BİLGİSAYAR KONTROLLÜ HEDEF TAKİBİ

Nazım ÖZGEN

YÜKSEK LİSANS TEZİ
ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AĞUSTOS 2008
ANKARA

Nazım ÖZGEN tarafından hazırlanan “Bilgisayar Kontrollü Hedef Takibi” adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Prof.Dr.Müzeyyen SARITAŞ
Tez Danışmanı, Elektrik-Elektronik Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Elektrik-Elektronik Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Prof.Dr.Nihal Fatma GÜLER
Elektronik-Bilgisayar Anabilim Dalı, Gazi Üniversitesi
Prof.Dr.İrfan KARAGÖZ
Elektrik-Elektronik Mühendisliği Anabilim Dalı, Gazi Üniversitesi
Prof.Dr.Müzeyyen SARITAŞ
Elektrik-Elektronik Mühendisliği Anabilim Dalı, Gazi Üniversitesi

Tarih : 07/08/2008

Bu tez ile G.Ü. Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onamıştır.

Prof. Dr. Nermin ERTAN
Fen Bilimleri Enstitüsü Müdürü

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Nazım ÖZGEN

BİLGİSAYAR KONTROLLÜ HEDEF TAKİBİ
(Yüksek Lisans Tezi)

Nazım ÖZGEN

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Ağustos 2008

ÖZET

Günümüzde gelişen bilgisayar teknolojisi ve görüntü işleme teknikleri ile askeri ve sivil alanda hedef takibi kolaylaşmıştır.

Hedef takibinde en çok kullanılan metotlar özellikli nokta tanımlayıcıları (Moravec, Harris, KLT ve SIFT), arka plan çıkartımı ve bölümleme metotlarıdır. Bu yöntemlerin kendilerine göre avantaj ve dezavantajları bulunmaktadır. Bu çalışmada, sabit/değişken arka planda sabit/hareketli hedefi en iyi takip edebilecek metot olan SIFT metodu seçilmiştir. MATLAB ortamında hazırlanmış olan SIFT programı hedef takibinde kullanılmıştır. SIFT parametrelerinin hedef takibine etkileri incelenmiş ve SIFT parametrelerinin optimum değerleri elde edilmiştir. Bu değerler; kademe sayısı s=3, düşük kontrasta sahip noktaların ayıklanması için gerekli eşik değer D(x)=0.02, kenarlarda tespit edilen noktaların ayıklanması için gerekli eşik değer R=10, örnekleme alanı (nxn)=4x4, kilit nokta tanımlayıcıları için gerekli her örnek noktanın temsil edileceği yön sayısı r=8'dir. Bu optimum değerlerin literatür ile uyumlu olduğu bulunmuştur.

Ayrıca bu çalışmada, birbirini takip eden resim kareleri arasındaki hatalı eşleşmeleri yok etmek için ek bir program hazırlanmıştır. Bu program

kullanıldığında, optimum olmayan SIFT parametre değerlerindeki hedef takibinde oluşan sapmaların azaldığı tespit edilmiştir.

Resim çözünürlüğü ve bilgisayar işlemci hızı değiştirilerek hedef takibine etkileri incelenmiştir. Resim çözünürlüğü arttırıldığında, SIFT kilit nokta sayısının, SIFT kilit noktalarının tespit süresinin ve hedefi bulma süresinin exponansiyel olarak arttığı gözlenmiştir. Bilgisayar işlemci hızı arttırıldığında ise SIFT kilit noktalarını bulma süresinin ve hedefi bulma süresinin azaldığı tespit edilmiştir.

Sabit/hareketli kamera ile, sabit/değişken arka plandaki sabit/hareketli hedef takibinde, optimum SIFT parametre değerleri kullanılarak güvenilir hedef takibi yapılabileceği sonucuna varılmıştır.

Bilim Kodu : 905.1.035
Anahtar Kelimeler : Hedef takibi, SIFT, kamera ile takip
Sayfa Adedi : 124
Tez Yöneticisi : Prof.Dr.Müzeyyen SARITAŞ

COMPUTER BASED TARGET TRACKING
(M.Sc. Thesis)

Nazım ÖZGEN

GAZİ UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
August 2008

ABSTRACT

Today by development of the computer technology and the image processing techniques, target tracking becomes easier in military and civilian area.

The most commonly used methods for trackig point detectors (Moravec, Harris, KLT and SIFT), background substraction and segmentation methods. They have some advantages and disadvantages with respect to each other. At this workshop, we choose SIFT method which is the best static/mobile target tracking method in the constant/variable background. SIFT program implemented in MATLAB was used in target tracking. The effects of SIFT parameters in target tracking were analyzed and optimum values of these parameters were obtained. These values are; the number of levels s=3, the necessary threshold value for discarding low contrast points |D(x)|=0.02, the necessary threshold value for discarding edge points R=10, sampling area nxn=4x4 and the number of SIFT descriptors r=8. These optimum values were found compatible with the literature.

Moreover, additional program has been written to minimize the probable false matchings between the picture frames. When the

program was used without optimum SIFT parameters, it was obtained that the deflection in target tracking was decreased.

The target tracking was analyzed by varying picture resolution and the speed of computer processor. When picture resolution was increased; it was seen that the number of SIFT keypoints and the duration of finding SIFT keypoints and the target were increased exponentially. When the speed of computer processor speed was increased, the time required to find SIFT keypoints and the target were decreased.

It was concluded that, the static/mobile target tracking has been provded using static/mobile camera with optimum SIFT parameters in the constant/variable background.

Science Code :905.1.035
Key Words : Object tracking, SIFT, tracking using camera
Page Number : 124
Adviser : Prof.Dr.Müzeyyen SARITAŞ

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren tez danışmanım Prof. Dr. Müzeyyen SARITAŞ'a, hayatım boyunca hiçbir fedakarlıktan kaçınmayan ve manevi destekleriyle beni hiçbir zaman yalnız bırakmayan babam Süleyman ÖZGEN'e, annem Şerefnur ÖZGEN'e ve kardeşim Emre ÖZGEN'e teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

RESİMLERİN LİSTESİ

Resim **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklama
SIFT	Scale Invariant Feature Transform
KLT	Kanade-Lucas Transform

1. GİRİŞ

Görüntü, nesnelerden yansıyan ışık enerjisinin ışık sensörleri tarafından algılanarak elektriksel sinyale dönüştürülmesi ile elde edilir. Etrafımızda gördüğümüz ortamın sayısal bilgi haline dönüştürülmesi ile, görüntüler üzerinde çeşitli analizler ve değişiklikler yapabilme olanağı sağlanmaktadır. Bu kapsamda çok sayıda görüntü işleme, bilgisayarla görme metotları geliştirilmiştir [1-20] ve artan işlemci hızları da daha önceden denenemeyen farklı metotların denenmesine imkan sağlamaktadır. Yapılacak analizlerin en önemlilerinden biri de etrafımızda değişen ortama ve görüntü algılayıcıların hareketine rağmen, görüntü algılayıcısı tarafından takip edilmesi istenen nesnenin ardışık resim karelerinde takip edilebilmesidir.

Hedef takibi, takip edilecek nesnenin bulunduğu ortamlar, nesnenin alabileceği konumlar ve görüntünün alındığı kameranın durumu açısından farklılıklar arz etmektedir. Burada nesne; az-orta-çok şiddetteki gürültüde, kendisiyle kıyaslandığında etrafında az-orta-çok sayıda karışık-net nesneler arasında bulunabilir. Nesneler üç boyutlu olmalarına karşın, alınan görüntüler iki boyutla sınırlı olduğundan nesne, sonraki resim karelerinde boyutsal, döngüsel değişimlere maruz kalabilir. Ayrıca, hem nesneden hem de kameradan kaynaklanabilecek hareket ile nesnenin resim içindeki konumu da sürekli değişebilmektedir.

Bu birbirinden farklı birçok olumsuz faktörün etkisi, farklı uygulamalarda değişik seviyelerde gözlenmektedir. Bu tezde, bütün bu değişimler göz ardı edilmeden hedef takibi yapılması amaçlanmaktadır. Hedef takibi konusunda literatürde karşımıza değişik yöntemler çıkmaktadır. Özel uygulamalar için geliştirilmiş özel algoritmalar da bulunmaktadır. Örneğin insan hareketlerinin analizi üzerine geniş çapta incelemeler yapılmıştır [21-23]. Yine göz takibi, el, kol gibi uzuvların takibini konu alan uygulamalar bilinen bir ortamda ve gürültü seviyesinde sabit kamera önünde çok hareketli olmayan nesnelerin

takibi kategorisine girmektedir. Bu tip uygulamalarda sistemi etkileyecek çok olumsuz bir etken bulunmamaktadır.

Bu çalışmada, çok düşük çözünürlükteki video görüntülerinde, kamera ve nesnenin bulundukları ortamlar ve aldıkları konumlar değişken bir yapıda olmasına rağmen kullanıcıya sunulan hedef penceresinin içinde kalan alanın ardışık resim karelerinde takip edilmesi amaçlanmıştır.

Hedef takibinde boyutsal, döngüsel ve parlaklık değişimlerinden en az etkilenen özellikli nokta tanımlayıcı olan SIFT [24-32] algoritması kullanılmıştır. Bu algoritma ile örneklenen resim karelerinin özellikli noktaları tanımlanarak, bir önceki resim karesindeki karşılıkları ile eşleştirilmiştir. Ayrıca SIFT algoritmasının olası hatalı eşleşmelerini ortadan kaldırmak için ek önlem alınmıştır.

Tezdeki bölümlerin içerikleri şu şekildedir. İkinci bölümde, hedef takibinde kullanılan metotların literatürdeki önemli gelişim süreçleri, avantaj ve dezavantajları ile birlikte sunulmuştur. Tezde kullanılan yöntem olan özellikli nokta tanımlayıcılar ve bu nokta tanımlayıcılardan SIFT daha ayrıntılı analiz edilmiştir. Üçüncü bölümde SIFT metodu anlatılmıştır. Dördüncü bölümde tezde kullanılan algoritma parametrelerinin hedef takibine etkileri incelenmiştir. Beşinci bölümde, farklı ortamlardaki nesne ve kameranın farklı konumları için yapılan testlerin sonuçları sunulmuş olup, altıncı bölümde elde edilen sonuçlara ve ileriye dönük yapılabilecek çalışmalara yer verilmiştir.

2. HEDEF TAKİP METOTLARI

2.1. Hedef Takibi

Yüksek hızlara ve özelliklere sahip bilgisayarların geliştirilmesi, yüksek kalitede ve düşük maliyetli video kameraların piyasada bulunması ile otomatik video analizi ihtiyacı; nesne takibi algortimalarına olan ilgiyi arttırmıştır. Bu video analizleri, aşağıdaki üç temel alanda kullanılmaktadır:

- Tanımlanmış bir hareket davranışının tespitinde
- Tespit edilen nesneleri ardışık video kareleri bazında takip etmede
- Nesne takibi analizleri ile nesnelerin davranış modellerini ortaya koymada

Bu yüzden nesne takibi aşağıdaki işlerle ilişkilidir:

- Hareket tabanlı tanımlama,
- Şüpheli hareketlerin veya farklı hareketlerin tespiti,
- Video veri tabanında bulunan videoları belirli özelliklere göre sınıflandırma,
- İnsan – bilgisayar etkileşimi (göz bebeğinin, mimiklerin takibi),
- Araç seyrüseferi (video tabanlı yol takibi, istenmeyen bir nesnenin ortadan kaldırılması).

En basit tanımıyla takip; nesnenin konumu ardışık resimler düzleminde değişirken, nesnenin güzergahını izlemek olarak tanımlanabilir. Diğer bir deyişle; takipçi, değişik video karelerindeki izlenecek olan nesneye değişmeyen etiketler atamaktadır.

Nesneleri takip etmede karşılaşılan zorluklar aşağıda sıralanmıştır.

Döngüsel değişim : Kamera hareketli olduğundan cisimlerin farklı yönleri görülür.

Boyutsal değişim : Cisimlerin boyutları onlardan uzaklaştıkça ya da yaklaştıkça resim içinde kapladıkları boyutları değişir.

Yer değişimi : Nesnenin resim içindeki yeri değişir.

Üç boyutlu bir nesneyi iki boyutlu olarak temsil ederken bilgi kaybı yaşanır.

Resimdeki gürültü.
Görüntü parlaklığının değişmesi.

2.2. Nesne Tanımlama Metotları

Bütün izleme metotları, her resim parçasının veya görüş alanına ilk defa giren nesnelerin tanınmasını ve nesneye ait özelliklerin tanımlanmasını gerektirmektedir. Bu konudaki genel görüş, alınan tek bir resim parçası üzerinden nesnelerin tanımlanmasıdır. Ancak bazı nesne tanımlama yöntemleri, yanlış tanımlama oranını düşürmek için ardışık dizilerdeki resim parça dizileri üzerinden bu işi yapmaktadır. Bu bilgiler genellikle resim parçaları arasındaki farkı alma şeklindedir. En yaygın olan nesne tanımlama metotları (Şekil 2.1) aşağıdaki şekilde tanımlanabilir.

- Arka plan çıkartımı
- Bölümleme
- Özellikli nokta tanımlayıcıları

Çizelge 2.1. Nesne tanımlama metotları

Kategoriler	**Yapılan Çalışmalar**
Arka Plan Çıkartımı	Gaussların Karışımı Öz Arka Plan Duvar Akışı Dinamik Doku Arka Plan
Bölümleme	Ortalama Kaydırma Grafik Kesimleme Aktif Çerçeve
Özellikli Nokta Tanımlayıcılar	Moravec Operatörü Harris Algılayıcı SIFT

2.2.1. Arka plan çıkartımı

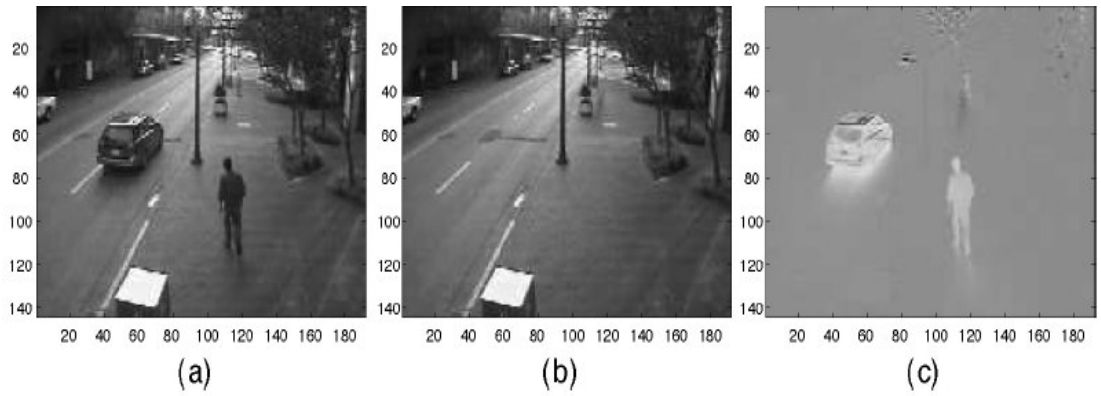
Bu metot, sabit kamera ile sabit ortamda hareket eden nesnelerin tespitinde ve takibinde çok kullanışlı bir metottur.

Ayrıca bu metot, arka plan modeli (Şekil 2.1) olarak tanımlanan görünümün temsil edildiği ve bir sonraki resimde bu arka planda oluşan sapmaların tespit edildiği bir yöntemdir. Tanımlanan arka planda önemli bir değişim olması, hareketli bir nesnenin varlığını göstermektedir.

Ardışık resim kareleri arasındaki farkı alma yöntemi 1970'lerin sonunda (Jain ve Nagel 1979) çalışılmaya başlanmıştır [33]. Bununla birlikte arka plan çıkartımı metodu, ancak 1997'de Wren'in yaptığı çalışma ile ünlenmiştir [34].

Wren'in çalışmasına göre, sabit arka plandaki her pikselin renk değeri, I(x,y), üç boyutlu tek bir Gauss fonksiyonu ile modellenmiştir. İlk resim karesindeki her (x,y) piksel için arka plana ait model çıkartıldıktan sonra, arka plana ait oluşturulan modeldeki sapmalar ön plan pikselleri olarak tanımlanır.

Gao tarafından 2000 yılında tek bir Gauss ile oluşturulan modelin, ardışık hareketler, gölgeler ve yansımalar karşısında yetersiz kaldığı ortaya konulmuştur [35]. Arka planın modellenmesinde önemli bir iyileştirme her pikselin arka plan rengi tanımlanmasıyla oluşturulan istatiksel modeller kullanılarak gerçekleştirilmiştir. Stauffer ve Grimson, 2000 yılında piksellere ait renkleri modellemek için karışık Gauss fonksiyonları kullanmışlardır [36]. Bu metotta mevcut resim karesindeki her piksel arka plan modelindeki ile karşılaştırılır. Eğer eşleşme bulunursa, eşleşen Gauss fonksiyonunun standart sapması ve ortalaması güncellenir. Aksi takdirde yeni Gauss fonksiyonunun standart sapması ve ortalaması, mevcut pikselin rengine ve bazı başlangıç standart sapma değerine eşitlenir.



