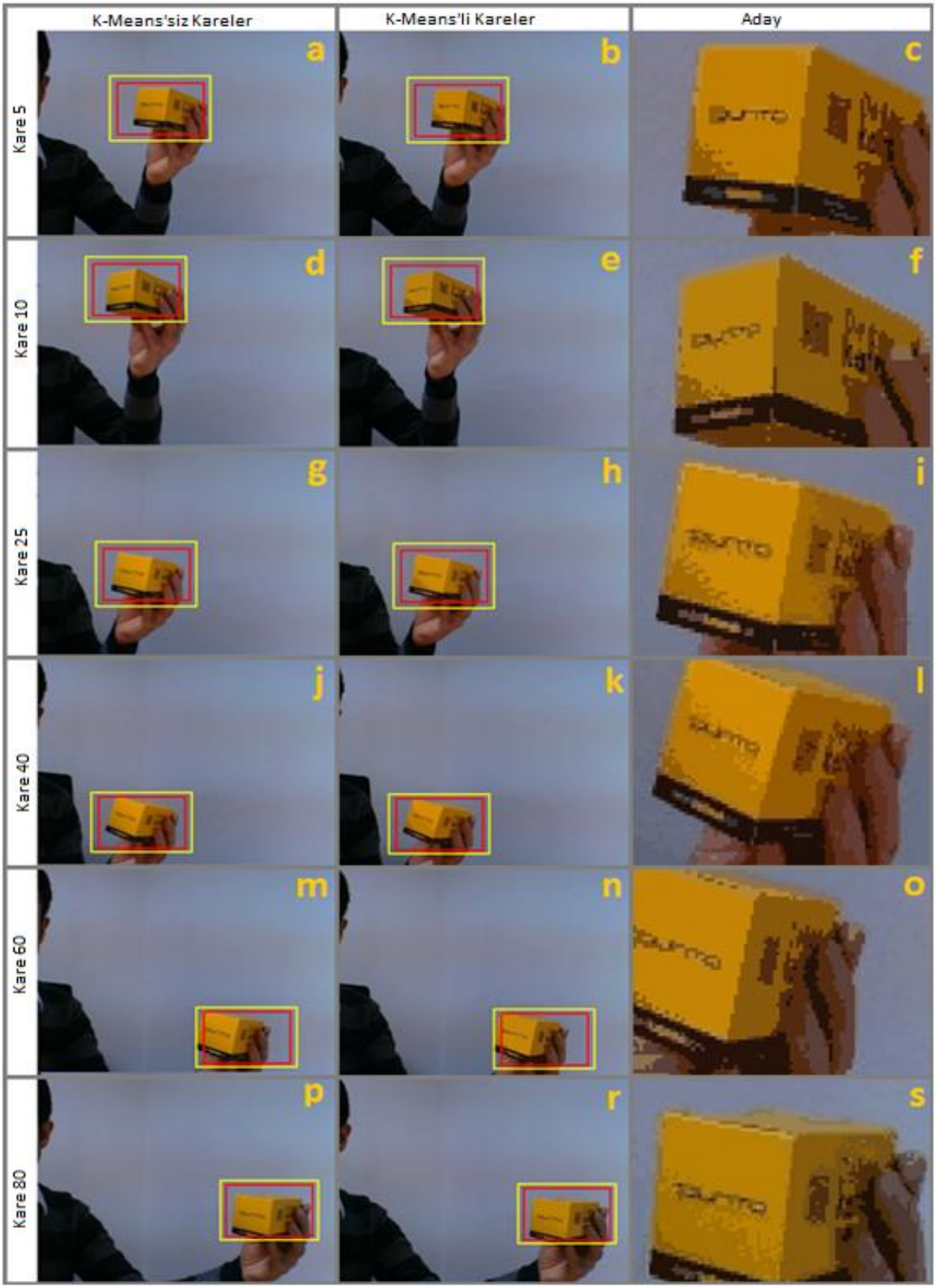
Bölütleme yöntemiyle birlikte kullanılması önerilen histogram geri projeksiyon algoritması, ilgilenilen aday görüntünün koordinat olasılığını hesaplamaktadır. Hedefin yönü, uzunluk ve genişlik dağılım olasılığı ile açı yönelimi hakkında daha fazla bilgi vermesi açısından histogram geri projeksiyon yöntemi, takip algoritmasının daha az hatalı çalışmasını sağlamaktadır.

K-Means bölütleme daha güvenilir bir işleme için gereklidir ve görüntü işleme uygulamalarında sıklıkla kullanılmaktadır [25, 26]. K-Means yöntemi ile Öklit mesafesine dayalı hesaplanan kümeler elde edilir [27]. K-Means uygulanmış ve uygulanmamış algoritma sonuçları deneylerle karşılaştırılmıştır.

Bu çalışmada takip hızını artıran ve hataları azaltan Bhattacharyya katsayısı yöntemi, adayı görüntü ve 16 bölütlü hedef görüntü histogramını karşılaştırmak için kullanılmıştır.

Parametrik olmayan ve tekrarlamalı özelliği sayesinde Mean-shift algoritması hedefin merkezindeki temel değişimleri hesaplamak için tercih edilmiştir. Ayrıca önceden belirlenmiş dikdörtgen tarama bölgesiyle yumuşatılmış ve etkili bir takip sağlar [28].

Şekil 6.2, K-Means uygulanmış ve uygulanmamış sonuçlar arasındaki farkları göstermektedir. 5, 10, 25, 40, 60 ve 80. Kareler için elde edilen aday görüntüler ve bu görüntülerin bütün resim içindeki görüntüleri Şekil 6.2'de görülmektedir. Bu çalışma sonucunda, geliştirilmiş K-Means yönteminin nesne takip hatalarını azalttığı izlenmiştir. K-Means ile elde edilen Bhattacharyya katsayısındaki iyileşmeler, Çizelge 6.2'de hesaplanmıştır. Çizelgede görüldüğü üzere 5. karede K-Means uygulandığı durumda hesaplanan katsayı 0,999965 olurken uygulanmadığı durumda 0,966735 olmaktadır. Video dizisindeki bütün karelerin Bhattacharyya katsayıları incelendiğinde iyileşmeler açıkça görülmektedir. Dizideki kare numaraları arttıkça nesne hızı değiştiği için benzerlik oranı değişmekte ve Bhattacharyya katsayısı azalmaktadır. Takip edilen nesnenin hızı yükselmesine rağmen, önerilen yöntem ile gerçek zamanlı bir takip gerçekleştirilebilir. Gerekli donanımsal iyileştirmeler yapıldığı takdirde, gözetleme, sürücü yardım sistemleri, akıllı bina sistemleri ve medikal sistemler gibi gerçek zamanlı uygulamalarda önerilen yöntem çok yaygın olarak kullanılabilir.



