T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KAMERA KULLANILARAK GÖRÜNTÜ İŞLEME YOLUYLA GERÇEK ZAMANLI GÜVENLİK UYGULAMASI

YÜKSEK LİSANS TEZİ

Hazırlayan
Atınç YILMAZ

Tez Danışmanı
Prof.Dr. Ali OKATAN

Temmuz, 2007
İSTANBUL

HALIC UNIVERSITY
THE INSTITUDE OF SCIENCE
COMPUTER ENGINEERING

REAL TIME SECURITY APPLICATION WITH IMAGE PROCESSING USING CAMERA

MASTER THESIS

Prepared By
Atınç YILMAZ

Supervisor
Prof. Dr. Ali OKATAN

July, 2007
İstanbul

T.C.

HALİÇ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Bilgisayar Mühendisliği Programı Yüksek Lisans öğrencisi Atınç Yılmaz tarafından hazırlanan **"Kamera Kullanılarak Görüntü İşleme Yoluyla Gerçek Zamanlı Güvenlik Uygulaması"** adlı bu çalışma jürimizce Yüksek Lisans Tezi olarak Kabul Edilmiştir.

Tez Savunma Tarihi : 10.07.2007

(Jüri Üyesinin Ünvanı , Adı , Soyadı ve Kurumu) :	İmzası :
Jüri Üyesi: Prof.Dr.Ali OKATAN (Danışman)	.....
Jüri Üyesi : Prof.Dr.Bekir KARAOĞLU (Elektronik ve Hab.ABD Öğr.Üyesi)	.....
Jüri Üyesi :Yrd.Doç.Dr. Murat BEKEN	.....

İÇİNDEKİLER

ÖNSÖZ

Tez konumun belirlenmesinde ve çalışmalarımın her aşamasında yardımlarını esirgemeyip değerli fikir ve katkılarıyla ışık tutan ve yönlendiren, tez metnini inceleyerek biçim ve içerik bakımından son şeklini almasında katkıda bulunan danışman hocam Sayın Prof. Dr. Ali OKATAN'a (Haliç Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği); tez metnini inceleyerek biçim ve içerik bakımından son şeklini almasında katkıda bulunduğu ve her türlü yardımlarını esirgemediği için Sayın Öğretim Görevlisi Oğuz KARAN'a (Haliç Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği) ve katkılarından dolayı Sayın Okutman Sabri Serkan Güllüoğlu'na (Haliç Üniversitesi) teşekkürlerimi sunarım.

Ayrıca aileme yoğun çalışma zamanlarımda desteklerini esirgemediği için teşekkür ederim.

Atınç Yılmaz
Temmuz, 2007

ŞEKİLLER LİSTESİ

T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

Kamera Kullanılarak Görüntü İşleme Kullanılarak Gerçek Zamanlı Güvenlik Uygulaması

Hazırlayan
Atınç Yılmaz

Tez Danışmanı
Prof.Dr.Ali Okatan

Temmuz, 2007

ÖZET

Görüntü işleme; dijital olarak alınan görüntülerin işlenerek özelliklerinin ve yapılarının değiştirilmesini, geliştirilmesini ve bu görüntüler vasıtasıyla analizlerin yapılmasını sağlayan teknolojidir. Modern teknoloji, herhangi bir görüntünün (fotoğraf veya video) girdi olarak kullanılarak istenilen özellikte bir başka görüntünün veya girdi olarak kullanılan görüntü ile ilgili verilerin elde edilmesini mümkün kılmaktadır. Görüntü işleme ile bir görüntünün rengi, parlaklığı, boyutu, yapısı gibi özellikleri uygun yazılımlar kullanılarak değiştirilebilir, geliştirilebilir ve analiz edilebilir.

Bu yazılımlar, dijital ortama aktarılan görüntülerdeki bozuklukların giderilmesi ve daha kaliteli görüntü almak için kullanılabileceği gibi nesnelerin tanımlanması, hareketli ve hareketsiz nesnelerin ayrıştırılması gibi bir çok amaç için de kullanılabilir. Farklı formatlarda görüntülerin kullanıldığı her sektöre uygun çözümlerin üretilmesini sağlayan görüntü işleme; güvenlikten astronomiye, savunma sanayiinden kalite kontrolüne kadar sayısız alanda kullanılabilir.

Görüntülerde hareketin analizi konusu, gelişen teknolojiyle beraber askeri, biyolojik, coğrafik, tarımsal inceleme ve uydu fotoğraflarının yorumlanması gibi bir çok uygulama alanında yer bulmuştur. Bunlar gibi dış ortam problemlerinin çözümlenmesinde hareket analizi yöntemlerinin kullanılması, verimliliği artırıp harcanan zaman ve malzemeden kazanç sağlamaktadır.

Hareket analizi, görüntülerin yorumlanması, işlenmesi ve sıkıştırılması gibi uygulama alanlarında kullanılan temel tekniklerden biridir.

Bu çalışmada, kamera kullanılarak cihazdan alınan ardışık görüntülerde hareketin analizi için kullanılan yöntemler incelenip uygulanarak; elde edilen sonuçlar tartışılmıştır. Bu yöntemler Arka Plan Farkı Yöntemleri ve İstatiksel Yöntemler olmak

üzere iki kategori de incelenmiştir. Uygulamada görüntüler arasında piksel farkları karşılaştırılarak gerçek zamanlı bir güvenlik uygulaması gerçekleştirilmiştir.

Anahtar Kelimeler: Görüntü işleme, hareket tespiti, güvenlik uygulaması, arka plan farkı yöntemi, istatistiksel yöntem.

HALIC UNIVERSITY
THE INSTITUDE OF SCIENCE
COMPUTER ENGİNEERING

MASTER THESIS

Real Time Security Application With Image Processing Using Camera

Prepared By
Atınç Yılmaz

Supervisor
Prof.Dr.Ali Okatan

July, 2007

ABSTRACT

Image Processing is a technology that image properties and structures are processed with changes, improvement for data analysis. The modern technology provide important data with using photo and video inputs. Using image processing an image color, brightness, dimensions and any other structure can be changed, improved and made analysis by using appropriate softwares.

These software are used for the image reconstruction which are transferred to digital athmosphere. These images have some spoiled parts and for this we can rearrange these photos with imag processing. Image processing is used in security, astronomy, defensive, industry and quaility control centers.

The motion anaysis subject is being in some application areas that is interpreted for military, biyological, geographic, agrarian photos. Like these, problems are analyzed with using motion analysis methods and that bring fertility. With this solution we can gain time and material.

Motion analysis is one of the major techniques which is used in applications related with interpretation, processing and compressing digital images.

In this thesis, methods that are widely used fort he motion detection in a moving picture sequence, were studied and implemented, and their performances were discussed. These methods were investigated under two major headings namely background subtraction methods and statistical methods. In this application i have compared pixel differences between images, for appliying security application.

Keywords: Image processing, motion detection, security application, background subtraction, statical method.

1. GİRİŞ

Bu çalışmada, görüntü işleme konusu incelenmiş ve hareket analizi şablonunun temeli olan, hareketin tespiti ve izlenmesinde kullanılan yöntemler üzerinde inceleme ve uygulamalar yapılmıştır. Uygulamada, C# programlama dili kullanılmıştır.

Görüntü işleme ölçülmüş veya kaydedilmiş olan elektronik (dijital) görüntü verilerini, elektronik ortamda (bilgisayar ve yazılımlar yardımı ile) amaca uygun şekilde değiştirmeye yönelik olarak yapılan bilgisayar çalışmasıdır.

Bilgisayar kullanımının insan hayatının hemen hemen her noktasına yoğun bir şekilde girmesi, beraberinde dış dünya problemlerinin algılanması ve çözümlenmesi zorunluluğunu getirmiştir. Bu noktada, hareket analizi konusu; yer değiştirme, hız, alan, derinlik, etiketleme, nesne takibi ve tanımlanması gibi değişik alanlarda etkili ve verimli bir şekilde kullanılabilme potansiyelinden dolayı bilgisayar dünyasında sıkça araştırılan bir konu olmuştur.

Görüntülerde hareketin analizi konusu, gelişen teknolojiyle beraber askeri, biyolojik, coğrafik, tarımsal inceleme ve uydu fotoğraflarının yorumlanması gibi bir çok uygulama alanında yer bulmuştur. Bunlar gibi dış ortam problemlerinin çözümlenmesinde hareket analizi yöntemlerinin kullanılması, verimliliği artırıp harcanan zaman ve malzemeden (algılayıcı donanımlar, alarm sistemleri, insan kaynakları v.b.) kazanç sağlamaktadır.

Hareket analizi algoritmaları, genel olarak, hareketin varlığının tespiti, hareketli nesnenin yerinin belirlenmesi, izlenmesi ve son olarak da hareketin tanımlanması olarak 4 kısımdan oluşmaktadır. Analizin başarısını belirleyen kısım, ilk kısım olan hareketin varlığının tespitidir. Bu aşamanın doğru ve etkili bir şekilde yapılması, sonraki aşamaların verimini doğrudan etkilemektedir.

Hareket analizi, görüntülerin yorumlanması, işlenmesi ve sıkıştırılması gibi uygulama alanlarında kullanılan temel tekniklerden biridir.
Bu çalışmada, web kamera cihazından alınan ardışık görüntülerde hareketin analizi için kullanılan yöntemler incelenip uygulanarak, elde edilen sonuçlar tartışılmıştır. Bu yöntemler. Arka Plan Farkı Yöntemleri ve İstatistiksel Yöntemler olmak üzere iki kategoride incelenmiştir.

2. GÖRÜNTÜ İŞLEME

2.1. Görüntü İşleme Nedir?

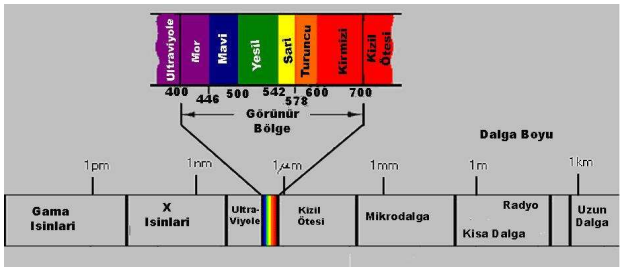
Görüntü işleme ölçülmüş veya kaydedilmiş olan elektronik (dijital) görüntü verilerini, elektronik ortamda (bilgisayar ve yazılımlar yardımı ile) amaca uygun şekilde değiştirmeye yönelik olarak yapılan bilgisayar çalışmasıdır.

Görüntü işleme, verilerin, yakalanıp ölçme ve değerlendirme işleminden sonra, başka bir aygıtta okunabilir bir biçime dönüştürülmesi ya da bir elektronik ortamdan başka bir elektronik ortama aktarmasına yönelik bir çalışma olan "Sinyal işlemeden" farklı bir işlemdir.

Görüntü işleme, daha çok, kaydedilmiş olan, mevcut görüntüleri işlemek, yani mevcut resim ve grafikleri, değiştirmek, yabancılaştırmak ya da iyileştirmek için kullanılır.

Elektronik veri işleme son 40 yılda inanılmaz bir hızla gelişmiştir. Bu gelişme bilgisayar teknolojisindeki gelişmelere paralel olarak meydana gelmiştir. Bilgisayarların giderek boyutlarının küçülmesi, bellek kapasitelerinin ve veri işleme hızlarının artışı görüntü işleme teknolojilerindeki gelişmeyi hızlandırmıştır.

Görüntü işleme açısından ele alındığında insan algılama sistemi; görüntü yakalama, gruplama ve analiz konusunda bilinen en karmaşık sistemdir. İnsan görme sistemi gözlerimizle başlar. Işığın çok kanallı ve pankromatik dalga boyları her biri birer algılama sistemi olan gözlerimiz yardımı ile algılanır. Görülebilen spektrum tanımı; insan gözünün görebileceği elektro manyetik dalga boyu aralığını tanımlar (Şekil-1.1). Buna karşın bir arının görebildiği spektral aralık ultraviyole bölgede başlar ve yeşil dalga boylarında sona erer. Spektrum uzunluk ölçme birimleri ile ölçülebilen periyodik davranış sergileyen enerji dalgalarını temsil eder. Görülebilen alana aitdalga boyları 0.4μm-0.7μm arasındadır.



Şekil 1.1. İnsan gözünün görebileceği elektro manyetik dalga boyu aralığı.

Gözlerimizle görülebilen alandaki elektro manyetik dalgaları algılayabiliriz ve beynimiz yardımı ile yorumlanabilir görüntü haline dönüştürebiliriz. Gözün ana bileşenleri; Kornea, göz bebeği, mercek, retina ve optik sinirlerdir.Kornea gözün dış kısmında olup geçirgen, kubbe formunda olup, ışığa odaklama fonksiyonuna sahiptir. Göz bebeği kendisini tutan kaslar yardımı ile ışık göze ulaştığında gözün açılıp kapanmasına yarar. Göz bebeği göz merceğini örter. Kaslar yardımı ile mercek göze giren ışığın şiddetine göre kalınlaşır veya incelir.

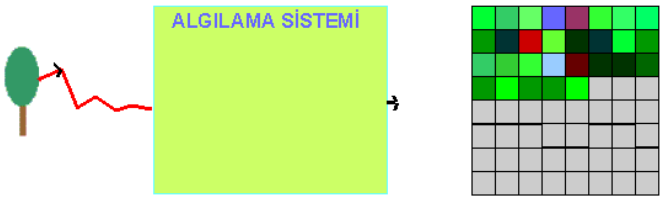
Gözlerin farklı kontrastlara adapte olabilme yeteneği parlaklık adaptasyonu (brightness adaption) olarak adlandırılır. İki parlaklık düzeyleri arasında ayırım yapabilme yeteneğine ise kontrast duyarlılığı adı verilir. Bu da gözün etrafını çevreleyen parlaklık düzeylerine bağlıdır. Güneşli bir günde farları yanan bir aracın farlarını görmek güçtür, fakat gece değildir.

Özet olarak; sayısal görüntü işleme için görme sistemlerimizin altında yatan temel mekanizmaların bilinmesi oldukça önemlidir. Kısaca göz bir fotoğraf makinası gibi düşünülebilir ve beynin görme bölümleri de karmaşık bir sayısal görüntü işleme sistemi olarak düşünülebilinir.

Görüntü işleme yaşam ver oldukça söz konusu olmuştur. İnsanlar ve hayvanlar gözleri ile analog temele dayanan görüntü işleme yapmaktadırlar. Bu olay beyin yardımı ile (akıllı sistem) on-line, paralel ve çok spektrumlu (multispektral) oluşmaktadır.

Resimlerin bilgisayar ortamında değerlendirilebilmeleri için veri formatlarının bilgisayar ortamına uygun hale getirilmeleri gerekmektedir. Bu dönüşüme sayısallaştırma (digitizing) adı verilir. Bir resmin fotografik sunumunu daha doğrusu sayısal forma dönüştürülmesi çeşitli şekillerde olanaklıdır. Buna farklı teknikler kullanılarak resmin sayısallaştırıldığı tarayıcılar örnek olarak verilebilir. Ya da Analog/Sayısal dönüşümün kullanılarak resmin sayısal hale dönüştürüldüğü sistemler (Frame-Grapper), uzaktan algılamada uçak ya da uydulara yerleştirilen çok kanallı tarayıcılar yine örnek olarak verilebilir.

Sayısal bir resim deyince akla analog bir sinyalin sayısal bir sinyale dönüştürülmesi gelmelidir. Bu da obje tarafından yayılan enerjinin (analog sinyal) bir algılayıcı tarafından öngörülen elektromanyetik aralıkta algılanarak sayısal sinyal haline dönüştürülmesi ile olanaklıdır.

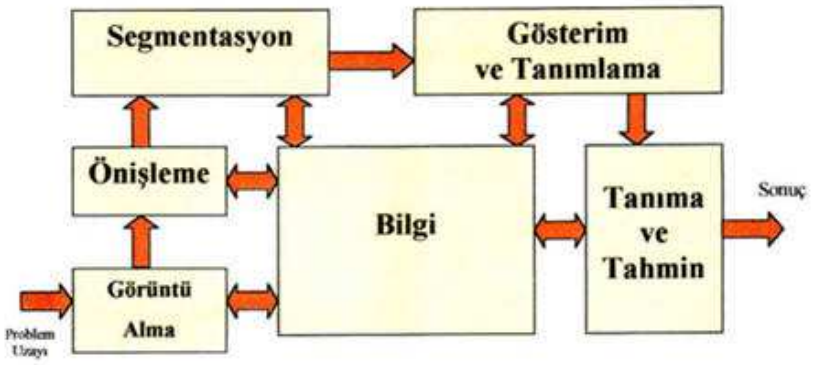


Şekil 1.2. Algılama Sistemi.

2.2. Görüntü İşleme Adımları ve Kullanım Alanları

Şekil 2.1. Görüntü İşleme.

Görüntü işleme; dijital olarak alınan görüntülerin işlenerek özelliklerinin ve yapılarının değiştirilmesini, geliştirilmesini ve bu görüntüler vasıtasıyla analizlerin yapılmasını sağlayan teknolojidir. Modern teknoloji, herhangi bir görüntünün (fotoğraf veya video) girdi olarak kullanılarak istenilen özellikte bir başka görüntünün veya girdi olarak kullanılan görüntü ile ilgili verilerin elde edilmesini mümkün kılmaktadır. Görüntü işleme ile bir görüntünün rengi, parlaklığı, boyutu, yapısı gibi özellikleri uygun yazılımlar kullanılarak değiştirilebilir, geliştirilebilir ve analiz edilebilir. Bu yazılımlar, dijital ortama aktarılan görüntülerdeki bozuklukların giderilmesi ve daha kaliteli görüntü almak için kullanılabileceği gibi nesnelerin tanımlanması, hareketli ve hareketsiz nesnelerin ayrıştırılması gibi bir çok amaç için de kullanılabilir. Farklı formatlarda görüntülerin kullanıldığı her sektöre uygun çözümlerin üretilmesini sağlayan görüntü işleme; güvenlikten astronomiye, savunma sanayisinden, kalite kontrolüne kadar sayısız alanda kullanılabilir.



Şekil 2.2. Görüntü İşleme Adımları

Görüntü işlemede ilk adım görüntüyü gerçek dünyadan bir film tabakasına veya bir hafıza birimine almamızı sağlayan resim alıcılarıdır. Bu cihazlarda bir resim algılayıcısı ve algılanan resmi sayısal hale getiren sayısallaştırıcı birim bulunmaktadır. Eğer resim sensörü resmi doğrudan sayısal hale dönüştürmüyorsa, elde edilen analog resim, bir Analog/Sayısal dönüştürücü yardımıyla sayısal hale dönüştürülmektedir.

Sayısal resim elde edildikten sonraki basamak ise ön-işleme'dir. Adından da anlaşıldığı gibi ön-işleme, elde edilen sayısal resmi kullanmadan önce daha başarılı bir sonuç elde edebilmek için, resmin bazı ön işlemlerden geçirilmesidir. Bu işlemlere örnek olarak; kontrastın ayarlanması, resimdeki gürültülerin azaltılması ve/veya yok edilmesi, resimdeki bölgelerin birbirinden ayrılması gibi işlemleri verebiliriz.

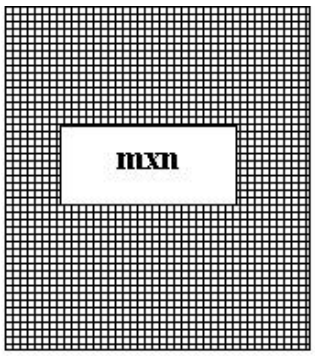
Ön-işlemler bittikten sonra segmantasyon (segmentation) basamağına geçilir. Segmantasyon, bir resimdeki nesne ve artalanın veya resim içerisinde ki ilgilenilen değişik özelliklere sahip bölgelerin birbirinden ayrıştırılması işlemidir. Segmantasyon görüntü işlemenin en zor uygulamasıdır ve segmantasyon tekniklerinin sonuçlarında belli bir hata oranı olabilmektedir. Segmentasyon bir resimde ki nesnenin sınırları, şekli veya o nesnenin alanı gibi ham bilgiler üretir. Eğer objelerin şekilleriyle ilgileniyorsak segmantasyonun bize o nesnenin kenarları, köşeleri ve sınırları hakkında bilgi vermesini bekleriz. Fakat resim içerisindeki nesnenin yüzey kaplaması, alanı, renkleri, iskeleti gibi iç özellikleriyle ilgileniliyorsa bölgesel segmantasyonun kullanılması gerekir. Karakter veya genel olarak örnek (pattern) tanıma gibi oldukça karmaşık problemlerinin çözümü için her iki segmantasyon metodunda bir arada kullanılması gerekebilmektedir.

Segmentasyondan sonraki basamak, resmin gösterimi ve resmin tanımlanması'dır. Ham bilgiler resimde ilgilenilen ayrıntı ve bilgilerin ön plana çıkarılması bu aşamada yapılır. En son kısım ise tanıma ve yorumlamadır. Bu aşamada ise resmin içerisindeki nesnelerin veya bölgelerin önceden belirlenen tanımlamalara göre etiketlendirilmesidir.

Görüntünün alınması ve gösterilmesi dışında görüntü işleme fonksiyonlarının çoğu temel görüntü işleme algoritmalarına göre yazılmış yazılımlardan ibarettir. Bilgisayarların bazı kısıtlamalarını aşmak ve işlemi hızının daha da arttırılmamasının istendiği durumlarda, görüntü işleme fonksiyonları, donanımla (hardware) elde edilmeye çalışılabilir.

2.3. Görüntü Türleri

Bir görüntünün temel bileşeni piksel-resim elemanı(pixel-picture element) dir. Dolayısı ile görüntü deyince mxn boyutlu piksellerden oluşan bir matris gelmelidir.