Şekil 2.1. Arka plan çıkartım metodu
a) Alınan resim
b) Arka plan
c) Arka plansız resim

Diğer bir yaklaşım olarak, 2000 yılında Elgammal ve Davis tarafından sadece renk bilgisinin kullanılmasının yanında, parametrik olmayan yoğunluk yaklaşımı da piksel bazında modelleme için kullanılmıştır [37]. Bu modelde, çıkarma işlemi boyunca piksel sadece arka plandaki karşılığı ile değil aynı zamanda komşu pikseller ile de karşılaştırılmaktadır. Bu yüzden bu metot kamerada oluşacak küçük oynamaların ve arka plandaki küçük hareketlerin etkisini ortadan kaldırmaktadır. Li ve Leung ise, 2002 yılında 5x5'lik alanlar halinde hem doku hem de renk bilgisini kullanarak arka plan çıkartımı fikrini ortaya atmışlardır. Böylece, parlaklık değişiminden daha az etkileşim sağlanmıştır [38].

Arka plan çıkartım metodu için Toyama tarafından 1999 yılında üç-sıralı algoritma önerilmiştir [39]. Toyoma, piksel seviyesi çıkarımına ek olarak alan ve resim karesi seviyelerindeki bilgiyi de kullanmıştır. Toyama, piksel seviyesinde, Wiener filtresi kullanarak beklenen arka plana ait olasılık tahmini yapmıştır. Alansal seviye için, ön alandaki alanlar eşit dağılıma sahip renk ile doldurulur. Resim karesi seviyesinde ise, resim karesindeki piksellerin çoğunluğu aniden değişirse, piksel seviyesinde oluşturulan arka plan modeli geçerliliğini yitirir. Bu aşamada arka plan modeli iptal olur ve yeni arka plan modeli oluşturulur.

Bağımsız piksellerin değişiminin modellenmesi yerine, Oliver tarafından öz uzaysal kompozisyonu kullanılarak bütünsel bir yaklaşım fikri ortaya atılmıştır. k adet resim karesi girişi için, nxm boyutunda I^i : i = 1 · · · k ve kxl boyutunda arka plan matrisi B oluşturulur. Bu yaklaşım ışık değişimlerine karşı pek hassas değildir.

Arka plan çıkartım metodundaki bir kısıtlama da bu metodun statik arka plan gerektirmesidir. Bu kısıtlamalarla ilgili Monnet (2003) ile Zhong ve Sclaroff (2003) tarafından çalışmalar yapılmıştır [40-41]. Her iki yaklaşımda da zamanla değişen bir arka plan (denizdeki dalgalar, hareket eden bulutlar ve yürüyen merdivenler, vb.) ile ilgilenilmiştir. Bu modeller; resim alanını, resimdeki hareket örüntüsünü öğrenen ve tahmin etmeye yarayan işlemler dizisi şeklinde modellemektedirler.

Özet olarak yukarıda, sabit kamera ile hedef izleme uygulamalarında, arka plan çıkartım metodu açıklanmıştır. [42-45]

Değişen parlaklık, gürültü ve periyodik olarak tekrar eden arka plan alanlarının hareketi altındaki ortamlarda nesneler doğru bir şekilde seçildiklerinden dolayı bu metot kullanılmaktadır. Ayrıca bu metot işlem yükü açısından etkindir. Bu metodun en önemli kısıtlaması sabit bir arka plan gerektirmesidir. Kameranın hareketli olması, arka planın bozulmasına sebep olmaktadır. Bu yüzden sabit kamera uygulamalarında etkin bir yöntemdir. Hareketli kameralara uygulanması ile ilgili çalışmalar da bulunmaktadır [46-47]. Ancak bu çözümler, küçük hareket olduğu ve yüzeysel görünüm varsayımları altında oluşturulmuştur.

2.2.2. Bölümleme

Burada temel amaç, resimde aynı özelliklere sahip bölgeleri tespit etmektir. Bir çok yöntem bulunmaktadır. Burada en önemlilerinden bahsedilmiştir.

Bölümleme algoritmalarındaki amaç, resmi benzer alanlara bölmektir (Şekil 2.2). Her bölümleme algoritması iki temel probleme çözüm bulmalıdır. Bunlar, iyi bir bölüm için uygun kriter ve etkin bir bölümleme için uygun bir metottur [48].

Ortalama kaydırma yöntemi

Comaniciu ve Meer tarafından, 2002 yılında ortaya konulmuştur [49]. Algoritma, resim bilgisinden rasgele seçilen birçok merkez ile başlatılır. Bu yöntem ile, hem renk bilgilerinin hem de konum bilgilerinin birleştirildiği bir uzay oluşturulur. Bu uzayda renkli bir görüntüdeki her piksel, iki adet konum (x,y) ve üç adet renk (r,g,b) bilgisine sahiptir. Gri tonlu bir resimde ise renk bilgisi bire düşmektedir. Her küme merkezi, bu merkezi baz alan çok boyutlu elips etrafında kaydırılır. Yeni küme merkezleri ile eski küme merkezleri tarafından ortalama-kayma vektörü tanımlanır. Bu vektör merkezlere ait konumlar değişmeyene kadar tekrarlanır. Bu tekrarlar esnasında bazı küme merkezleri birleşecektir. Ortalama kayma yöntemi (Şekil 2.2.b) ölçeklendirilebilir. Bu metoda ait renk, yoğunluk bant genişliği ve en küçük alan boyut kriteri gibi parametrelerinin iyi seçimi, bölümlemenin performansını etkilemektedir.

Grafik-kesimlerini kullanarak bölümleme

Resim bölümleme, grafik kesimleme problemi ile formüle edilebilir (Şekil 2.2.c). Buna göre; resim, pikseller V = {u, v, ...} ve ağırlıklandırılmış kenarlar kullanılarak N adet alt alana bölümlenir. İki alt alan arasındaki ağırlıklandırılmış kenarlar “kesim” olarak adlandırılır. Ağırlık; noktalar arasındaki renk, parlaklık veya doku benzerlikleridir. Wu ve Leahy, 1993’te en küçük kesim kriterini kullanmışlardır [50]. Buradaki amaç, kesimleri en aza indiren alanları bulmaktır. Bu yaklaşımda ağırlıklar renk benzerliklerine göre tanımlanmıştır.

Shi ve Malik 2000 yılında; kesimin, kesimdeki kenar ağırlıkları toplamına ve her alan içindeki noktaların toplam ağırlığının resimdeki bütün noktalara oranına bağlı olması gerektiğini vurgulamışlardır [51].

Aktif çerçeveler

Bu yöntem, nesne sınırlarının kapalı bir çerçeve ile çevrilmesi esasına dayanmaktadır. Çerçevenin değişimi, nesne alanı içindeki enerji fonksiyonu tarafından yapılır. Enerji fonksiyonunun tanımı aşağıdaki gibidir :

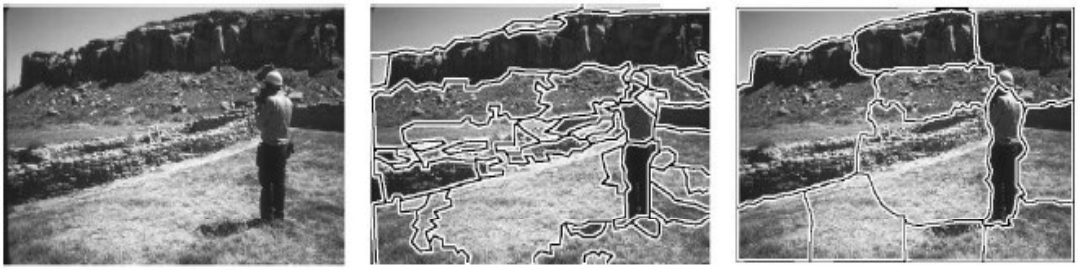
$$E(\Gamma) = \int_0^1 [E_{int}(v) + E_{im}(v) + E_{ext}(v)]ds \qquad (2.1)$$

s çerçevenin yay uzunluğu, E_{int} düzleştirme sınırlamaları, E_{im} resim tabanlı enerji ve E_{ext} ek sınırlamalarıdır.

Çerçevenin başlatılması, çerçeve tabanlı metotlardaki önemli bir husustur. Çerçeve tipik olarak nesne sınırlarını kapsayacak şekilde yerleştirilir ve nesne sınırlarına kadar daraltılır [52-53]. Çerçeve nesne sınırlarının içinde veya dışında yerleştirilmiş olabilir. Buna göre daraltılması veya genişletilmesi gerekebilir. Ancak bu, nesne veya arka plan hakkında geçmiş bilgiye sahip olmayı gerektirir. Çoklu resim kareleri veya referans resim karesi çerçevenin başlatılması için kullanılabileceği düşünülmüştür. Paragios and Deriche 2000 yılında çerçeveyi başlatabilmek için arka plan çıkarımını kullanmışlardır [54].

Bu yöntemde nesnelerin değişik yönlerinden elde edilen örnek kümesi ile yapılır. Hangi ayırt edici özelliğin seçileceği ve eğitimin nasıl bir eğitimin yapılacağı örnekler kümesi, çok büyük bir öneme sahiptir. Seçilen ayırt edici özelliğe göre eğitim gerçekleştirilir. 1994 yında Ronfard [55], 1996 yılında Zhu ve Yuille [56] ve 2004 yılında Yilmaz [57] tarafından renk üzerine,

Paragios ve Deriche [58] tarafından ise 2002 yılında doku üzerine çalışmalar yapılmıştır.



(a) (b) (c)

Şekil 2.2. Bölümleme metodu
a) Resim
b) Ortalama kaydırma yöntemi ile bölümleme
c) Grafik kesimlerini kullanarak bölümleme

Bu yöntem genellikle geniş bir örnek yelpazesi gerektirmektedir. Bu yüzden iki veya daha fazla farklı ayırt edici özelliğin kullanıldığı, küçük örnek kümelerine sahip sınıflandırıcılar da kullanılır.

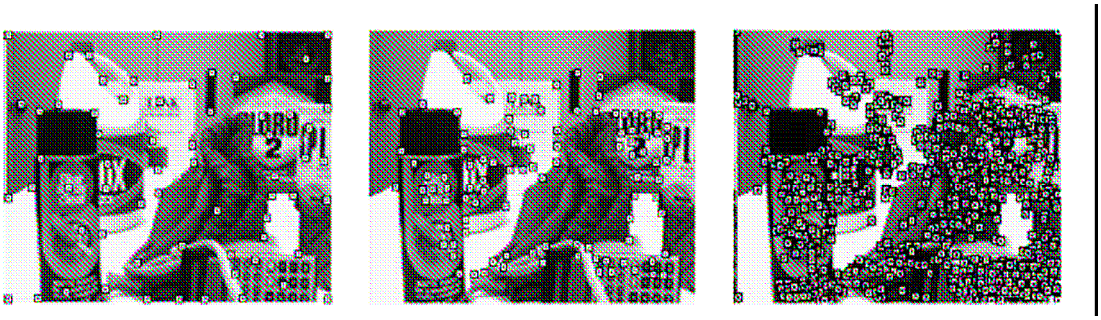
2.2.3. Özellikli nokta tanımlayıcıları

Seçilen noktaların istenilen kalitede olması; noktaların boyutsal, döngüsel değişimlerden, parlaklık, gürültü ve kamera açısı değişiminden etkilenmemeleri ile orantılıdır. Tezin gerçekleştirilmesinde özellikli nokta tanımlayıcılarından SIFT (Scale Invariant Feature Transform) maetodu kullanılmıştır.

Özellikli noktalarda aranan nitelikler aşağıdaki şekilde tanımlanabilir. Buna göre, özellikli bir nokta,

- Açık, belli, tercihen matematiksel olarak iyi tanımlanmış olmalıdır.
- Resim düzleminde iyi tanımlanmış konuma sahip olmalıdır.
- Bu noktanın etrafındaki resim yapısı bilgi içeriği yönünden zengin olmalıdır.
- Resimde oluşabilecek değişimlere (boyutsal, döngüsel, parlaklık, görünüş) karşı kararlı, başka bir deyişle yüksek tekrar edilebilirliğe sahip olmalıdır.

Literatürde özellikli nokta bulan nokta tanımlayıcıları; Moravec operatörü [59], Harris Algılayıcısı [60], KLT dedektörü [61], ve SIFT dedektörü [62]. (Bkz. Şekil 2.3)



(a) (b) (c)

Şekil 2.3. Özellikli nokta tanımlayıcılarının buldukları noktalar
a) Harris algılayıcısı b) KLT c) SIFT

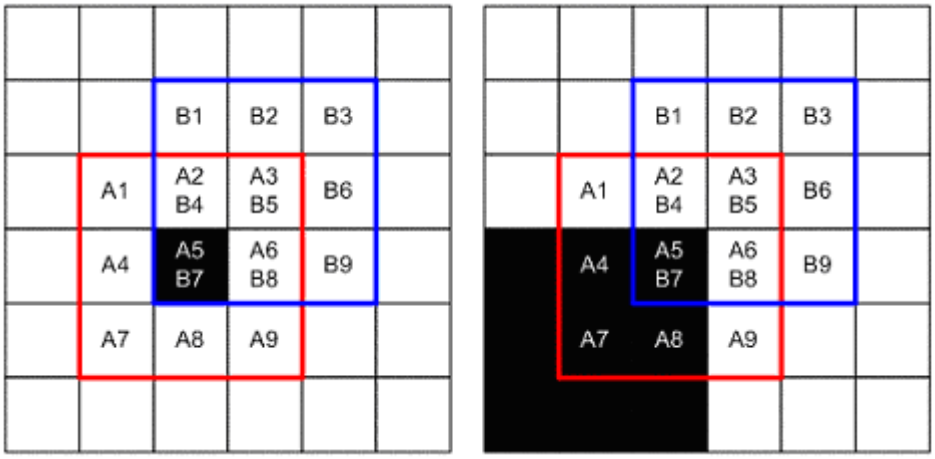
Moravec opeatörü

Hans P. Moravec, bu operatörü 1977 yılında "Standford El Arabası Seyrüseferi" projesini [63-64] yaparken geliştirmiştir. Moravec "özellikli noktalar" kavramını ortaya atmıştır. Buna göre resimlerdeki farklı alanların bu özellikli noktalar ile tanımlanabileceğini ve ardışık resim karelerinde bu noktaların eş bölgelerin eşleştirilmesinde kullanılabileceğini savunmuştur. Bu, ona, aracının bulunduğu alandaki nesnelerin varlığının ve konumunun belirlemesine izin veren önemli bir adımdı.

Moravec operatörü, bir çeşit kenar tanımlayıcı olarak düşünülebilir. Çünkü operatör, her yöndeki yoğunluk değişimi en büyük olan noktaları özellikli nokta olarak tanımlamaktadır. Ancak Moravec, özellikle kenarları tanımlamaya çalışmamıştır. O, sadece ardışık resim kareleri arasında bağımsız alanlar bulmaya çalışmıştır.

Moravec operatöründe, resimdeki yoğunluk değişimlerini tespit etmek için nxn'lik küçük bir pencere tanımlanır. Bu pencere resim düzleminde bir

noktaya yerleştirilir ve bu noktanın sekiz yönüne (dikey, yatay ve dört köşe) bir piksel miktarı kaydırılır. Yoğunluk değişimi, ilk konumdaki pencerenin içinde kalan pikseller ile kaydırılmış penceredeki karşılıklarının farklarının karesi hesaplanarak bulunur. Bu durum, Şekil 2.4'te gösterilmiştir. Kırmızı pencere pencerenin ilk konumunu, mavi ise kaydırılmış konumunu temsil etmektedir. Beyaz karelerin yoğunluğu 255, siyah karelerin yoğunluğu ise 0'dır.



(a) (b)

Şekil 2.4. 3x3'lük pencere için sağ üst köşe yönündeki yoğunluk değişimi hesabı

$$V = \sum_{i=1}^{9} (A_i - B_i)^2 = 2 * 255^2 \quad (2.2)$$

$$V = \sum_{i=1}^{9} (A_i - B_i)^2 = 3 * 255^2 \quad (2.3)$$

Şekil 2.4.(a)'daki yoğunluk değişimi Eş.2.1'de ve Şekil 2.4.(b)'deki yoğunluk değişimi ise Eş. 2.2'de belirtilmiştir.