Şekil 6.2. K-Means uygulanmış ve uygulanmamış nesne takip sonuçları (*a, d, g, j, m, p görüntüleri K-Means uygulanmamış görüntülerdir. b, e, h, k, n, r görüntüleri K-Means uygulanmış görüntülerdir. c, f, i, l, o, s görüntüleri K-Means uygulanmış aday görüntü parçalarıdır)*

Çizelge 6.2. K-Means uygulanmış ve uygulanmamış sonuçlar için Bhattacharyya katsayıları

	K-Means Uygulanmamış	K-Means Uygulanmış
Kare 5	0,966735	0,999965
Kare 10	0,739096	0,957452
Kare 25	0,806795	0,916243
Kare 40	0,764425	0,853334
Kare 60	0,913240	0,971650
Kare 80	0,799920	0,895239

Bhattacharyya katsayısı yöntemi başarılı bir takip uygulaması için geliştirilmeye açık bir yöntemdir. Bu çalışma sonucu donanım özelliklerinin sınırları ölçüsünde gerçek zamanlı uygulamalarında kullanışlı bir teknik olan katsayı yöntemi üzerine iyileştirmeler ve farklı teknikler uygulanmıştır. Sonuçlar ışığında K-Means bölütleme ve ön filtreleme çalışmalarıyla daha etkili bir takip algoritması elde edilebilir.

KAYNAKLAR

1. Birchfield, S. (1998)**.** “Elliptical Head Tracking Using Intensity Gradients and Color Histograms”, ***IEEE Conference on Computer Vision and Pattern Recognition***, 232-237.

2. Khan, S. and Shah, M. (2000). “Tracking People in Presence of Occlusion”, ***Asian Conference on Computer Vision***, 263-266.

3. Wren, C., Azarbayejani, A., Darrell, T. and Pentland, A. (1997). “P_nder: Real-time tracking of the human body”, ***PAMI***, 19-7.

4. Toyama, K., Krumm, J., Brumitt, B. and Meyers, B. (1999). “Wall_ower: Principles of background maintenance”, ***ICCV***.

5. Heath, M., Sarkar, S., Sanoki,T., and Bowyer, K. (1998). "Comparison of Edge Detectors: A Methodology and Initial Study", ***Computer Vision and Image Understanding***, 69(1),38-54.

6. Stauffer, C. and Grimson. (2000). “Learning patterns of activity using real time tracking”, ***IEEE Trans. Pattern Analysis and Machine Intelligence,*** 22(8), 747-767.

7. Flail, E. L. (1979). “Computer Image Processing and Recognition”, ***Academic Press***, New York.

8. Ekin, A., Tekalp, A. M. and Mehrotra, R. (2001). ”Automatic extraction of low-level object motion descriptors***”, Proceedings International Conference on Image Processing, Thessaloniki***, 2, 633 -636.

9. Beymer, D. and Konolige, K. (1999). “Real-time tracking of multiple people using continuous detection” ***IEEE International Conference on Computer Vision (ICCV) Frame-Rate Workshop***.

10. Zaibi, R., Yardimci, Y. and Cetin, A. E. (2000). ”Small Moving Object Detection In Video Sequences”, ***Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-2000),*** 2071-2074.

11. Christopher, R. W., Azarbayejani, A., Trevor, D. and Alex, P. (1997). “Pfinder: Real-time tracking of the human body,” ***IEEE Transactions on Pattern Analysis and Machine Intelligence***, 19(7), 780-785.

12. Khan, Z. H., Gu, I. Y. and Backhouse, A. G. (2011). “A robust particle filter-based method for tracking single visual object through complex scenes using dynamical object shape and appearance similarity”, ***Journal of Signal Processing Systems***, 65, 63-79.

13. Huang, J. and Li, Z. (2011). “Automatic detection of object of ınterest and tracking in active video”, ***Journal of Signal Processing Systems***, 65, 49-62.

14. Li, P. (2011). “An efficient particle filter–based tracking method using graphics processing unit (gpu)”, ***Journal of Signal Processing Systems***, 68, 317-332.

15. Avidan, S. (2005). “Ensemble tracking”, ***Computer vision and pattern recognition,*** 494-501.

16. Enkelmann, W. (2001). “Video-based driver assistance: From basic functions to applications”, ***International Journal of Computer Vision***, 45(3), 201-221.

17. Intille, S. S., Davis, J. W. and Bobick, A. F. (1997). “Real-time closed-world tracking”, ***Computer vision and pattern recognition***, 697-703.

18. Liu, D., Hua, G. and Chen, T. (2008). “Videocut: Removing irrelevant frames by discovering the object of interest”, ***European conference on computer vision***, 1, 441-453.

19. Simakov, D., Caspi, Y., Shechtman, E. and Irani, M. (2008). “Summarizing visual data using bidirectional similarity”, ***Computer vision and pattern recognition***, 1-8.

20. Lautissier, J., Legrand, L., Lalande, A., Walker, P. and Brunotte, F. (2003). “Object tracking in medical imaging using a 2D active mesh system”, ***the 25th Annual International Conference of the IEEE BMBS***, 739-742.

21. Cheng, T.Y. and Herman, C. (2014). “Motion tracking in infrared imaging for quantitative medical diagnostic applications”, ***Infrared Physics & Technology***, 62, 70-80.

22. Martínez, B. A. and Jiménez, J. J. (2011). “Tracking by means of geodesic region models applied to multidimensional and complex medical images”, ***Computer Vision and Image Understanding***, 115(8), 1083-1098.

23. Singhal, N., Lee, Y. Y., Kim, C. S. and Lee, S. U. (2009). “Robust image watermarking using local Zernike moments” ***J. Vis. Commun. Image R.***, 20, 408-419.

24. Quine, B. M., Tarasyuk, V., Mebrahtu, H. and Hornsey, R. (2007). “Determining star-image location: A new sub-pixel interpolation technique to process image centroids” ***Computer Physics Communications***, 177, 700-706.