Şekil 2.3. Pikseller

Bir pikselin iki temel özelliği söz konusudur:
1.Radyometrik özelliği: Pikselin algılandığı elekromanyetik spekrumdaki gri değeri
2.Geometrik özelliği: Görüntü matrisinde sahip olduğu matris koordinatları

Bir resmin sayısallaştırılmasının açıklanması amacı ile öncelikle Siyah-Beyaz resim göz önünde bulundurulmuştur. Siyah-Beyaz resim sadece iki gri değerden oluşan birresimdir. Böylesi bir görüntüde her bir piksel ya siyah ya da beyaz olarak oluşur. Burada sembolik olarak beyaz pikseller 1, siyah pikseller 0 değeri ile gösterilecektir.

Şekil 2.4.' te görüntüye ait piksellerin 0 ve 1 kodlanmış hali verilmiştir. Bu şekilde 0 ve 1 kodlanmış piksellerden oluşan görüntülere ikili görüntü (binary image) adı verilir.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	0	0	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Şekil 2.4. İkili Görüntü

	1	2	3	……..		M
1						
2		(x-1,y-1)	(x-1,y)	(x-1,y+1)		
3		(x,y-1)	(x,y)	(x,y+1)		
⋮		(x+1,y-1)	(x+1,y)	(x+1,y+1)		
N						

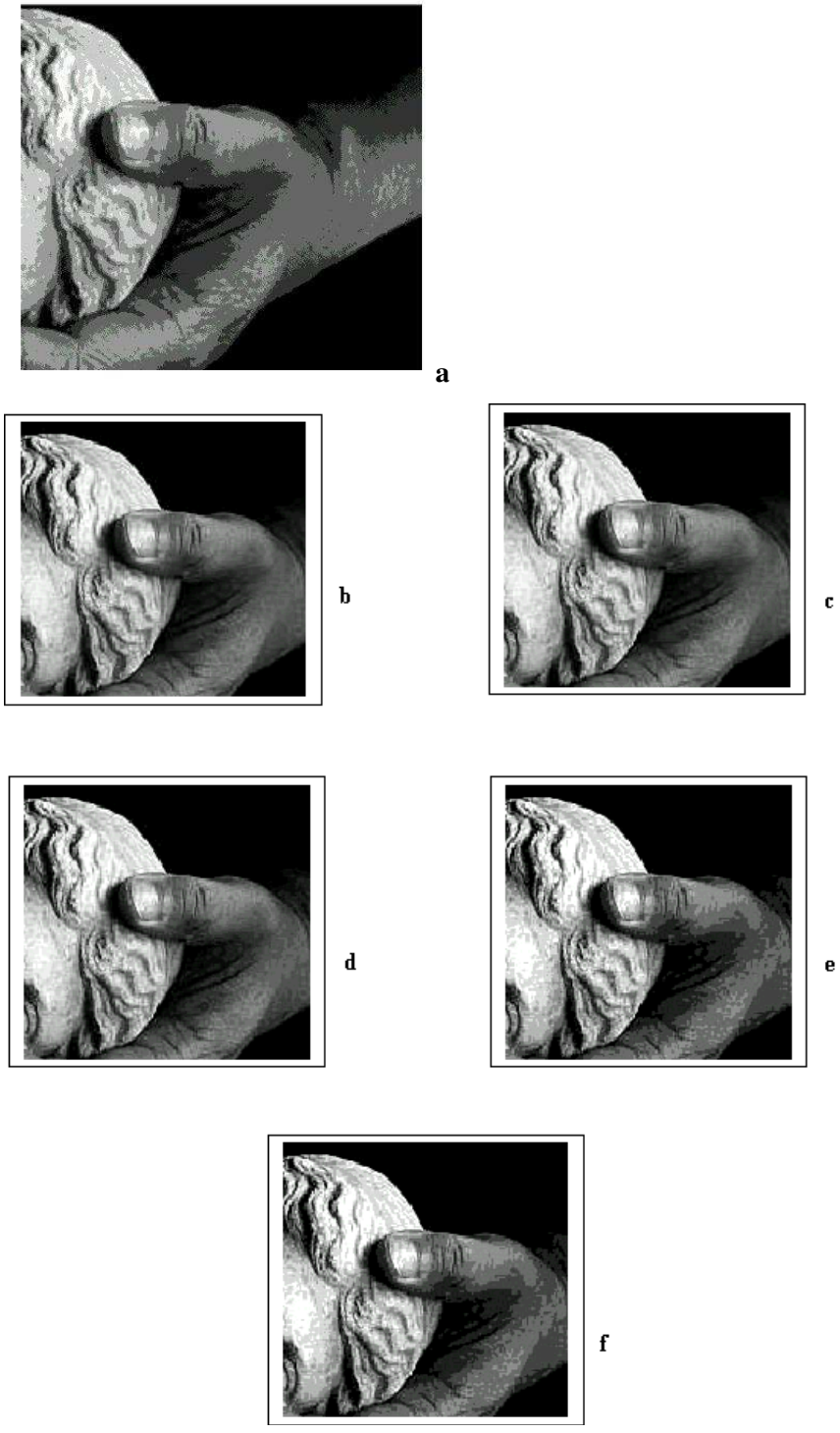
Şekil 2.5. Bir görüntünün koordinat sistemi.

Bir görüntü büyütüldükçe görüntüde detay yorumlama olanağı gittikçe yok olur. Aşağıdaki şekillerde bu açıkça görülecektir. 1 numaralı şekil orijinal görüntü olmak üzere bir görüntüyü büyütmeye başladığımızda en son 6 numaralı şekli elde ederiz. Bu da bizi piksele götürür. Şekillerdeki görüntü ile yukarıda anlatılan ikili görüntü arasındaki temel farklılık her pikselin gri değerinin farklı olmasıdır. Oysa ikili görüntüde pikseller 0 veya 1 değeri alabiliyordu. Daha başka bir deyişle ya siyah ya da beyaz olabiliyorlardı. O halde gri değerlerin farlı olmasının nedeni nedir?

Gri tonlu görüntülerde; görüntü farklı gri ton değerlerinden oluşur. Gri değer aralıkları: G={0,1,2,……..255} şeklinde ifade edilir. Bunun anlamı şudur: Bir gri tonlu görüntüde 256 tane farklı gri ton değeri daha doğrusu gri değer bulunabilir. Burada 256 gri değer bir byte olarak tanımlanabilir (1 Byte=8 Bit ve 28=256) [3].

0 gri değeri kural olarak siyah renge, 255 gri değeri ise beyaza karşılık gelir. Bu değerler arasında ise gri tonlar oluşur.

Aşağıdaki test görüntüleri sırası ile a)128, b)64, c) 32, d)16, e) 8 , f)4 gri değer düzeyi ile oluşturulmuştur.



Şekil 2.6. Görüntünün gri değer farkları.

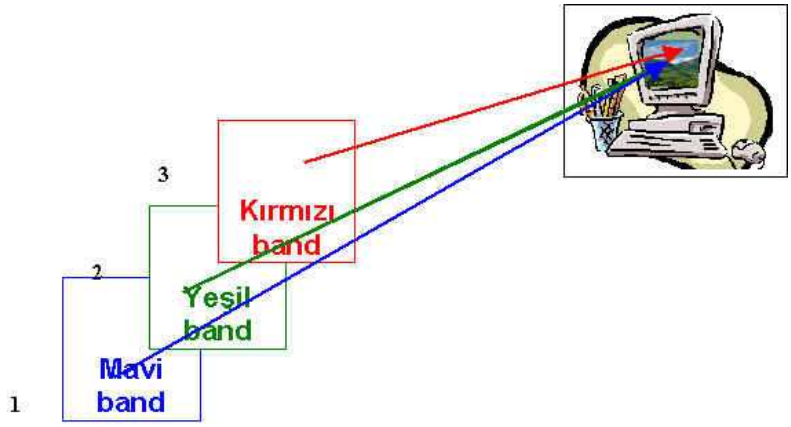
Bunun anlamı şudur. a) görüntüsünde 0-255 arasındaki gri değer düzeyleri orijinal görüntünün yarısı kadardır. Ya da f) görüntüsü ele alınacak olursa: orijinal görüntüde her 64 gri değer düzeyine f) görüntüsünde bir gri değer düşmektedir. Söz konusu kriterler görüntünün kalitesini etkileyen kriterlerdir.

Soru: Eğer işlemi devam ettirip orijinal görüntüyü 2 gri ton düzeyinde gösterseydik sonuçta ne elde ederdik?

Görüntünün kalitesini etkileyen diğer bir unsur da görüntüdeki piksel sayısıdır. Örneğin orijinal görüntü 233x224 pixelden oluşmasına karşın a) görüntüsü 117x112, b) görüntüsü 58x56 pikselden oluşmaktadır.

Şekil 2.7. Görüntünün piksel sayısı

Renkli görüntüler bilgisayar ekranlarında 24 bit lik veri olarak görüntülenir. Görüntüleme R(Kırmızı), G(Yeşil), B(Mavi) kodlanmış aynı objeye ait üç adet gri düzeyli görüntünün üst üstle ekrana iletilmesi ile oluşur. Elektro-manyetik spektrumda 0,4-0,5 µm dalga boyu mavi renge; 0,5-0,6 µm dalga boyu yeşil renge; 0,6-0,7 µm dalga boyu kırmızı renge karşılık gelir. Bu dalga boylarında elde edilmiş üç gri düzeyli görüntü bilgisayar ekranında sırası ile kırmızı-yeşil-mavi kombinasyonunda üst üste düşürülecek olursarenkli görüntü elde edilmiş olur.



Şekil 2.8. Kırmızı – yeşil – mavi bandlar.

Renkli görüntü kavramı; 1 band bir anlamda kırmızı filtrelenmiş, başka bir deyişle orijinal görüntüdeki gri değerler kırmızının tonları şeklinde ifade edilmiş, benzer şekilde 2 ve 3 bandlar da da yeşilin ve mavinin tonları şeklinde ifade edilip üstüste çakıştırılmış ve oluşan renk karışımından da doğal renkler elde edilmiştir ; şeklinde de açıklanabilir. Öyle ise band kombinasyonu şekilden de görüleceği üzere 3-2-1 dir.

3. HAREKET ANALİZİ VE TESPİTİ

3.1. Hareket Analizi Nedir?

Bilgisayar kullanımının insan hayatının hemen hemen her noktasına yoğun bir şekilde girmesi, beraberinde dış dünya problemlerinin algılanması ve çözümlenmesi zorunluluğunu getirmiştir. Bu noktada, hareket analizi konusu; yer değiştirme, hız, alan, derinlik, etiketleme, nesne takibi ve tanımlanması gibi değişik alanlarda etkili ve verimli bir şekilde kullanılabilme potansiyelinden dolayı bilgisayar dünyasında sıkça araştırılan bir konu olmuştur.

Görüntülerde hareketin analizi konusu, gelişen teknolojiyle beraber askeri, biyolojik, coğrafik, tarımsal inceleme ve uydu fotoğraflarının yorumlanması gibi bir çok uygulama alanında yer bulmuştur. Bunlar gibi dış ortam problemlerinin çözümlenmesinde hareket analizi yöntemlerinin kullanılması, verimliliği artırıp harcanan zaman ve malzemeden (algılayıcı donanımlar, alarm sistemleri, insan kaynakları v.b.) kazanç sağlamaktadır.

Hareket analizi algoritmaları, genel olarak, hareketin varlığının tespiti, hareketli nesnenin yerinin belirlenmesi, izlenmesi ve son olarak da hareketin tanımlanması olarak 4 kısımdan oluşmaktadır. Analizin başarısını belirleyen kısım, ilk kısım olan hareketin varlığının tespitidir. Bu aşamanın doğru ve etkili bir şekilde yapılması, sonraki aşamaların verimini doğrudan etkilemektedir.

Bu çalışmada, hareket analizi şablonunun temeli olan, hareketin tespiti ve izlenmesinde kullanılan yöntemler üzerinde inceleme ve uygulamalar yapılmıştır. Uygulamada, C# programlama dili kullanılmıştır.

3.2. Hareket Tespiti Araştırmalarının Tarihsel Gelişimi

Hareket tespitine ilgi 1970'lerin sonlarından itibaren artmaya başlamıştır. 1979'da Philadelphia'da yapılan ilk özel çalışma konferansı bu konunun gelişmesindeki ilk adım olarak gösterilmektedir. Bu adımdan sonra bir çok panel, sempozyum ve özel çalışma konferansı yapılmıştır. NATO Advanced Study Instute " Images Sequence Processing and Dynamic Scene Analysis" adlı özel çalışma konferansını 1982'de yapmıştır. Aynı yıl ilk uluslararası konferans "Time-Varying Processing and Moving Object Recognation" adı altında Florence, İtalya'da yapılmıştır. 1986'da Charleston, Güney Carolina'da IEEE Computer Society'nin "Motion" konulu konferansına çok sayıda makalenin katılması artan ilginin göstergesi olmuştur. Artan bu ilgi ve çalışmalar sonrasında yoğun talep ve ihtiyaçtan dolayı IEEE Computer Society 1989'da Irvine, California'da 1991'de Princeton, New Jersey'de "Visual Motion" adı altında iki konferans daha düzenlemiştir. Bunların sonucu olarak, zaman bağımlı görüntüler üzerine çeşitli özel konularda çok sayıda makale yayınlamış birçok dergi, gazete ve kitap bu konuyu ele almaya başlamıştır.

3.3. Hareket Tespiti – Segmentasyon Algoritmaları

Görsel tabanlı her türlü hareket analizi hareketin tespitiyle başlar. Hareket tespiti, hareketli bölgenin görüntüdeki diğer bölgelerden ayrılması (segmentasyon) temeline dayanır. Uygulanmış olan segmentasyon algoritmaları genel olarak 3 grupta incelenebilir.

- Arka Plan Farkı (Background Subtraction) Yöntemleri
- İstatiksel Yöntemler
- Görsel Akış(Optical Flow) Yöntemleri

3.3.1. Arka Plan Farkı (Background Subtraction) Yöntemleri

Arka plan farkı uygulamalardaki kullanım kolaylığından dolayı çok popüler bir yaklaşımdır. Bu yöntemde, hareketli bölgeler, incelenen görüntü ile referans görüntünün aynı, koordinatlardaki piksel farklarının, önceden belirlenmiş bir eşik değeriyle(Threshold) kıyaslanması sonucu tespit edilir. Ancak bu algoritmanın, en ufak ışık değişimlerine bile çok hassas olması, araştırmacıları, belirli zaman aralığında alınan ardışık görüntülerdeki her piksel değerinin ortalamasını veya medyanını arka plan modeli olarak kabul etmesi gibi yaklaşımlara itmiştir. Bu yaklaşım ışık değişimlerine olan hassasiyeti azaltmışsa da, çoğu araştırmacı ani değişimlere daha çabuk adapte olan daha kararlı arka plan modellemesi için değişik yöntemlere başvurmuştur. Örneğin, fark almada kullanılacak referans görüntüyü, bir önceki referans görüntüye fark aldığı görüntünün veya görüntülerin değişik miktarlarda ağırlığını ekleyerek modelleme yöntemi, bunlardan biridir. Bazı araştırmacılar ise, hareket tespitindeki kararlılığı arttırmak için fark almada tek arka plan modeli kullanmak yerine iki hatta üç arka plan modeli kullanma çözümünü önermişlerdir. Değişken ortamlara çok iyi uyum sağlayabilen bir başka yöntemde ise, bir veya daha fazla arka plan kullanmak yerine, piksel bazında fark almada, birden fazla ardışık görüntü kullanılıp değişken pikseller belirlenmektedir. Ancak, görüntüler ardışık alındığından, olası hareketli bölgeler içinde bozulmalar meydana gelmektedir. Ayrıca bu yöntemlerden daha eski ve farklı olarak, incelenen görüntüdeki her pikselin yoğunluğunu Kalman Filtresi kullanarak modelleyip arka planı tahmin etmeye yönelik çalışmalar yapan araştırmacılar da olmuştur.

Segmentasyon Algoritmalarından olan Arka Plan Farkı yöntemleri 4 farklı grupta incelenebilir

1. Basit Fark Alma Yöntemi
2. Medyan Filtreleme Kullanılarak Arka Plan Modelleme Yöntemi
3. Ağırlıklı Toplam Yöntemi
4. Çift Arka Plan Yöntemi

3.3.1.1. Basit Fark Alma Yöntemi

İki görüntü arasındaki değişimi tespit etmek için kullanılan en temel ve en basit yöntemdir. Bu yönteme göre değişim, t_1 ve t_2 zamanlarında alınan f_1 ve f_2 görüntülerinin, temel görüntü birimlerinin(piksel) matematiksel farklarının alınmasıyla tespit edilir. Bu görüntülerden biri, durağan bileşenlerden oluşan referans yani arka plan görüntüsü, diğeri ise, aynı durağan bileşenlerle beraber hareketli bir nesne veya nesnelerin olduğu görüntüdür. Bu iki görüntünün farkının alınması, değişimlerin gözlenmesi için oluşturulan fark resminde, durağan bileşenleri eleyerek, durağan olmayan bileşenleri ortaya çıkarmaktadır. t_o ve t_k zamanlarında alınan iki görüntü arasındaki değişiklikleri gösteren fark resminin oluşturulması şöyle ifade edilebilir.

$$D(x,y)=\begin{cases}1, \left| f(x,y,t_k)-f(x,y,t_0)\right| > Th \\ 0, DigerDurumlarda\end{cases} \qquad \text{Eşitlik 3.1.}$$

Th ile gösterilen değer, iki piksel arasındaki farkın, harekete ait olup olmadığını belirleyen, tamamen deneysel olarak önceden belirlenmiş eşik değeridir. Fark resmindeki "1" değerleri, incelenen resimlerin o koordinattaki, eşik değeri Th'ye göre belirgin farkı sembolize eder. Bir anlamda değişimi ikilik resim ile ifade eder.

Dinamik görüntü analizinde D(x,y)'deki tüm "1" değerleri, nesne hareketi olarak kabul edilir. Bu yaklaşım, ancak ışık yoğunluğu sabit ise doğrudur. Pratikte, ışık yoğunluğu her zaman sabit olmadığından, bu durum çoğunlukla D(x,y)'de, gürültü diye tabir edilen, harekete ait olmayan aktif piksel gruplarının oluşmasına yol açmaktadır. Gürültünün temizlenmesi için, tipik olarak D(x,y)'deki birbirine 4- veya 8- komşuluk ile bağlı "1" değerleri sayılıp, küçük alanlara sahip adacıkların elenmesi yöntemi kullanılmaktadır. Her ne kadar bu yaklaşım, küçük ve/veya yavaş-hareketli

nesneleri elese de, fark resminde geriye kalan alanların, gerçek harekete ait olma olasılıklarını büyük ölçüde arttırmaktadır. Bu yöntem için geliştirilen algoritmai sırasıyla şu işlemleri yapar:

T_o ve t_k zamanlarında alınan görüntüler Th değerine göre piksel bazında eşitliğe göre değerlendirilerek fark resmi oluşturulur:

Her resim elemanı (x,y) için
    Eğer mutlak fark > Th ise D(x,y)=1
    Değil ise D(x,y)=0

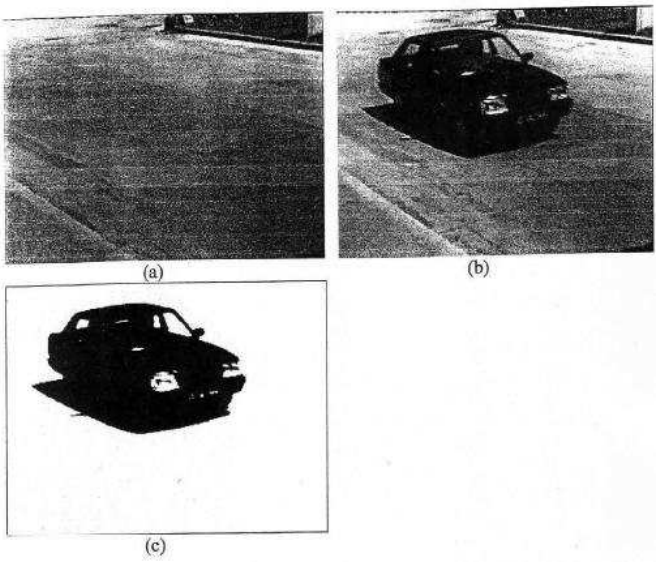
Oluşturulan fark resminde, Th_c eşik değerine göre 8- komşuluk taramasıyl "say" ve " sil" adlı iki rekürsif fonksiyon ile gürültü analizi yapılarak, belirlenen bölgeler temizlenir.

Her resim elemanı (x,y) için
    Eğer D(x,y)=1 ise fonksiyon(alan hesapla)
    Eğer alan > Th_c ise fonksiyon(alanı temizle)

Fonksiyon(alan hesapla)
Her resim elemanının 8- komşuluğu için
    Eğer komşu resim elemanı=1 ise fonksiyon(alan hesapla)
Alan bilgisini gönder

    Fonksiyon(alanı temizle)
    Her resim elemanının 8- komşuluğu için
    Eğer komşu resim elemanı=1 ise fonksiyon(alanı temizle)

Bu işlemlerin ardından, Şekil xx de görülceği gibi, gürültüden bir ölçüde arınmış, aktif piksel gruplarına ait bir fark resmi elde edilmiş olur.



Şekil 3.1. Basit fark alma yöntemi, a) t_o anında alınan görüntü b) t_k anında alınan görüntü c) Th=30 için fark resmi

3.3.1.2. Medyan Filtreleme Kullanılarak Arka Plan Modelleme Yöntemi

Basit fark alma yönteminde bahsedilen, değişken ışık yoğunluğundan kaynaklanan gürültü sorunu, arka planın modellenmesi veya güncellenmesi ihtiyacını doğurmuştur. Bu çözüme en basit yaklaşım, her bir pikselin, daha önce gelen K tane değerin ortalamasıyla ifade edildiği arka plan modelidir. Ancak, çoğu araştırmacıya göre modelde pikselin ortalamasını değil de medyan değerini kullanmak daha kararlı bir sonuç vermektedir.