Moravec operatörü, resimdeki her pikselin kenarlılık ölçümünde de kullanılabilir. Moravec operatörü resimdeki her piksele uygulanarak kenarlılık haritası oluşturulur.

Şekil 2.5'te 3x3'lük pencere kullanılarak oluşturulan kenarlılık haritası gösterilmiştir. Haritada bulunan değerler 255^2'ye bölünmüş değerlerdir.

- Kenar, maksimum olan değerdir.
- İzole edilmiş pikselin değeri kenardaki pikselin değeri ile aynıdır.
- Resmin kenarlarında Movarec operatörünün uygulanamadığı 'X' ile işaretlenen bölgeler vardır.

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	0	0	0	0	0	0	0	0	0	1	1	1	X	X
X	X	0	0	0	0	0	1	1	0	0	1	2	1	X	X
X	X	0	0	0	0	0	2	1	0	0	1	1	1	X	X
X	X	0	0	0	0	0	0	0	0	0	0	0	0	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Şekil 2.5. Moravec operatörü kullanılarak oluşturulan kenarlılık haritası

İzole edilmiş (Piksel yoğunluk farkı, komşu pikseller ile kıyaslandığında fazla olan, bağımsız) piksel de Şekil 2.5'te görüldüğü gibi kenar noktası gibi bulunmaktadır. Dolayısıyla Moravec operatörünün bulduğu noktalar gürültüye karşı hassastırlar. Noktaları bulmak için ne kadar büyük pencere kullanılırsa, kenar noktalarının izole edilmiş noktaya göre daha fazla yoğunluk değişimi olacağından, gürültüye karşı o kadar daha fazla dayanıklılık kazanılmış olacaktır. Her nokta için yoğunluk değişimi hesaplandığından, gri tonlu resimlerde istenmeyen bazı noktalar kenar noktası olarak bulunacaktır. Bu problem, belli bir eşik değer konularak ortadan kaldırılır. Bu eşik değerin seçimi önemlidir. Çünkü hatalı noktaları ayıklayabilecek kadar büyük, doğru noktaları elemeyecek kadar da küçük olmalıdır. Ayrıca, resmin kenarlarında bulunan noktaların yoğunluk değişimi de hesaplanamamaktadır.

Moravec Operatörü Algoritması

Resimdeki yoğunluk dağılımı I(x,y) olmak üzere;

Girdiler: Gri tonlu resim, Pencere büyüklüğü, Eşik değeri T

Çıktılar: Kenarlılık Haritası

1. Resimdeki (x,y) konumundaki her piksel için yoğunluk değişimi, pencere (u,v) miktarı kaydırılarak aşağıdaki şekilde hesaplanır:

$$V_{u,v}(x,y)= \sum_{a,b\ (penceredeki)} (I(x+u+a,y+v+b)-I(x+a,y+b))^2 \qquad (2.4)$$

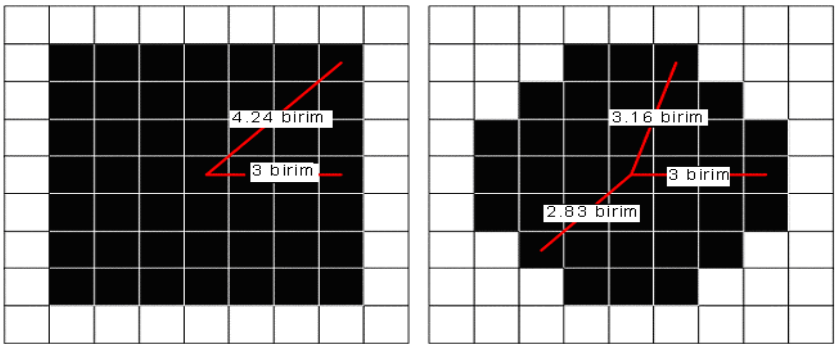
   (u,v) = (1,0), (1,1), (0,1), (-1,1), (-1,0), (-1,-1), (0,-1), (1,-1)

2. Her piksel için Kenarlılık Haritasını oluşturmak için gerekli olan kenarlılık ölçüsü olan C(x,y) = min ($V_{u,v}(x,y)$) hesaplanır.
3. C(x,y) değerlerinden T eşik değerinin altında olanlar sıfıra eşitlenir.
4. Sıfır olmayan noktalar kenar noktalarıdır.

Özellikli noktaları belirlemek için, 4x4'lük pikselin içinde yatay, dikey, çapraz yönlerde resim parlaklığı hesaplanır ve 4x4lük pencere içindeki en az parlaklık değişiminin olduğu nokta belirlenir.

Moravec Opeatöründe Karşılaşılan Problemler

1. Döngüsel değişimlere karşı dayanıklı değildir.



(a) (b)

Şekil 2.6. Kare ve dairesel penceredeki öklit mesafeleri
a) Öklit mesafesindeki en büyük değişim : 1.24 br. b) Öklit mesafesindeki en büyük değişim : 0.17 br.

2. Moravec tarafından kullanılan pencere karedir. Yoğunluk değişimine daha doğru bir yaklaşım yapmak için merkezden pencerenin kenarındaki her noktaya eşit uzaklığa sahip dairesel bir pencere gerekmektedir. Bu pencereye en yakın pencere, Şekil 2.6'da gösterilmiştir.

Yoğunluk değişimindeki doğruluğu daha da arttırabilmek için penceredeki pikseller ağırlıklandırılır. Pencerenin merkezindeki piksellerin katsayısı arttırılırken, kenardakilerin katsayısı azaltılır.

Dairesel yapıda bir pencerenin gerekliliği ve pencere merkezindeki piksellerin katsayılarının daha fazla olması, Gauss penceresi ile bu işin daha sağlıklı yapılabileceğini gündeme getirmiştir. Genellikle kullanılan 5x5'lik Gauss penceresi Şekil 2.7'deki gibidir.

w1 .004	w2 .015	w3 .026	w4 015	w5 .004
w6 015	w7 .059	w8 .095	w9 .059	w10 015
w11 .026	w12 .095	w13 .15	w14 .095	w15 .026
w16 015	w17 .059	w18 .095	w19 .059	w20 015
w21 .004	w22 015	w23 .026	w24 015	w25 .004

Şekil 2.7. 5x5'lik gauss penceresi için genel katsayılar

Yoğunluk Değişimi : $V = \sum_{i=1}^{25} w_i (A_i - B_i)^2$ dır. (2.5)

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	0	0	0	0	0	0	0	0	0	1	1	1	X	X
X	X	1	2	1	0	0	1	1	0	0	1	2	1	X	X
X	X	2	2	2	0	0	2	1	0	0	1	1	1	X	X
X	X	1	1	1	0	0	0	0	0	0	0	0	0	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

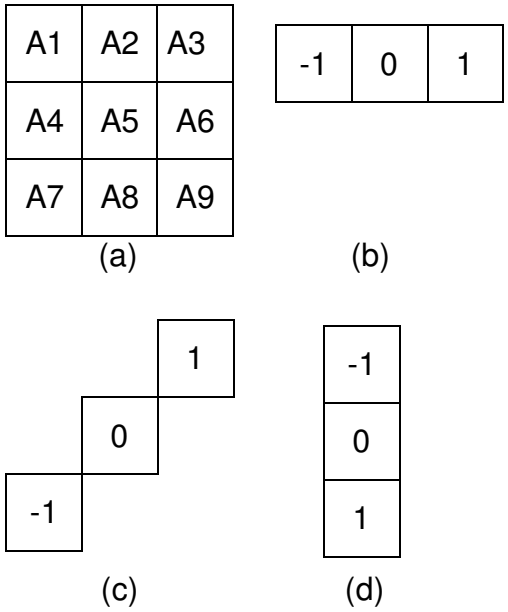
Şekil 2.8. Gürültü sebebiyle kenarlar boyunca hatalı oluşan tanımlamalar

3. Kenarlar boyunca ve izole edilmiş piksellerde hatalı noktalar bulmaktadır.
4. İşlem hızı yönünden minimum düzeydedir.

Harris algılayıcı

Chris Harris ve Mike Stephens, 1988 yılında robotların resim karelerini kullanarak etraflarındaki çevreyi tanımaları için yaptıkları çalışmalar esnasında Harris Algılayıcıyı geliştirmişlerdir [65]. Moravec gibi Harris ve Stephens de ardışık resim karelerinde karşılık gelen noktaları eşleştirebilecekleri bir metoda ihtiyaç duymaktaydılar. Bunun için kenar ve köşe noktalarını kullanabileceklerini düşünmüşlerdir.

Harris Algılayıcısında, Moravec operatöründe yapılan yoğunluk değişim ölçümlerinin sadece 45'er derecelik yönlerde değil istenilen yönlerde de yapılması sağlanmıştır. Burada birinci dereceden türevler kullanılmıştır ve türevler için Şekil 2.9'daki yaklaşımlar yapılmıştır. Şekil 2.10'da Moravec operatörünün de bu şekilde temsil edilebileceği gösterilmiştir.



Şekil 2.9. Harris algılayıcısında türev yaklaşımları

A5'teki yatay türev :

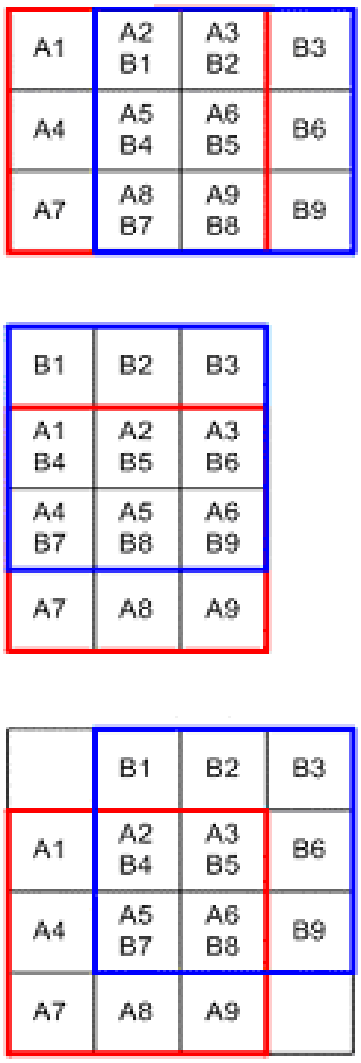
$$\frac{\partial I_{A5}}{\partial x} \approx (I_{A6} - I_{A4}) = I_{A5} \otimes (-1,0,1) \tag{2.6}$$

A5'teki üst sağ çapraz türev :

$$\frac{\partial I_{A5}}{\partial h} \approx (I_{A3} - I_{A7}) = I_{A5} \otimes \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \tag{2.7}$$

A5'teki dikey türev :

$$\frac{\partial I_{A5}}{\partial y} \approx (I_{A2} - I_{A8}) = I_{A5} \otimes (-1,0,1)^T \tag{2.8}$$



Şekil 2.10. Resim türevi kullanılarak Moravec operatörü yaklaşımı

Yatay Moravec Yoğunluk Değişimi :

$$\frac{\partial I_i}{\partial x} = I_i \otimes (-1,0,1)^T \approx B_i - A_i \text{ olmak üzere}$$

$$V_x = \sum_{i=1}^{9} w_i (A_i - B_i)^2 = \sum_{i=1}^{9} w_i (B_i - A_i)^2 \approx \sum_{i=1}^{9} \left(\frac{\partial I_i}{\partial x} \right)^2 \qquad (2.9)$$

Çapraz Moravec Yoğunluk Değişimi :

$$\frac{\partial I_i}{\partial h} = I_i \otimes \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \approx B_i - A_i \text{ olmak üzere} \tag{2.10}$$

$$V_h = \sum_{i=1}^{9} w_i (A_i - B_i)^2 = \sum_{i=1}^{9} w_i (B_i - A_i)^2 \approx \sum_{i=1}^{9} \left(\frac{\partial I_i}{\partial h} \right)^2 \tag{2.11}$$

Dikey Moravec Yoğunluk Değişimi :

$$\frac{\partial I_i}{\partial y} = I_i \otimes (-1,0,1)^T \approx B_i - A_i \text{ olmak üzere}$$

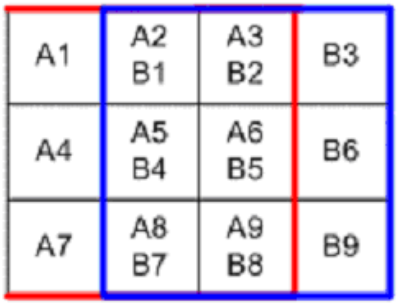
$$V_y = \sum_{i=1}^{9} w_i (A_i - B_i)^2 = \sum_{i=1}^{9} w_i (B_i - A_i)^2 \approx \sum_{i=1}^{9} \left(\frac{\partial I_i}{\partial y} \right)^2 \tag{2.12}$$

Yukarıda yapılan analizler, yoğunluk değişiminin resmin istenilen yöndeki türevi ile de hesaplanabileceğini ortaya koymuştur. Penceredeki (u,v) miktarı kayma için oluşacak yoğunluk değişimi aşağıda verilmiştir :

$$V_{u,v}(x,y) = \sum_{\text{Merkezi}(x,y)\text{olan}\forall i\_\text{penceresi}\_\text{için}} w_i \left(u \frac{\partial I_i}{\partial x} - v \frac{\partial I_i}{\partial y} \right)^2 \tag{2.13}$$

w_i : i konumundaki Gauss penceresine ait ağırlık

Moravec operatörünün kullandığı pencere karesel olduğundan yoğunluk değişimi gürültüden etkilenmektedir. Bu olumsuz etki Gauss penceresi kullanılarak giderilmiştir. Yoğunluk değişiminin 3x3'lük Gauss penceresi ile gerçekleştirilmiş hali Şekil 2.11'de gösterildiği gibidir. Ağırlıklar ise, $w_{u,v} = e^{-(u+v)^2/2\sigma^2}$ ile hesaplanır.



w1 0.04	W2 0.12	w3 0.04
w4 0.12	w5 0.36	w6 0.12
w7 0.04	w8 0.12	w9 0.04

Şekil 2.11. Gauss penceresini kullanarak dikey kaymanın hesaplanması

Ağırlıklandırılmış Yatay Yoğunluk Değişimi :

$\frac{\partial I_i}{\partial x} = I_i \otimes (-1,0,1) \approx B_i - A_i$ olmak üzere

$$V_x = \sum_{i=1}^{9} w_i (A_i - B_i)^2 = \sum_{i=1}^{9} w_i (B_i - A_i)^2 \approx \sum_{i=1}^{9} w_i \left(\frac{\partial I_i}{\partial x} \right)^2 \quad (2.14)$$

Moravec operatörü kenarlarda oluşacak gürültüye karşı hassastır ve çabuk etkilenmektedir. Harris algılayıcı, kenarlılık ölçümünü aşağıdaki gibi yeniden formüle etmektedir.

$$V_{u,v}(x,y) = \sum_{\text{Merkezi}(x,y)\text{olan}\forall i\_\text{penceresi}\_\text{için}} w_i \left(u \frac{\partial I_i}{\partial x} + v \frac{\partial I_i}{\partial y} \right)^2 \quad (2.15)$$

$$V_{u,v}(x,y) = \sum_{\text{Merkezi}(x,y)\text{olan}\forall i\_\text{penceresi}\_\text{için}} w_i \left(u^2 \frac{\partial I_i}{\partial x}^2 + 2uv \frac{\partial I_i}{\partial x} \frac{\partial I_i}{\partial y} + v^2 \frac{\partial I_i}{\partial y}^2 \right) \quad (2.16)$$

$Au^2 + 2Cuv + Bv^2$

$$A = \left(\frac{\partial I}{\partial x} \right)^2 \otimes w, B = \left(\frac{\partial I}{\partial y} \right)^2 \otimes w, C = \left(\frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \right) \otimes w \quad (2.17)$$

$\otimes$: Konvulasyon operatörü

Harris ve Stephens kenarlılık ölçümünün aşağıdaki gibi matris eşitliği şeklinde de yazılabileceğini ortaya koymuşlardır:

$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$ olmak üzere

$$V_{u,v}(x,y) = Au^2 + 2Cuv + Bv^2 = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \quad (2.18)$$

M matrisi, döngüsel olarak dayanıklı bir yapı oluşturulmasını sağlamıştır.