25. Li, J., Li, X. and Tao, D. (2005). “KPCA for semantic object extraction in images”, ***Pattern Recognition,*** 41, 3244-3250.

26. Cao, J., Wu, Z., Wu, J. and Liu, W. (2013). “Towards information-theoretic k-means clustering for image indexing”, ***Signal Processing***, 93, 2026-2037.

27. Lin, C., Chen, C., Lee, H. and Liao, J. (2014). “Fast K-means algorithm based on a level histogram for image retrieval”, ***Expert Systems with Applications***, 41, 3276-3283.

28. Ghassabeh, Y. A., Linder, T. and Takahara, G. (2013). “On some convergence properties of the subspace constrained mean shift”, ***Pattern Recognition***, 46, 3140-3147.

29. Klette, R. and Zunic, J. (2012). “ADR shape descriptor – Distance between shape centroids versus shape diameter”, ***Computer Vision and Image Understanding***, 116, 690-697.

30. Hu, M. (1962). “Visual pattern recognition by moment invariants”, ***IRE Trans. Inform. Theory 8*** 179-187.

31. Horn, B. K. P. (1986). “Robot vision”, ***MIT Press***.

32. Finlayson, G., Hordley, S., Schaefer, G. and Tian, G. Y. (2005). “Illuminant and device invariant colour using histogram equalization”. ***Pattern Recognition***, 38, 179-190.

33. Sim, K. S., Tso, C. P. and Tan, Y. Y. (2007). “Recursive sub-image histogram equalization applied to gray scale images”, ***Pattern Recognition Letters***, 28, 1209-1221.

34. Humied, I. A., Abou-Chadi, F. E. Z. and Rashad, M. Z. (2012). “A new combined technique for automatic contrast enhancement of digital images”, ***Egyptian Informatics Journal***, 13, 27-37.

35. Martinez Canada, P., Morillas, C., Urena, R., Gómez López, J. M. and Pelayo, F.J. (2013). “Embedded system for contrast enhancement in low-vision”, ***Journal of Systems Architecture***. 59(1), 30-38.

39. Bhattacharyya, A. (1943). “On a measure of divergence between two statistical populations defined by their probability distributions”, ***Bulletin of the Calcutta Mathematical Society***, 35, 99–109.

40. Yao, A., Wang, G., Lin, X. and Chai, X. (2010). “An incremental Bhattacharyya dissimilarity measure for particle filtering”, ***Pattern Recognition***, 43(4), 1244-1256.

41. Khalid, M. S., Ilyas, M. U., Sarfanaz, M. S. and Ajaz, M. A. (2006). “Bhattacharyya Coefficient in Correlation of Gray-Scale Objects”, ***Journal of Multimedia***, 1(1), 56-61.

42. Lu, N. and Freng, Z. (2008). “Mathematical model of blob matching and modified Bhattacharyya coefficient”, ***Image and Vision Computing***, 26(10), 1421-1434.

43. Ayed, I. B., Chen, H., Punithakumar, K., Ross, I. and Li, S. (2012). “Max-flow segmentation of the left ventricle by recovering subject-specific distributions via a bound of the Bhattacharyya measure”, ***Medical Image Analysis***, 16(1), 87-100.

45. Wang, F., Yu, S. and Yang, J. (2010). “Robust and efficient fragments-based tracking using mean shift”, ***International Journal of Electronics and Communications***, 64, 614-623.

46. Li, S. X., Chang, H. X. and Zhu, C. F. (2010). “Adaptive pyramid mean shift for global real-time visual tracking”, ***Image and Vision Computing***, 28, 424-437.

47. Leichter, I., Lindenbaum, M. and Rivlin, E. (2010). “Mean Shift tracking with multiple reference color histograms”, ***Computer Vision and Image Understanding***, 114, 400-408.

48. Hidayetullah, P. and Konik, H. (2011). “Camshift improvement on multi-hue object and multi-object tracking”, ***IEEE European workshop on visual information processing***, 143-148.

EKLER

EK-1. K-Means Algoritması Kaynak Kodları

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;
using System.Collections;
using System.Collections.Specialized;

using ImageProcessor;

namespace CVMeanshift
{
    public unsafe class KMeans
    {
        public class Distance
        {
            public Distance(float d) { _d = d; }
            public float Measure
            {
                get { return _d; }
                set { _d = value; }
            }
            private float _d;
        }

        public class Cluster
        {
            public Cluster(float R, float G, float B)
            {
                _centroid1 = R;
                _centroid2 = G;
                _centroid3 = B;
            }
            public float CentroidR
            {
                get { return _centroid1; }
                set { _centroid1 = value; }
            }
            public float CentroidG
            {
                get { return _centroid2; }
                set { _centroid2 = value; }
            }
            public float CentroidB
            {
                get { return _centroid3; }
                set { _centroid3 = value; }
            }
            private float _centroid1;
```

EK-1. (devam) K-Means Algoritması Kaynak Kodları

```
        private float _centroid2;
        private float _centroid3;
    }

    public KMeans(Bitmap bmp, int numCluster, Colour.Types model)
    {
        _image = (Bitmap)bmp.Clone();
        _processedImage = (Bitmap)bmp.Clone();
        _model = model;

        _previousCluster = new Dictionary<string, Cluster>();
        _currentCluster = new Dictionary<string, Cluster>();
        FindTopXColours(numCluster); //find top X colours in the image
        //create clusters for top X colours
        for (int i = 0; i < _topColours.Length; i++)
        {
            PixelData pd = Colour.GetPixelData(_topColours[i].R, _topColours[i].G,
_topColours[i].B, model);

            _previousCluster.Add(_topColours[i].Name, new Cluster(pd.Ch1,
pd.Ch2, pd.Ch3));
            _currentCluster.Add(_topColours[i].Name, new Cluster(pd.Ch1, pd.Ch2,
pd.Ch3));
        }
    }

    public Bitmap ProcessedImage { get { return _processedImage; } }

    public bool Converged { get { return _converged; } }

    public void Iterate()
    {
        _colourClusterAllocation = new Hashtable(); //for keeping track of colour<-
>cluster allocation
        _pixelDataClusterAllocation = new Hashtable();
        _clusterColours = new Hashtable();

        UnsafeBitmap fastBitmap = new UnsafeBitmap(_image);
        fastBitmap.LockBitmap();
        Point size = fastBitmap.Size;
        BGRA* pPixel;

        for (int y = 0; y < size.Y; y++)
        {
            pPixel = fastBitmap[0, y];
            for (int x = 0; x < size.X; x++)
            {
                PixelData pd = Colour.GetPixelData(pPixel, _model);
```

EK-1. (devam) K-Means Algoritması Kaynak Kodları