Bu yöntemde t_k zamanında alınan $f(x,y,t_k)$ görüntüsünün kıyaslanacağı arka plan modeli $B_m(x,y),[t_o,t_{k-1}]$ aralığında alınan, $f(x,y,t_o),f(x,y,t_1)\ldots.f(x,y,t_{k-2}),f(x,y,t_{k-1})$ görüntülerinden, her piksel için medyan değeri kullanılarak elde edilir. K tane piksel değerinin medyanı, değerlerin küçükten büyüğe sıralanmasının ardından, K/2'inci eleman seçilmesiyle bulunur.K'nın tek sayı olması durumunda, k/2 değerinin sıralamadaki bir sonraki tam sayıya denk gelen elemanı seçilir. Örneğin gelen 9 değer (10,20,20,20,15,20,20,25,100) olsun. Değerler küçükten büyüğe doğru (10,15,20,20,20,20,20,25,100) gibi sıralandığında, bu veri grubunun medyan değeri 5. büyük değer yani "20" olur. Arka plan $B_m(x,y)$ modellendikten sonra, $f(x,y,t_k)$ ile arasındaki fark resmi eşitlik 3.1 dekine benzer şekilde belirlenir

$$D(x,y)=\begin{cases}1, \left| f(x,y,t_k) - B_m(x,y) \right| > Th \\ 0, DigerDurumlarda\end{cases}$$ Eşitlik 3.2.

Eşitlikteki Th değeri, yine önceden deneysel olarak belirlenmiş bir eşik değerdir. Ancak bu eşik değeri, bir önceki yöntemdeki eşik değeri ile kıyaslandığında,nispeten daha küçük bir değerdir. Bu yöntem ile geliştirilen sistemin akışı şu şekildedir:

Öncelikle, önceden belirlenen sayıda görüntü (k=20…100) daha sonra piksel meydanların belirlenmesi amacıyla geçici olarak sırasıyla 3 boyutlu bir matrise aktarılır.

0'dan K-1 'e her matris için
Her resim elemanı (x,y) için
Geçici$_{xyk}$= resim elemanı

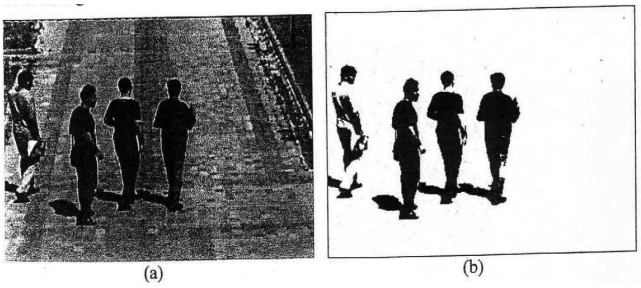
Üç boyutlu matrise aktarılan K adet görüntünün piksel bilgilerinden yararlanılmak suretiyle medyan analizi yapılarak, sonuçlar iki boyutlu bir matrise aktarılır.

Her resim elemanı (x,y) için
Her geçici$_{xyk}$ için
Küçükten büyüğe sırala
Medyan$_{xy}$ = geçici$_{xyk/2}$

Oluşturulan arka plan modeli ile incelenecek görüntü arasındaki farktan, eşitlik(3.2) e göre değişim resmi oluşturulur.

Her resim elemanı(x,y) için
Eğer mutlak fark > Th ise D(x,y)=1
Değilse D(x,y)=0

Yukarıda işleyişi verilen algoritmada anlatıldığı gibi, önyükleme için gelen 20 görüntüden her pikselin modellemesi yapıldıktan sonra elde edilen fark resmi şekil 3.2. deki gibi olmaktadır.



Şekil 3.2. Medyan filtreleme yöntemi a)t_k zamanında alınan görüntü b)th=30 için elde edilen fark resmi.

3.3.1.3. Ağırlıklı Toplam Yöntemi

Bu yönteme göre gelen görüntü F_k ile uyarlanabilir arka plan görüntüsü B_k'nın piksel tabanlı farkı hareketi belirler. Aradaki fark önceden belirlenmiş Th eşik değerini aşarsa gelen görüntüdeki o pikselin hareket bilgisi taşıdığı varsayılır. Aktif pikseller "1", pasif pikseller ise "0" ile etiketlenerek ikili bir fark resmi oluşturulur.

$$D(x,y)=\begin{cases}1, \left|f(x,y)-B_k(x,y)\right|>Th\\ 0, DigerDurumlarda\end{cases} \qquad \text{Eşitlik 3.3}$$

Güvenilir bir hareket tespitini garantilemek için arka plan görüntüsü sık sık güncelleştirilmelidir. Arka plan güncellenmesi yeni gelen görüntünün o anki arka plana, belirli bir entegrasyonu ile gerçekleşmektedir. Entegrasyon da, 1.dereceden rekürsif bir filtreyle 3.4 teki gibi yapılmalıdır.

$$B_{k+1}(x,y)=\alpha F_k(x,y)+(1-\alpha)B_k(x,y) \qquad \text{Eşitlik 3.4}$$

Eşitlikteki "α" 1'den küçük bir adaptasyon katsayısını temsil etmektedir. "α" katsayısı büyüdükçe adaptasyon hızı artmakta ancak hareketli objenin arkasında yapay bir kuyruk belirlemeye başlamaktadır.

Arka plan güncellenirken kontrol edilmesi gereken iki durum vardır. Eğer bir piksel arka arkaya "m" kereden fazla aktif olarak etiketlenmiş ise veya sürekli durum değiştiriyorsa arka plan olarak işaretlenmelidir. Çünkü bu durum arka plana durağan bir objenin dahil olduğuna veya yaprak v.b. gibi periyodik hareket eden bir objeye ait olduğuna işarettir.

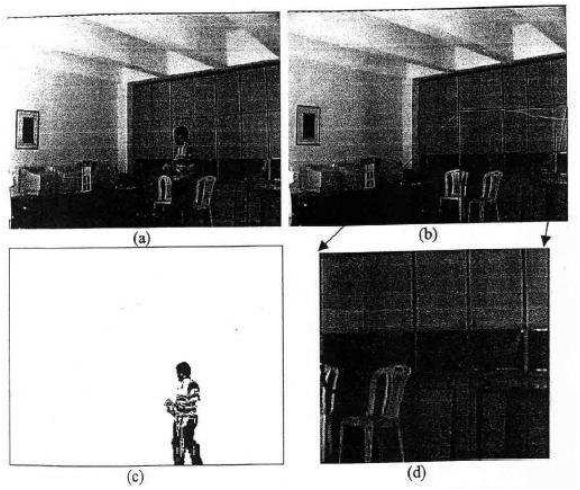
Bu yöntemin uygulanması için geliştirilen algoritmanın işleyişi aşağıdaki gibidir: Alınan ilk görüntü arka plan olarak kabul edilerek işleme ikinci görüntüden itibaren başlanır ve ilk fark resmi oluşturulur. Ardışık aktifliği sayılıp iki boyutlu bir matriste tutulan her piksel güncellenme aşamasından önce kontrol edilir. Arka plan eşitlik 3.4' e göre güncellendikten sonra işlem bir sonraki görüntüyü almak üzere başa döner.

Her resim elemanı (x,y) için
Eğer mutlak fark > Th ise D(x,y) = 1
Değil ise D(x,y) = 0

Eğer ardışık aktiflik > m ise $B_k = F_k$
Arka plan güncelle

Çok değişken ışık yoğunluğunun söz konusu olduğu durumlarda ihtiyaç duyulması halinde daha temiz ve belirgin sonuç elde etmek için daha önce anlatıldığı gibi gürültü analizi veya belirli bir değerden küçük alanların temizlenmesi yoluna başvurulmalıdır.

Şekil 3.3 de görülen sonuç algoritmaya verilen α =0,125 ve Th=20 değerleri için yapılan işlemlerin ardından elde edilen fark resmidir.



Şekil 3.3. Ağırlıklı toplam yöntemi a) incelenen görüntü b) α =0,03125 için hesaplanan ağırlıklı arka plan c) Th=20 için oluşturulan fark resmi d) Arka plan adaptasyonu

3.3.1.4. Çift Arka Plan Yöntemi

Bu yöntem arka plan farkının yeniden uyarlanıp medyan filtreleme yöntemleriyle birleştirilmiş olan yeni bir hareket tespiti yaklaşımıdır. Hareketin inceleneceği görüntü F_k biri uzun süreli diğeri kısa süreli olmak üzere iki arka plan ile karşılaştırılır. Uzun süreli arka plan B_{LT} ilk alınan görüntüdür ve belirli aralıklarla medyan filtreleme yöntemiyle güncellenir. Kısa süreli arka plan F_{k-1} , incelen görüntüden önceki görüntüdür. Hareket her pikselin komşuluk farkları toplamı bilgisine göre tespit edilir. Komşuluk farkları toplamı şöyle ifade edilebilir.

$$D[F_1, F_2] = \sum_x \sum_y \sum_{i=-1}^{1} \sum_{j=-1}^{1} [F_1(x+i, y+j) - F_2(x+i, y+j)] \qquad \text{Eşitlik 3.5}$$

Eşitliğe göre incelenen görüntüyle uzun süreli arka plan arasındaki fark resmi Dif_b eşitlik 3.6' dan incelenen görüntüyle bir önceki arasındaki fark resmi Dif_t 'de eşitlik 3.7' den bulunabilir.

$$Dif_b = D[F_k, B_{LT}] \quad \text{Eşitlik 3.6}$$

$$Dif_t = D[F_k, F_{k-1}] \quad \text{Eşitlik 3.7}$$

Bu fark resimleri piksellerin hareketli bölgeye ait olup olmama olasılıklarını göstermektedir. Bu olasılıkları değerlendirmek için Th_b ve Th_t gibi iki eşik değerine ihtiyaç duyulmaktadır. Bu değerlerin yeterince büyük seçilmesi olasılıkları küçük olanları elemektedir. Bu değerlendirmeler aşağıdaki gibidir.

Eşitlik 3.8 ve Eşitlik 3.9

$$M_b(x, y) = \begin{cases} 1, Dif_b(x, y) > Th_b \\ 0, DigerDurumlarda \end{cases}$$

$$M_t(x, y) = \begin{cases} 1, Dif_t(x, y) > Th_t \\ 0, DigerDurumlarda \end{cases}$$

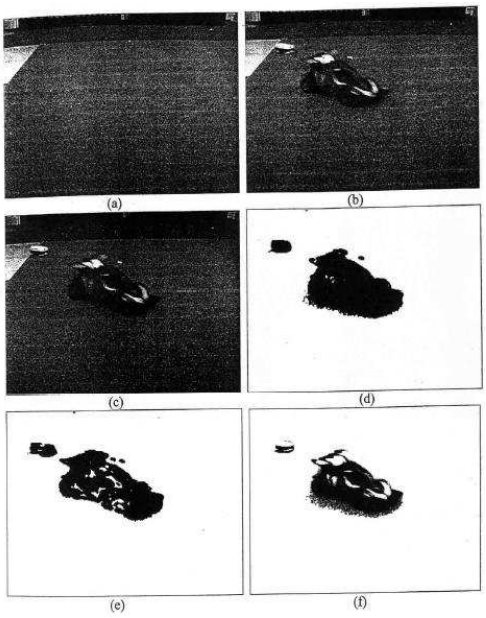
M_b resmi hareketli ve statik bölgelerden oluşurken, M_t resmi ise çoğunlukla hareketli bölgelerden oluşmaktadır. Ancak M_t'de ardışık iki görüntü olmasından dolayı elenen bazı pikseller olabilmektedir. Bu yüzden M_t ve M_b resmi bir sınıflandırma prosedürüne tabi tutulur. Bu prosedür sonucu biri B_{mov} diğeri B_{si} olan iki sonuç resmi elde edilir. Prosedürdeki işlem sırası:

M_b 'deki piksel grupları etiketlenir
M_t'deki piksel grupları etiketlenir
İki resimde etiketlenen alanların kesişmeleri bulunur
B_{si} ve B_{mov} resimleri oluşturulur

B_{mov} hareketli piksel gruplarını ihtiva eder bu piksel grupları M_b ve M_t'nin her ikisinde de aktif olduğu piksellerdir. B_{st} medyan filtreden yeni bir arka plan gelene kadar o anki uzun süreli arka planın güncellenmesinde kullanılır. Arka plan güncellemesinin ardından da hareket tespiti tamamlanmış olur. Bu yöntem için geliştirilen sistemin işleyişi şu şekildedir:

İncelenen görüntüyle arka planlar arasındaki komşuluk farkları toplamı eşitlik 3.5' teki gibi bulunur. Bulunan bu toplamların Th_b ve Th_i eşik değerlerine göre kıyaslanmasıyla Şekil 3.4 ve Şekil 3.4' deki gibi M_t ve M_b resimleri oluşturulur.

Her resim elemanı (x,y) için
8-komşuluk fark toplamını Dif_b için bul
8-komşuluk fark toplamını Dif_t için bul
Eğer $Dif_b > Th_b$ ise $M_b=1$ Değil ise $M_b = 0$
Eğer $Dif_t > Th_t$ ise $M_t=1$ Değil ise $M_t=0$



Şekil 3.4. Çift arka plan yöntemi a) B_{LT} b) I_{k-1} c) M_b d)M_t e) B_{mov}

Gelen iki resimdeki kesişen ve kesişmeyen alanlar belirlenerek rekürsif bir fonksiyonla sonuç resimleri oluşturulur. Aynı zamanda arka planın güncellenmesi de yapılır.

Her resim elemanı (x,y) için
Eğer M_b ve M_t kesişiyor ise B_{mov}=1, B_{st}=0
Değil ise B_{mov} =0 , B_{st} =1
Arka planı güncelle $B_{LT} = F_k$

Uzun süreli arka plan belirli bir süreyle işlem sırasında sürekli yenilenir. Bunun için medyan filtresinden faydalanılır. Gelen görüntüler üç boyutlu bir matrise aktarılarak seçilen süre sonunda arka plan pikselleri daha önceki bölümlerde anlatıldığı gibi medyanı alınmış şekilde güncellenir. İki güncelleme süreci arasında, B_{st}, de aktif olan koordinatlara denk gelen F_k değerleri arka planın aynı koordinatlarına aktarılarak güncellenme sürekli kılınır.

3.3.2. İstatistiksel Yöntemler

Son yıllarda temel arka plan farkından esinlenilerek, bazı istatistiksel metotlarla, değişim gösteren alanlar tespit edilebilmektedir. Bu istatistiksel yaklaşımlarda, bir piksel veya piksel grubunun karakteristiğine bakılarak, daha ileri düzeyde bir arka plan modellemesi yapılabilmektedir. Bu yöntemlerin en büyük avantajı, bu istatistiksel bilgilerin işlem sırasında güncellenebilmesidir. Bloklara bölünmüş ardışık iki görüntü arasındaki farklılığın, bloklara piksel ortalamaları ve standart sapmaları kullanarak, Maximum Likelihood yöntemiyle araştırılması buna belirgin bir örnektir. Kimi araştırmacılar, değişimin, kenar tespitinin ardından olası piksel hareketinin istatistiksel olarak oylanmasını temel alan Hough Transform kullanılarak incelenmesini savunmuşlardır. İstatiksel olarak karakterize edilen değerlerin çoğunlukla renk ve kenarlar olmasından dolayı, piksellerin, çoğunlukla renk ve kenarlar olmasından dolayı, piksellerin, YUV ve RBG renk uzayındaki Gauss yayımlarının karışımı veya diagonal kovaryansı olarak modellendiği yöntemlere de son zamanlarda sıkça rastlanmaya başlamıştır. Bu yöntemlere ek olarak Haritaoğlu ve ark,2000 hareket tespiti için incelenecek bölgenini kısa bir süre görüntüsü aldıktan sonra, her pikselin, minimumunu, maksimumunu ve ardışık görüntülerdeki maksimum farkını kullanıp o pikselleri istatistiksel olarak modelleyerek farklı bir metot geliştirmişlerdir.

Segmentasyon Algoritmalarından olan İstatistiksel Metot yöntemleri 4 farklı grupta incelenebilir

1. İstatistiksel Basit Fark Alma
2. Maksimum Fark Yöntemi
3. Ağırlık Merkezi Analizi Yöntemi
4. Hough Transform Yöntemi

3.3.2.1. İstatiksel Basit Fark Alma

Bu yöntem basit fark alma yönteminin istatistiksel yaklaşımıdır. Basit fark alma yönteminde görüntüler arasındaki fark önceden deneysel olarak belirlenmiş bir eşik değerine göre değerlendirilmekteydi. Bu yöntemde ise eşik değeri istatistiksel olarak ifade edilmektedir. Eşik değeri , $[t_o,t_{k-1}]$ aralığında alınan $f(x,y,t_o)$, $f(x,y,t_1)$… $f(x,y,t_{k-2})$, $f(x,y,t_{k-1})$ görüntülerinden her piksel kullanılarak oluşturulacak olan fark

resmindeki aktif pikseller eşitlik 3.10 a göre eşitlik 3.11' den hesaplanan ortalama (μ_{xy}) ve eşitlik 3.12 den hesaplanan standart sapmasına (σ_{xy}) göre belirlenir.

$$D(x,y)=\begin{cases}1, \left| f(x,y,t_k)-\mu_{xy}\right| > N\sigma_{xy} \\ 0, DigerDurumlarda\end{cases} \qquad \text{Eşitlik 3.10}$$

$$\mu_{xy}=\frac{1}{K}\sum_{k=0}^{K-1} f(x,y,t_k) \qquad \text{Eşitlik 3.11}$$

$$\sigma_{xy}=\sqrt{\frac{1}{K}\sum_{k=0}^{K-1}\left[f(x,y,t_k)-\mu_{xy}\right]^2} \qquad \text{Eşitlik 3.12}$$

Bu yöntemin basit fark alma yönteminden ayrıldığı diğer bir nokta ise, hareketin iki piksel arasındaki farktan değil de incelenen görüntünün işlemdeki pikseli ile o pikselin hesaplanan μ değeri arasındaki farktan belirlenmesidir. Ortalaması ve standart sapması hesaplanan piksellerin bir normal dağılma(Normal Distribution) sahip olduğu varsayılırsa eşitlik 3.12' de eşik değeri olarak ifade edilen $N\sigma_{xy}$ ' deki N katsayısının 2 olarak alınması hesaplamaya katılan değerlerin %95'ini, 3 olarak alınması %99'unu kapsayacaktır. Böylece eşik değeri deneysellikten kurtulup istatistiksel bir tabana oturtulmuştur.

Bir görüntü dizisinden ortalama ve standart sapma tek piksel için hesaplanacağı gibi MxM piksele sahip bloklara bölünmüş görüntünün her bloğu için de hesaplanabilir. Bu durumda farklılık karakteristiği belirlenmiş bloklar için değerlendirilir. Bir görüntüyü piksel piksel ifade etmek yerine blok blok ifade etmek ani ışık değişimlerine olan hassasiyeti azaltmaktadır. Koordinatı (x,y) olan, MxM piksele sahip bloklara bölünmüş görüntüdeki her blok piksellerinin ortalaması ile eşitlik 3.13' teki gibi ifade edilebilir.

$$\mu_{xy}=\frac{1}{MxM}\sum_{i=\frac{M-1}{2}}^{\frac{M-1}{2}}\sum_{j=\frac{M-1}{2}}^{\frac{M-1}{2}} f(x+i,y+j) \qquad \text{Eşitlik 3.13}$$

Her bloğun ortalaması hesaplandıktan sonra daha sonra değerlendirilmek üzere üç boyutlu bir matrise aktarılır. Görüntüdeki tüm bloklar bu şekilde incelendikten sonra bir sonraki görüntü alınarak aynı işlemden geçirilir. $[t_o,t_{k-1}]$ aralığındaki tüm görüntülerin işlemden geçirilmesinin sonucunda üç boyutlu matrise bloklara ait K adet ortalama bilgisi aktarılmış olur. Her blok için 3.10 ve 3.11 eşitliklerinin uygulanmasıyla her bloğun karakteristiğine ulaşılmış olur. Son adım olarak da $F(x,y,t_k)$ görüntüsünden her blok için eşitlik 3.12 uygulanarak fark resmi oluşturulur. Bu işlemlerin uzun süre devam edeceği göz önünde bulundurulursa elde edilen piksel veya blok karakteristiklerini güncellemek zorunludur. Güncelleme işlemi için "Running Avarage" denilen bir yöntem kullanılmaktadır. Bu yöntemde ilk başta elde edilen ortalamalara her yeni gelen görüntünün belli miktardaki ağırlığını eşitlik 3.14' e göre ekleyerek ortalamalar güncellenmektedir.

$$\mu_{xyt_{k+1}} = \alpha f(x,y,t_k) + (1-\alpha)\mu_{xyt_k} \qquad \text{Eşitlik 3.14}$$

Eşitlikteki "α" adaptasyon katsayısı 0.05 gibi çok küçük bir değerdir. Ancak α katsayısının deneysel olarak belirlenmesi yerine 1/(MxM) olarak alınması istatistiksel anlamda daha uygundur. Ortalamaların güncellenmesinin ardından yeni standart sapmaların hesaplanması için üç boyutlu matristeki değerlerin de güncellenmesi gerekir. Yeni gelen her değer matrise son eleman olarak atanırken elemanlar geriye doğru bir kaydırılarak matristeki ilk değer atılır. Ancak yeniden hesaplanan standart sapmaların eskisine göre ihmal edilebilir küçüklükte bir değişim göstereceği varsayılırsa tercihen bu işlemin atlanması mümkündür. Ayrıca standart sapmaların yeniden hesaplanması ek bir işlem zamanı getirdiğinden bu işlemin atlanması işlem zamanı anlamında da yarar sağlayabilmektedir. Bu yöntemlerin uygulamaları için geliştirilen algoritmaların akışı şu şekildedir:

İlk yöntem için $[t_o,t_{k-1}]$ aralığındaki K adet görüntüdeki her pikselin ortalaması ve standart sapması hesaplanır.