Harris Algılayıcısına Ait Algoritma

Girdiler : Gri tonlu resim, Gauss penceresi (Pencerenin yarıçapı Gauss fonksiyonu standart sapmasının 3 katı kadardır), k değeri, eşik değer T

Çıktılar : Algılanan köşelerin oluşturduğu harita

1. Resimdeki her (x,y) için oto korelasyon matrisi M hesaplanır.

$A = \left(\frac{\partial I}{\partial x}\right)^2 \otimes w, B = \left(\frac{\partial I}{\partial y}\right)^2 \otimes w, C = \left(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\right) \otimes w$ olmak üzere

$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$

2. Her piksel için kenarlılık ölçümü olan C(x,y) hesaplanır:

$\det(M) = \lambda_1\lambda_2 = AB - C^2$

$\text{trace}(M) = \lambda_1 + \lambda_2 = A + B$

k : sabit olmak üzere

$C(x,y) = \det(M) - k(\text{trace}(M))^2$

3. C(x,y) değerlerinden T eşik değerinin altında olanlar sıfıra eşitlenir.

4. Sıfır olmayan noktalar kenar noktalarıdır.

Harris algılayıcı, Gauss penceresinden geçirildiği için Moravec operatörüne göre daha ağır hesaplamalar içermektedir. Harris algılayıcısı da Movarec

operatörü gibi gürültüden etkilenmektedir. Gauss penceresini büyüterek bu etki azaltılabilir, ancak işlemsel yük artacaktır. Sonuç olarak, Harris algılayıcısı, Moravec operatörüne oranla daha yüksek tekrarlanabilme oranına ve işlemsel yüke sahiptir.

KLT

Lucas-Kanade metodu [66], t anında ve t+α anlarında alınan iki resim karesi arasındaki hareketi hesaplamaya çalışır. Hesaplamalar pikselin yoğunluk değişiminin çok fazla olmadığı temeline dayanmaktadır. Bu tip metodlar diferansiyel metod olarak adlandırılırlar. Çünkü bu tip algoritmalar Taylor serisi yaklaşımını kullanırlar. Yani pikselin bulunduğu koordinatlara göre parçalı türev kullanılmaktadır.

(x,y,z,t) konumundaki I(x,y,z,t) yoğunluğa sahip pikselin δx, δy, δz and δt miktarı kayması sonucunda oluşacak yeni konumdaki yoğunluk değeri $I(x,y,z,t) = I(x + \delta x, y + \delta y, z + \delta z, t + \delta t)$ olarak bulunur.

Kayma miktarının yeterince küçük olduğunu düşünürsek, Taylor serisini kullanarak yapılacak yoğunluk değişimi,

$$I(x + \delta x,\ y + \delta y,\ z + \delta z,\ t + \delta t) = I(x,y,z,t) = \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial z}\delta z + \frac{\partial I}{\partial t}\delta t + \text{Y.D.T.}$$

Y.D.T. yüksek dereceli terimleri temsil etmekte olup, göz ardı edilbilecek kadar küçüktür.

V_x, V_y, V_z resimdeki yoğunluk ($I(x,y,z,t)$) değişim hızı olmak üzere, yukarıdaki eşitlik ve Y.D.T'nin ihmal edilmesi ile aşağıdaki sonuçlara varılır :

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial z}\delta z + \frac{\partial I}{\partial t}\delta t = 0 \qquad (2.19)$$

veya

$$\frac{\partial I}{\partial x}\frac{\delta x}{\delta t}+\frac{\partial I}{\partial y}\frac{\delta y}{\delta t}+\frac{\partial I}{\partial z}\frac{\delta z}{\delta t}+\frac{\partial I}{\partial t}\frac{\delta t}{\delta t}=0 \qquad (2.20)$$

$$\frac{\partial I}{\partial x}Vx+\frac{\partial I}{\partial y}Vy+\frac{\partial I}{\partial z}Vz+\frac{\partial I}{\partial t}=0 \qquad (2.21)$$

I_x, I_y, I_z and I_t aşağıdaki şekillerde yazılabilir:

$$I_xV_x + I_yV_y + I_zV_z = - I_t \text{ veya } \nabla I \bullet \vec{V} = -I_t \qquad (2.22)$$

Burada üç bilinmeyenli bir denklem karşımıza çıkmaktadır ve denklemi çözebilmek için ek eşitliklere ihtiyaç vardır. Lucas ve Kanade tarafından resim içindeki küçük alanlarda sabit bir akış olduğu yaklaşımı benimsenmiştir.

(V_x, V_y, V_z)'nin, m gibi küçük bir pencere içinde sabit olduğu varsayılmıştır. m>1 ve sabittir. x,y,z pencerenin merkezinde ve 1..n'ye kadar numaralandırılmış piksellerdir. $n=m^3$'tür. Bu varsayımlardan sonra aşağıdaki eşitlik kümesi elde edilir :

$$I_{x1}V_x + I_{y1}V_y + I_{z1}V_z = -I_{t1}$$
$$I_{x2}V_x + I_{y2}V_y + I_{z2}V_z = -I_{t2}$$
.
.
$$I_{xn}V_x + I_{yn}V_y + I_{zn}V_z = -I_{tn}$$

Üç denkleme ihtiyaç duyulurken burada daha fazla miktarda denklem bulunmaktadır. Bu eşitlik aşağıdaki şekilde çözülebilir :

$$\begin{pmatrix} I_{x1} & I_{y1} & I_{z1} \\ I_{x2} & I_{y2} & I_{z2} \\ \vdots & \vdots & \vdots \\ I_{x3} & I_{y3} & I_{z3} \\ & & \end{pmatrix} \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} = \begin{pmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{pmatrix} \Rightarrow A\vec{v} = -b \qquad (2.23)$$

$$\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} = \begin{pmatrix} \sum I_{xi}^2 & \sum I_{xi}I_{yi} & \sum I_{xi}I_{zi} \\ \sum I_{xi}I_{yi} & \sum I_{yi}^2 & \sum I_{yi}I_{zi} \\ \sum I_{xi}I_{zi} & \sum I_{yi}I_{zi} & \sum I_{zi}^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum I_{xi}I_{ti} \\ -\sum I_{yi}I_{ti} \\ -\sum I_{zi}I_{ti} \end{pmatrix}, \; i=1..n \qquad (2.24)$$

Yukarıda yapılan hesaplamalar resmin dört yöndeki türevleri hesaplanarak iki resim arasındaki kayma miktarının bulunabileceğini göstermektedir. Ayrıca pencerenin merkezindeki piksellerde katsayısı daha yüksek olan ağırlık fonksiyonu *W(i,j,k)* eklenmelidir. Bunun için Gauss fonksiyonu kullanılır. Ayrıca ölçeksel uzay da oluşturulabilir.

KLT izleme metodunda da $M = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix}$ eşitliği kullanılır.

Temelde Harris ve KLT izleme metodu aynıdır. Yaklaşık olarak da aynı noktaları içerirler. M matrisi döngüsel değişimlere karşı kısmen korunumludur, ancak üç boyutlu değişimlere karşı korumasızdır.

SIFT

Farklı değişimlere karşı dayanıklı bir model elde edebilmek için SIFT (Scale Invariant Feature Transform) metodu dört adımda gerçekleştirilir. SIFT metodu ile ilgili ayrıntılı açıklamalar Bölüm 3'te yapılmıştır.

3. ÖLÇEKSEL DEĞİŞMEYEN ÖZELLİK DÖNÜŞÜMÜ (SIFT)

SIFT metodu, 1999 yılında David G. Lowe tarafından ortaya konulmuş olup, David Lowe teorisinin patentini 2004 yılında almıştır. David G. Lowe halen British of Colombia Üniversitesi Bilgisayar Mühendisliği Bölümü'nde ders vermekte ve araştırmalarına devam etmektedir.

SIFT algoritması dört aşamadan oluşan bir algoritmadır:
- *Ölçeksel uzaydaki uç noktaların (minimum - maksimum) elde edilmesi:* Algoritmanın ilk aşaması olan bu kısımda, tüm ölçekler ve resim konumları araştırılır. Boyutsal ve döngüsel değişime karşı dayanıklı potansiyel özellik noktalarının tanımlanması, DoG fonksiyonu ile etkili bir şekilde gerçekleştirilir.
- *Kilit noktaların konumlarının belirlenmesi:* Her aday noktada konum ve ölçek belirlemek için ayrıntılı bir model oluşturulur. Kilit noktalar, özellik noktalarının kararlılığına göre seçilir.
- *Döngüsel değişime karşı dayanıklılık kazanılması:* Kilit noktanın bulunduğu her konum için bir veya daha fazla yön atanır. Daha sonra yapılacak olan tüm işlemler; yön, ölçek ve konum olarak ataması yapılmış kilit noktalar baz alınarak yapılacağından bu değişimlere karşı dayanıklılık kazanılmış olunur.
- *Kilit nokta tanımlayıcıların bulunması:* Her kilit nokta için etrafındaki alanda ve seçilen ölçeksel büyüklükte resim gradyentleri ölçülür. Bu gradyentler, parlaklık ve şekil değişimlerine karşı kararlılık kazanılmasını temin edecek şekilde temsil edilirler.

Bu algoritma SIFT olarak adlandırılır. Çünkü bu algoritma ile resim bilgileri ölçeksel bazda değişmeyen koordinatlara dönüştürülür. SIFT ile resmi ölçeksel ve konumsal bazda tam anlamıyla kapsayabilmek için fazla miktarda özellik noktası bulunur. Resmin içeriğine ve SIFT algoritmasında seçilen parametrelere göre değişiklik göstermesine rağmen, tipik bir 500x500 piksellik resim için yaklaşık 2000 civarında SIFT noktası bulunur.

3.1. SIFT Yönteminin Diğer Yöntemlere Göre Üstünlükleri ve Dezavantajları

Ön bilgi (model,geçmişe ait bilgi,vs.) gerektirmemesi, arka plan çıkartımında olduğu gibi sabit kamera görüntüsü yerine hareketli kamera görüntülerinde de işlevsel olabilmesi arka plan çıkartımı ve bölümleme yöntemlerine göre özellikli noktalarla yapılan takibin avantajlarıdır. Ancak her seferinde noktasal bazda özellikli nokta bulma gerekliliği, işlem süresini olumsuz olarak etkilemektedir.

Özellikli nokta ile nesne takibindeki yöntemler arasında SIFT metodunun boyutsal, döngüsel, parlaklık ve üç boyutlu değişimlere karşı en dayanıklı yöntem olduğu 2005 yılındaki Mikolajczyk K. ve Schmid, C. tarafından yapılan çalışmalar sonucunda tespit edilmiştir [67]. Buna göre;

- 50 derecelik üç boyutlu dönüşümde en iyi sonucu veren algoritma SIFT algoritmasıdır.
- Dokusal ve yapısal görünüme sahip ortamlardaki tanımlayıcılar içinde en iyi performansı SIFT tanımlyıcıları göstermektedir.
- 2-2.5 arasındaki ölçeksel değişimler ve 30-45 derecelik döngüsel değişimlerde en iyi performansı yine SIFT sağlamaktadır.
- Kenar tanımlamaya dayalı olan tanımlayıcıların performansı gürültülü ortamlarla karşılaşıldığında düşmektedir. Bu konuda SIFT algoritması çok daha iyi bir performans sergilemektedir. Ayrıca SIFT ortamda meydana gelebilecek olan ışık değişimlerinde de iyi sonuçlar vermektedir.

SIFT metodu, noktasal bazda özellik vektörü çıkaran tüm algoritmalar gibi, noktasal bazda özellik vektörü çıkarmada yüksek doğruluk oranına sahip olmasına karşın ağır miktarda hesaplama gerektirmektedir. Ancak artan mikroişlemci hızları ile bu açık kapanmıştır. Burada resmin içeriği ve boyutu yani tespit edilecek ve işleme tabi tutulacak noktaların sayısı da büyük önem arz etmektedir. Ne kadar çok SIFT kilit noktası bulunursa, sistemin doğruluğu o kadar artmakta ancak işlem süresi de aynı oranda uzamaktadır. SIFT kilit

nokta sayısı oluşturulan sistemin işlem süresini ve doğruluğunu etkilemektedir.

SIFT algoritmasının ;

- Boyutsal
- Döngüsel
- Parlaklık
- Gürültü
- Üç boyutlu değişimlere karşı en dayanıklı algoritma olduğu görülmektedir.

Bu sebeple özellikli nokta bulmak için SIFT algoritması kullanılmıştır.

3.1.1. Ölçeksel uzaydaki uç noktaların (minimum- maksimum) elde edilmesi

İlk olarak ölçeksel uzay oluşturulur. Daha sonra kilit nokta olabilecek noktaların tespiti yapılır. Orijinal resim, farklı standart sapmaya sahip Gauss filtrelerinden geçirilir. Elde edilen Gauss filtresinden geçirilmiş resimler birbirleri arasında ardışık olarak çıkartma işlemine tabi tutulur ve böylece DoG (Diffrence of Gaussians) diye tabir edilen önce Gauss fitresinden geçirilmiş sonra birbirleri arasında çıkarma işlemine tabi tutulmuş resimler elde edilir. Bu yeni oluşturulan resimlerden elde edilen uç noktalar (minimum-maksimum) ile nesnenin resim içindeki ölçek değişikliğine karşı dayanıklı hale gelinmiş olunur.

Ölçeksel uzayda sabit, değişmeyen kilit noktalar bulmak için birçok teknik kullanılabileceği düşünülmekteydi. Bunlardan biri DoG (Difference of Gaussians) metoduydu. Koenderink (1984) [68] ve Lindeberg (1994) [69]; yaptıkları çeşitli varsayımlar sonucunda ölçeksel uzayın Gauss fonksiyonu kullanılarak oluşturulabileceğini tespit etmişlerdir. Bu sebeple SIFT algoritmasında DoG metodu kullanılmıştır. Ölçeksel uzay Lindeberg'in isminin baş harfi olan "L" harfi ile temsil edilmektedir.

I: orijinal resim, G: değişken orantıya sahip Gauss fonksiyonu

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (3.1)$$

olmak üzere, ölçeksel uzay fonksiyonu şu şekilde tanımlanır:

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \quad (3.2)$$

DoG fonksiyonu olan $D(x, y, \sigma)$, biri diğerine göre k kadar orantılı büyüklüğe sahip Gauss filtreli iki resim arasındaki farkın sonucunda elde edilir:
$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$
$\sigma\nabla^2 G = \partial G/\partial \sigma \approx [G(x, y, k\sigma) - G(x, y,\sigma)] / (k\sigma-\sigma)$
ve dolayısıyla $G(x, y, k\sigma) - G(x, y,\sigma) = (k-1).\sigma^2\nabla^2 G$ elde edilir.

Bu eşitlik, resmi Gauss filtresinden geçirmek ile Laplasını almak arasında ciddi bir benzerlik bulunduğunu göstermektedir. Lindeberg σ^2 ile normalize edilmiş bir Laplasın ancak doğru ölçeksel kararlı bir yapı sağlayacağını göstermiştir. Detaylı deneysel karşılaştırmalar sonucunda; Mikolajczyk (2002) tarafından σ^2 ile normalize edilmiş bir Laplasa ($\sigma^2\nabla^2 G$) ait uç noktaların, Hessian veya Harris kenar fonksiyonu gibi diğer fonksiyonlar ile karşılaştırıldığında en kararlı resim özelliklerini oluşturulduğu bulunmuştur [70].

DoG fonksiyonu, (k-1) sabiti kadar farklı olsa da σ^2 ile normalize edilmiş bir Laplas fonksiyonunu içermektedir. (k-1) sabiti de bütün ölçekler bazında sabittir ve dolayısıyla hesaplanacak olan uç noktanın konumunu etkilememektedir.

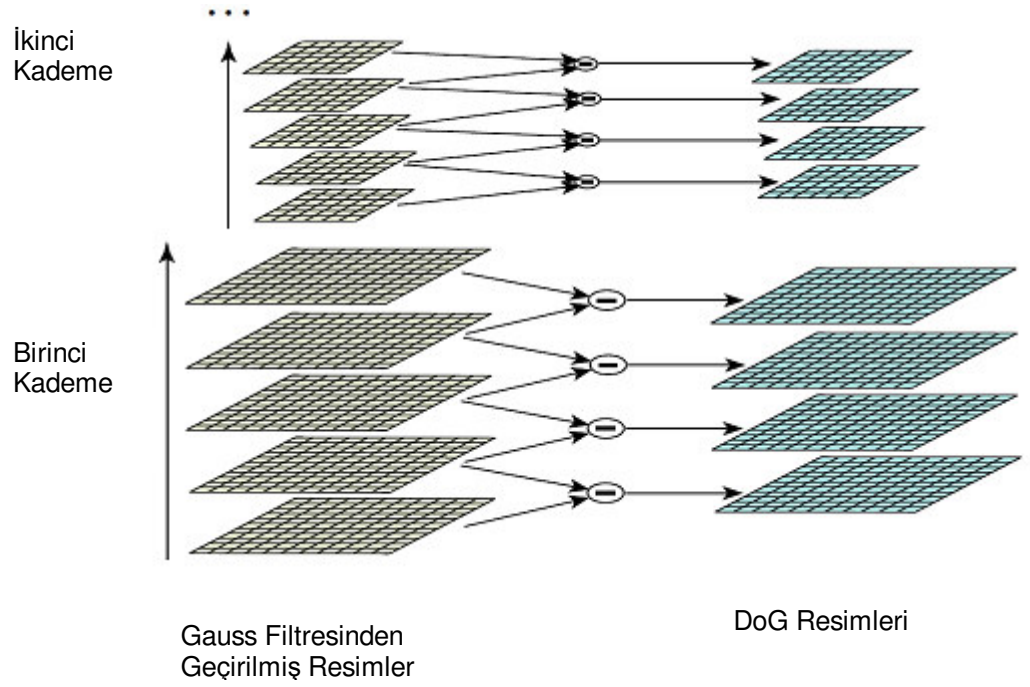
SIFT algoritmasında bu işlem şu şekilde gerçekleştirilir:
1) Orijinal resim farklı standart sapmaya sahip olan Gauss filtrelerinden geçirilir.