```
                AllocateToCluster(pd);

                //increment the pointer
                pPixel++;
            }
        }
        fastBitmap.UnlockBitmap();

        CalculateClusterCentroids();

        _processedImage = (Bitmap)_image.Clone();

        //segment the image based on the cluster
        fastBitmap = new UnsafeBitmap(_processedImage);
        fastBitmap.LockBitmap();
        for (int y = 0; y < size.Y; y++)
        {
            pPixel = fastBitmap[0, y];
            for (int x = 0; x < size.X; x++)
            {
                PixelData pd = Colour.GetPixelData(pPixel, _model);
                Color newClr = (Color)_clusterColours[pd.Name];

                pPixel->red = newClr.R;
                pPixel->green = newClr.G;
                pPixel->blue = newClr.B;

               //increment the pointer
                pPixel++;
            }
        }

        fastBitmap.UnlockBitmap();

        CheckConvergence();

    }

    private void CheckConvergence()
    {
        //if current and previous cluster centroids are the same then converged
        bool match = true;
        foreach (KeyValuePair<string, Cluster> cluster in _currentCluster)
        {
            if (((int)cluster.Value.CentroidR !=
(int)_previousCluster[cluster.Key].CentroidR)
                && ((int)cluster.Value.CentroidG !=
(int)_previousCluster[cluster.Key].CentroidG)
```

EK-1. (devam) K-Means Algoritması Kaynak Kodları

```
                && ((int)cluster.Value.CentroidB !=
(int)_previousCluster[cluster.Key].CentroidB))
            {
                match = false;
                break;
            }
        }
        if (!match)
        {
            foreach (KeyValuePair<string, Cluster> cluster in _currentCluster)
            {
                _previousCluster[cluster.Key].CentroidR=cluster.Value.CentroidR;
                    _previousCluster[cluster.Key].CentroidG=cluster.Value.CentroidG;
                    _previousCluster[cluster.Key].CentroidB = cluster.Value.CentroidB;
            }
        }
        _converged = match;
    }

    private void CalculateClusterCentroids()
    {
        foreach (KeyValuePair<string, Cluster> cluster in _currentCluster)
        {
            List<PixelData> clrList =
(List<PixelData>)_pixelDataClusterAllocation[cluster.Key];
            float cR = 0;
            float cG = 0;
            float cB = 0;
            foreach (PixelData clr in clrList)
            {
                cR += clr.Ch1;
                cG += clr.Ch2;
                cB += clr.Ch3;

                if (!_clusterColours.ContainsKey(clr.Name))
                {
                    _clusterColours.Add(clr.Name,
Color.FromArgb((int)cluster.Value.CentroidR, (int)cluster.Value.CentroidG,
(int)cluster.Value.CentroidB));
                }
            }
            float count = clrList.Count + 1; //total of colours plus 1 for the existing
centroid
            cluster.Value.CentroidR = (cluster.Value.CentroidR + cR) / count;
//average to find new centroid
            cluster.Value.CentroidG = (cluster.Value.CentroidG + cG) / count;
            cluster.Value.CentroidB = (cluster.Value.CentroidB + cB) / count;
        }
```

EK-1. (devam) K-Means Algoritması Kaynak Kodları

```
    }

    private void AllocateToCluster(PixelData pd)
    {
        //find distance of this colour from each cluster centroid
        Dictionary<string, Distance> distances = new Dictionary<string,
Distance>();

        foreach (KeyValuePair<string, Cluster> c in _currentCluster)
        {
            float d = (float)Math.Sqrt(
                (double)Math.Pow((c.Value.CentroidR - pd.Ch1), 2) +
                (double)Math.Pow((c.Value.CentroidG - pd.Ch2), 2) +
                (double)Math.Pow((c.Value.CentroidB - pd.Ch3), 2)
            );
            distances.Add(c.Key, new Distance(d));
        }

        //allocate this colour to the closest cluster based on distance
        List<KeyValuePair<string, Distance>> list = new List<KeyValuePair<string,
Distance>>();
        list.AddRange(distances);

        list.Sort(delegate(KeyValuePair<string, Distance> kvp1,
KeyValuePair<string, Distance> kvp2)

        { return Comparer<float>.Default.Compare(kvp1.Value.Measure,
kvp2.Value.Measure); });

        //assign to closest cluster
        if (_pixelDataClusterAllocation.ContainsKey(list[0].Key))
        {
            ((List<PixelData>)_pixelDataClusterAllocation[list[0].Key]).Add(pd);
        }
        else
        {
            List<PixelData> clrList = new List<PixelData>();
            clrList.Add(pd);
            _pixelDataClusterAllocation.Add(list[0].Key, clrList);
        }
    }

    private void FindTopXColours(int numColours)
    {
        Dictionary<string, ColourCount> colours = new Dictionary<string,
ColourCount>();
        UnsafeBitmap fastBitmap = new UnsafeBitmap(_image);
        fastBitmap.LockBitmap();
```

EK-1. (devam) K-Means Algoritması Kaynak Kodları

```
        Point size = fastBitmap.Size;
        BGRA* pPixel;

        for (int y = 0; y < size.Y; y++)
        {
            pPixel = fastBitmap[0, y];
            for (int x = 0; x < size.X; x++)
            {
                //get the bin index for the current pixel colour
                Color clr = Color.FromArgb(pPixel->red, pPixel->green, pPixel->blue);

                if (colours.ContainsKey(clr.Name))
                {
                    ((ColourCount)colours[clr.Name]).Count++;
                }
                else
                    colours.Add(clr.Name, new ColourCount(clr, 1));

                //increment the pointer
                pPixel++;
            }
        }

        fastBitmap.UnlockBitmap();

        //instantiate using actual colours found - which might be less than
numColours
        if (colours.Count < numColours)
            numColours = colours.Count;

        _topColours = new Color[numColours];