0'dan K-1'e her resim için
Her resim elemanı için
Ortalamayı hesapla
Standart sapmayı hesapla

Alınan $f(x,y,t_k)$ görüntüsünün her pikseli hesaplanmış olan ortalama ve standart sapma değerlerine göre eşitlik 3.12' deki gibi incelenerek far resmi oluşturulur.

Her blok için
Her resim elemanı için
Eğer mutlak fark > Nσ ise D(x,y) =1
Değil ise D(x,y)=0

İkinci yöntem için $[t_o,t_{k-1}]$ aralığındaki K adet görüntüdeki MxM piksele sahip her blok için ortalama hesaplanır ve üç boyutlu bir matrise aktarılır.

0'dan K-1 e Her resim için
Her blok için
Resim elemanı için
Ortalamayı hesapla
$geçici_{xyk}$=ortalama

Alınan $f(x,y,t_{k)}$ görüntüsünün her bloğu üç boyutlu matristeki bilgilerden hesaplanmış olan ortalama ve standart sapma değerlerine göre eşitlik 3.12 'deki gibi incelenerek fark resmi oluşturulur. Ardından da güncelleme işlemi yapılır.

0'dan K-1'e kadar her $geçici_{xyk}$ için
Ortalamayı hesapla
Standart sapmayı hesapla

Her blok için
Eğer mutlak fark > Nσ ise
Her resim elemanı için D(x,y) = 1
Değil ise
Her resim elemanı için D(x,y) = 0
Ortalamayı eşitlik 3.14' e göre güncelle.

Yatayda i, düşeyde j sayıda bloğa bölünmüş $f_1(x,y)$ ve $f_2(x,y)$ gibi iki görüntü arasındaki farkın (veya benzerliğin) belirlenmesi için öncelikle, "i x j" bloğa bölünmüş bu iki görüntüdeki her bloğun gri seviye karakteristiğinin çıkarılması gerekir. Bunun için de eşitlik 3.13 ve eşitlik 3.11' deki gibi MxM piksele sahip blokların ortalamaları (μ_{1ij},μ_{2ij}) ve standart sapmaların hesaplanmasıyla bloklar arasındaki benzerlik orası eşitlik 3.15 e göre bulunabilir.

$$L_{ij} = \frac{\left[\frac{\sigma_{1_{ij}} + \sigma_{2_{ij}}}{2} + \left(\frac{\mu_{1_{ij}} - \mu_{2_{ij}}}{2} \right)^2 \right]^2}{\sigma_{1_{ij}} \sigma_{2_{ij}}} \qquad \text{Eşitlik 3.15}$$

Eşitlik 3.15' in tanımsız olmaması için paydadaki "σ_{1ij}, σ_{2ij}" çarpımının "0" değerini almamasına dikkat edilmesi gerekmektedir. Bloklar arasındaki benzerlik oranının bulunmasının ardından bu oranın belirlenmiş bir eşik değerine göre eşitlik 3.16' daki gibi kıyaslanmasıyla fark resmi D_{ij} oluşturulabilir. Eşik değerinden küçük benzerlik oranına sahip olan blokların pikselleri "0" değeri alırken aktif bloklarınki ise "1" değerini alır.

$$D_{ij}(x,y) = \begin{cases} 1, L_{ij} > Th \\ 0, DigerDurumlarda \end{cases} \qquad \text{Eşitlik 3.16}$$

Benzerlik oranının bulunacağı $f_1(x,y)$ ve $f_2(x,y)$ görüntüleri, t_{k-1} ve t_k zamanlarında ardışık olarak alınmış iki görüntü olabileceği gibi (dinamik referanslı), birinin referans(arka plan) diğerinin hareketli nesne içeren görüntü olduğu iki görüntü de olabilir(statik referanslı). Görüntüler Ardışık alındığında fark resminde objenin hızına veya alınan görüntülerin zaman farkına bağlı olarak nesne üzerinde yapay bir kuyruk olumla olasılığı yüksektir. Diğer durumda ise fark resminde ani ışık yoğunluğu değişimlerine bağlı olarak gürültüye rastlanabilmektedir. Bu durumda gürültü analizi veya belirli bir değerden küçük alanların temizlenmesi yoluna uzun vadede ise herhangi bir arka plan güncellemesi yöntemine başvurulması uygundur.Benzerlik oranı yöntemi için geliştirilen algoritmanın akışı şu şekildedir. MxM piksele sahip bloklara bölünmüş iki görüntüdeki her bloğun ortalama ve standart sapması hesaplanarak blokları temsil eden ve bu değerleri tutan iki boyutlu matrislere aktarılır.

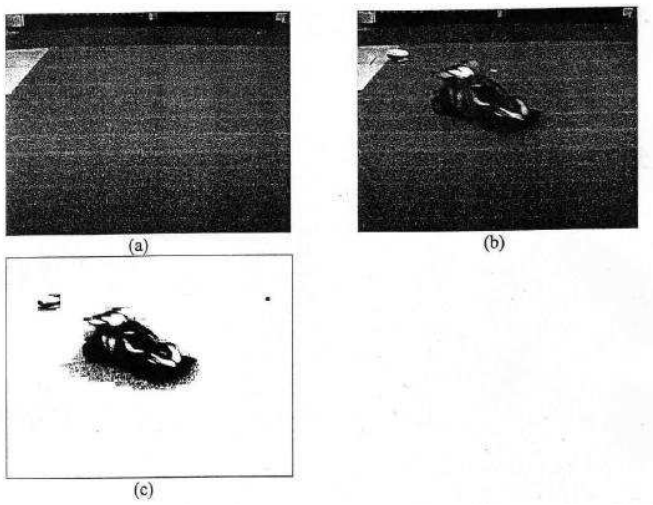
Her iki resim için
Her $blok_{ij}$ için
Ortalamayı hesapla
Standart sapmayı hesapla
$O1_{ij}$=Ortalama, $S1_{ij}$ = Standart Sapma
$O2_{ij}$ = Ortalama, $S2_{ij}$ Standart Sapma

İki görüntüye ait tüm blokların karakteristiğinin aktarıldığı matrislerden faydalanılarak iki görüntüde aynı koordinata denk gelen matris değerlerinden eşitlik 3.15' teki gibi benzerlik ve iki boyutlu bir matrise aktarılır. Bu benzerlik oranları da eşitlik 3.16' ya göre değerlendirilerek fark resmi oluşturulur.

Her $O1_{ij}$,$S1_{ij}$,$O2_{ij}$,$S2_{ij}$ elemanı için
Benzerlik oranı hesapla
L_{ij} = Benzerlik oranı

Her $blok_{ij}$ için
Eğer $L_{ij} > Th$ ise
Her eleman (x,y) için
$D_{ij}(x,y) = 1$
Değil ise
Her eleman (x,y) için $D_{ij}(x,y) = 0$

Benzerlik oranı bulunacak görüntülerin dinamik veya statik referanslı olması durumunda elde edilecek fark resimleri yukarıda verilmiş olan algoritmaya göre Şekil 3.6 ve Şekil 3.7' deki olmaktadır.



Şekil 3.5. Statik referanslı benzerlik oranı yöntemi a)statik referans görüntüsü b) incelenen görüntü c) elde edilen fark resmi

Şekil 3.6. Dinamik referanslı benzerlik oranı yöntemi a)dinamik referans görüntüsü b) incelenen görüntü c) elde edilen fark resmi

3.3.2.2. Maksimum Fark Yöntemi

Bu yöntemde de, diğer istatistiksel yöntemlerde olduğu gibi t_k zamanında alınan $f(x,y,t_k)$ görüntüsündeki değişimin belirlenebilmesi için piksellerin önceden karakterize edilmiş olması gereklidir. Bu yöntem pikselleri karakterize etmek için $[t_o,t_{k-1}]$ zaman aralığına alınan $f(x,y,t_o)$, $f(x,y,t_1)$… $f(x,y,t_{k-2})$, $f(x,y,t_{k-1})$ görüntülerinden her piksel için elde ettiği şu üç parametreyi kullanmaktadır.

N:$[t_o,t_{k-1}]$ zaman aralığında ölçülen en küçük değeri

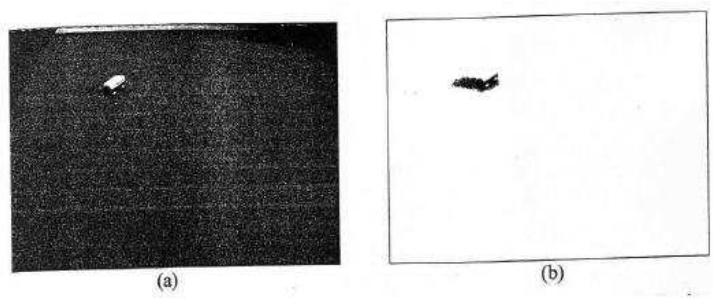
M:$[t_o,t_{k-1}]$ zaman aralığında alınan en büyük değeri

MD: :[t_o,t_{k-1}] zaman aralığında alınan, ardışık görüntüler arasındaki maksimum mutlak fark

Bu parametrelere göre f(x,y,t_k) görüntüsündeki her piksel eşitlik 3.16 daki gibi aktif veya durağan olarak işaretlenir.

$$D_{xy} = \begin{cases} 1, \left|N_{xy} - f(x,y,t_k\right| > MD_{xy} V \left|M_{xy} - f(x,y,t_k)\right| > MD_{xy} \\ 0, DigerDurumlarda \end{cases} \qquad \text{Eşitlik 3.16}$$

Ancak eşitlik 3.16 ışık yoğunluğu değişiminden oluşacak şekilde gürültüden dolayı temiz bir hareketle obje görüntüsü elde etmek için yeterli olmayabilir. Fark alma işleminden sonra oluşan ikilik fark resminin bir dizi morfolojik işlemden geçirilmesi gerekmektedir. Tek piksellik gürültülerin temizlenmesinin ardından hareketli bölgelerin belirlenmesi için bağlı bileşenler analizi (connected - component analysis) yapılır ve istenenden küçük alana sahip bölgeler yok edilir. Böylece şekil 3.8.b'de görülebileceği gibi, fark resmindeki gürültülerden büyük oranda kurtulunmuş olur. Gürültü seviyesinin en alt seviyede tutulması için piksel karakteristiklerinin bir başka deyişle arka plan modelinin yeterince sık güncellenmesi gerekmektedir. Zira gürültü seviyesinin üst seviyelerde olması işlem zamanının istenmeyen ölçüde arttırmakta dolayısıyla da birim zamanda işlenen görüntü sayısını azaltmaktadır.



Şekil 3.7. Maksimum fark yöntemi a)hareketin incelendiği görüntü b) gürültülerden temizlenmiş fark resmi

Bu yöntemin uygulanması için geliştirilen algoritmanın işleyişi sırasıyla şu şekildedir:

$[t_o,t_{k-1}]$ zaman aralığında alınan görüntülerden her piksel için bulunan maksimum, minimum ve maksimum fark değerleri iki boyutlu matrislere aktarılır.

0'dan K-1'e her resim için
Her resim elemanı için
N_{xy} = Minimum eleman
M_{xy} = Maksimum eleman
MD_{xy} = Maksimum fark

t_k zamanında alınan $f(x,y,t_k)$ görüntüsündeki değişim matrislere aktarılan değerlere göre eşitlik 3.16' dan tespit edilerek fark resmi oluşturulur.

Her resim elemanı için
Eğer mutlak farklardan biri > MD_{xy} ise D_{xy}=1
Değil ise D_{xy}=0

Fark resmi oluşturulduktan sonra tek piksellik gürültüler temizlenir. Bağlı bileşenler analizi yapılır ve alanı belirlenmiş bir eşik değerinin altında olan bölgeler elemine edilir.

Her resim elemanı için
Eğer D_{xy} = 1 ise ve her komşu eleman = 0 ise D_{xy} = 0
Eğer D_{xy} = 1 ise
Eğer fonksiyon(say) < Th ise
Fonksiyon(sil)

Son adım olarak belirli periyotlarda (7-10 sn) N,M ve MD değerleri güncellenerek gürültüye daha az hassas dinamik bir yapı oluşturulabilir.

3.3.2.3. Ağırlık Merkezi Analizi Yöntemi

Dinamik görüntü analizinde nesne tanımlamada sıkça kullanılan moment teoremine göre eğer F(x,y) fonksiyonu, xy düzleminde sürekliyse ve "0" dan farklı en az bir değere sahipse her dereceden momenti vardır ve bu moment dizisi (m_{pq}), F(x,y) fonksiyonuna özeldir. Sayısal görüntülerin iki boyutlu birer fonksiyon olmalarından dolayı, her görüntünün kendine özel bir ağırlık merkezi olmalıdır. Moment teoremine göre, (p+q) dereceden moment şöyle tanımlanmıştır.

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q F(x,y)dxdy \qquad \text{Eşitlik 3.17}$$

Eşitlik 3.17, f(x,y) gibi sayısal görüntüdeki (p+q) dereceden momentleri için yeniden düzenlenirse eşitlik 3.18'deki gibi olur.

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y) \qquad \text{Eşitlik 3.18}$$

Sayısal görüntülerde ağırlık merkezi koordinatı (x_c,y_c), eşitlik (3.18) kullanılarak hesaplanan momentler yardımıyla bulunur.

$$x_c = \frac{m_{10}}{m_{00}} \qquad \text{Eşitlik 3.19}$$

$$y_c = \frac{m_{01}}{m_{00}} \qquad \text{Eşitlik 3.20}$$

Sayısal görüntülerde ağırlık merkezi global olabileceği gibi bloklara bölünmüş görüntülerdeki her blok için de bulunabilir. Gerek genel ağırlık merkezi, gerekse lokal ağırlık merkezi, gürültülere çok fazla hassas olmamasından ve bulunduğu alana özel olduğundan, özellikle nesne tanımlama algoritmalarında sıkça kullanılmaktadır.

Bu yöntem ağırlık merkezinin bu özelliklerini kullanarak bir hareket tespiti yaklaşımında bulunmuştur. Yönteme göre, [t_o,t_{k-1}] aralığında alınan ve MxM piksele sahip yatayda i, düşeyde j sayıda bloğa bölünmüş $f(x,y,t_o)$,…, $f(x,y,t_{k-1})$

görüntülerinin her bloğunun ağırlık merkezi koordinatı bulunur. Her blok için elde edilen K adet değerden, eşitlik 3.10 ve eşitlik 3.11' den ortalama (μ_{xc} , μ_{yc}) ve standart sapmalar (σ_{xc},σ_{yc}) bulunarak her bloğun ağırlık merkezi karakteristiği çıkarılmış olur.

İncelenecek f(x,y,t_k) görüntüsü de yatayda i, düşeyde j sayıda bloğa bölünerek, her bloğun ağırlık merkezi koordinatı (x_c,y_c) bulunur. Her koordinatın "x" ve "y" bileşenleri, karakteristiği çıkarılmış istatistiksel model ile eşitlik 3.21' deki gibi ayrı ayrı karşılaştırılarak fark resmi oluşturulur.

$$D_{ij}(x,y)=\begin{cases}1, \left|x_c(i,j)-\mu_{x_c}(i,j)\right| > N\sigma_{x_c}(i,j)\\ 1, \left|y_c(i,j)-\mu_{y_c}(i,j)\right| > N\sigma_{y_c}(i,j)\\ 0, DigerDurumlarda\end{cases} \qquad \text{Eşitlik 3.21}$$

Eşitlik 3.21 'de eşik değeri olarak kullanılan $N\sigma_{xc}(i,j)$ ve $N\sigma_{yc}(i,j)$ değerlerindeki N katsayısı 2 ile 3 arasında bir değerdir. Çünkü ortalaması ve standart sapması hesaplanan değerlerin normal dağılıma sahip olduğu düşünülerek 3-sigma kuralına uyulmuştur. Fark resmi, karşılaştırma sonucu bloğun alacağı "1" veya "0" değeri bloğa ait tüm piksellere uygulanması suretiyle oluşturulur. Oluşturulan fark resminde gürültü analizine ihtiyaç yoktur ancak tercihen alan analizi yapılarak belli bir alandan küçük olan bölgeler elemine edilebilir. Bu yöntem için geliştirilen algoritmanın işlem sırası ve uygulanan yöntemler şu şekildedir:

Öncelikle, blok ağırlık merkezlerinin karakteristiğinin belirlenmesi için, [t_o,t_{k-1}] aralığında alınan f(x,y,t_o), f(x,y,t_1)…. f(x,y,t_{k-2}), f(x,y,t_{k-1}) görüntüleri, MxM piksele sahip, yatayda i ve düşeyde j sayıda bloğa bölünür. Her bloğun ağırlık merkezi koordinatları bulunarak üç boyutlu matrislere aktarılır.

0'dan K-1'e her resim için
Her blok (i,j) için
Cmx_{ijk} = Ağırlık Merkezi x bileşeni
cmx_{ijk} = Ağırlık Merkezi y bileşeni

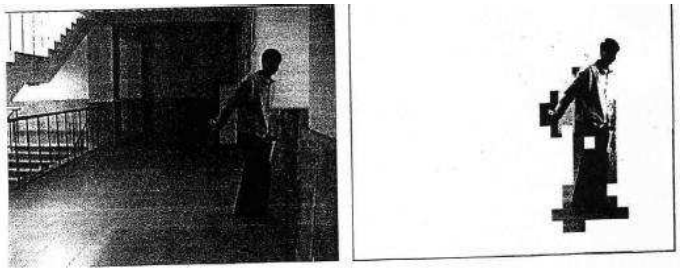
Üç boyutlu matrislere aktarılan değerlerden, blokların ağırlık merkezi ortalamaları ve standart sapmaları bulunarak iki boyutlu matrislere aktarılır.

Her cmx_{ijk} ve cmx_{ijk} elemanı için
Her μ_{xij}, $\mu_{yij,}$ σ_{xij} ve σ_{yij} elemanı için
μ_{xij} = Ağırlık Merkezi x bileşeni ortalaması
μ_{yij} = Ağırlık Merkezi y bileşeni ortalaması
σ_{xij} = Ağırlık Merkezi x bileşeni standart sapması
σ_{yij} = Ağırlık Merkezi y bileşeni standart sapması

$f(x,y,t_k)$ görüntüsü yatayda i, düşeyde j sayıda bloğa bölünerek, her bloğun ağırlık merkezi koordinatı (x_c,y_c) bulunarak, blok karakteristikleri ile karşılaştırılır ve fark resmi oluşturulur.

Her blok (i,j) için
Eğer $x_c-\mu_{xij} > N\sigma_{xij}$ ise veya $y_c-\mu_{yij} > N\sigma_{yij}$ ise
Her blok elemanı (x,y) için
D(x,y) = 0

Bu yöntem için geliştirilen algoritmanın elde ettiği fark resmi Şekil 3.9' da görülmektedir.



Şekil 3.8. Ağırlık merkezi analizi yöntemiyle 11x11 piksele sahip bloklara bölünmüş görüntülerden edilen fark resmi

3.3.2.4. Hough Transform Yöntemi

Hareket tespiti yöntemlerinden biri de $f_1(x,y)$ ve $f_2(x,y)$ görüntülerindeki hareketli nesneye ait hareket parametrelerinin (dx,dy), Hough Transform (HT) kullanılarak bulunmasıdır. HT' nin dayandığı temel nokta, resim üzerindeki her noktanın, parametre uzayında, bir eğri olarak ifade edilmesidir. HT kullanılarak yapılan hareket tespitinde, hareketli nesneye ait tüm piksel gruplarının belli bir doğrultuda aynı şekilde hareket ettiği varsayılır. Diğer bir varsayım ise hareketli tek bir nesnenin olduğu ve nesnenin şeklinin değişmediğidir.

Bu yöntemde öncelikle gri seviyede alınan ve tek bir hareketli nesneye sahip olan $f_1x,y)$ ve $f_2(x,y)$ görüntüleri kenar tesbiti algoritmalarıyla ikilik resme dönüştürülür. (($f_{1o}(x,y)$ ve $f_{2o}(x,y)$). Bu ikilik resimler birbirlerinden eşitlik 3.22 ve 3.23' deki gibi çıkarılarak hareketli nesnenin belirginleşmesi sağlanır.

$$f_1(x,y) = f_{1_0}(x,y) - f_{2_0}(x,y) \qquad \text{Eşitlik 3.22}$$

$$f_2(x,y) = f_{2_0}(x,y) - f_{1_0}(x,y) \qquad \text{Eşitlik 3.23}$$

Piksel gruplarıyla işlem yapılacağı için tek piksellik gürültülerin temizlenmesi gerekmektedir. Gürültülerin temizlenmesinin ardından, her resimden ikişer piksel alınarak aralarındaki yer değiştirmeler eşitlik 3.24 ve 3.25' den hesaplanır.

$$dx = \left| f_{2_{mx}} - f_{ix} \right| = \left| f_{2_{mx}} - f_{1_{jx}} \right| \qquad \text{Eşitlik 3.24}$$

$$dy = \left| f_{2_{my}} - f_{1_{iy}} \right| = \left| f_{2_{my}} - f_{1_{jy}} \right| \qquad \text{Eşitlik 3.25}$$

Her iki resimdeki piksel çiftinin kendi içindeki yer değiştirmesi eşit çıkarsa akümülatör A(dx,dy)'deki bu yer değiştirme değerleri puanlanarak saklanır. Puanlama işlemi, gelen her yer değiştirme değerinin akümülatördeki ilgili değeri "1" arttırması şeklinde olur. Resimlerdeki tüm aktif piksel çiftleri değerlendirildikten sonra, akümülatörde en çok oyu alan dx ve dy yer değiştirmeleri, nesnedeki piksel

gruplarının yani nesnenin yer değiştirme miktarıdır. İki resimdeki aktif pikseller, bulunan dx ve dy' lere göre karşılaştırılarak hareketli nesne belirlenir.