2) Gauss filtresinden geçirilen resimler, Şekil 3.1'deki gibi kademeler halinde gruplandırılırlar.

3) Bir sonraki kademenin ilk resmi 2 kat daha düşük örneklemeye tabi tutularak oluşturulur ve bu kademedeki ilk resim farklı standart sapmaya sahip olan Gauss filtrelerinden geçirilir. Bu sayede bir üst kademe oluşturulur.

4) Her kademedeki komşu Gauss filtreli resimler birbirinden çıkartılır ve böylece DoG resimleri oluşturulmuş olur.

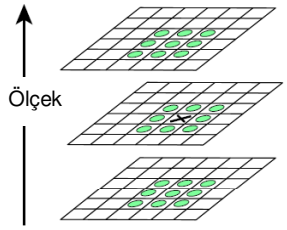
5) k, her kademe başına sabit miktarda Gauss filtresinden geçirilmiş resim elde etmek için seçilir.



Şekil 3.1. Ölçeksel uzayın oluşturulması

DoG resimleri oluşturulduktan sonra, DoG resminin (D(x, y, σ)) lokal minimum ve maksimum noktalarını bulmak için her nokta aynı ölçekteki 8 komşusu ve 1 "bir" ölçek alt ve üstündeki 9 komşusu ile, yani toplam 26 nokta ile karşılaştırılır. Eğer bu değer bütün bu noktalar içindeki minimum veya maksimum nokta ise bu nokta uç nokta olarak adlandırılır (Bkz. Şekil 3.2).

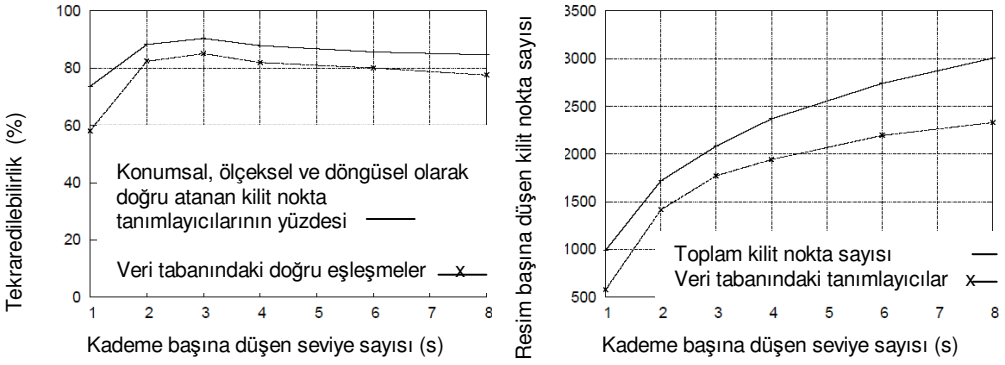
26 adet nokta içinden sadece bir noktanın komşularına göre minimum veya maksimum olup olmadığı kontrol edilmektedir. Dolayısıyla bu kontrol sonucunda tespit edilen noktaların hesaplanması için geçen süre üst kademelerde büyük ölçüde düşer.



Şekil 3.2. Uç noktaların bulunması

Lowe tarafından yapılan çalışmalar ile; s her kademede bulunacak olan seviye sayısı olmak üzere, k= 2^1/s olarak seçilir. Her kademede s+3 adet Gauss filtresinden geçirilmiş resim bulunur. Bu resimlerin birbirleri arasında çıkarma işlemine tabi tutulması sonucunda ise s+2 adet DoG resmi elde edilmiş olur.

Kilit noktaların tekrar edilebilirliğinin sağlanması için, Lowe s değerinin 3 olarak seçilmesi gerektiğini bulmuştur. Bu en uygun s değerini bulabilmek için Lowe tarafından çeşitli denemeler yapılmıştır. Bu kapsamda temelde iki deney yapılmıştır. Birincisi oluşturulan veri tabanındaki resim parçalarının eşleştirilmesi, ikincisi ise farklı resim karelerindeki konumsal, döngüsel ve ölçeksel değişimlere karşı tanımlanan kilit noktaların doğru olarak tanımlanmasıdır. Bu iki deney kullanılarak, kademe başına düşen seviye sayısı s değeri ile kilit noktaların tekrar edilebilirliği arasındaki ilişki Şekil 3.3'te gösterilmiştir.



Şekil 3.3. Kademelerde bulunacak resim sayısının (a) tekrar edilebilirliğe ve (b) kilit nokta sayısına olan etkisi

Gauss fonksiyonunun standart sapması olan σ değerinin büyük olması, kilit noktaların tekrar edilebilirliğini arttırmasına karşın Gauss pencere büyüklüğünü ve işlem süresini aynı ölçüde arttırmaktadır. Bu sebeple Lowe en ideal σ değeri olarak 1.6'nin alınması gerektiğini savunmuştur.

3.1.2. Kilit noktaların konumlarının belirlenmesi

Bir önceki aşamada, kararlı olmamasına rağmen tespit edilen bir çok kilit nokta adayı olmuştur. Bu aşamada kararlı olmayan yani düşük kontrasta sahip (dolayısıyla gürültüden daha çok etkilenen) veya kenarlarda zayıf olarak tespit edilmiş kilit noktaların ayıklanması sağlanır.

Bir önceki aşamada bulunan kilit nokta adayları arasından hangilerinin gerçek kilit nokta olduğunun tespit edilip, bu kilit noktaların doğru konumlara yerleştirilmesi gerekmektedir. Lowe bu aşamayı ilk algoritmayı oluşturduğunda (1999) birinci aşamada tespit edilen her kilit nokta adayını ölçeksel ve konumsal bazda aynen kilit nokta olarak atayarak, gerçekleştirmiştir. Ancak daha sonra 2002 yılında Brown ile birlikte yaptığı çalışmalar [71] sonucunda, kilit noktaları atarken kilit nokta adayı olarak bulunan uç noktaların interpolasyonu ile hesaplanan konumlarının atanmasının daha doğru eşleşmelere ve daha kararlı bir yapıya ulaştığını görmüştür.

İnterpolasyon DoG fonksiyonunun (D(x,y,σ)) kilit aday nokta baz olmak üzere ikinci dereceden Taylor serisi açılımı yapılarak gerçekleştirilir ve aşağıdaki eşitlik elde edilir.

$$D(x) = D + \frac{1}{2}(\frac{\partial D}{\partial x})^T + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \tag{3.3}$$

Burada DoG fonksiyonu olan D(x) ve türevi kilit aday nokta için hesaplanır. **x**= $(x,y,\sigma)^T$ bu noktadan olan sapma uzaklığıdır.

Uç noktalar, Eş. 3.3'teki D(x)'in türevinin sıfıra eşitlenmesi ile bulunur. Buna göre de uç noktaların interpolasyona uğramış konumları aşağıdaki şekilde ifade edilebilir:

$$\hat{\mathrm{x}} = -\frac{\partial^2 \mathrm{D}}{\partial \mathrm{x}^2}^{-1} \frac{\partial \mathrm{D}}{\partial \mathrm{x}} \tag{3.4}$$

Brown tarafından da önerildiği gibi; D fonksiyonunun Hessian matrisi ve türevi, komşu örnek noktaların farkları kullanılarak 3x3'lük bu lineer sistem en düşük işlem sayısı ile gerçekleştirilir. Eğer uç noktanın sapma uzaklığı herhangi bir yönde 0.5 ten büyük ise, bu, uç noktanın başka bir kilit nokta adayına doğru yöneldiğini gösterir. Bu durumda kilit nokta adayı değiştirilir ve interpolasyon bu nokta dahil edilmeden yeniden gerçekleştirilir. Aksi takdirde uç noktanın konumu hesaplanırken kilit nokta adayının sapması da interpolasyon hesabına katılmış olur.

Düşük kontrasta sahip noktaların ayıklanması

$$\mathrm{D}(\hat{\mathrm{x}}) = \mathrm{D} + \frac{1}{2}\frac{\partial \mathrm{D}}{\partial \mathrm{x}}^{\mathrm{T}} \hat{\mathrm{x}} \tag{3.5}$$

Düşük kontrasta sahip noktaların yok edilmesi için, ikinci dereceden Taylor Serisi açılımı olan Eş. 3.5, uç noktalarda hesaplanır ve eğer bu nokta eşik

değerden (0.03) daha düşükse ($|D(\hat{x})| < 0.03$) elenir. (Resimdeki piksellerin değerlerinin [0,1] arasında olduğu kabul edilmiştir.) Eşik değerden daha büyükse; bu nokta korunur ve ölçeği σ olan bu noktanın konumuna sapma değeri eklenerek, yeni konumu bulunur. Bu eşik değer Lowe tarafından ortaya konulmuştur.

Kenarlarda tespit edilen noktaların ayıklanması

Resimde bulunan nesnelerin kenarlarında fazla miktarda kilit nokta adayı bulmasına karşın, kenarlarda tespit edilen kilit noktalar düşük miktardaki gürültüden bile çok fazla etkilenmektedirler ve bu da kararlı olmayan kilit noktalarımızın bulunmasına yol açmaktadır. Bu durumda kararlılığı arttırmak için kenarlarda zayıf olarak bulunan noktaları ayıklamamız, ancak güçlü olanları ise elememiz gerekmektedir.

Kenarlarda zayıf olarak konumlanan noktalar DoG fonksiyonunda eğrisel prensibe sahiptirler. Heissan matrisinin öz değerleri D fonksiyonunun eğrisel prensibi ile doğru orantılıdır. Öncelikli olarak, ikinci dereceden Heissan matrisini, H, çözümleyerek, öz değerlerini bulmalıyız.

Öz değerlerin birbirlerine olan oranı $r=\alpha/\beta$ (α: büyük olan öz değer, β: küçük olan öz değer) SIFT için gerekli olan kriteri vermektedir.

Heissan matrisinin izdüşümü Tr(H)= Dxx+Dyy bize öz değerlerin toplamına,
Heissan matrisinin determinantı Det(H) = Dxx.Dyy-Dxy.Dxy ise öz değerlerin çarpımına götürür.

$R=Tr(H)2/Det(H) = (\alpha+\beta)2/(\alpha.\beta) = (r+1)2/r$ 'dir.

Lowe burada Harris ve Stephans tarafından 1988 yılında ortaya konan R'nin öz değerlerin değerlerinden çok, öz değerlerin birbirlerine olan oranlarına bağlı olduğu yaklaşımını kullanmıştır [72]. R'nin en küçük değerini aldığı

durum, öz değerler birbirine eşit olduğu durumdur. Başka bir deyişle, öz değerler arasındaki fark ne kadar büyükse R değeri de o kadar büyüktür.

R değerinin (rth+1)2/rth (Lowe 2004'te rth= 10 olarak beyan etmiştir) büyük olması bu noktanın zayıf olarak konumlandırılmış olduğunu gösterir ve bu nokta elenir.

(a) (b) (c)

Şekil 3.4. Kilit noktalarının algoritmalar sonrasındaki görünümleri.
(a) Ölçeksel uzayda uç noktaların tanımlanmasından sonraki görünüm
(b) Düşük kontrasta sahip noktalar elendikten sonraki görünüm
(c) Kenarlarda bulunan zayıf noktalar elendikten sonraki görünüm

3.1.3. Döngüsel değişime karşı dayanıklılık kazanılması

Algoritmanın bu aşaması nesnenin iki boyutlu eksen etrafında dönmesine karşı dayanıklılık kazanılması için önemlidir. Bu aşamada her kilit nokta için bir veya birden fazla döngüsel atama yapılmaktadır.

Daha önce σ ölçeğindeki Gauss filtresinden geçirilmiş resim olan
$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$ kullanılarak gradyent büyüklüğü m(x,y) ve açı θ(x,y) hesaplanır:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (3.6)$$

$$\theta(x,y) = \tan^{-1} \frac{(L(x+1,y) - L(x-1,y))}{(L(x,y+1) - L(x,y-1))} \quad (3.7)$$

Büyüklük ve yön hesaplaması, Gauss filtresinden geçirilmiş resimde bulunan kilit noktaların komşu her pikseli için yapılır. Kilit noktaların etrafında her biri 10 derecelik bir yönü kapsayan 36'lık bir döngüsel histogram oluşturulur. Histograma eklenen komşu penceredeki her örnek nokta, kendi gradyent büyüklüğü ile ağırlıklandırılır.

Histogramda en büyük değere sahip nokta ve bu en büyük değerin %80'i içinde kalan noktalar için kilit nokta tanımlaması oluşturulur. Böylece bir kilit nokta için aynı yerde ve aynı ölçekte birden fazla döngüsel atama yapılmış olur. Noktaların sadece yaklaşık %15'i için çoklu döngüsel atama yapılmış olmasına rağmen bu işlem eşleşmelerdeki kararlılığı önemli ölçüde arttırmaktadır.

3.1.4. Kilit noktalara ait tanımlayıcıların tespit edilmesi

Daha önceki aşamalarda nesnelerin ölçeksel, döngüsel ve konumsal değişimlere karşı dayanıklılık kazanıldı. Bu parametreler 2-boyutlu düzlemde tekrar edilebilir özellikler olduklarından bu parametrelerin değişmesine karşı kilit noktalar kararlılıklarını koruyabilmektedirler. Bu aşama ile ise diğer değişimlere (parlaklık değişimleri, 3-boyutlu görünüm,vb.) karşı dayanıklılık kazanılır.

Belli olan yöntem, uygun ölçekte kilit noktalar etrafındaki enerji yoğunluklarını örneklemek ve normalize edilmiş korelasyonu kullanarak bunlar arasında eşleştirme yapmaktır. Ancak resim parçalarının basit bir korelasyonu üç boyutlu değişimlere ve bozulmalara karşı çok hassastır.

Daha iyi bir yöntem ise Edelman, Intrator, ve Poggio (1997) tarafından ortaya konulmuştur [73]. Retinadaki karmaşık nöronların işleyişini model olarak benimsemişlerdir. Bu karmaşık nöronlar, döngüsel değişimlere gradyent cevabı vermekte, ancak gradyentin konumunun hassas bir şekilde konumlandırılmasından ziyade görme alanında küçük sapmalara izin

verebilmektedirler. Edelman, Intrator, ve Poggio; bu karmaşık nöronların işleyişinin 3-boyutlu nesnelerin eşleştirilmesinde ve tanınmasında kullanılabileceğini iddia etmişler ve bilgisayarda oluşturdukları 3-boyutlu nesne ve hayvan modellerini kullanarak yaptıkları detaylı deneyler sonucunda; gradyentlerin konumlarında değişiklik olmasına izin vererek yapılan gradyent eşleşmelerinin 3-boyutlu değişim altında daha iyi sonuç verdiğini bulmuşlardır. Örneğin derinlik bazında 20 derece dönen 3-boyutlu nesnenin tanınma doğruluğu %35'den %94'e yükselmiştir. Lowe da bu fikirden esinlenmiştir.

İlk olarak, kilit nokta etrafında gradyent büyüklükleri ve açıları kilit noktanın ölçeksel büyüklüğü kullanılarak 4x4'lük alanda örneklenir. Döngüsel değişmezliği sağlamak için kilit nokta tanımlayıcının konumu ve gradyent açıları, kilit nokta açısı baz alınarak döndürülür. Verimliliğin sağlanması için, bu gradyentler, bölüm 3.1.3'te anlatıldığı gibi hesaplanır. Bunlar her örnek nokta için Şekil 3.5'te görüldüğü gibi küçük oklar şeklinde temsil edilirler.

Kilit nokta ölçeğinin 1.5 katı büyüklüğündeki standart sapmaya sahip olan Gauss fonksiyonu, her örnek nokta için katsayı olarak atanır. Bunun amacı, penceredeki küçük değişimlere karşılık kilit noktadaki ani değişimleri önlemektir.

Bu aşamada yapılanlar, döngüsel dayanıklılık kazanım işlemlerindekine çok benzemektedir. Bu tanımlayıcılar 8 yönü temsil eden sekizlik döngüsel histogramlardır.