        List<KeyValuePair<string, ColourCount>> summaryList = new
List<KeyValuePair<string, ColourCount>>();
        summaryList.AddRange(colours);

        summaryList.Sort(delegate(KeyValuePair<string, ColourCount> kvp1,
KeyValuePair<string, ColourCount> kvp2)
        { return Comparer<int>.Default.Compare(kvp2.Value.Count,
kvp1.Value.Count); });


        for (int i = 0; i < _topColours.Length; i++)
        {
            _topColours[i] = Color.FromArgb(summaryList[i].Value.Colour.R,
summaryList[i].Value.Colour.G, summaryList[i].Value.Colour.B);
        }
    }
```

EK-1. (devam) K-Means Algoritması Kaynak Kodları

```
        private Color[] _topColours;
        private Colour.Types _model;
        private Dictionary<string, Cluster> _previousCluster;
        private Dictionary<string, Cluster> _currentCluster;
        private Hashtable _colourClusterAllocation;
        private Hashtable _pixelDataClusterAllocation;
        private Hashtable _clusterColours;
        private bool _converged = false;
        private Bitmap _image;
        private Bitmap _processedImage;
    }
}
```

EK-2. Bhattacharyya Katsayısı Algoritması Kaynak Kodları

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace CVIntro
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        string _path = "";
        Processor _imgProc = new Processor();
        Histogram _hist1 = null;
        Histogram _hist2 = null;

        private void Form1_Load(object sender, EventArgs e)
        {
            _path = "..\\..\\";
            LoadImages();
        }

        //Load BMP images from a pre-defined location
        private void LoadImages()
        {
            //start clean
            listView1.Clear();
            listView1.Refresh();
            imageList1.Images.Clear();

            //setup the list
            imageList1.ImageSize = new Size(50, 50);
            imageList1.ColorDepth = ColorDepth.Depth32Bit;
            imageList1.TransparentColor = Color.White;

            //get the location
            string[] images = Directory.GetFiles(_path, "*.bmp");

            //load the list image list
            foreach (string img in images)
            {
```

EK-2. (devam) Bhattacharyya Katsayısı Algoritması Kaynak Kodları

```
            System.Drawing.Image img1 = System.Drawing.Image.FromFile(img);

imageList1.Images.Add(GetThumbnailImage(imageList1.ImageSize.Width, img1));
        }

        //populate the list view
        int j = 0;
        foreach (string img in images)
        {
            FileInfo fi = new FileInfo(img);
            this.listView1.Items.Add(fi.Name);
            this.listView1.Items[j].ImageIndex = j;
            j++;
        }

        //display the list
        this.listView1.View = View.LargeIcon;
        this.listView1.LargeImageList = imageList1;
    }

    private Image GetThumbnailImage(int width, Image img)
    {
        Image thumb = new Bitmap(width, width);
        Image tmp = null;

        //If the original image is small than the Thumbnail size, just draw in the
center
        if (img.Width < width && img.Height < width)
        {
            using (Graphics g = Graphics.FromImage(thumb))
            {
                int xoffset = (int)((width - img.Width) / 2);
                int yoffset = (int)((width - img.Height) / 2);
                g.DrawImage(img, xoffset, yoffset, img.Width, img.Height);
            }
        }

        else //Otherwise we have to get the thumbnail for drawing
        {
            Image.GetThumbnailImageAbort myCallback = new
Image.GetThumbnailImageAbort(ThumbnailCallback);

            if (img.Width == img.Height)
            {
                thumb = img.GetThumbnailImage(width, width, myCallback,
IntPtr.Zero);
            }
            else
```

EK-2. (devam) Bhattacharyya Katsayısı Algoritması Kaynak Kodları

```
            {
                int k = 0;
                int xoffset = 0;
                int yoffset = 0;

                if (img.Width < img.Height)
                {
                    k = (int)(width * img.Width / img.Height);
                    tmp = img.GetThumbnailImage(k, width, myCallback, IntPtr.Zero);
                    xoffset = (int)((width - k) / 2);
                }

                if (img.Width > img.Height)
                {
                    k = (int)(width * img.Height / img.Width);
                    tmp = img.GetThumbnailImage(width, k, myCallback, IntPtr.Zero);
                    yoffset = (int)((width - k) / 2);
                }

                using (Graphics g = Graphics.FromImage(thumb))
                {
                    g.DrawImage(tmp, xoffset, yoffset, tmp.Width, tmp.Height);
                }
            }
        }

        using (Graphics g = Graphics.FromImage(thumb))
        {
            g.DrawRectangle(Pens.Green, 0, 0, thumb.Width - 1, thumb.Height - 1);
        }

        return thumb;
    }

    private bool ThumbnailCallback()
    {
        return true;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (listView1.SelectedItems.Count > 0)
        {
            pictureBox1.Image = Image.FromFile(_path + "\\" +
listView1.SelectedItems[0].Text);
            DisplayHistogram(ref _hist1, pictureBox1, pictureBox3);
        }
    }
```

EK-2. (devam) Bhattacharyya Katsayısı Algoritması Kaynak Kodları

```
        private void button2_Click(object sender, EventArgs e)
        {
            if (listView1.SelectedItems.Count > 0)
            {
                pictureBox2.Image = Image.FromFile(_path + "\\" +
listView1.SelectedItems[0].Text);
                DisplayHistogram(ref _hist2, pictureBox2, pictureBox4);
            }
        }

        private void DisplayHistogram(ref Histogram hist, PictureBox picBox,
PictureBox histBox)
        {
            Image bmp = picBox.Image;
            if (bmp != null)
            {
                hist = CreateHistogram((Bitmap)bmp);
                histBox.Refresh(); //force paint event
            }
        }

        private int Bin1
        {
            get { return Convert.ToInt32(textBox1.Text); }
        }

        private int Bin2
        {
            get { return Convert.ToInt32(textBox2.Text); }
        }

        private int Bin3
        {
            get { return Convert.ToInt32(textBox3.Text); }
        }

        private Histogram CreateHistogram(Bitmap img)
        {
            return _imgProc.Create1DHistogram(img, Bin1, Bin2, Bin3);
        }

        private void DisplayModel(Histogram model, int height, int width,
PaintEventArgs e, Color color, int binRange)
        {
            if (model != null)
            {
                Pen p = new Pen(color, 1);
                SolidBrush b = new SolidBrush(color);
```