Yer değiştirme hesaplamaları ile yer değiştirme değerlerinin akümülatör A(dx,dy)'ye aktarılması HT yönteminin motorudur ve yöntemin başarısı bu işlemlerin doğruluğuna bağlıdır. Bu yöntem için geliştirilen algoritmanın işleyişi sırasayla şöyledir:

(1)$f_1(x,y)$ ve $f_2(x,y)$ görüntülerinin kenar tespiti yöntemiyle $f_{1o}(x,y)$ ve $f_{2o}(x,y)$ ikilik resimlerini oluştur.

(2)Tek piksellik gürültüleri temizler.

(3)Arka planı temizleyerek yeni ikilik resimler oluştur.

$$f_1(x,y) = f_{1_0}(x,y) - f_{2_0}(x,y)$$
$$f_2(x,y) = f_{2_0}(x,y) - f_{1_0}(x,y)$$

(4)Tek piksellik gürültüleri temizle

(5)

$$f_1(x,y)' den(f_{1_{ix}}, f_{1_{iy}}) ve(f_{1_{jx}}, f_{1_{jy}}),$$
$$f_2(x,y)' den(f_{2_{mx}}, f_{2_{my}}) ve(f_{2_{mx}}, f_{2_{my}})$$

olmak üzere ikişer piksel seç.

(6)Eğer

$$(\left|f_{1_{jx}} - f_{1_{ix}}\right| = \left|f_{2_{nx}} - f_{2_{mx}}\right|) VE (\left|f_{1_{jy}} - f_{1_{iy}}\right| = \left|f_{2_{ny}} = f_{2_{my}}\right|)$$

ise dx ve dy 'yi hesapla ve (7)'ye git
değilse (5)' e git

(7) Akümülatör A(dx,dy)'yi "1" arttır.

(8)Eğer $f_1(x,y)$ ve $f_2(x,y)$'deki bütün pikseller işleme alındı ise (9)'a git değil ise (5)'e git

(9)A(dx,dy)'deki en yüksek oy alan dx ve dy'den $f_2(x,y)$'deki nesneye ait pikselleri işaretle.

3.3.3. Görsel Akış (Optical Flow) Yöntemleri

Bu yöntem genel olarak görüntüler arasındaki birbirine bağımlı hareketi tanımlamak için kullanılır. Görsel akış algoritmaları, görüntü dizisindeki değişen alanları belirlemek için, hareket eden nesnenin akış vektör karakteristiğini kullanır. Geçmiş yıllarda görsel akış hesaplamalarıyla ilgili literatüre bir çok yöntem sunulmuştur. Bu yöntemler kabaca gradyan, korelasyon, enerji ve faz temelli yaklaşımlar olarak 4 grupta sınıflandırılabilirler. Bu yaklaşımlardan en popüler olanı gradyan ve korelasyon temelli yaklaşımlardır. Bu yaklaşımlarla bir çok araştırmacı ardışık görüntülerdeki küçük değişimleri saymak, belirlemek dolayısıyla hareketi tespit etmek için, deplasman vektör alanını hesaplayan algoritmalar geliştirmiştir. Expection Maximization (EM) hesaplamalarıyla veya gruplandırılmış akış vektörlerinin Gauss yayılımlarının karakterize edilmesiyle bütünleşik olarak kullanan araştırmacılarda bu yönteme değişik bir bakış açısı getirmiştir. Ancak bazı araştırmacılara göre Görsel Akış yöntemi, hareketli nesneyi kamera koşullarından bağımsız olarak tespit edebilmesine karşın, çoğu hesaplamalarının karmaşıklığı ve hantallığı, gürültüye karşı aşırı hassasiyeti yüzünden özel ekipmanlar kullanılmadan uygulanması çok zor olan bir yöntemdir.

4. GÜVENLİK SİSTEMLERİ

4.1. Akıllı Güvenlik Sistemleri

Görüntü işleme teknolojisinin en çok kullanıldığı alanlardan biri de güvenliktir. Güvenliğin önemli olduğu bina ve ortamlarda etkin çözümler sunan sistemler sayesinde izinsiz girişlere, hırsızlığa ve bir çok tehlikeye karşı en yüksek seviyede korunmanız görüntü işleme teknolojisi ile mümkündür. Hızlı görüntü yakalama

kabiliyeti olan kameralar ve talepler doğrultusunda şekillendirilebilecek yazılımlar sayesinde binalara yapılacak izinsiz girişler, hırsızlık olayları ve şüpheli paketler akıllı güvenlik sistemleri sayesinde anında fark edilir ve ilgili birimler uyarılır. Geleneksel güvenlik kameraları ile bazen günler sonra fark edilen ve büyük kayıplara yol açabilecek hırsızlık ve benzeri olaylardan akıllı güvenlik sistemleri sayesinde anında haberdar olmak mümkündür.

Şekil 4.1. InfoDif'in akıllı güvenlik sistemleri

Yazılımlar tanımlanan alanlardaki hareketliliği, nesneleri, sistem tarafından tanımlanan anormal durumları takip ve tespit eder. İstenmeyen veya anormal durumlar oluştuğunda sesli veya görüntülü alarm harekete geçer ve güvenliği tehlikeye sokacak veya hırsızlık gibi kayıplara yol açacak olayların önlenmesine olanak sağlar.

5. KAMERA KULLANILARAK GÖRÜNTÜ İŞLEME YOLUYLA GERÇEK ZAMANLI GÜVENLİK UYGULAMASI

5.1. Uygulamanın algoritması

Bu uygulamada c# kullanılarak web kamera ile hareket tespitinin nasıl yapılabileceği gösterilmiştir. Uygulama içerisinde web kamera ile belli zaman periyotlarında görüntüler alınarak ilk yakalanan görüntü ile bir sonraki zamanda yakalanan görüntüler karşılaştırılır. Eğer aralarında büyük farklar var ise bu görüntüler diske kaydedilir.

Bu uygulamanın yapılabilmesi için çözülmesi gereken 2 büyük problem vardı:

1- Web kamera ile nasıl irtibat kurulabileceği ve görüntülerin web kamera ile nasıl yakalanabilecekleri.

2- Alınan iki görüntü nasıl karşılaştırılacak.

Bu problemlerin şu şekilde çözümlenme metodu bulundu.

1- Web kamera ile irtibat kurmanın 2 yolu vardır.

a) DirectX' in bileşenlerinden DirectShow kullanılarak.

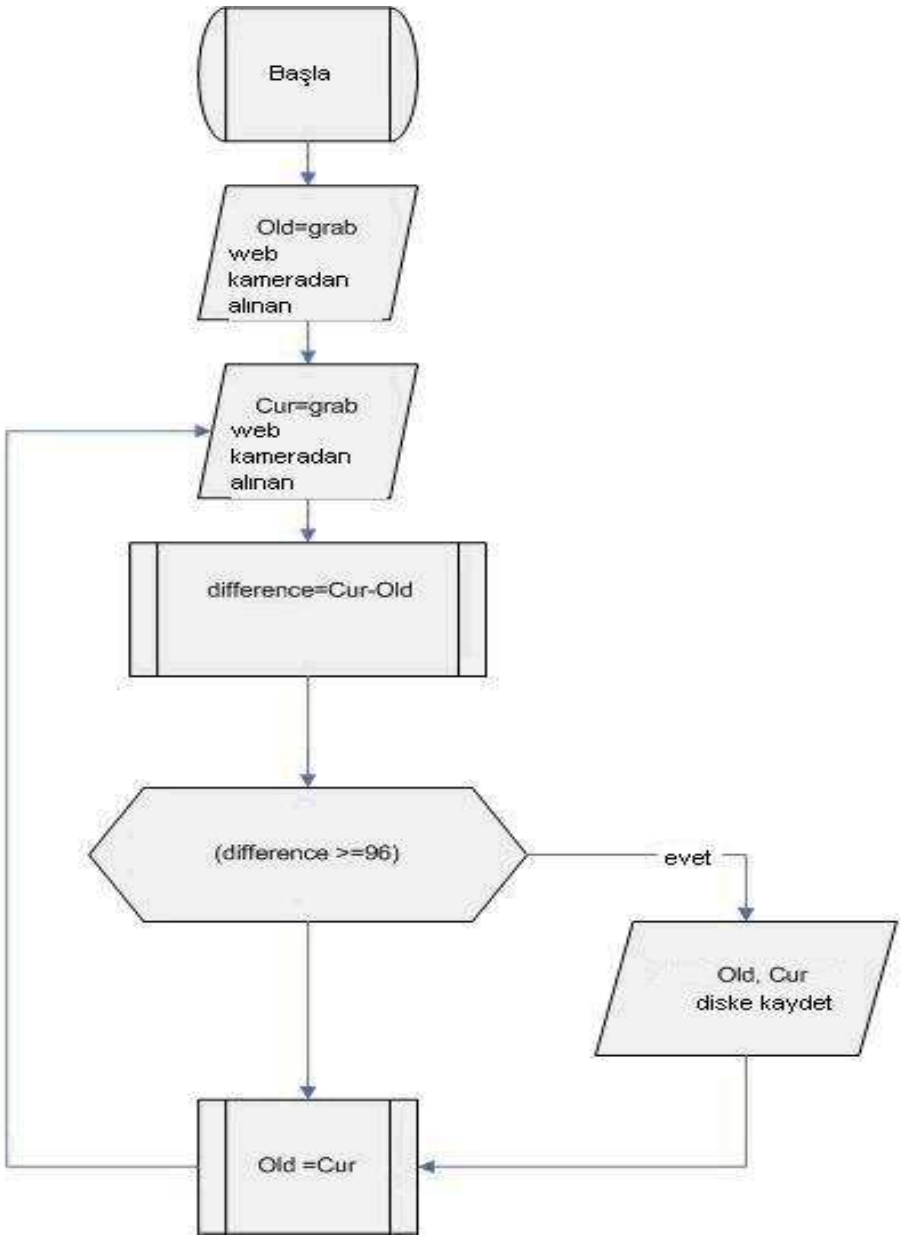
b) Cam server gibi 3rdPARTY kullanılarak.

2- Alınan 2 görüntünün karşılaştırılmasının 2 yolu vardır.

a) Görüntülerin piksel farklarının karşılaştırılması.

b) Filtreler kullanılarak köşegenler yakalanır. Daha sonra obje yeniden tanımlanır.

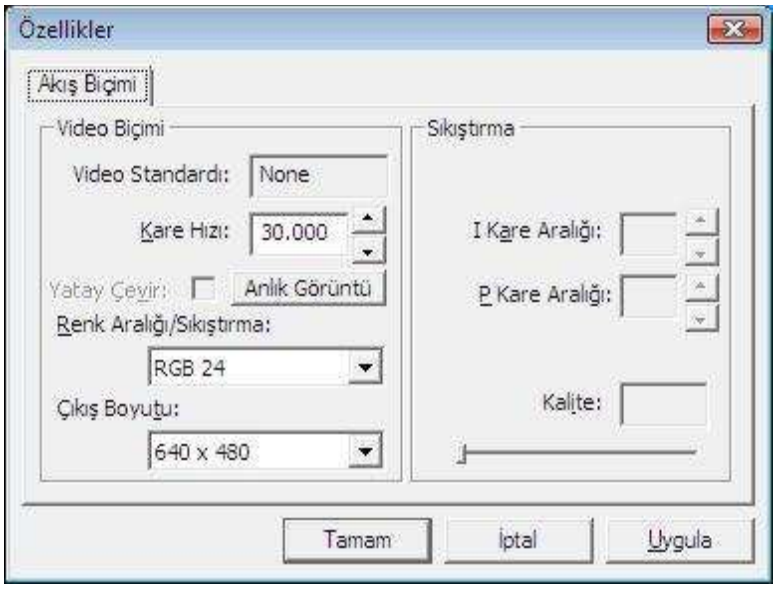
Programda kullanılan algoritma şu şekildedir:



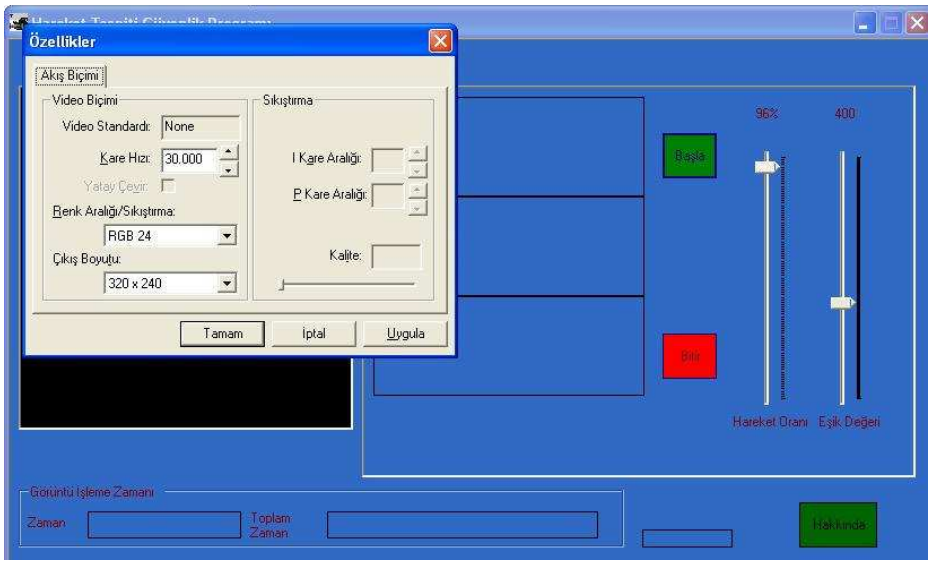
Web kameradan yakalanan ilk görüntü "old" olarak, ikinci olarak web kameradan yakalanan diğer görüntü "cur" olarak tanımlıdır. "Cur" ve "Old" olarak tanımlanan görüntülerin piksel farkları karşılaştırılır. Eğer farkları belirlenen değerden (bu değer web kameranın kalitesine ve o anda ki ortamın ışık durumuna bağlıdır) büyükse bu iki görüntüde kaydedilir.

5.2. Uygulama Yazılımı

Program çalıştırıldığında ilk önce web kameradan alınan görüntünün kare hızı, renk aralığı ve alınan görüntünün çıkış boyutu kullanıcı tarafından seçilir.

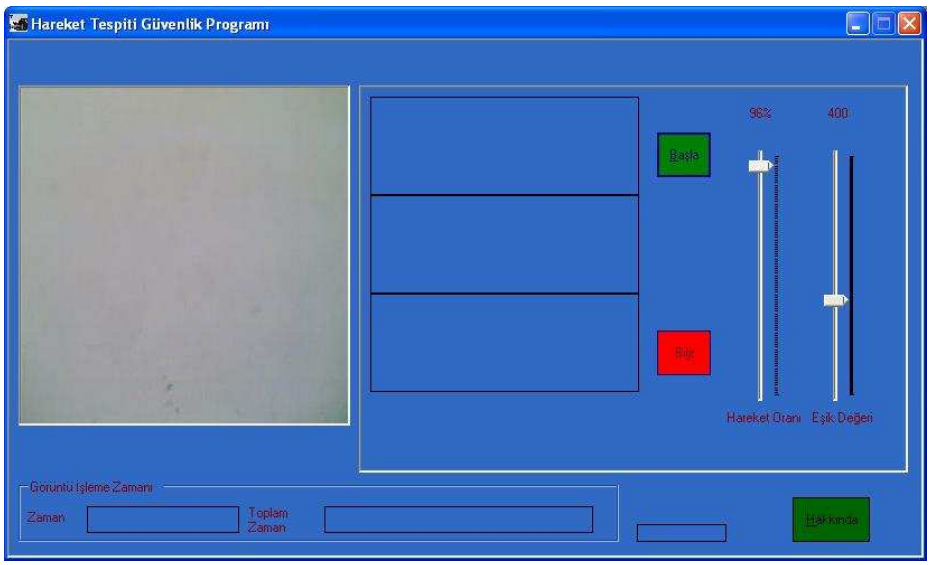


Şekil 5.1. Açılışta kullanıcı tercihleri seçer.



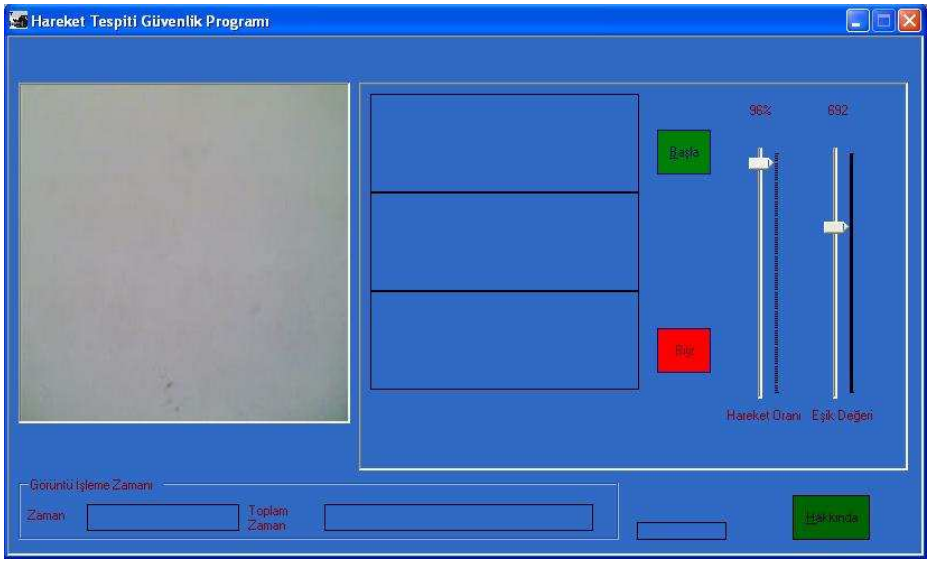
Şekil 5.2. Programın açılışı.

Kullanıcı tarafından yapılan tercihlerden sonra program web kameradan görüntü almaya başlar. Başla butonuna basılmadığından dolayı henüz program alınan görüntüleri karşılaştırmaya başlamamıştır.



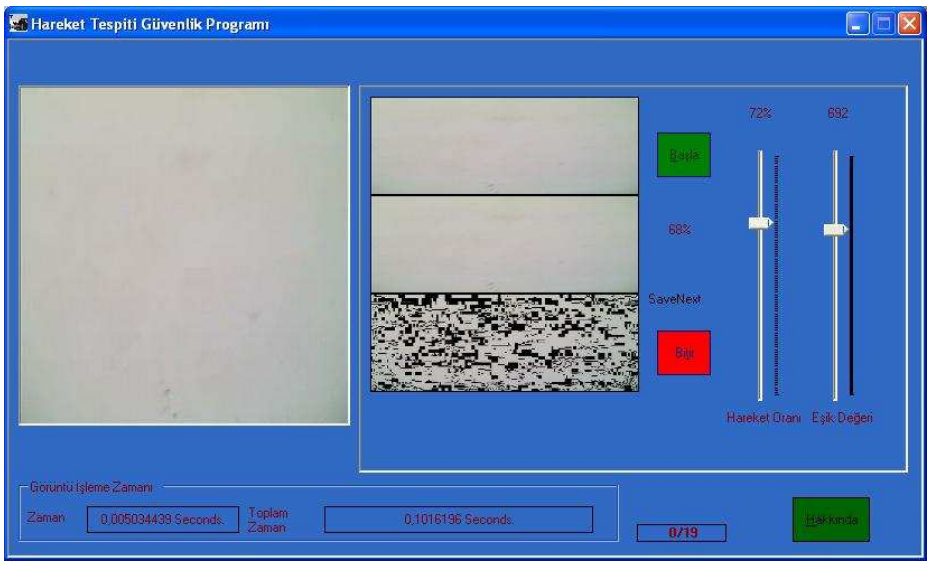
Şekil 5.3. Program web kameradan görüntü almaya başlıyor.

Kullanıcı başla butonuna bastığında program alınan görüntüleri işlemeye ve belli zaman aralığında alınan görüntüleri karşılaştırıp pikseller arasındaki farka göre konumda hareket olup olmadığına karar vermeye başlar.