Sekizlik vektörler, kilit noktanın etrafında bulunan 4x4'lük alanın içinde kalan her bir nokta için tanımlanır. Dolayısıyla her bir kilit nokta için tanımlanan vektörün uzunluğu 4x4x8=128 olur.
Bu aşamada yapılan işlemler aşağıda açıklanmıştır :

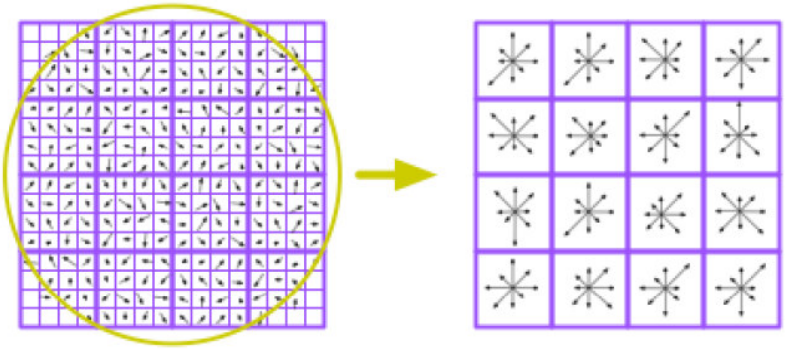
1. Kilit nokta etrafındaki resim gradyent büyüklüğü ve açısı örneklenir.
2. Kilit nokta açısına göre koordinatları ve gradyent büyüklükleri döndürülür.

3. Gradyent büyüklüğü, standart sapması 1.5* σ olan Gauss ile ağırlıklandırılır.
4. Bu örnekler 4x4'lük alanda biriktirilir.

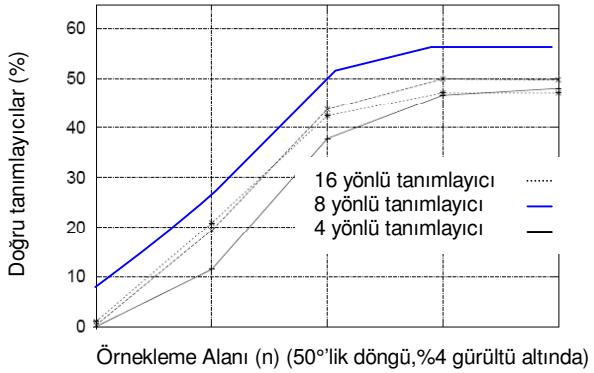
Şimdi ise parlaklık değişimlerinin etkisinin nasıl en düşük seviyeye düşürüldüğü incelenecektir:

İlk olarak elde edilen özellik vektörü birim uzunluğa göre normalize edilir. Resmin kontrastındaki değişim, her piksel değerinin belli bir sabitle çarpılması sonucunu doğurmaktadır. Bu gradyentlerin kaybı katsayı değişimine yol açacağından bu kontrast değişimi vektör normalizasyonu ile iptal edilmiş olur. Ancak lineer olmayan parlaklık değişimleri 3-boyutlu yüzeyleri değişik oranlarda etkilemektedir. Bu yüzden büyük gradyent büyüklüklerinin etkisini azaltmak için daha önce normalize edilmiş olan birim özellik vektörünün 0.2'lik eşik değerinden daha büyük olması şartı uygulanır. Bu eşik değerin üstündeki özellik vektörleri birim uzunluğa yeniden normalize edilir. Bu; eşleşme yapılırken, büyük büyüklüğe sahip gradyentlerden ziyade gradyentlerin açılarının dağılımının daha büyük bir öneme sahip olmasını sağlar. Buradaki 0.2'lik eşik değeri Lowe tarafından deneysel olarak tespit edilmiştir.

Kilit nokta tanımlayıcılarını oluştururken, seçilen iki parametreye göre kilit nokta tanımlayıcının karmaşıklık oranı değişir. Bu parametreler; örnekleme alanı, nxn, ve bir kutucukta kaç adet, r, açısal değişimin alınacağıdır. Bu parametrelere göre sistemin doğruluğu Lowe tarafından Şekil 3.6'da ortaya konulmuştur. Buna göre 4x4'lük 8 açısal değişime sahip vektörün en düşük karmaşıklıkta en iyi sonucu verdiği gözlenmektedir.

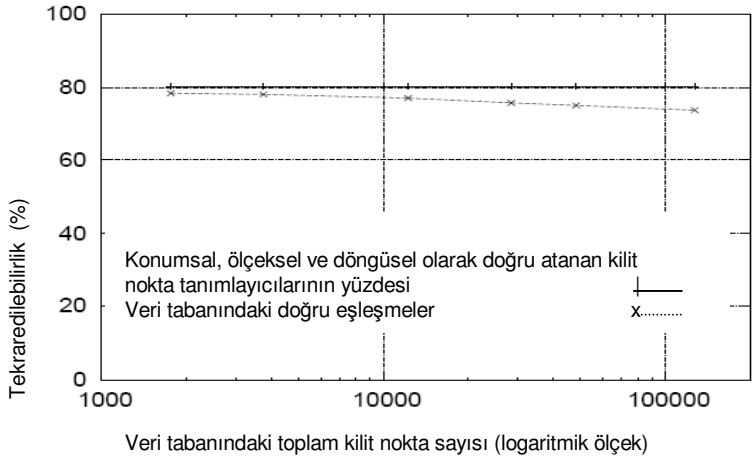


Şekil 3.5. 4x4'lük kilit nokta tanımlayıcıları



Şekil 3.6. Kilit nokta tanımlayıcı vektör büyüklüğünün (n) 50 derecelik açı ve %4'lük gürültü altındaki sonuçları

Lowe tarafından Şekil 3.7'de rasgele seçilen ölçeksel ve döngüsel değişime, 30 derecelik derinlik değişimine ve %2'lik gürültü oranına sahip 112 resim kullanılarak SIFT noktalarının kararlılığı gösterilmiştir. Burada kalın çizgi kilit noktaların resimler arasında doğru pozisyon , ölçek ve açıda tekrar edilebilirliği; kesikli çizgi ise daha önceden tanımlanmış kilit noktalar ile resimlerde bulunan kilit noktalar arasındaki eşleşmeleri temsil etmektedir. Görüldüğü üzere veritabanındaki kilit nokta sayısı çok fazla artmasına rağmen tekrar edilebilirlikte çok ciddi bir düşüş yaşanmamaktadır.



Şekil 3.7. SIFT noktalarının kararlılığı

3.2. SIFT Metodu Parametreleri

- Her kademede bulunacak toplam seviye sayısı s
- Başlangıç standart sapma değeri σ
- Düşük kontrasta sahip noktaların ayıklanması için gerekli eşik değer
- Kenarlarda tespit edilen noktaların ayıklanması için gerekli eşik değer
- Kilit nokta tanımlayıcıları için gerekli örnekleme alanının büyüklüğü
- Kilit nokta tanımlayıcıları için gerekli her örnek noktanın kaç yönde temsil edileceği

Bu parametrelerin Lowe tarafından yapılan seçimlerine ve etkilerine, SIFT metodu anlatılırken değinilmiştir. Tezde seçilen parametre değerlerinden ise Bölüm 4'te bahsedilmiştir.

4. SIFT METODUNUN HEDEF TAKİBİNDE UYGULANMASI

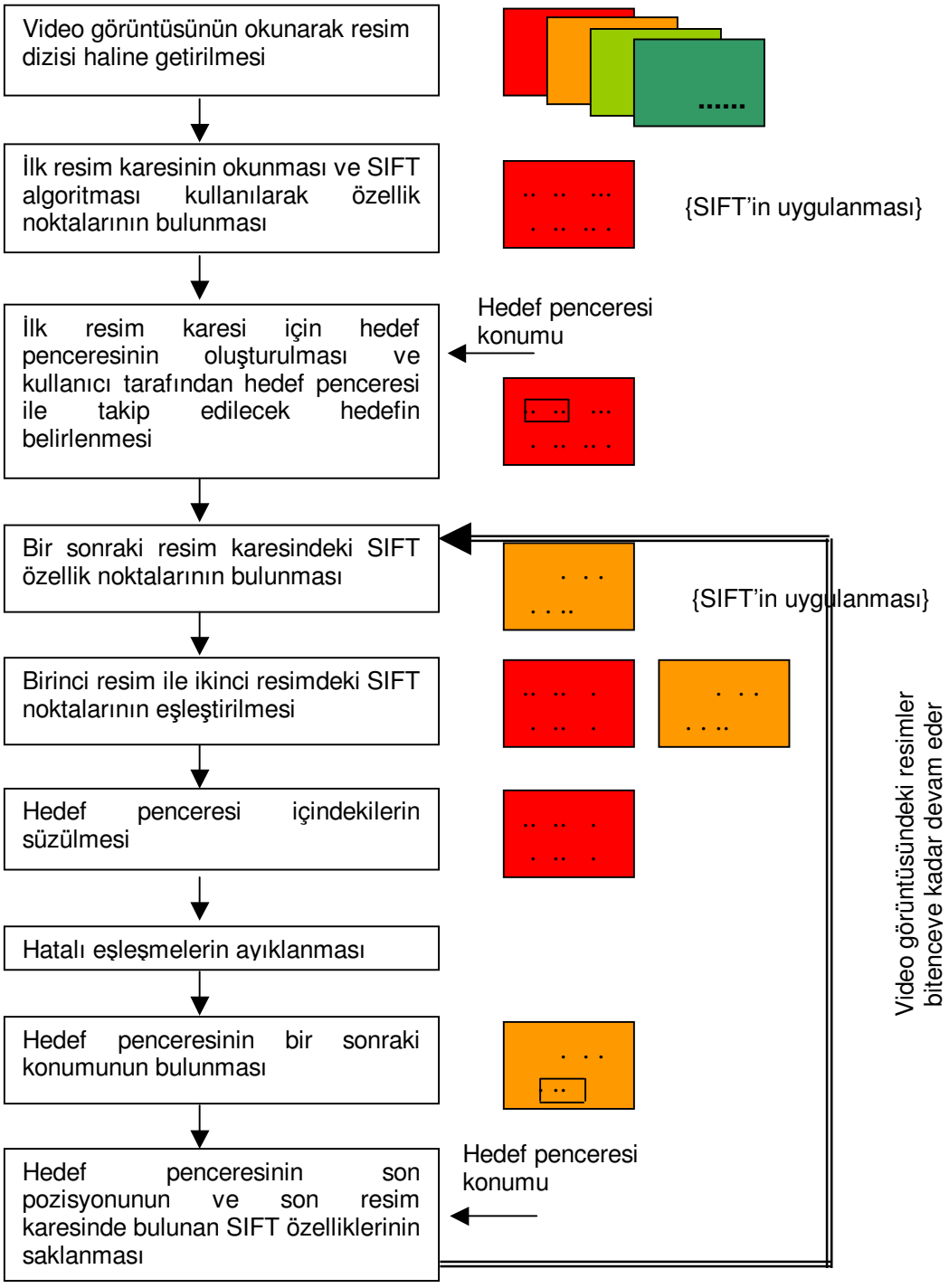
Bu çalışmada seçilen takip metodunun temeli noktasal bazda özellik vektörlerinin çıkarılmasına dayanmaktadır. Literatürde en yaygın olarak kullanılan metotlar Harris, KLT ve SIFT'dir.

Yukarıdaki bu metotların arasından boyutsal, döngüsel ve parlaklık değişimlerinden en az etkilenen David Lowe'un, 1999 yılında bulduğu ve 2004 yılında değişikliklerde bulunduğu SIFT metodu seçilmiştir. SIFT metodunun diğer noktasal bazda özellik vektörü bulan metotlar ile karşılaştırılması 3'üncü bölümde anlatılmıştır.

Çekilen video görüntüsü, istenilen örnekleme oranında örneklenerek ardışık resim dizileri oluşturulmaktadır. Oluşturulan her bir resim karesi için SIFT özellik noktaları hesaplanmaktadır. Ardışık iki resim karesindeki bu özellik noktaları karşılaştırılarak birbiriyle eşleşen noktalar bulunmaktadır. İlk resimdeki hedef penceresi içinde kalan özellik noktalarının ikinci resimdeki karşılıkları bulunarak, hedef penceresinin bir sonraki konumu atanmaktadır. Bu çalışmada hatalı eşleşmeleri ayıklayan ek bir algoritma da oluşturulmuştur.

SIFT metodu, noktasal bazda özellik vektörü çıkaran tüm algoritmalar gibi, noktasal bazda özellik vektörü çıkarmada yüksek doğruluk oranına sahip olmasına karşın, çok fazla hesaplama gerektirmektedir. Ancak, artan mikroişlemci hızları ile bu açık kapanmıştır. Burada, resmin içeriği ve boyutu yani tespit edilecek ve işleme tabi tutulacak noktaların sayısı da büyük önem kazanmaktadır. Ne kadar çok SIFT kilit noktası bulunursa, sistemin doğruluğu o kadar artmakla birlikte işlem süresi de aynı oranda uzamaktadır. SIFT kilit nokta sayısı, oluşturulan sistemin işlem süresini ve doğruluğunu etkilemektedir.

Hedef takibinde kullanılan programın akış şeması aşağıda verilmiştir.



Şekil 4.1. Hedef takibinde kullanılan programın akış şeması

Kullanıcı tarafından belirlenen hedef takibini, bir önceki resim karesinde seçtiğimiz hedefin yerinin şu anki resim karesinde bulunması olarak da tanımlayabiliriz. Burada, işlem gören materyaller resimlerdir. Video, ardışık resim dizilerinden oluşmaktadır. Akış şemasının (Şekil 4.1) ilk basamağını oluşturan aşamada, video görüntüsü, istenilen örnekleme frekansında ardışık resim kareleri haline getirilir.

İkinci basamakta kullanıcı, görüntü alınmasına başla komutunu verdikten sonra resim dizisindeki ilk resim okunur ve bu resme ait özellikli noktalar (SIFT kilit noktaları) SIFT algoritması kullanılarak bulunur.

Üçüncü basamakta, resim ve resim üzerinde rasgele bir konuma atanan hedef penceresi, kullanıcı ara yüzünde belirir. Bu ilk resim karesinin üzerine, daha önceden tanımlanmış olan bir konuma hedef penceresi yerleştirilir ve kullanıcıya hedef seçimi yapması için fırsat sunulur.

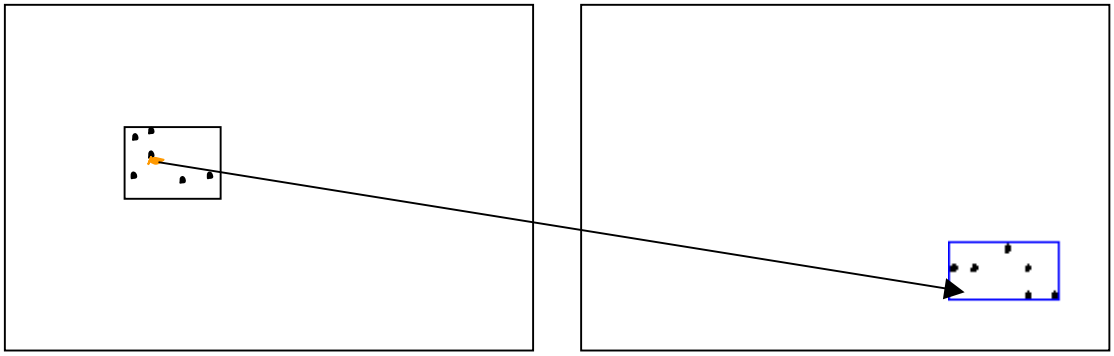
Bundan sonraki aşamalar, döngüsel olarak resim dizisindeki son resme kadar sürekli tekrar edecektir. Bu döngünün aşamaları aşağıda açıklanmıştır:

- Daha önce bulunan bir önceki resim karesine ait SIFT kilit noktaları birinci resim SIFT noktaları olarak atanır.
- Bir sonraki resim karesindeki özellikli noktalar (SIFT kilit noktaları) SIFT algoritması kullanılarak bulunur.
- Birinci resim karesinde bulunan SIFT kilit noktalarının ikinci resim karesindeki karşılıkları bulunur.
- Hedef penceresi içinde olan SIFT kilit noktaları tespit edilir.
- Olası hatalı eşleşmeleri en düşük seviyeye indirecek algoritma kullanılır. Bu kısım bölüm 4.1'de açıklanacaktır.