EK-2. (devam) Bhattacharyya Katsayısı Algoritması Kaynak Kodları

```
            float[] copy = new float[model.Data.Length];
            Array.Copy(model.Data, copy, model.Data.Length);
            Array.Sort(copy);
            float max = copy[copy.Length - 1];
            float scale = height / max;
            if (float.IsNaN(scale))
                scale = 1;
            //Approximation: divide by total bins and remove 4 from width to account
for picbox border
            float w = (float)(width - 4) / (float)(model.Data.Length);
            for (int count = 0; count < model.Data.Length; count++)
            {
                if (model.Data[count] > 0)
                {
                    e.Graphics.DrawRectangle(p, (float)count * w, (float)height -
(model.Data[count] * scale), w, model.Data[count] * scale);
                    e.Graphics.FillRectangle(b, (float)count * w, (float)height -
(model.Data[count] * scale), w, model.Data[count] * scale);

                }
            }
        }
    }

    private void pictureBox3_Paint(object sender, PaintEventArgs e)
    {
        DisplayModel(_hist1, pictureBox3.Height, pictureBox3.Width, e, Color.Red,
Bin1 * Bin2 * Bin3);
    }

    private void pictureBox4_Paint(object sender, PaintEventArgs e)
    {
        DisplayModel(_hist2, pictureBox4.Height, pictureBox4.Width, e,
Color.Blue, Bin1 * Bin2 * Bin3);
    }

    private float Bhattacharyya(Histogram hist1, Histogram hist2)
    {
        double coeff = 0;
        float t = 0;
        for (int i = 0; i < hist1.Data.Length; i++)
        {
            coeff += Math.Sqrt((double)hist1.Data[i] * (double)hist2.Data[i]);
            t += hist1.Data[i];
        }
        return (float)coeff;
    }
```

EK-2. (devam) Bhattacharyya Katsayısı Algoritması Kaynak Kodları

```
        private void button3_Click(object sender, EventArgs e)
        {
            if ((_hist1 != null) && (_hist2 != null))
            {
                MessageBox.Show("Bhattacharyya Coefficient: " +
Bhattacharyya(_hist1, _hist2).ToString());
            }
            else
            {
                MessageBox.Show("First select two images");
            }
        }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {
            DisplayHistogram(ref _hist1, pictureBox1, pictureBox3);
            DisplayHistogram(ref _hist2, pictureBox2, pictureBox4);
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
            DisplayHistogram(ref _hist1, pictureBox1, pictureBox3);
            DisplayHistogram(ref _hist2, pictureBox2, pictureBox4);
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            DisplayHistogram(ref _hist1, pictureBox1, pictureBox3);
            DisplayHistogram(ref _hist2, pictureBox2, pictureBox4);
        }
    }
}
```

EK-3. Nesne Takip Algoritması Kaynak Kodları

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Threading;

using ImageProcessor;
using Tracker;
using UIGraphic;
using Statistic;

namespace tracking
{
   public partial class Form1 : Form
   {
      KMeans _kMeans;
      int iter_say = 0;


      public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         _path = "..\\..\\";
         //_path = Application.StartupPath.ToString();
         //Load images
         _isTracking = false;
         webCamCapture1.ImageCaptured += new
WebCam_Capture.WebCamCapture.WebCamEventHandler(webCamCapture1_I
mageCaptured);
      }

      //create the target selector
      private void CreateSelector()
      {
         if (_selector == null)
         {
            _selector = new Selector(pictureBox1.Width, pictureBox1.Height);
            _selector.TargetROISelected = new
Selector.TargetSelected(this.CreateTargetModel);
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
            //wire preview window events to selector's events
            //picPreview.MouseDown += new
MouseEventHandler(_selector.OnMouseDown);
            //picPreview.MouseMove += new
MouseEventHandler(_selector.OnMouseMove);
            //picPreview.MouseUp += new
MouseEventHandler(_selector.OnMouseUp);

            pictureBox1.MouseDown += new
MouseEventHandler(_selector.OnMouseDown);
            pictureBox1.MouseMove += new
MouseEventHandler(_selector.OnMouseMove);
            pictureBox1.MouseUp += new
MouseEventHandler(_selector.OnMouseUp);
        }
    }

    //create the model when the target is slected
    private void CreateTargetModel()
    {
        //create the model
        Tracker.CreateTargetModel((Bitmap)pictureBox1.Image, Bin1, Bin2, Bin3,
_selector.TargetWindow, _selector.SearchWindow);
        //show target histogram
        DisplayHistogram(ref _histTarget, pictureBox1, picTargetHist);
    }

    private void DisplayHistogram(ref Histogram hist, PictureBox picBox,
PictureBox histBox)
    {
        Image bmp = picBox.Image;
        if (bmp != null)
        {
            hist = CreateHistogram((Bitmap)bmp);
            histBox.Refresh(); //force paint event
        }
    }

    private int Bin1
    {
        get { return Convert.ToInt32(textBox1.Text); }
    }
    private int Bin2
    {
        get { return Convert.ToInt32(textBox2.Text); }
    }

    private int Bin3
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
        {
            get { return Convert.ToInt32(textBox3.Text); }
        }

        //creates the histograms
        private Histogram CreateHistogram(Bitmap img)
        {
            return _imgProc.Create1DHistogram(img, Bin1, Bin2, Bin3);
        }

        //Displays the histogram
        private void DisplayModel(Histogram model, int height, int width,
PaintEventArgs e, Color color, int binRange)
        {
            if (model != null)
            {
                Pen p = new Pen(color, 1);
                SolidBrush b = new SolidBrush(color);
                float[] copy = new float[model.Data.Length];
                Array.Copy(model.Data, copy, model.Data.Length);
                Array.Sort(copy);
                float max = copy[copy.Length - 1];
                float scale = height / max;
                if (float.IsNaN(scale))
                    scale = 1;
                //Approximation: divide by total bins and remove 4 from width to account
for picbox border
                float w = (float)(width - 4) / (float)(model.Data.Length);
                for (int count = 0; count < model.Data.Length; count++)
                {
                    if (model.Data[count] > 0)
                    {
                        e.Graphics.DrawRectangle(p, (float)count * w, (float)height -
(model.Data[count] * scale), w, model.Data[count] * scale);
                        e.Graphics.FillRectangle(b, (float)count * w, (float)height -
(model.Data[count] * scale), w, model.Data[count] * scale);

                    }
                }
            }
        }

        private void DisplayInformation(Dictionary<string, string> information)
        {
            lblInfo.Text = "";
            foreach (KeyValuePair<string, string> kv in information)
            {
                lblInfo.Text += kv.Key + ": " + kv.Value + Environment.NewLine;
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
        }
        lblInfo.Refresh();
    }