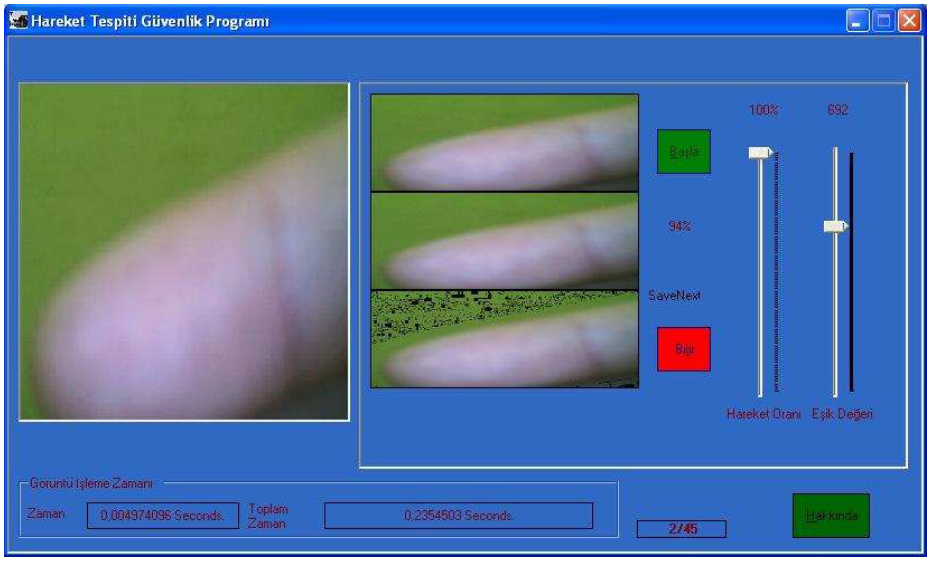
Şekil 5.4. Eşik Değeri Seçimi

Uygulamada hareketi tespit etmek için gerekli olan eşik değeri ortamdaki ışık durumuna göre veya güvenlik için yeterli olan hareketi tespit etmek için kullanıcı tarafından programda seçilir



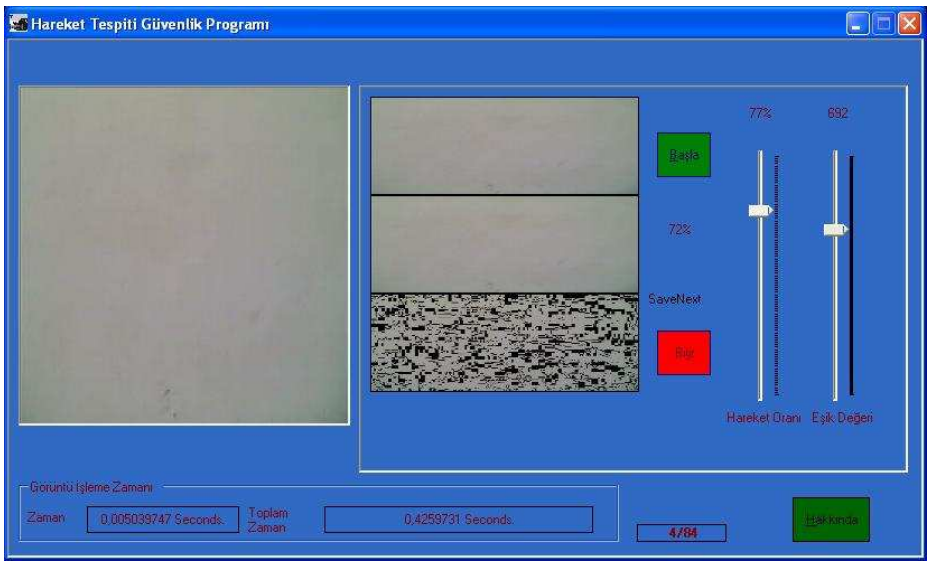
Şekil 5.5. Programda başla butonuna basıldıktan sonra görüntüler karşılaştırılıyor.

Program görüntüler arasındaki piksel farkının belli bir değeri geçmesi durumunda ses ile uyarıp alarm verir. Hareket başladıktan sonra web kameradan alınan görüntüler kaydedilmeye başlar.

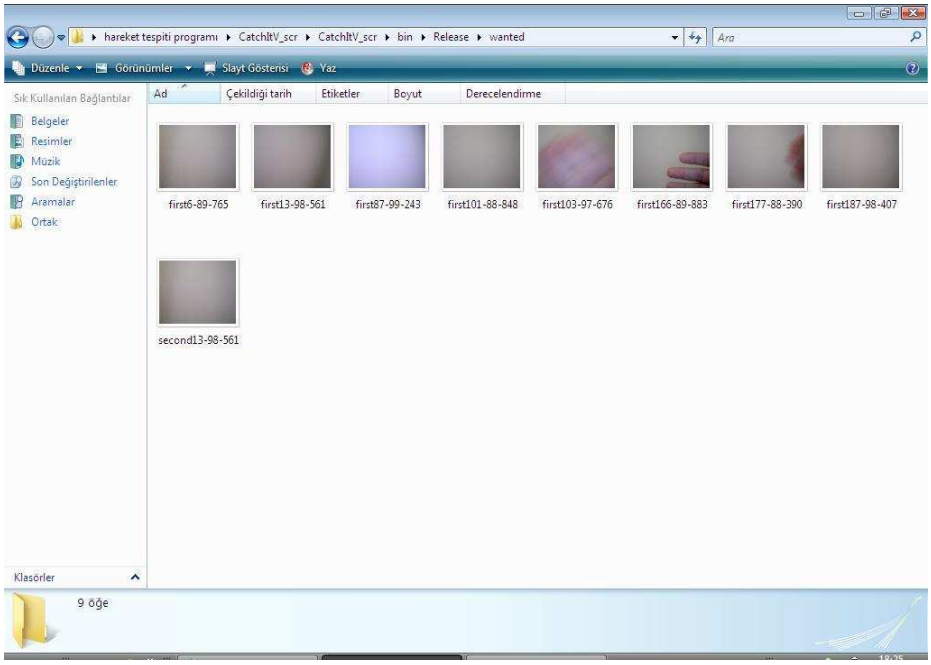


Şekil 5.6. Program hareket algıladığında sesli bir alarm veriyor.

Daha önce belirlenen şekilde hareket tespitinden sonraki web kameradan alınan görüntüler programda belirlenen dizin içinde saklanır . Böylece ortamda beklenmeyen şekilde gözlemlenen hareket resimlenerek hareketin kaynağı ortaya çıkmış olur.

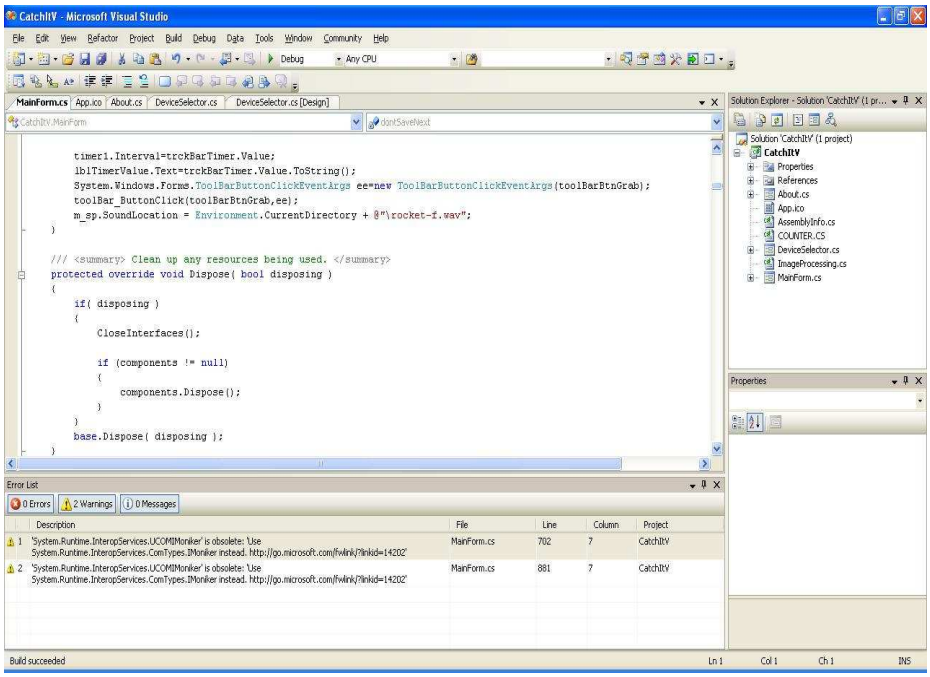


Şekil 5.7. Bitir butonuna basarak program alınan görüntüleri karşılaştırmayı durdurur.



Şekil 5.8. Program hareket tespitinden sonra alınan görüntüleri belirlenen dizin içerisinde kaydediyor.

Şekil 5.9. Hakkında



Şekil.5.10. Uygulamanın C# ile derlenmesi.

6. EKLER

EK1

MainForm.cs

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Runtime.InteropServices;
using System.IO;
using System.Media;
using System.Diagnostics;
using ImgProcessing;
using Timing;
using System.Threading;

using DShowNET;
using DShowNET.Device;


namespace CatchItV
{
public class MainForm : System.Windows.Forms.Form, ISampleGrabberCB
{
        bool dontSaveNext=true,firstRun=true;
        private  Int32 cPercent=96,minSave=96,count=0,count2=0;
        string saveTime="";
        private Counter counter=new Counter();
        private  float sec=0;
        Bitmap a;
        Bitmap wanted;
```

```
        private System.Windows.Forms.Splitter splitter1;
        private System.Windows.Forms.ToolBar toolBar;
        private System.Windows.Forms.Panel videoPanel;
        private System.Windows.Forms.Panel stillPanel;
        private System.Windows.Forms.PictureBox pictureBox;
        private System.Windows.Forms.ToolBarButton toolBarBtnTune;
        private System.Windows.Forms.ToolBarButton toolBarBtnGrab;
        private System.Windows.Forms.ToolBarButton toolBarBtnSave;
        private System.Windows.Forms.ToolBarButton toolBarBtnSep;
        private System.Windows.Forms.ImageList imgListToolBar;
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Label lblPercent;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.Label lblSavePercent;
        private System.Windows.Forms.Label lblSaveOrNot;
        private System.Windows.Forms.PictureBox pictureBox2;
        private System.Windows.Forms.Button btnStart;
        private System.Windows.Forms.GroupBox groupBoxImageProcTime;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label lblTotalTime;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label lblTime;
        private System.Windows.Forms.Label lblCount;
        private System.Windows.Forms.Button btnStop;
        private System.Windows.Forms.Button btnAbout;
        private System.Windows.Forms.Label lblTimerValue;
        private System.Windows.Forms.TrackBar trckBarTimer;
        private System.Windows.Forms.TrackBar trckBarSaveValue;
        private System.ComponentModel.IContainer components;
    private SoundPlayer m_sp = new SoundPlayer();

        public MainForm()
        {
                // Required for Windows Form Designer support
```

```
                InitializeComponent();

                trckBarSaveValue.Value=cPercent;
                lblSavePercent.Text=cPercent+"%";

                timer1.Interval=trckBarTimer.Value;
                lblTimerValue.Text=trckBarTimer.Value.ToString();
                System.Windows.Forms.ToolBarButtonClickEventArgs ee=new
ToolBarButtonClickEventArgs(toolBarBtnGrab);
                toolBar_ButtonClick(toolBarBtnGrab,ee);
        m_sp.SoundLocation = Environment.CurrentDirectory + @"\rocket-f.wav";
        }
        protected override void Dispose( bool disposing )
        {
                if( disposing )
                {
                        CloseInterfaces();

                        if (components != null)
                        {
                                components.Dispose();
                        }
                }
                base.Dispose( disposing );
        }

                #region Windows Form Designer generated code

                private void InitializeComponent()
                {
        this.components = new System.ComponentModel.Container();
        System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(MainForm));
        this.toolBar = new System.Windows.Forms.ToolBar();
```

```
            this.toolBarBtnTune = new System.Windows.Forms.ToolBarButton();
            this.toolBarBtnGrab = new System.Windows.Forms.ToolBarButton();
            this.toolBarBtnSep = new System.Windows.Forms.ToolBarButton();
            this.toolBarBtnSave = new System.Windows.Forms.ToolBarButton();
            this.imgListToolBar = new
System.Windows.Forms.ImageList(this.components);
            this.videoPanel = new System.Windows.Forms.Panel();
            this.splitter1 = new System.Windows.Forms.Splitter();
            this.stillPanel = new System.Windows.Forms.Panel();
            this.lblTimerValue = new System.Windows.Forms.Label();
            this.trckBarTimer = new System.Windows.Forms.TrackBar();
            this.btnStop = new System.Windows.Forms.Button();
            this.btnStart = new System.Windows.Forms.Button();
            this.pictureBox2 = new System.Windows.Forms.PictureBox();
            this.lblSaveOrNot = new System.Windows.Forms.Label();
            this.lblSavePercent = new System.Windows.Forms.Label();
            this.trckBarSaveValue = new System.Windows.Forms.TrackBar();
            this.lblPercent = new System.Windows.Forms.Label();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.pictureBox = new System.Windows.Forms.PictureBox();
            this.timer1 = new System.Windows.Forms.Timer(this.components);
            this.groupBoxImageProcTime = new System.Windows.Forms.GroupBox();
            this.label4 = new System.Windows.Forms.Label();
            this.lblTotalTime = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.lblTime = new System.Windows.Forms.Label();
            this.lblCount = new System.Windows.Forms.Label();
            this.btnAbout = new System.Windows.Forms.Button();
            this.stillPanel.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.trckBarTimer)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox2)).BeginInit();
```

```
((System.ComponentModel.ISupportInitialize)(this.trckBarSaveValue)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
        ((System.ComponentModel.ISupportInitialize)(this.pictureBox)).BeginInit();
        this.groupBoxImageProcTime.SuspendLayout();
        this.SuspendLayout();
        //
        // toolBar
        //
        this.toolBar.Buttons.AddRange(new
System.Windows.Forms.ToolBarButton[] {
        this.toolBarBtnTune,
        this.toolBarBtnGrab,
        this.toolBarBtnSep,
        this.toolBarBtnSave});
        this.toolBar.Cursor = System.Windows.Forms.Cursors.Hand;
        this.toolBar.DropDownArrows = true;
        this.toolBar.Enabled = false;
        this.toolBar.ImageList = this.imgListToolBar;
        this.toolBar.Location = new System.Drawing.Point(0, 0);
        this.toolBar.Name = "toolBar";
        this.toolBar.ShowToolTips = true;
        this.toolBar.Size = new System.Drawing.Size(774, 42);
        this.toolBar.TabIndex = 0;
        this.toolBar.Visible = false;
        this.toolBar.ButtonClick += new
System.Windows.Forms.ToolBarButtonClickEventHandler(this.toolBar_ButtonClic
k);
        //
        // toolBarBtnTune
        //
        this.toolBarBtnTune.Enabled = false;
        this.toolBarBtnTune.ImageIndex = 0;
```

```
            this.toolBarBtnTune.Name = "toolBarBtnTune";
            this.toolBarBtnTune.Text = "Tune";
            this.toolBarBtnTune.ToolTipText = "TV tuner dialog";
            //
            // toolBarBtnGrab
            //
            this.toolBarBtnGrab.ImageIndex = 1;
            this.toolBarBtnGrab.Name = "toolBarBtnGrab";
            this.toolBarBtnGrab.Text = "Grab";
            this.toolBarBtnGrab.ToolTipText = "Grab picture from stream";
            //
            // toolBarBtnSep
            //
            this.toolBarBtnSep.Enabled = false;
            this.toolBarBtnSep.Name = "toolBarBtnSep";
            this.toolBarBtnSep.Style =
System.Windows.Forms.ToolBarButtonStyle.Separator;
            //
            // toolBarBtnSave
            //
            this.toolBarBtnSave.Enabled = false;
            this.toolBarBtnSave.ImageIndex = 2;
            this.toolBarBtnSave.Name = "toolBarBtnSave";
            this.toolBarBtnSave.Text = "Save";
            this.toolBarBtnSave.ToolTipText = "Save image to file";
            //
            // imgListToolBar
            //
            this.imgListToolBar.ImageStream =
((System.Windows.Forms.ImageListStreamer)(resources.GetObject("imgListToolBa
r.ImageStream")));
            this.imgListToolBar.TransparentColor = System.Drawing.Color.Transparent;
            this.imgListToolBar.Images.SetKeyName(0, "");
            this.imgListToolBar.Images.SetKeyName(1, "");
```

```
            this.imgListToolBar.Images.SetKeyName(2, "");
            //
            // videoPanel
            //
            this.videoPanel.BackColor = System.Drawing.Color.Black;
            this.videoPanel.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.videoPanel.Location = new System.Drawing.Point(8, 40);
            this.videoPanel.Name = "videoPanel";
            this.videoPanel.Size = new System.Drawing.Size(296, 304);
            this.videoPanel.TabIndex = 1;
            this.videoPanel.Resize += new
System.EventHandler(this.videoPanel_Resize);
            //
            // splitter1
            //
            this.splitter1.Location = new System.Drawing.Point(0, 42);
            this.splitter1.Name = "splitter1";
            this.splitter1.Size = new System.Drawing.Size(5, 363);
            this.splitter1.TabIndex = 2;
            this.splitter1.TabStop = false;
            //
            // stillPanel
            //
            this.stillPanel.AutoScroll = true;
            this.stillPanel.AutoScrollMargin = new System.Drawing.Size(8, 8);
            this.stillPanel.AutoScrollMinSize = new System.Drawing.Size(32, 32);
            this.stillPanel.BackColor = System.Drawing.SystemColors.MenuHighlight;
            this.stillPanel.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.stillPanel.Controls.Add(this.lblTimerValue);
            this.stillPanel.Controls.Add(this.trckBarTimer);
            this.stillPanel.Controls.Add(this.btnStop);
            this.stillPanel.Controls.Add(this.btnStart);
            this.stillPanel.Controls.Add(this.pictureBox2);
            this.stillPanel.Controls.Add(this.lblSaveOrNot);
```

```
            this.stillPanel.Controls.Add(this.lblSavePercent);
            this.stillPanel.Controls.Add(this.trckBarSaveValue);
            this.stillPanel.Controls.Add(this.lblPercent);
            this.stillPanel.Controls.Add(this.pictureBox1);
            this.stillPanel.Controls.Add(this.pictureBox);
            this.stillPanel.Location = new System.Drawing.Point(312, 40);
            this.stillPanel.Name = "stillPanel";
            this.stillPanel.Size = new System.Drawing.Size(459, 302);
            this.stillPanel.TabIndex = 3;
            //
            // lblTimerValue
            //
            this.lblTimerValue.Location = new System.Drawing.Point(400, 16);
            this.lblTimerValue.Name = "lblTimerValue";
            this.lblTimerValue.Size = new System.Drawing.Size(40, 24);
            this.lblTimerValue.TabIndex = 10;
            this.lblTimerValue.Text = "400";
            //
            // trckBarTimer
            //
            this.trckBarTimer.LargeChange = 300;
            this.trckBarTimer.Location = new System.Drawing.Point(403, 48);
            this.trckBarTimer.Maximum = 1000;
            this.trckBarTimer.Minimum = 7;
            this.trckBarTimer.Name = "trckBarTimer";
            this.trckBarTimer.Orientation =
System.Windows.Forms.Orientation.Vertical;
            this.trckBarTimer.Size = new System.Drawing.Size(45, 240);
            this.trckBarTimer.SmallChange = 10;
            this.trckBarTimer.TabIndex = 9;
            this.trckBarTimer.Value = 400;
            this.trckBarTimer.ValueChanged += new
System.EventHandler(this.trckBarTimer_ValueChanged);
            //
```

```
            // btnStop
            //
            this.btnStop.BackColor = System.Drawing.Color.DarkGreen;
            this.btnStop.Enabled = false;
            this.btnStop.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.btnStop.Location = new System.Drawing.Point(264, 248);
            this.btnStop.Name = "btnStop";
            this.btnStop.Size = new System.Drawing.Size(48, 40);
            this.btnStop.TabIndex = 8;
            this.btnStop.Text = "Bi&tir";
            this.btnStop.UseVisualStyleBackColor = false;
            this.btnStop.Click += new System.EventHandler(this.btnStop_Click);
            //
            // btnStart
            //
            this.btnStart.BackColor = System.Drawing.Color.DarkGreen;
            this.btnStart.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.btnStart.Location = new System.Drawing.Point(264, 40);
            this.btnStart.Name = "btnStart";
            this.btnStart.Size = new System.Drawing.Size(48, 40);
            this.btnStart.TabIndex = 7;
            this.btnStart.Text = "&Başla";
            this.btnStart.UseVisualStyleBackColor = false;
            this.btnStart.Click += new System.EventHandler(this.btnStart_Click);
            //
            // pictureBox2
            //
            this.pictureBox2.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.pictureBox2.Location = new System.Drawing.Point(8, 184);
            this.pictureBox2.Name = "pictureBox2";
            this.pictureBox2.Size = new System.Drawing.Size(240, 88);
            this.pictureBox2.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
```

```
            this.pictureBox2.TabIndex = 6;
            this.pictureBox2.TabStop = false;
            //
            // lblSaveOrNot
            //
            this.lblSaveOrNot.Location = new System.Drawing.Point(256, 208);
            this.lblSaveOrNot.Name = "lblSaveOrNot";
            this.lblSaveOrNot.Size = new System.Drawing.Size(80, 32);
            this.lblSaveOrNot.TabIndex = 5;
            //
            // lblSavePercent
            //
            this.lblSavePercent.Location = new System.Drawing.Point(344, 16);
            this.lblSavePercent.Name = "lblSavePercent";
            this.lblSavePercent.Size = new System.Drawing.Size(40, 24);
            this.lblSavePercent.TabIndex = 4;
            //
            // trckBarSaveValue
            //
            this.trckBarSaveValue.Enabled = false;
            this.trckBarSaveValue.Location = new System.Drawing.Point(344, 48);
            this.trckBarSaveValue.Maximum = 100;
            this.trckBarSaveValue.Name = "trckBarSaveValue";
            this.trckBarSaveValue.Orientation =
System.Windows.Forms.Orientation.Vertical;
            this.trckBarSaveValue.Size = new System.Drawing.Size(45, 240);
            this.trckBarSaveValue.TabIndex = 3;
            this.trckBarSaveValue.ValueChanged += new
System.EventHandler(this.trckBarSaveValue_ValueChanged);
            //
            // lblPercent
            //
            this.lblPercent.Location = new System.Drawing.Point(272, 120);
            this.lblPercent.Name = "lblPercent";
```

```
            this.lblPercent.Size = new System.Drawing.Size(40, 32);
            this.lblPercent.TabIndex = 2;
            //
            // pictureBox1
            //
            this.pictureBox1.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.pictureBox1.Location = new System.Drawing.Point(8, 96);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(240, 88);
            this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox1.TabIndex = 1;
            this.pictureBox1.TabStop = false;
            //
            // pictureBox
            //
            this.pictureBox.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.pictureBox.Location = new System.Drawing.Point(8, 8);
            this.pictureBox.Name = "pictureBox";
            this.pictureBox.Size = new System.Drawing.Size(240, 88);
            this.pictureBox.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox.TabIndex = 0;
            this.pictureBox.TabStop = false;
            //
            // timer1
            //
            this.timer1.Interval = 600;
            this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
            //
            // groupBoxImageProcTime
            //
```