Birinci resimdeki hedef penceresi içinde kalan ve hatalı eşleşmelerden arınmış SIFT klit noktalarının birinci resimdeki ve ikinci resimdeki karşılıklarının ağırlık merkezleri hesaplanarak, hedef penceresinin ikinci

resimdeki konumu aradaki fark kadar kaydırılarak bulunur (Bkz. Şekil 2.21). Hedef penceresinin yeni konumu kullanıcı ara yüzünde belirir. Böylece hedefin yeni yeri belirlenmiş olur. Bu aşamanın sonunda kullanıcıya takip etmek istediği hedefi değiştirme olanağı sunulur.

Hedef penceresinin son konumu ve ikinci resimdeki SIFT kilit noktaları bir sonraki döngüde kullanılmak üzere saklanır.



(a) (b)

Şekil 4.2. Hedef penceresinin ikinci resimdeki konumunun hesaplanması
(a) Birinci Resim (b) İkinci Resim

4.1. Hatalı Eşleşmelerin En Düşük Seviyeye Getirilmesi ile Takip Algoritmasına Dayanıklılık Kazandırılması:

Bu çalışmada, ardışık resim karelerinde ortamdaki olumsuz koşullar, birbirine çok aşırı benzeyen noktaların varlığı ve çekilen görüntülerden kaynaklanabilecek sebeplerden dolayı hatalı eşleşmelerin de oluştuğu gözlenmiştir. Takip edilen nesnenin bir sonraki konumu hesaplanırken, bu istenmeyen noktalar da hesaba katıldığında takip edilen nesnenin yeni konumunda sapmalar oluşmaktadır. Hedef penceresinin yeni konumu hesaplanırken kullanılan özellikli noktaların (SIFT kilit noktalarının) sayısı hatalı olarak tespit edilen nokta sayısına göre ihmal edilebilecek düzeyde ise bu sapma miktarının düşük olduğu, ancak bu sayının ihmal edilebilir düzeyde olmadığı durumlarda takip edilmesi istenilen nesneden ciddi oranda sapmaların olduğu gözlenmiştir. Dolayısıyla oluşabilecek hatalı eşleşmelerin

nesne takibi üzerindeki bu olumsuz etkisini ortadan kaldırmak için hatalı eşleşmeleri ayıklama işlemi uygulanmıştır.

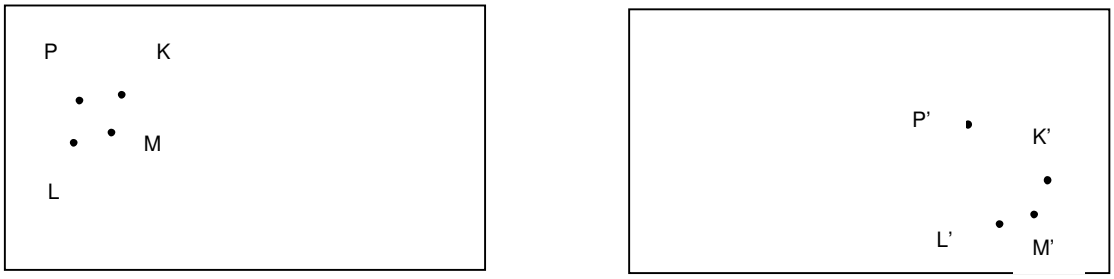
Birinci resim karesindeki hedef penceresi içinde kalan her bir SIFT kilit noktasının hatalı bir eşleşme yapıp yapmadığı aşağıdaki şekilde kontrol edilir:

- P,K,L,M noktaları : Birinci resimde hedef penceresi içinde kalan SIFT kilit noktaları,
- P',K',L',M' noktaları : Birinci resimde hedef penceresi içinde kalan SIFT kilit noktalarının ikinci resimdeki karşılıkları olmak üzere;

1. Birinci resim karesinde hedef penceresi içinde kalan SIFT noktası (P) seçilir.
2. P noktasının K,L,M noktalarına olan uzaklıkları hesaplanır.
3. P noktasının ikinci resimdeki karşılığı olan P' noktasının; K',L',M' kilit noktalarına uzaklıkları da hesaplanır.
4. |P'K'| - |PK| > Eşik Değer
   |P'L'| - |PL| > Eşik Değer
   |P'M'| - |PM| > Eşik Değer
   } ise P-P' eşleşmesi HATALI EŞLEŞME'dir.

Yukarıdaki hatalı eşleşme kontrolü sadece P-P' eşleşmesi için yapılmıştır. Söz konusu kontrol birinci resimde hedef penceresi içinde kalan tüm SIFT kilit noktaları için yapılmalıdır.

Hedef penceresi içinde kalan ve hatalı eşleşmelerden arınan SIFT noktalarının birinci resimdeki ağırlık merkezinin konumu hesaplanır. Aynı işlem, bu noktaların ikinci resimdeki karşılıkları için de yapılır. Hedef penceresi, iki resimde hesaplanan ağırlık merkezleri arasındaki fark kadar kaydırılır ve hedef penceresinin ikinci resimdeki yeri bulunmuş olur.



(a) (b)

Şekil 4.3. Hatalı eşleşmelerin ayıklanması
(a) Birinci Resim (b) İkinci Resim

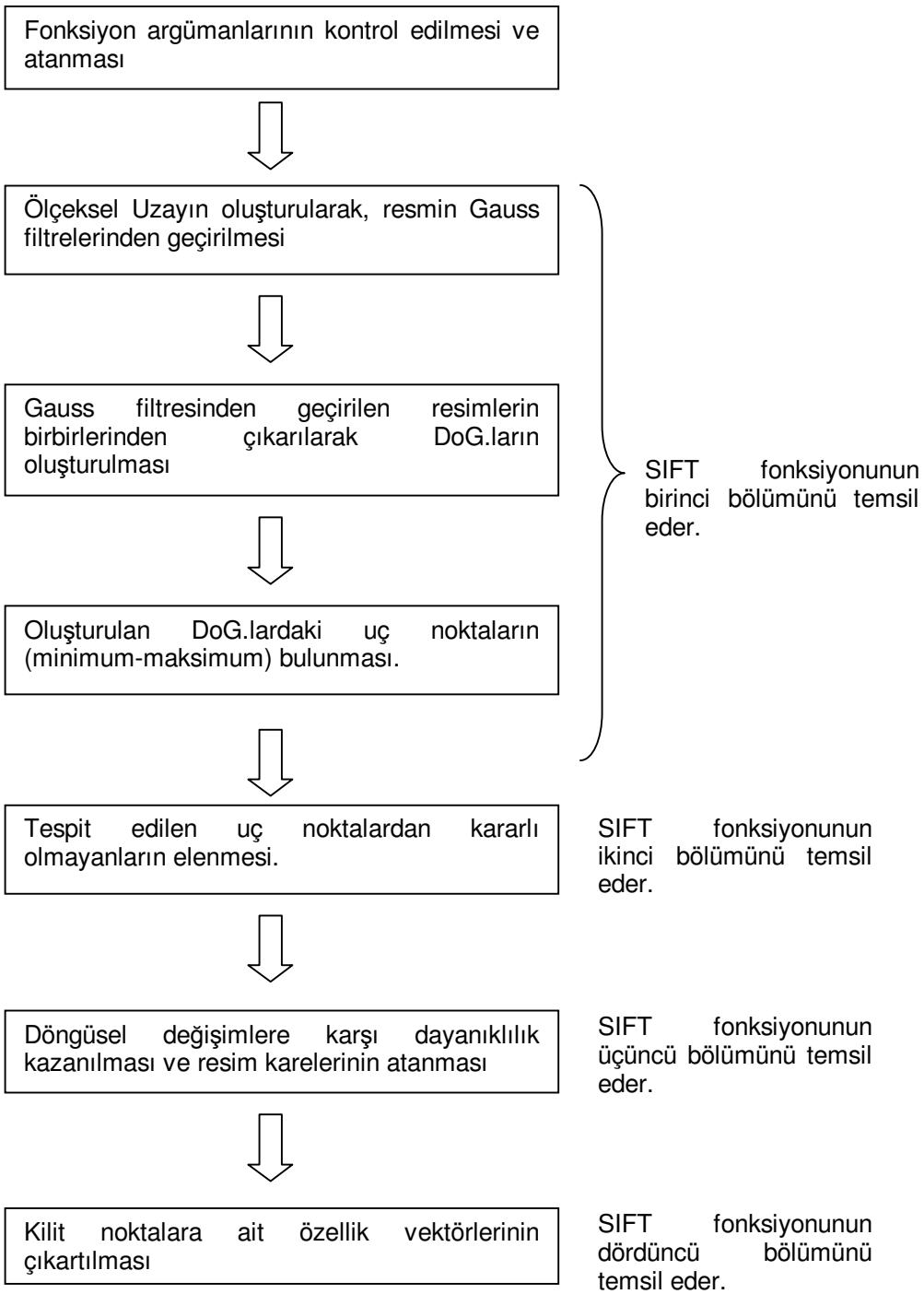
4.2. Örnekleme Frekansının Seçilmesi

Bir video görüntüsü ardışık resim karelerinin insan gözünün resimlerdeki kesiklikleri ayırt edemeyecek frekanslarda ardışık olarak sıralanması olarak nitelendirilebilir. Buradaki örnekleme frekansından kasıt da video görüntüsünden elde edilecek resim karelerinin hangi sıklıkla örnekleneceğidir.

Bu seçim, sistemin uygulanacak alandaki ihtiyacına ve resim karelerinin değişim hızına göre belirlenir. Bu tezde bir çok farklı örnekleme frekansı kullanılmıştır.

4.3. SIFT Fonksiyonunun Akış Şeması

SIFT kilit noktalarını hesaplayan yapı temel olarak aşağıdaki şekilde tanımlanabilir:



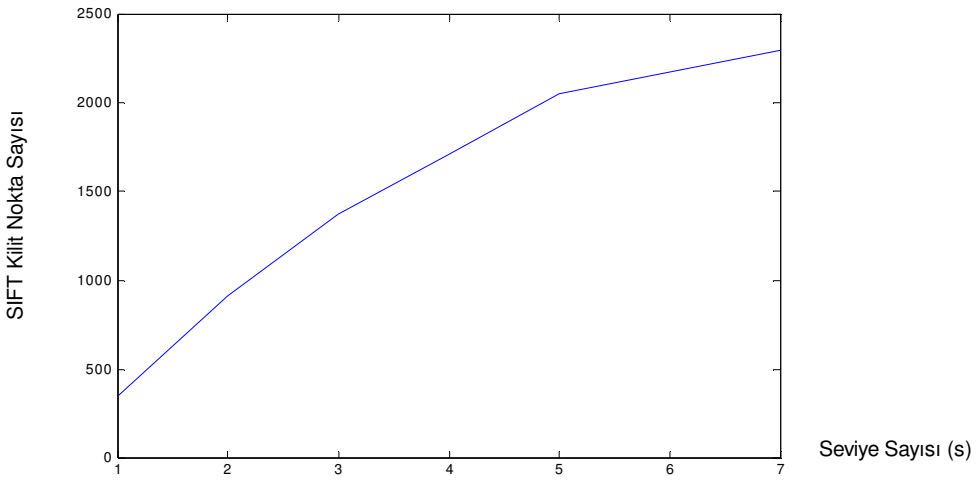
Şekil 4.5. Sift Fonksiyonunun İşleyişi

4.3.1. Seçilen SIFT metodu parametreleri ve etkileri

Bu çalışmada, SIFT parametrelerinin (s, |D(x)|, R, nxn, r) etkileri, 320x240 çözünürlüğe sahip resimler üzerinde 1.6 G Hz.lik işlemci kullanılarak incelenmiştir.

Ölçeksel uzaya ait parametreler

- Ölçeksel uzaydaki her kademede bulunacak toplam seviye sayısı s ile ifade edilmiştir. Resim karesinde tespit edilen kilit nokta sayısının s değeri ile değişimi incelenmiş ve Şekil 4.6'da sunulmuştur. Buna göre s değeri arttıkça, tespit edilen SIFT kilit nokta sayısı da doğru orantılı olarak artmaktadır. Ancak, her kademede bulunacak seviye sayısındaki bu artışın işlem süresini de arttırdığı tespit edilmiştir. İşlem süresi ile kademe sayısı arasındaki ilişki ise Çizelge 4.1'de verilmiştir. Ayrıca s değerinin 3'ten sonra alacağı değerlerde doğruluk oranının artmadığı, aksine düşüşlerin yaşandığı da Resim 4.1'de görülmektedir.



Şekil 4.6. S değerinin SIFT kilit nokta sayısına etkisi

Yukarıda belirtilen nedenlerden dolayı, programda s değeri 3 olarak alınmıştır. S değerinin 3'ün altındaki değerlerinde; daha az SIFT kilit noktası

bulunduğundan dolayı hedef penceresinin, takip edilen nesneden sapmalar meydana getirdiği, ancak işlemsel yükün düştüğü gözlenmiştir. Bu değerin artması ile ise hedef takibindeki doğruluğun önemli ölçüde değişmediği hatta bazen düşürmediği; buna karşılık işlem yükünü arttırdığı tespit edilmiştir.

Çizelge 4.1. Kademe sayısı ile işlem süresi arasındaki ilişki

Kademe Sayısı (s)	SIFT Kilit Noktalarının Tespit Süresi	Hedef Konumunun Tespit Süresi
1	4.06 s	0.25 s
2	9.40 s	3.05 s
3 (Optimum Değer)	11.8 s	6.08 s
4	17.00 s	10.50 s
5	17.80 s	14.70 s

(a)

Resim 4.1. Farklı s değerlerinin hedef takibine etkisi: a) s=3 b) s=2 c) s=1 ç) s=4 d) s=5

(b)

(c)

Resim 4.1. (Devam) Farklı s değerlerinin hedef takibine etkisi: a) s=3 b) s=2 c) s=1 ç) s=4 d) s=5

(ç)

(d)

Resim 4.1. (Devam) Farklı s değerlerinin hedef takibine etkisi: a) s=3 b) s=2 c) s=1 ç) s=4 d) s=5

İstenmeyen kilit noktaların elenmesine ait parametreler

- Bu çalışmada, düşük kontrasta sahip noktaların ayıklanması için gerekli eşik değer 0.02 olarak seçilmiştir. Bu eşik değer arttırıldığında elenen SIFT kilit nokta sayısı artmaktadır. Bu parametrenin ölçeksel uzay oluşturulurken geçen zamanla bağıntısı bulunmamakla birlikte, döngüsel atama yapma ve kilit nokta tanımlayıcılarının tanımlanması aşamaları daha az kilit nokta için yapılacağından dolayı işlem yükü nispeten azalmakta ve işlem hızı artmatadır. Ancak, bu artışın çok ciddi miktarlarda olmadığı gözlenmiştir. Ayrıca tanımlanan kilit nokta sayısı da çok büyük miktarda değişmemektedir. D(x) ile tanımlanan kilit nokta sayısı arasındaki ilişki Çizelge 4.2'de sunulmuş ve optimum değer tabloda verilmiştir.

Çizelge 4.2. D(x) ile tanımlanan kilit nokta sayısı arasındaki ilişki

D(x)	Tespit Edilen Kilit Nokta Sayısı
0.002	1312
0.02	1312 (Optimum Değer)
0.2	1309
0.5	1276
2	956

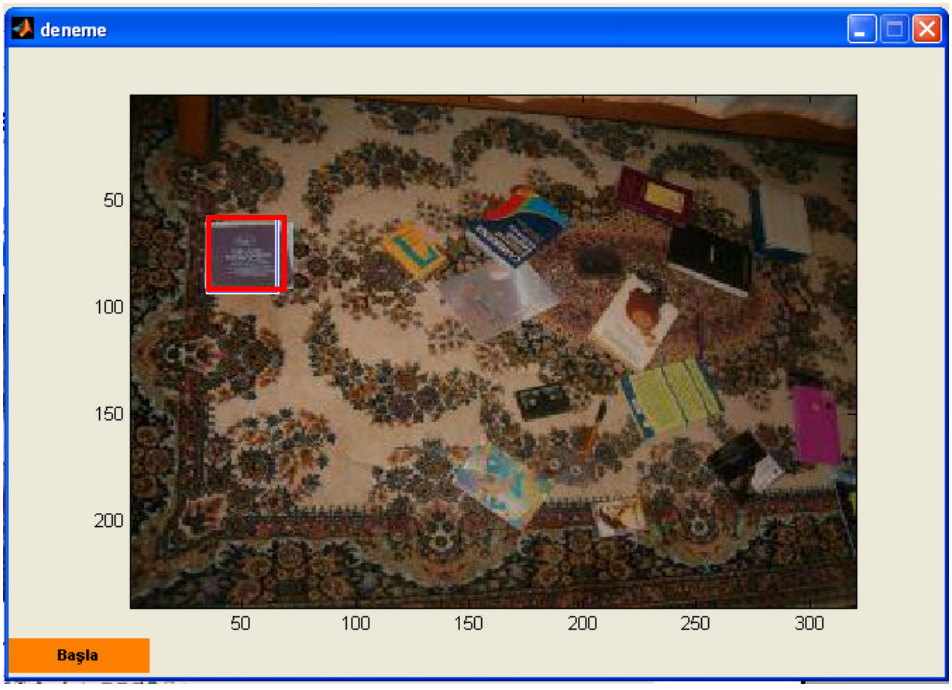
D(x) parametresinin hedef tespitinin doğruluğuna etkisi ise Çizelge 4.3'te görülmektedir.