    //Track the object
    private void TrackObject(Bitmap bmp)
    {
        //run track on the tracker to find centroid
        Window roi;
        Window searchRoi;
        Dictionary<string, string> information;

        Tracker.Track(bmp, out roi, out searchRoi, out information);

        Bitmap b = Tracker.ProcessedImage;

        //Draw the centroid if valid
        if (roi.CentreX > 0 && roi.CentreY > 0)
        {
            //Display Info
            DisplayInformation(information);
        }
        //picPreview.Image = b;
        //picPreview.Refresh();

        pictureBox2.Image = b;

        //Display the candidate histogram
        if (_tracker.GetType().Name != "Centroid")
            //DisplayHistogram(ref _histCandidate, picPreview, picCandidateHist);
        DisplayHistogram(ref _histCandidate, pictureBox2 , picCandidateHist);
        //Drawer.DrawWindow(picPreview, roi, Color.Red);
        //Drawer.DrawWindow(picPreview, searchRoi, Color.Yellow);
    }
    private ITracker Tracker
    {
        get
        {
            if (_tracker == null)
                CreateTracker();

            return _tracker;
        }
    }

    private void CreateTracker()
    {
        if (radTrackerCentroid.Checked)
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
            _tracker = new TrackerFactory(TrackerType.CENTROID).Tracker;
        else if (radTrackerMS.Checked)
            _tracker = new TrackerFactory(TrackerType.MEANSHIFT).Tracker;
        else if (radTrackerMSK.Checked)
            _tracker = new
TrackerFactory(TrackerType.MEANSHIFTKERNEL).Tracker;
        else if (radTrackerMSKBG.Checked)
            _tracker = new
TrackerFactory(TrackerType.MEANSHIFTKERNELBG).Tracker;
    }

    #region "CONTROL EVENTS"
    private void picTargetHist_Paint(object sender, PaintEventArgs e)
    {
        DisplayModel(_histTarget, picTargetHist.Height, picTargetHist.Width, e,
Color.Red, Bin1 * Bin2 * Bin3);
    }

    private void picCandidateHist_Paint(object sender, PaintEventArgs e)
    {
        DisplayModel(_histCandidate, picCandidateHist.Height,
picCandidateHist.Width, e, Color.Blue, Bin1 * Bin2 * Bin3);
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        if (_count < _trackSequenceCount)
        {
            //set the next image

            Bitmap bmp = (Bitmap)Image.FromFile(_path + "\\" + _count +
_imageFileExtension);
            TrackObject(bmp); //do tracking
            _count++; //move to next image
        }
        else
        {
            //btnTrack.Text = "Start Tracking";
            _count = 0;
            timer1.Enabled = false;
            timer1.Stop();
        }
    }

    private void btnLoadSequence_Click(object sender, EventArgs e)
    {
        folderBrowserDialog1.SelectedPath = Application.StartupPath;  //
"C:\\CodeProject\\AviSequence";
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
        if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
        {

            Cursor.Current = Cursors.WaitCursor;

            //_isTracking = true;
            _path = folderBrowserDialog1.SelectedPath;

            //picPreview.Image = null;
            //picPreview.Refresh();
            _histTarget = null;
            _histCandidate = null;
            picTargetHist.Refresh();
            picCandidateHist.Refresh();
            lblInfo.Text = "";
            _count = 0;
            string imagePath = "";
            Bitmap b = null;

            string[] files = Directory.GetFiles(_path, "*.jpg");

            if (files.Length > 0)
            {
                imagePath = files[0];
                b = (Bitmap)Image.FromFile(imagePath);
            }
            else
            {
                files = Directory.GetFiles(_path, "*.bmp");
                imagePath = files[0];
                b = (Bitmap)Image.FromFile(imagePath);
                _imageFileExtension = ".bmp";
            }
            //picPreview.Image = b;
            FileInfo fi = new FileInfo(imagePath);
            _imageFileExtension = fi.Extension;
            //picPreview.Refresh();

            _trackSequenceCount = files.Length;

            CreateSelector();

            Cursor.Current = Cursors.Default;
        }
    }
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
        private void btnTrack_Click(object sender, EventArgs e)
        {
            ////start the frame grabber
            //if (btnTrack.Text == "Start Tracking")
            //{
            //    _isTracking = true;
            //    //btnTrack.Text = "Stop Tracking";

            //    timer1.Enabled = true;
            //    timer1.Start();
            //}
            //else if (btnTrack.Text == "Stop Tracking")
            //{
            //    timer1.Stop();
            //    timer1.Enabled = false;
            //    btnTrack.Text = "Start Tracking";
            //}
        }
        #endregion

        string _imageFileExtension = ".jpg";
        int _trackSequenceCount = 0;
        int _count = 0;
        string _path = "";
        Processor _imgProc = new Processor();
        Histogram _histTarget = null;
        Histogram _histCandidate = null;
        Selector _selector = null;
        ITracker _tracker = null;
        bool _isTracking = false;

        private void button1_Click(object sender, EventArgs e)
        {
            webCamCapture1.Start(0);
            _histTarget = null;
            _histCandidate = null;
            CreateSelector();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            webCamCapture1.Stop();
        }

        void webCamCapture1_ImageCaptured(object source,
WebCam_Capture.WebcamEventArgs e)
        {
            pictureBox1.Image = e.WebCamImage;
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
            pictureBox3.Image = e.WebCamImage;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            timer2.Enabled = true;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            timer2.Enabled = false;
        }

        private void timer2_Tick(object sender, EventArgs e)
        {
            TrackObject((Bitmap)pictureBox1.Image);
        }

        private void button5_Click(object sender, EventArgs e)
        {
            _kMeans = new KMeans((Bitmap)pictureBox3.Image,
Convert.ToInt32(txtNumClusters.Text), tracking.Colour.Types.RGB);
            timer_KMeans.Enabled = true;
            iter_say = 0;
        }

        private void timer_KMeans_Tick(object sender, EventArgs e)
        {
            if (!_kMeans.Converged)
            {
                _kMeans.Iterate();
                iter_say += 1;
                label3.Text = "Iterasyon Sayısı: " + iter_say.ToString();
                pictureBox4.Image = _kMeans.ProcessedImage;
            }
            else
            {
                timer_KMeans.Enabled = false;
            }

        }

        private void button6_Click(object sender, EventArgs e)
        {
            pictureBox5.Image = pictureBox2.Image;
        }

        private void button7_Click(object sender, EventArgs e)
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();