```
            this.groupBoxImageProcTime.Controls.Add(this.label4);
            this.groupBoxImageProcTime.Controls.Add(this.lblTotalTime);
            this.groupBoxImageProcTime.Controls.Add(this.label8);
            this.groupBoxImageProcTime.Controls.Add(this.lblTime);
            this.groupBoxImageProcTime.ForeColor = System.Drawing.Color.DarkRed;
            this.groupBoxImageProcTime.Location = new System.Drawing.Point(8,
344);
            this.groupBoxImageProcTime.Name = "groupBoxImageProcTime";
            this.groupBoxImageProcTime.Size = new System.Drawing.Size(536, 56);
            this.groupBoxImageProcTime.TabIndex = 17;
            this.groupBoxImageProcTime.TabStop = false;
            this.groupBoxImageProcTime.Text = "Görüntü İşleme Zamanı";
            //
            // label4
            //
            this.label4.ForeColor = System.Drawing.Color.DarkRed;
            this.label4.Location = new System.Drawing.Point(202, 22);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(64, 29);
            this.label4.TabIndex = 19;
            this.label4.Text = "Toplam Zaman";
            this.label4.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.label4.UseMnemonic = false;
            //
            // lblTotalTime
            //
            this.lblTotalTime.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.lblTotalTime.ForeColor = System.Drawing.Color.DarkRed;
            this.lblTotalTime.Location = new System.Drawing.Point(272, 24);
            this.lblTotalTime.Name = "lblTotalTime";
            this.lblTotalTime.Size = new System.Drawing.Size(240, 24);
            this.lblTotalTime.TabIndex = 18;
```

```
            this.lblTotalTime.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter;
            this.lblTotalTime.UseMnemonic = false;
            //
            // label8
            //
            this.label8.ForeColor = System.Drawing.Color.DarkRed;
            this.label8.Location = new System.Drawing.Point(6, 22);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(51, 24);
            this.label8.TabIndex = 17;
            this.label8.Text = "Zaman";
            this.label8.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.label8.UseMnemonic = false;
            //
            // lblTime
            //
            this.lblTime.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.lblTime.ForeColor = System.Drawing.Color.DarkRed;
            this.lblTime.Location = new System.Drawing.Point(60, 24);
            this.lblTime.Name = "lblTime";
            this.lblTime.Size = new System.Drawing.Size(136, 24);
            this.lblTime.TabIndex = 16;
            this.lblTime.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
            this.lblTime.UseMnemonic = false;
            //
            // lblCount
            //
            this.lblCount.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.lblCount.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(178)), true);
```

```
this.lblCount.ForeColor = System.Drawing.Color.DarkRed;
this.lblCount.Location = new System.Drawing.Point(576, 368);
this.lblCount.Name = "lblCount";
this.lblCount.Size = new System.Drawing.Size(80, 16);
this.lblCount.TabIndex = 18;
this.lblCount.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
this.lblCount.UseMnemonic = false;
//
// btnAbout
//
this.btnAbout.BackColor = System.Drawing.Color.DarkGreen;
this.btnAbout.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.btnAbout.Location = new System.Drawing.Point(672, 360);
this.btnAbout.Name = "btnAbout";
this.btnAbout.Size = new System.Drawing.Size(69, 40);
this.btnAbout.TabIndex = 19;
this.btnAbout.Text = "&Hakkında";
this.btnAbout.UseVisualStyleBackColor = false;
this.btnAbout.Click += new System.EventHandler(this.btnAbout_Click);
//
// MainForm
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.BackColor = System.Drawing.SystemColors.MenuHighlight;
this.ClientSize = new System.Drawing.Size(774, 405);
this.Controls.Add(this.btnAbout);
this.Controls.Add(this.lblCount);
this.Controls.Add(this.groupBoxImageProcTime);
this.Controls.Add(this.stillPanel);
this.Controls.Add(this.splitter1);
this.Controls.Add(this.videoPanel);
this.Controls.Add(this.toolBar);
this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
```

```
        this.MaximizeBox = false;
        this.Name = "MainForm";
        this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
        this.Text = "Hareket Tespiti Güvenlik Programı";
        this.Activated += new System.EventHandler(this.MainForm_Activated);
        this.Closing += new
System.ComponentModel.CancelEventHandler(this.MainForm_Closing);
        this.stillPanel.ResumeLayout(false);
        this.stillPanel.PerformLayout();
        ((System.ComponentModel.ISupportInitialize)(this.trckBarTimer)).EndInit();
        ((System.ComponentModel.ISupportInitialize)(this.pictureBox2)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.trckBarSaveValue)).EndInit();
        ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
        ((System.ComponentModel.ISupportInitialize)(this.pictureBox)).EndInit();
        this.groupBoxImageProcTime.ResumeLayout(false);
        this.ResumeLayout(false);
        this.PerformLayout();

            }
            #endregion

        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.Run(new MainForm());
        }

        private void MainForm_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
```

```
        {
                this.Hide();
                CloseInterfaces();
        }

                /// <summary> detect first form appearance, start grabber.
</summary>
        private void MainForm_Activated(object sender, System.EventArgs e)
        {
                if( firstActive )
                        return;
                firstActive = true;

                if( ! DsUtils.IsCorrectDirectXVersion() )
                {
                        MessageBox.Show( this, "DirectX 8.1 NOT installed!",
"DirectShow.NET", MessageBoxButtons.OK, MessageBoxIcon.Stop );
                        this.Close(); return;
                }

                if( ! DsDev.GetDevicesOfCat( FilterCategory.VideoInputDevice, out
capDevices ) )
                {
                        MessageBox.Show( this, "No video capture devices found!",
"DirectShow.NET", MessageBoxButtons.OK, MessageBoxIcon.Stop );
                        this.Close(); return;
                }

                DsDevice dev = null;
                if( capDevices.Count == 1 )
                        dev = capDevices[0] as DsDevice;
                else
                {
                        DeviceSelector selector = new DeviceSelector( capDevices );
```

```
                selector.ShowDialog( this );
                dev = selector.SelectedDevice;
            }

            if( dev == null )
            {
                this.Close(); return;
            }

            if( ! StartupVideo( dev.Mon ) )
                this.Close();
        }

        private void videoPanel_Resize(object sender, System.EventArgs e)
        {
            ResizeVideoWindow();
        }



        private void toolBar_ButtonClick(object sender,
System.Windows.Forms.ToolBarButtonClickEventArgs e)
        {
            Trace.WriteLine( "!!BTN: toolBar_ButtonClick" );

            int hr;
            if( sampGrabber == null )
                return;

            if( e.Button == toolBarBtnGrab )
            {
                Trace.WriteLine( "!!BTN: toolBarBtnGrab" );

                if( savedArray == null )
```

```
            {
                int size = videoInfoHeader.BmiHeader.ImageSize;
                if( (size < 1000) || (size > 16000000) )
                    return;
                savedArray = new byte[ size + 64000 ];
            }

            toolBarBtnSave.Enabled = false;
            Image old = pictureBox.Image;
            pictureBox.Image = null;
            if( old != null )
                old.Dispose();

            toolBarBtnGrab.Enabled = false;
            captured = false;
            hr = sampGrabber.SetCallback( this, 1 );
        }
        else if( e.Button == toolBarBtnSave )
        {
            Trace.WriteLine( "!!BTN: toolBarBtnSave" );

            SaveFileDialog sd = new SaveFileDialog();
            sd.FileName = @"DsNET.bmp";
            sd.Title = "Save Image as...";
            sd.Filter = "Bitmap file (*.bmp)|*.bmp";
            sd.FilterIndex = 1;
            if( sd.ShowDialog() != DialogResult.OK )
                return;

            pictureBox.Image.Save( sd.FileName, ImageFormat.Bmp );
        // save to new bmp file
        }
        else if( e.Button == toolBarBtnTune )
        {
```

```
                    if( capGraph != null )
                        DsUtils.ShowTunerPinDialog( capGraph, capFilter,
this.Handle );
                }

        }


        void OnCaptureDone()
        {
                Trace.WriteLine( "!!DLG: OnCaptureDone" );
                try
                {
                    toolBarBtnGrab.Enabled = true;
                    int hr;
                    if( sampGrabber == null )
                        return;
                    hr = sampGrabber.SetCallback( null, 0 );

                    int w = videoInfoHeader.BmiHeader.Width;
                    int h = videoInfoHeader.BmiHeader.Height;
                    if( ((w & 0x03) != 0) || (w < 32) || (w > 4096) || (h < 32) || (h >
4096) )
                        return;
                    //get Image
                    int stride = w * 3;
                    GCHandle handle = GCHandle.Alloc( savedArray,
GCHandleType.Pinned );
                    int scan0 = (int) handle.AddrOfPinnedObject();
                    scan0 += (h - 1) * stride;
                    Bitmap b = new Bitmap( w, h, -stride,
PixelFormat.Format24bppRgb, (IntPtr) scan0 );
                    handle.Free();
```

```
                    pictureBox1.Image = b;
                    b=new Bitmap(pictureBox1.Image);

                    if(firstRun)
                    {
                        firstRun=false;
                        pictureBox.Image=b;
                        a=new Bitmap(b);
                    }
                    else
                    {
                        count++;
                        wanted=new Bitmap(b.Width,b.Height);
                        Int32 percent;
                        counter.Start();
                        ImageProcessing imageProcessing=new
ImageProcessing(a,b,wanted);
                        imageProcessing.CompareUnsafeFaster(out percent);
                        percent=(Int32)(percent*100/(b.Width*b.Height));
                        counter.Stop();
                        pictureBox2.Image=wanted;
                        if((percent>=cPercent)&&!dontSaveNext)
                        {

        //imageProcessing.Save(Application.StartupPath+"\\wanted\\Wanted"+count
+"-"+percent+"-"+System.DateTime.Now.Millisecond.ToString()+".jpg");
                                    saveTime=count+"-"+percent+"-
"+System.DateTime.Now.Millisecond.ToString()+".jpg";



                m_sp.Load();
```

```
m_sp.Play();


b.Save(Application.StartupPath+"\\Wanted\\first"+saveTime,System.Drawing.Imaging.ImageFormat.Jpeg);

a.Save(Application.StartupPath+"\\Wanted\\second"+saveTime,System.Drawing.Imaging.ImageFormat.Jpeg);
                    count2++;
                    Microsoft.VisualBasic.Interaction.Beep();
                    dontSaveNext=true;
                    lblSaveOrNot.Text="don'tSaveNext";

                }
                else
                {
                    dontSaveNext=false;
                    lblSaveOrNot.Text="SaveNext";
                }
                this.lblTime.Text=counter.ToString();
                sec+=counter.Seconds;
                lblTotalTime.Text=sec.ToString()+" Seconds.";
                counter.Clear();

                lblCount.Text=count2+"/"+count;
                lblPercent.Text=percent+"%";

                if((percent-minSave)>5)
                {
                    try
                    {
                        trckBarSaveValue.Value=percent+5;
                        minSave=percent;
```

```
                    }
                    catch{}
                }

                //else if(percent<minSave)
                try
                {
                    minSave=(minSave+percent)/2;
                    trckBarSaveValue.Value=minSave+5;
                }
                catch{}


            }

            pictureBox.Image=b;
            a=new Bitmap(pictureBox.Image);

            savedArray=null;



        }
        catch
        {

        }
}



bool StartupVideo( UCOMIMoniker mon )
{
        int hr;
```

```
                try {
                        if( ! CreateCaptureDevice( mon ) )
                                return false;

                        if( ! GetInterfaces() )
                                return false;

                        if( ! SetupGraph() )
                                return false;

                        if( ! SetupVideoWindow() )
                                return false;

                        #if DEBUG
                                DsROT.AddGraphToRot( graphBuilder, out rotCookie
);              // graphBuilder capGraph
                        #endif

                        hr = mediaCtrl.Run();
                        if( hr < 0 )
                                Marshal.ThrowExceptionForHR( hr );

                        bool hasTuner = DsUtils.ShowTunerPinDialog( capGraph,
capFilter, this.Handle );
                        toolBarBtnTune.Enabled = hasTuner;
        m_sp.Load();
        m_sp.Play();

                        return true;
                }
                catch
                {
                        return false;
                }
```

```
        }

                bool SetupVideoWindow()
        {
                int hr;
                try {
                        hr = videoWin.put_Owner( videoPanel.Handle );
                        if( hr < 0 )
                                Marshal.ThrowExceptionForHR( hr );


                        hr = videoWin.put_WindowStyle( WS_CHILD |
WS_CLIPCHILDREN );
                        if( hr < 0 )
                                Marshal.ThrowExceptionForHR( hr );

                        ResizeVideoWindow();

                        hr = videoWin.put_Visible( DsHlp.OATRUE );
                        if( hr < 0 )
                                Marshal.ThrowExceptionForHR( hr );

                        hr = mediaEvt.SetNotifyWindow( this.Handle,
WM_GRAPHNOTIFY, IntPtr.Zero );
                        if( hr < 0 )
                                Marshal.ThrowExceptionForHR( hr );
                        return true;
                }
                catch
                {
                        return false;
                }
        }
```

```
        bool SetupGraph()
        {
                int hr;
                try {
                        hr = capGraph.SetFiltergraph( graphBuilder );
                        if( hr < 0 )
                                Marshal.ThrowExceptionForHR( hr );

                        hr = graphBuilder.AddFilter( capFilter, "Ds.NET Video
Capture Device" );
                        if( hr < 0 )
                                Marshal.ThrowExceptionForHR( hr );

                        DsUtils.ShowCapPinDialog( capGraph, capFilter, this.Handle
);

                        AMMediaType media = new AMMediaType();
                        media.majorType      = MediaType.Video;
                        media.subType        = MediaSubType.RGB24;
                        media.formatType = FormatType.VideoInfo;                //
???
                        hr = sampGrabber.SetMediaType( media );
                        if( hr < 0 )
                                Marshal.ThrowExceptionForHR( hr );

                        hr = graphBuilder.AddFilter( baseGrabFlt, "Ds.NET Grabber"
);
                        if( hr < 0 )
                                Marshal.ThrowExceptionForHR( hr );

                        Guid cat = PinCategory.Preview;
                        Guid med = MediaType.Video;
```

```
                    hr = capGraph.RenderStream( ref cat, ref med, capFilter, null,
null ); // baseGrabFlt
                    if( hr < 0 )
                            Marshal.ThrowExceptionForHR( hr );

                    cat = PinCategory.Capture;
                    med = MediaType.Video;
                    hr = capGraph.RenderStream( ref cat, ref med, capFilter, null,
baseGrabFlt ); // baseGrabFlt
                    if( hr < 0 )
                            Marshal.ThrowExceptionForHR( hr );

                    media = new AMMediaType();
                    hr = sampGrabber.GetConnectedMediaType( media );
                    if( hr < 0 )
                            Marshal.ThrowExceptionForHR( hr );
                    if( (media.formatType != FormatType.VideoInfo) ||
(media.formatPtr == IntPtr.Zero) )
                            throw new NotSupportedException( "Unknown
Grabber Media Format" );

                    videoInfoHeader = (VideoInfoHeader)
Marshal.PtrToStructure( media.formatPtr, typeof(VideoInfoHeader) );
                    Marshal.FreeCoTaskMem( media.formatPtr ); media.formatPtr
= IntPtr.Zero;

                    hr = sampGrabber.SetBufferSamples( false );
                    if( hr == 0 )
                            hr = sampGrabber.SetOneShot( false );
                    if( hr == 0 )
                            hr = sampGrabber.SetCallback( null, 0 );
                    if( hr < 0 )
                            Marshal.ThrowExceptionForHR( hr );
```

```
                return true;
        }
        catch
        {
                return false;
        }
    }



    bool GetInterfaces()
    {
        Type comType = null;
        object comObj = null;
        try {
            comType = Type.GetTypeFromCLSID( Clsid.FilterGraph );
            if( comType == null )
                throw new NotImplementedException( @"DirectShow
FilterGraph not installed/registered!" );
            comObj = Activator.CreateInstance( comType );
            graphBuilder = (IGraphBuilder) comObj; comObj = null;

            Guid clsid = Clsid.CaptureGraphBuilder2;
            Guid riid = typeof(ICaptureGraphBuilder2).GUID;
            comObj = DsBugWO.CreateDsInstance( ref clsid, ref riid );
            capGraph = (ICaptureGraphBuilder2) comObj; comObj = null;

            comType = Type.GetTypeFromCLSID( Clsid.SampleGrabber
);
            if( comType == null )
                throw new NotImplementedException( @"DirectShow
SampleGrabber not installed/registered!" );
            comObj = Activator.CreateInstance( comType );
            sampGrabber = (ISampleGrabber) comObj; comObj = null;
```

```
            mediaCtrl      = (IMediaControl)     graphBuilder;
            videoWin       = (IVideoWindow)      graphBuilder;
            mediaEvt       = (IMediaEventEx)     graphBuilder;
            baseGrabFlt    = (IBaseFilter)               sampGrabber;
            return true;
        }
        catch
        {
            return false;
        }
        finally
        {
            if( comObj != null )
                Marshal.ReleaseComObject( comObj ); comObj = null;
        }
}


bool CreateCaptureDevice( UCOMIMoniker mon )
{
        object capObj = null;
        try {
            Guid gbf = typeof( IBaseFilter ).GUID;
            mon.BindToObject( null, null, ref gbf, out capObj );
            capFilter = (IBaseFilter) capObj; capObj = null;
            return true;
        }
        catch
        {
            return false;
        }
        finally
        {
```

```
                if( capObj != null )
                        Marshal.ReleaseComObject( capObj ); capObj = null;
        }

    }



    {
        int hr;
        try {
            #if DEBUG
                if( rotCookie != 0 )
                    DsROT.RemoveGraphFromRot( ref rotCookie
);
            #endif

            if( mediaCtrl != null )
            {
                hr = mediaCtrl.Stop();
                mediaCtrl = null;
            }

            if( mediaEvt != null )
            {
                hr = mediaEvt.SetNotifyWindow( IntPtr.Zero,
WM_GRAPHNOTIFY, IntPtr.Zero );
                mediaEvt = null;
            }

            if( videoWin != null )
            {
                hr = videoWin.put_Visible( DsHlp.OAFALSE );
```

```
                    hr = videoWin.put_Owner( IntPtr.Zero );
                    videoWin = null;
                }

                baseGrabFlt = null;
                if( sampGrabber != null )
                    Marshal.ReleaseComObject( sampGrabber );
sampGrabber = null;

                if( capGraph != null )
                    Marshal.ReleaseComObject( capGraph ); capGraph =
null;

                if( graphBuilder != null )
                    Marshal.ReleaseComObject( graphBuilder );
graphBuilder = null;

                if( capFilter != null )
                    Marshal.ReleaseComObject( capFilter ); capFilter =
null;

                if( capDevices != null )
                {
                    foreach( DsDevice d in capDevices )
                        d.Dispose();
                    capDevices = null;
                }
            }
            catch
            {}
        }


        void ResizeVideoWindow()
```

```
{
        if( videoWin != null )
        {
                Rectangle rc = videoPanel.ClientRectangle;
                videoWin.SetWindowPosition( 0, 0, rc.Right, rc.Bottom );
        }
}

                protected override void WndProc( ref Message m )
{
        if( m.Msg == WM_GRAPHNOTIFY )
                {
                if( mediaEvt != null )
                        OnGraphNotify();
                return;
                }
        base.WndProc( ref m );
}


void OnGraphNotify()
{
        DsEvCode      code;
        int p1, p2, hr = 0;
        do
        {
                hr = mediaEvt.GetEvent( out code, out p1, out p2, 0 );
                if( hr < 0 )
                        break;
                hr = mediaEvt.FreeEventParams( code, p1, p2 );
        }
        while( hr == 0 );
}
```

```
        int ISampleGrabberCB.SampleCB( double SampleTime,
IMediaSample pSample )
    {
        Trace.WriteLine( "!!CB: ISampleGrabberCB.SampleCB" );
        return 0;
    }


    int ISampleGrabberCB.BufferCB( double SampleTime, IntPtr pBuffer, int
BufferLen )
    {
        if( captured || (savedArray == null) )
        {
            Trace.WriteLine( "!!CB: ISampleGrabberCB.BufferCB" );
            return 0;
        }

        captured = true;
        bufferedSize = BufferLen;
        Trace.WriteLine( "!!CB: ISampleGrabberCB.BufferCB  !GRAB! size
= " + BufferLen.ToString() );
        if( (pBuffer != IntPtr.Zero) && (BufferLen > 1000) && (BufferLen
<= savedArray.Length) )
            Marshal.Copy( pBuffer, savedArray, 0, BufferLen );
        else
            Trace.WriteLine( "    !!!GRAB! failed " );
        this.BeginInvoke( new CaptureDone( this.OnCaptureDone ) );
        return 0;
    }


    private bool                    firstActive;
```

```
        private IBaseFilter                 capFilter;

        private IGraphBuilder               graphBuilder;

        private ICaptureGraphBuilder2       capGraph;
        private ISampleGrabber              sampGrabber;

        private IMediaControl               mediaCtrl;

        private IMediaEventEx               mediaEvt;

        private IVideoWindow                videoWin;

        private IBaseFilter                 baseGrabFlt;

        private VideoInfoHeader             videoInfoHeader;
        private bool                        captured = true;
        private int                             bufferedSize;

        private byte[]                      savedArray;

        private ArrayList                   capDevices;

        private const int WM_GRAPHNOTIFY    = 0x00008001;       // message
from graph
```

```
        private const int WS_CHILD                  = 0x40000000;       // attributes
for video window
        private const int WS_CLIPCHILDREN     = 0x02000000;
        private const int WS_CLIPSIBLINGS     = 0x04000000;
        private delegate void CaptureDone();

        #if DEBUG
                private int             rotCookie = 0;

        #endif
        private void timer1_Tick(object sender, System.EventArgs e)
        {
                System.Windows.Forms.ToolBarButtonClickEventArgs ee=new
ToolBarButtonClickEventArgs(toolBarBtnGrab);
                toolBar_ButtonClick(toolBarBtnGrab,ee);

        }

        private void btnStart_Click(object sender, System.EventArgs e)
        {
                btnStart.Enabled=false;
                btnStop.Enabled=true;
                timer1.Enabled=true;
        }

        private void trckBarSaveValue_ValueChanged(object sender,
System.EventArgs e)
        {
                cPercent=trckBarSaveValue.Value;
                lblSavePercent.Text=cPercent+"%";

        }

        private void btnStop_Click(object sender, System.EventArgs e)
```

```
        {
                btnStart.Enabled=true;
                btnStop.Enabled=false;
                timer1.Enabled=false;
                firstRun=true;
        }

        private void btnAbout_Click(object sender, System.EventArgs e)
        {
                About about=new About();
                about.ShowDialog(this);
        }

        private void trckBarTimer_ValueChanged(object sender, System.EventArgs
e)
        {
           timer1.Interval=trckBarTimer.Value;
                lblTimerValue.Text=trckBarTimer.Value.ToString();
        }




}

internal enum PlayState
{
        Init, Stopped, Paused, Running
}

}
```

EK2

ImageProcessing.cs

```
using System;
using System.Drawing;
using System.Drawing.Imaging;


namespace ImgProcessing
{

        public unsafe sealed class ImageProcessing
        {
                Bitmap flag,flag2,flag3;
                int width,width2,width3;
                BitmapData bitmapData = null,bitmapData2= null,bitmapData3= null;
                Byte* pBase = null,pBase2=null,pBase3=null;

                #region ImageProcessing Constructor
                public ImageProcessing(Bitmap picOld,Bitmap picNew,Bitmap
target)
                {
                        this.flag=picOld;
                        this.flag2=picNew;
                        this.flag3=target;
                }
                public ImageProcessing(Bitmap source,Bitmap target)
                {
                        this.flag=source;
                        this.flag2=target;
                }
                #endregion

                #region internal methods
                #region PixelSize
```

```
        public Point PixelSize
        {
            get
            {
                GraphicsUnit unit = GraphicsUnit.Pixel;
                RectangleF bounds = flag.GetBounds(ref unit);

                return new Point((int) bounds.Width, (int)
bounds.Height);
            }
        }

        public Point PixelSize2
        {
            get
            {
                GraphicsUnit unit = GraphicsUnit.Pixel;
                RectangleF bounds = flag2.GetBounds(ref unit);

                return new Point((int) bounds.Width, (int)
bounds.Height);
            }
        }
        public Point PixelSize3
        {
            get
            {
                GraphicsUnit unit = GraphicsUnit.Pixel;
                RectangleF bounds = flag3.GetBounds(ref unit);

                return new Point((int) bounds.Width, (int)
bounds.Height);
            }
        }
```

```
#endregion

#region PixelAt
public Pixel* PixelAt(int x, int y)
{
        return (Pixel*) (pBase + y * width + x * sizeof(Pixel));
}
public Pixel* PixelAt2(int x, int y)
{
        return (Pixel*) (pBase2 + y * width2 + x * sizeof(Pixel));
}
public Pixel* PixelAt3(int x, int y)
{
        return (Pixel*) (pBase3 + y * width3 + x * sizeof(Pixel));
}

#endregion

#region LockBitMap
public void LockBitmap()
{
        GraphicsUnit unit = GraphicsUnit.Pixel;
        RectangleF boundsF = flag.GetBounds(ref unit);
        Rectangle bounds = new Rectangle((int) boundsF.X,
                (int) boundsF.Y,
                (int) boundsF.Width,
                (int) boundsF.Height);


        width = (int) boundsF.Width * sizeof(Pixel);
        if (width % 4 != 0)
        {
                width = 4 * (width / 4 + 1);
        }
```

```
                bitmapData =
                        flag.LockBits(bounds, ImageLockMode.ReadWrite,
PixelFormat.Format24bppRgb);

                pBase = (Byte*) bitmapData.Scan0.ToPointer();

        }
        public void UnlockBitmap()
        {
                flag.UnlockBits(bitmapData);
                bitmapData = null;
                pBase = null;

        }
        public void LockBitmap2()
        {
                GraphicsUnit unit = GraphicsUnit.Pixel;
                RectangleF boundsF = flag2.GetBounds(ref unit);
                Rectangle bounds = new Rectangle((int) boundsF.X,
                        (int) boundsF.Y,
                        (int) boundsF.Width,
                        (int) boundsF.Height);


                width2 = (int) boundsF.Width * sizeof(Pixel);
                if (width2 % 4 != 0)
                {
                        width2 = 4 * (width2 / 4 + 1);
                }

                bitmapData2 =
                        flag2.LockBits(bounds, ImageLockMode.ReadWrite,
PixelFormat.Format24bppRgb);
```

```
            pBase2 = (Byte*) bitmapData2.Scan0.ToPointer();
        }

        public void UnlockBitmap2()
        {
            flag2.UnlockBits(bitmapData2);
            bitmapData2= null;
            pBase2= null;
        }
        public void LockBitmap3()
        {
            GraphicsUnit unit = GraphicsUnit.Pixel;
            RectangleF boundsF = flag3.GetBounds(ref unit);
            Rectangle bounds = new Rectangle((int) boundsF.X,
                (int) boundsF.Y,
                (int) boundsF.Width,
                (int) boundsF.Height);


            width3 = (int) boundsF.Width * sizeof(Pixel);
            if (width3 % 4 != 0)
            {
                width3 = 4 * (width3 / 4 + 1);
            }

            bitmapData3 =
                flag3.LockBits(bounds, ImageLockMode.ReadWrite,
PixelFormat.Format24bppRgb);

            pBase3 = (Byte*) bitmapData3.Scan0.ToPointer();
        }

        public void UnlockBitmap3()
```

```
{
        flag3.UnlockBits(bitmapData3);
        bitmapData3= null;
        pBase3= null;
}
#endregion

#region Save
public void Save(string filename)
{
        flag3.Save(filename, ImageFormat.Jpeg);
}
#endregion

#region Dispose
public void Dispose()
{
        flag=null;
        flag2=null;
        flag3=null;
}
#endregion

public Bitmap bitmap
{
        get
        {
                return(this.flag3);
        }
}


#endregion
```

```
#region Magic code for CompareUnsafeFaster
public void CompareUnsafeFaster(out Int32 percent)
{
        Point size = PixelSize;
    percent=0;
        LockBitmap();
        LockBitmap2();
        LockBitmap3();

        for (int y = 0; y < size.Y; y++)
        {
            Pixel* pPixel = PixelAt(0, y);
            Pixel* pPixel2 = PixelAt2(0, y);
            Pixel* pPixel3 = PixelAt3(0, y);
            for (int x = 0; x < size.X; x++)
            {
             if(!(pPixel->green==pPixel2->green))//((pPixel->red==pPixel2->red)))//&&(pPixel->green==pPixel2->green)&&(pPixel->blue==pPixel2->blue)))
                {
                    pPixel3->red = pPixel2->red;
                    pPixel3->green = pPixel2->green;
                    pPixel3->blue = pPixel2->blue;
                    percent++;
                }
                pPixel++;
                pPixel2++;
                pPixel3++;
            }
        }
        UnlockBitmap3();
        UnlockBitmap2();
        UnlockBitmap();
    }
```

```
#endregion

#region Magic code for ComplementUnsafeFaster
public void ComplementUnsafeFaster()
{
        Point size = PixelSize;

        LockBitmap();
        LockBitmap2();



        for (int y = 0; y < size.Y; y++)
        {
                Pixel* pPixel = PixelAt(0, y);
                Pixel* pPixel2 = PixelAt2(0, y);

                for (int x = 0; x < size.X; x++)
                {

                        pPixel2->red =(byte)(255-(int)pPixel->red);
                        pPixel2->green = (byte)(255-(int)pPixel-
>green);
                        pPixel2->blue = (byte)(255-(int)pPixel->blue);

                        pPixel++;
                        pPixel2++;

                }
        }
        UnlockBitmap2();
        UnlockBitmap();
        flag3=new Bitmap(flag2);
```

```
        }

        #endregion

        #region Magic code for ColorBallUnsafeFaster
        public void ColorBallUnsafeFaster(double red,double green,double
blue)
        {
            Point size = PixelSize;
            double fr, fg, fb;
            if(red<0)
            {
                fr=red/100+1;
            }
            else
            {
                fr=red/100;
            }
            if(green<0)
            {
                fg=green/100+1;
            }
            else
            {
                fg=green/100;
            }
            if(blue<0)
            {
                fb=blue/100+1;
            }
            else
            {
                fb=blue/100;
            }
```

```
LockBitmap();
LockBitmap2();


for (int y = 0; y < size.Y; y++)
{
	Pixel* pPixel = PixelAt(0, y);
	Pixel* pPixel2 = PixelAt2(0, y);

	for (int x = 0; x < size.X; x++)
	{

		if(  red < 0 )
		{
			pPixel2->red =(byte)((int)pPixel->red *
fr);
		}
		else
		{
			pPixel2->red = (byte)((int)pPixel->red +
(255 - (int)pPixel->red) * fr);
		}
		if(  green < 0 )
		{
			pPixel2->green = (byte)((int)pPixel-
>green * fg);
		}
		else
		{
			pPixel2->green = (byte)((int)pPixel-
>green + (255 - (int)pPixel->green) * fg);
		}
		if(  blue < 0 )
		{
```

```
                                        pPixel2->blue =(byte)((int) pPixel->blue
* fb);
                                    }
                                    else
                                    {
                                        pPixel2->blue= (byte)((int)pPixel->blue
+ (255 - (int)pPixel->blue) * fb);
                                    }

                                    pPixel++;
                                    pPixel2++;

                                }
                            }
                            UnlockBitmap2();
                            UnlockBitmap();
                            flag3=new Bitmap(flag2);

                        }

                        #endregion

                        #region Magic code for Brightness
                        public void Brightness(double brightness)
                        {
                            Point size = PixelSize;
                            double f;
                            if(brightness<0)
                            {
                                f=brightness/100+1;
                            }
                            else
                            {
                                f=brightness/100;
```

```
                }

                LockBitmap();
                LockBitmap2();


                for (int y = 0; y < size.Y; y++)
                {
                        Pixel* pPixel = PixelAt(0, y);
                        Pixel* pPixel2 = PixelAt2(0, y);

                        for (int x = 0; x < size.X; x++)
                        {
                                if(  brightness < 0 )
                                {
                                        pPixel2->red =(byte)( (int)pPixel->red *
f);
                                        pPixel2->green =(byte)( (int)pPixel-
>green * f);
                                        pPixel2->blue =(byte)( (int)pPixel->blue
* f);
                                }
                                else
                                {
                                        pPixel2->green = (byte)((int)pPixel->red
+ (255 - (int)pPixel->red) * f);
                                        pPixel2->green = (byte)((int)pPixel-
>green + (255 - (int)pPixel->green) * f);
                                        pPixel2->blue= (byte)((int)pPixel->blue
+ (255 - (int)pPixel->blue) * f);
                                }

                                pPixel++;
                                pPixel2++;
```

```
                }
            }
            UnlockBitmap2();
            UnlockBitmap();
            flag3=new Bitmap(flag2);

        }

        #endregion

        #region MakeGreyUnsafeFaster
        public void MakeGreyUnsafeFaster()
        {
            Point size = PixelSize;

            LockBitmap();
            LockBitmap2();

            for (int y = 0; y < size.Y; y++)
            {
                Pixel* pPixel = PixelAt(0, y);
                Pixel* pPixel2 = PixelAt2(0, y);
                for (int x = 0; x < size.X; x++)
                {
                    byte value = (byte) ((pPixel->red + pPixel-
>green + pPixel->blue) / 3);
                    pPixel2->red =  value;
                    pPixel2->green = value;
                    pPixel2->blue = value;
                    pPixel++;
                    pPixel2++;
                }
            }
```

```
                    UnlockBitmap2();
                    UnlockBitmap();
                    flag3=new Bitmap(flag2);
                }
                #endregion

        }

        #region Pixel struct
        public struct Pixel
        {
                public byte blue;
                public byte green;
                public byte red;
        }
        #endregion
}
```

EK3

About.cs

```
namespace CatchItV
{
   using System;
   using System.Drawing;
   using System.Collections;
   using System.ComponentModel;
   using System.Windows.Forms;

   public class About : System.Windows.Forms.Form
   {
      public Button ButtonOK;
      private Label label2;
      private PictureBox pictureBox1;
      public Label label1;
```

```
public Label label3;
private System.ComponentModel.Container components = null;



public About()
{

   InitializeComponent();
                        string t=label2.Text+Application.ProductVersion;
                        label2.Text=t;

}


protected override void Dispose(bool disposing)
{
                        if(disposing)
                        {
                                if (components != null)
                                {
                                        components.Dispose();
                                }
                        }
                        base.Dispose(disposing);
}

private void InitializeComponent()
                {
   System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(About));
   this.ButtonOK = new System.Windows.Forms.Button();
   this.label2 = new System.Windows.Forms.Label();
   this.pictureBox1 = new System.Windows.Forms.PictureBox();
```

```
            this.label1 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
            this.SuspendLayout();
            //
            // ButtonOK
            //
            this.ButtonOK.BackColor = System.Drawing.Color.Transparent;
            this.ButtonOK.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.ButtonOK.Font = new System.Drawing.Font("Microsoft Sans Serif",
20F, ((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold |
System.Drawing.FontStyle.Italic))), System.Drawing.GraphicsUnit.Point,
((byte)(178)));
            this.ButtonOK.ForeColor = System.Drawing.SystemColors.ActiveCaption;
            this.ButtonOK.Location = new System.Drawing.Point(0, 0);
            this.ButtonOK.Name = "ButtonOK";
            this.ButtonOK.Size = new System.Drawing.Size(809, 41);
            this.ButtonOK.TabIndex = 1;
            this.ButtonOK.Text = "> &Çık <";
            this.ButtonOK.UseVisualStyleBackColor = false;
            this.ButtonOK.Click += new System.EventHandler(this.ButtonOK_Click);
            //
            // label2
            //
            this.label2.BackColor = System.Drawing.Color.Transparent;
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 15F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(178)));
            this.label2.ForeColor = System.Drawing.Color.DarkRed;
            this.label2.Location = new System.Drawing.Point(26, 58);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(338, 74);
            this.label2.TabIndex = 3;
```

```
            this.label2.Text = "Hareket Tespiti Güvenlik Uygulaması (Görüntü İşleme
İle)";
            this.label2.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
            //
            // pictureBox1
            //
            this.pictureBox1.BackColor = System.Drawing.Color.Transparent;
            this.pictureBox1.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.pictureBox1.Cursor = System.Windows.Forms.Cursors.No;
            this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(520, 99);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(247, 157);
            this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox1.TabIndex = 6;
            this.pictureBox1.TabStop = false;
            //
            // label1
            //
            this.label1.BackColor = System.Drawing.Color.Maroon;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(178)));
            this.label1.ForeColor = System.Drawing.Color.MistyRose;
            this.label1.Location = new System.Drawing.Point(0, 339);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(809, 41);
            this.label1.TabIndex = 7;
            this.label1.Text = "ayilmaz@halic.edu.tr";
            //
            // label3
```

```
            //
            this.label3.BackColor = System.Drawing.Color.Transparent;
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(178)));
            this.label3.ForeColor = System.Drawing.Color.DarkRed;
            this.label3.Location = new System.Drawing.Point(13, 289);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(404, 42);
            this.label3.TabIndex = 8;
            this.label3.Text = "Copyright © 2007 Atınç Yılmaz™";
            //
            // About
            //
            this.BackColor = System.Drawing.Color.Black;
            this.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("$this.BackgroundImage")));
            this.ClientSize = new System.Drawing.Size(498, 368);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.pictureBox1);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.ButtonOK);
            this.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(178)));
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "About";
            this.Opacity = 0.75;
            this.ShowInTaskbar = false;
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
```

```
            this.Text = "Hakkında";
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
            this.ResumeLayout(false);

        }

        protected void ButtonOK_Click (object sender, System.EventArgs e)
        {
            this.Close();
        }
    }
}
```

7. SONUÇ

Bu çalışmada görüntü işleme kullanılarak hareket tespiti yöntemleri incelenmiş ve buna uygun bir algoritma geliştirilerek uygulama yapılmıştır. Uygulama için görüntünün dış dünyadan alınması için bir web kamera kullanılmıştır. Uygulanan algoritmada web kameradan ilk alınan görüntü "Old" adı altında tutulurken, sonrasında web kameradan alınan görüntü "Cur" adı altında tutulur. Sonrasında bu iki görüntü ortamın ve ortamdaki ışık miktarının durumuna göre belirlenen eşik değerine göre pikselleri karşılaştırarak ortamda hareket olup olmadığı bulunur. Algoritmaya göre uygulama içerisinde hareket tespit edildiğinde sesli bir uyarı verilip, daha önce belirlenen dizin içersinde hareket tespiti yapılan görüntüler kaydedilir. Uygulama .Net teknolojilerinden C# programlama dili ile yazılıp derlenmiştir. Yapılan uygulamada hareket tespiti segmentasyon algoritmalarından olan arka plan farkı yöntemlerinden basit fark alma yöntemi uygulanmıştır. Uygulamada web kameradan alınan görüntüler piksel farkları alınarak karşılaştırılıp ortamdan hareket tespiti yapılmaktadır.

Bu uygulamada karşılaşılan en büyük güçlük, dış ortam görüntülerindeki değişken ışık yoğunluğu olmuştur. İç ortam görüntüleri alınırken, ortamın ışık yoğunluğu, sabit ışık kaynakları yardımıyla değişkenliği kontrol edilebilirken, dış ortamda ise bunun gibi bir seçenek bulunmamaktadır. Karşılaşılan başka bir sorun ise işlem sırasındaki fark görüntülerinde meydana gelen gürültüler olmuştur. Düzgün bir sonuca ulaşabilmek için temizlenmesi gereken bu gürültüler, algoritmaların fazladan fonksiyon kullanmasından dolayı işlem zamanı uzamaktadır. Hareket tespiti yöntemlerindeki en önemli nokta eşik değerinin seçilmesidir.

Hareket tespiti yöntemlerinde karşılaşılan sorunları gidermek için aşağıdaki işlemlerin yapılmasıyla yöntemlerin başarısının artabileceği öngörülmektedir:

Görüntüler arasındaki farkın bulunmasında büyük rol oynayan eşik değerinin, ışık yoğunluğu değişimlerinden mümkün olduğunca bağımsız hale getirilmesi gereklidir. Bunun için incelenecek olan görüntünün işleme girmeden önce ışık yoğunluğu değişimlerini bastırabilecek görüntü işleme adımlarından geçirilmesi yöntemi kullanılabilir.

İşlem sırasında oluşturulan fark resimlerinde oluşan gürültülerin temizlenmesi işleminde kullanılan prosedür ve fonksiyonların düşük seviye programlama düzenlemeleriyle algoritmaya verilen yük hafifletebilir.

Görüntü alımı sırasında kontrast ve parlaklık seviyesini sabit tutabilen kamera veya donanımların kullanılması da sisteme daha kararlı görüntüler yollanarak sistemin verimi arttırılabilir.

8. KAYNAKLAR

BAYRAM, Bülent ; **"Sayısal Görüntü İşleme"**, İstanbul, 2005

CUTLER, R. Ve Davis, L.; **Wiev-Based detection and analysis of periodic motion.** International Conference on Pattern Recognation. Brisbane, Australia, 1998

FORSYTH, David A. ; **Computer Vision – A modern Approach**, 2000

GONZALEZ, R. C. Ve WOODS, R. E.. **Digital Image Processing.** Addison-Wesley Publishing Company Inc., UK. ,1992

GONZALEZ, Rafael C. ; **Digital Image Processing By Gonzalez, Prectice Hall** , 2001

GÖÇERİ, Engin ; **"Çok Boyutlu Görüntüler için JPEG2000 Standardını Destekleyen Görüntü İşleme Uygulaması",** Kütahya, 2007

JARABA, E. H., URUNUELA, C. O. VE SENAR, J ; **Detection motion classifaction with a double-background and a Neighbourhood-based differerence.** Pattern Recognition Letters., 2003

JOHN, C. Russ ; **The Image Processing Hand Book, Third Edition**, CRC Press, 1999, ISBN:0-8493-2532-3

KALVIAINEN, H., XU, L. Ve OJA, E. ; "**Motion Detection Using Randomized Hough Transform"**. Proceedings of the 7th Scandinavian Conference on Image Analysis, Aalborg, Denmark. , 1991

KARMANN, K. P. VE BRANDT, A. V. ; **"Moving Object Recognation Using an Adaptive Background Memory, Time-varying Image Processing and Moving Object Recognation"**. (V.Cappellini ed.), Elsevier, Amsterdam, 1990

MEYER, D., DENZLER, J: ve NIEMANN, H. ; **Model based extraction of articulated objects in image sequences for gait analysis.** IEEE International Conference on Image Processing. Washington DC , 1997.

PRATT, William K. ; **Digital Image Processing-3rd Edition**, Addison Wesley, 2007

YANG, Y. H. Ve LEVINE, M. D. ; **The background primal sketch: an approach for tracking moving objects.** Machine Vision Applications, 1992

http://www.goruntuisleme.org

9. ÖZGEÇMİŞ

1 Mayıs 1983 yılında İzmit Kocaeli' nde dünyaya geldim. İlkokulu Kocaeli Ulugazi İlköğretim Okulu' nda (1989-1994), ortaokul ve lise öğrenimimi Kocaeli Anadolu Lisesi' sinde (1994-1999) ve lise öğrenimimin son 2 yılını Kadıköy Kenan Evren Anadolu Lisesi' nde (1999-2001) tamamladım. 2001-2005 yılları arasında lisans eğitimimi Haliç Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği bölümünde aldım. Yüksek lisansımı ise 2005-2007 yılları arasında Haliç Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği ana bilim dalında yaptım. Halen Haliç Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği bölümünde Araştırma Görevlisi olarak görev almaktayım. İlgi alanlarım arasında görüntü işleme, örüntü tanıma, veritabanı sistemleri, yapay sinir ağları, mantık tasarım devreleri ve .Net programlama yer almaktadır.

Atınç YILMAZ