            //openFileDialog.InitialDirectory = "c:\\";
            //openFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files
(*.jpg)|*.jpg|All valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            openFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files
(*.jpg)|*.jpg|All valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            openFileDialog.FilterIndex = 2;
            openFileDialog.RestoreDirectory = false;

            if (DialogResult.OK == openFileDialog.ShowDialog())
            {
                pictureBox5.Image =
(Bitmap)Bitmap.FromFile(openFileDialog.FileName, false);
            }
        }

        private void button8_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();

            //saveFileDialog.InitialDirectory = "c:\\";
            saveFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files (*.jpg)|*.jpg|All
valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            saveFileDialog.FilterIndex = 2;
            //saveFileDialog.RestoreDirectory = true;

            if (DialogResult.OK == saveFileDialog.ShowDialog())
            {
                //m_Bitmap.Save(saveFileDialog.FileName);
                pictureBox5.Image.Save(saveFileDialog.FileName);

            }
        }

        private void button11_Click(object sender, EventArgs e)
        {
            pictureBox6.Image = pictureBox2.Image;
        }

        private void button10_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();

            //openFileDialog.InitialDirectory = "c:\\";
            //openFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files
(*.jpg)|*.jpg|All valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
            openFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files
(*.jpg)|*.jpg|All valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            openFileDialog.FilterIndex = 2;
            openFileDialog.RestoreDirectory = false;

            if (DialogResult.OK == openFileDialog.ShowDialog())
            {
                pictureBox6.Image =
(Bitmap)Bitmap.FromFile(openFileDialog.FileName, false);
            }
        }

        private void button9_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();

            //saveFileDialog.InitialDirectory = "c:\\";
            saveFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files (*.jpg)|*.jpg|All
valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            saveFileDialog.FilterIndex = 2;
            //saveFileDialog.RestoreDirectory = true;

            if (DialogResult.OK == saveFileDialog.ShowDialog())
            {
                //m_Bitmap.Save(saveFileDialog.FileName);
                pictureBox6.Image.Save(saveFileDialog.FileName);

            }
        }

        private void button12_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();

            //saveFileDialog.InitialDirectory = "c:\\";
            saveFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files (*.jpg)|*.jpg|All
valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            saveFileDialog.FilterIndex = 2;
            //saveFileDialog.RestoreDirectory = true;

            if (DialogResult.OK == saveFileDialog.ShowDialog())
            {
                //m_Bitmap.Save(saveFileDialog.FileName);
                picTargetHist.Image.Save(saveFileDialog.FileName);

            }
        }
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
        private void button13_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();

            //saveFileDialog.InitialDirectory = "c:\\";
            saveFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files (*.jpg)|*.jpg|All
valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            saveFileDialog.FilterIndex = 2;
            //saveFileDialog.RestoreDirectory = true;

            if (DialogResult.OK == saveFileDialog.ShowDialog())
            {
                //m_Bitmap.Save(saveFileDialog.FileName);
                picCandidateHist.Image.Save(saveFileDialog.FileName);

            }
        }

        private void button16_Click(object sender, EventArgs e)
        {
            pictureBox3.Image = pictureBox2.Image;
        }

        private void button15_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();

            //openFileDialog.InitialDirectory = "c:\\";
            //openFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files
(*.jpg)|*.jpg|All valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            openFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files
(*.jpg)|*.jpg|All valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            openFileDialog.FilterIndex = 2;
            openFileDialog.RestoreDirectory = false;

            if (DialogResult.OK == openFileDialog.ShowDialog())
            {
                pictureBox3.Image =
(Bitmap)Bitmap.FromFile(openFileDialog.FileName, false);
            }
        }

        private void button14_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();

            //saveFileDialog.InitialDirectory = "c:\\";
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
            saveFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files (*.jpg)|*.jpg|All
valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            saveFileDialog.FilterIndex = 2;
            //saveFileDialog.RestoreDirectory = true;

            if (DialogResult.OK == saveFileDialog.ShowDialog())
            {
               //m_Bitmap.Save(saveFileDialog.FileName);
               pictureBox3.Image.Save(saveFileDialog.FileName);

            }
         }

         private void button19_Click(object sender, EventArgs e)
         {
            pictureBox4.Image = pictureBox2.Image;
         }

         private void button18_Click(object sender, EventArgs e)
         {
            OpenFileDialog openFileDialog = new OpenFileDialog();

            //openFileDialog.InitialDirectory = "c:\\";
            //openFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files
(*.jpg)|*.jpg|All valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            openFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files
(*.jpg)|*.jpg|All valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            openFileDialog.FilterIndex = 2;
            openFileDialog.RestoreDirectory = false;

            if (DialogResult.OK == openFileDialog.ShowDialog())
            {
               pictureBox4.Image =
(Bitmap)Bitmap.FromFile(openFileDialog.FileName, false);
            }
         }

         private void button17_Click(object sender, EventArgs e)
         {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            //saveFileDialog.InitialDirectory = "c:\\";
            saveFileDialog.Filter = "Bitmap files (*.bmp)|*.bmp|Jpeg files (*.jpg)|*.jpg|All
valid files (*.bmp/*.jpg)|*.bmp/*.jpg";
            saveFileDialog.FilterIndex = 2;
            //saveFileDialog.RestoreDirectory = true;

            if (DialogResult.OK == saveFileDialog.ShowDialog())
            {
```

EK-3. (devam) Nesne Takip Algoritması Kaynak Kodları

```
                //m_Bitmap.Save(saveFileDialog.FileName);
                pictureBox4.Image.Save(saveFileDialog.FileName);

            }
        }

    }
}
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ŞAHİN, İsa
Uyruğu : T.C.
Doğum tarihi ve yeri : 03.08.1987, İzmir
Medeni hali : Bekâr
Telefon : 0 (546) 885 00 23
Faks : -
e-mail : sahin_isa_06@hotmail.com.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Yüksek lisans	Gazi Üniversitesi /E. E. M	Devam Ediyor
Lisans	Gazi Üniversitesi/ E. E. M	2010
Lise	Yozgat Erdoğan Akdağ A.Ö.L	2005

İş Deneyimi

Yıl	Yer	Görev
2010 - Çalışıyor	NOVOS Tıbbi Cihazlar	AR-GE Sorumlusu

Yabancı Dil

İngilizce

Yayınlar

-

Hobiler

Müzik, Film izlemek

GAZİ GELECEKTİR..