Çizelge 4.3. D(x) ile hedef takibinin doğruluğu

D(x)	Hedefin Takip Edilemediği Resim Karesi Sayısı	Doğruluk (%)
0.002	1	86
0.02	0	100
0.2	2	71
2	2	71
20	7	0

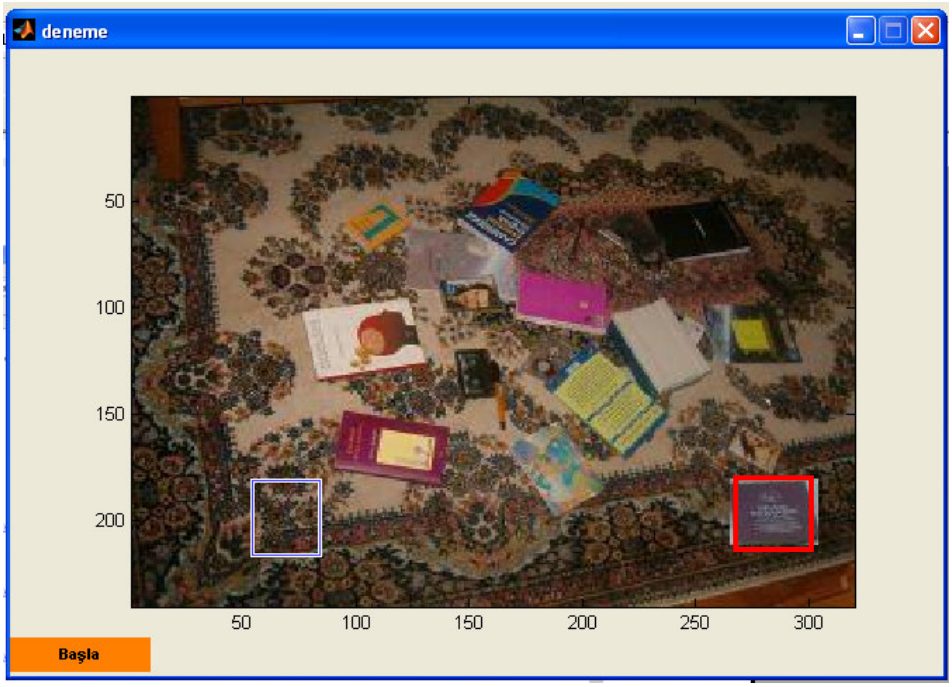
Resim 4.2'de ise eşleşme yapılamayan ara yüz görüntülerine yer verilmiştir.

(a)



(b)

Resim 4.2. D(x) parametresinin a) D(x)=0.002 b) D(x)=0.02 c) D(x)=2 iken hedef takibi üzerindeki etkisi.

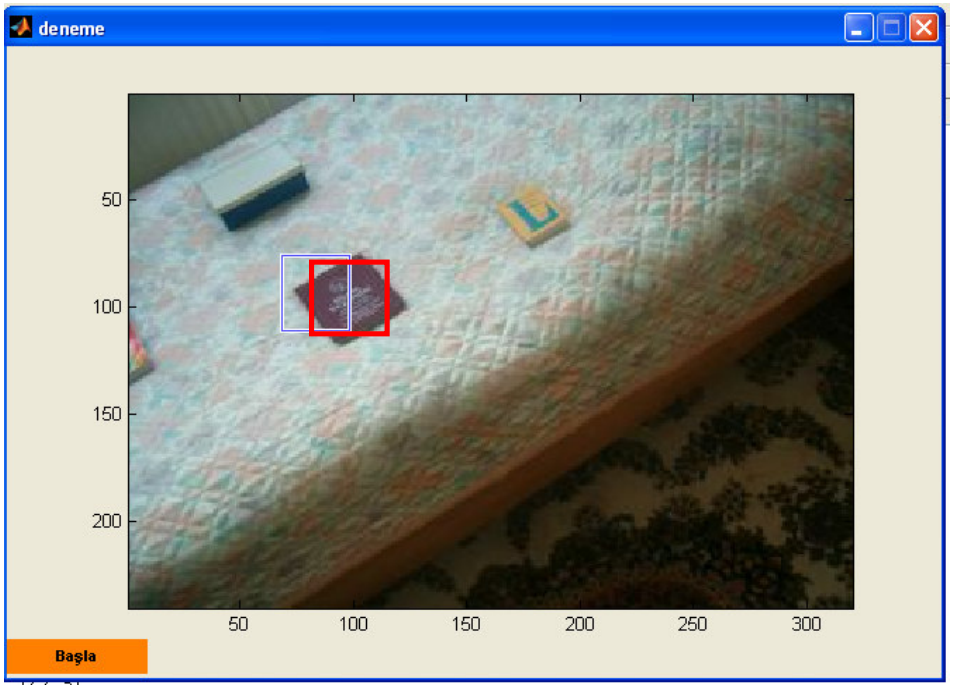


(c)

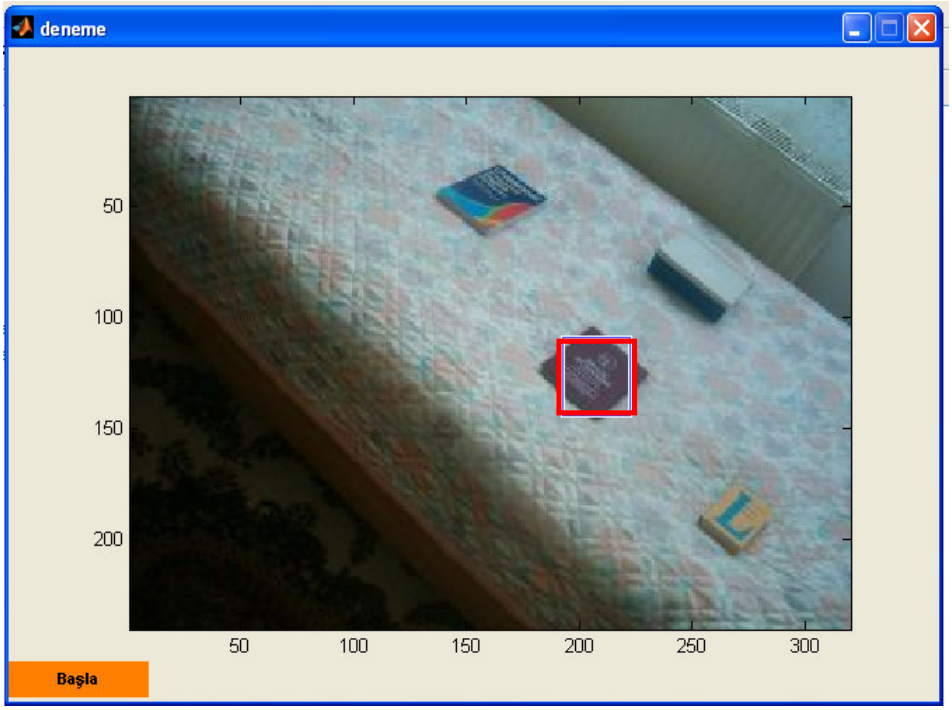
Resim 4.2. (Devam) D(x) parametresinin a) D(x)=0.002 b) D(x)=0.02 c) D(x)=2 iken hedef takibi üzerindeki etkisi.

- Kenarlarda tespit edilen noktaların ayıklanması için gerekli optimum eşik değer, 10 olarak bulunmuştur. Bu değer azaltıldığında veya arttırıldığında istenilen hedeften sapmalar gözlenmiştir. R eşik değeri değiştiğinde meydana gelen sapmalara örnek ise Resim 4.3'te verilmiştir.

R değeri aynı zamanda tespit edilen kilit nokta sayısını da etkilemektedir. Bu etki Çizelge 4.4'te görülmektedir.



(a)



(b)

Resim 4.3. Hedef takibinde a) R=2 ve b) R=10 c) R=17 değerlerinde meydana gelen sapmalar

(c)

Resim 4.3. (Devam) Hedef takibinde a) R=2 ve b) R=10 c) R=17 değerlerinde meydana gelen sapmalar

Çizelge 4.4. R değeri ile tanımlanan kilit nokta sayısı arasındaki ilişki

R	Tespit Edilen Kilit Nokta Sayısı
2	490
10 (Optimum Değer)	1611
17	1674

Kilit nokta tanımlayıcılarına ait parametreler

Burada kesin olarak tespit edilen kilit noktalara ait tanımlamalar yapılmaktadır. Bu kısımda iki parametre bulunmaktadır. Bunlardan biri, kilit noktaların tanımlanacağı alan olan örnekleme alanı (nxn); diğeri ise örnekleme alanında bulunan her noktanın temsil edileceği yön sayısıdır. Ayrıntılı bilgi için bölüm 3.1.5'e bakınız.

- Kilit nokta tanımlayıcıları için gerekli örnekleme alanının büyüklüğü 4x4 olarak atanmıştır. Örnekleme alanını arttırdığımızda veya azalttığımızda (n=4

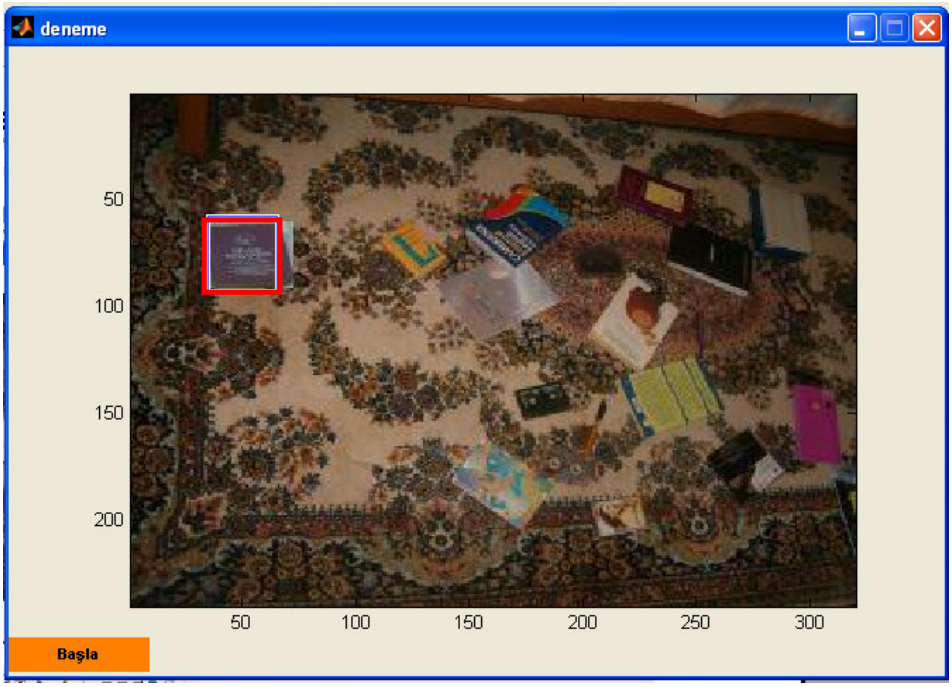
değerine göre) doğruluk oranı Çizelge 4.5'te görüldüğü gibi düşmektedir. Takip edilmesi istenilen hedeften sapmalar veya hedefi bulamama gözlenmektedir. Ayrıca toplam kilit nokta sayısı, nxn değeri ile ters orantılı olarak değişmektedir. İşlem süresi ise, nxn değeri ile doğru orantılı olarak artmaktadır.

Çizelge 4.5. Örnekleme alanı ile hedef takibinin doğruluğu arasındaki ilişki

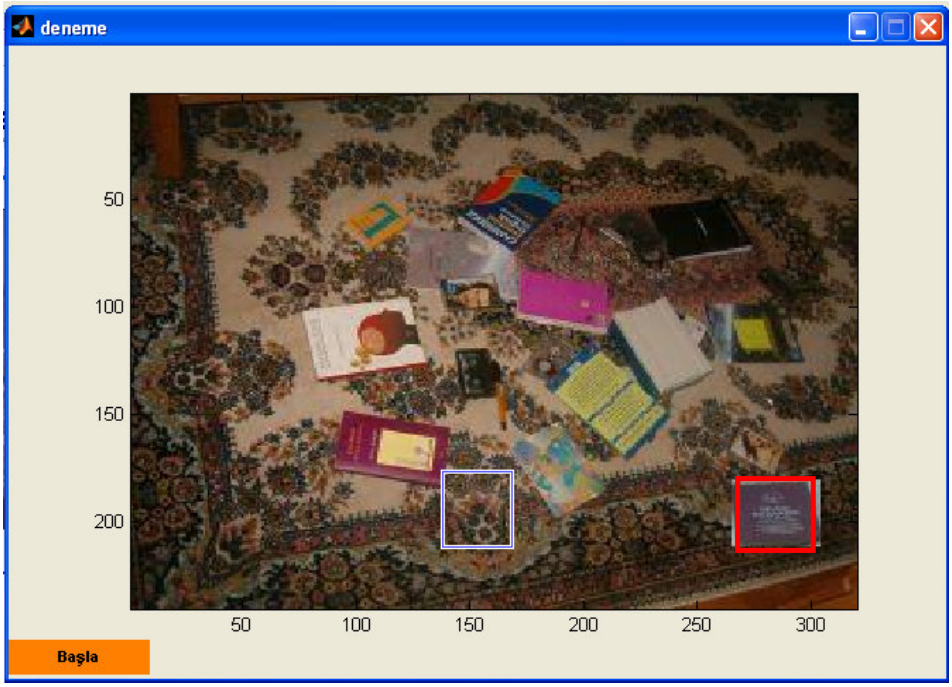
Örnekleme Alanı (nxn)	Doğruluk (%)
2x2	75
4x4 (Optimum Değer)	100
6x6	75
8x8	37.5

(a)

Resim 4.4. Örnekleme alanının a) n=2 b) n=4 c) n=6 için hedef takibine etkisi



(b)



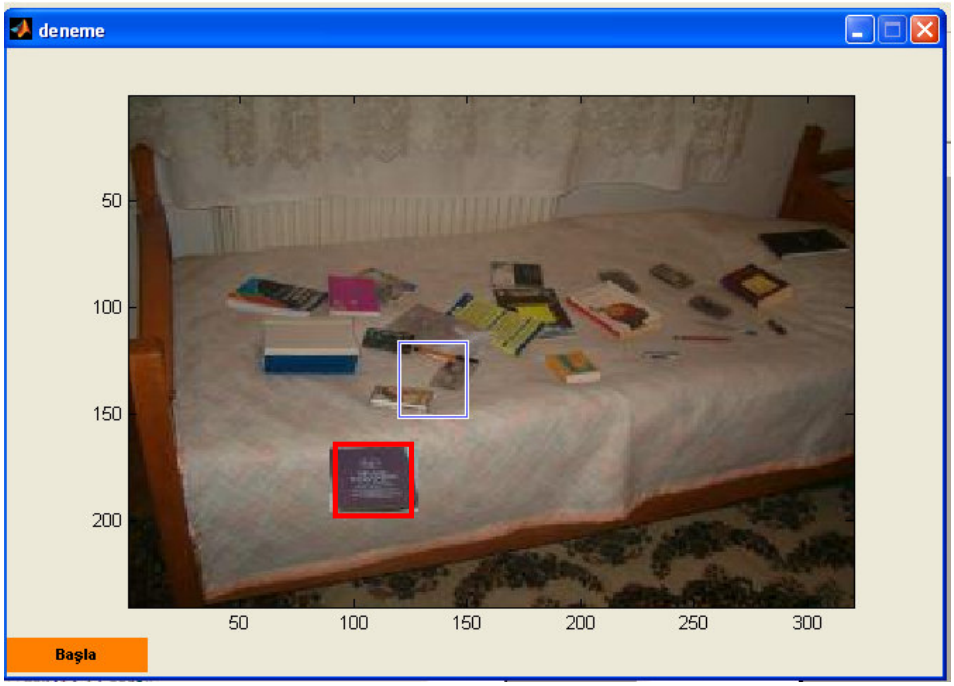
(c)

Resim 4.4. (Devam) Örnekleme alanının a) n=2 b) n=4 c) n=6 için hedef takibine etkisi

- Kilit nokta tanımlayıcıları için gerekli her örnek noktanın temsil edileceği yön sayısı 8 olarak seçilmiştir. Yön sayısı arttığında veya azaldığında ise istenilen hedef takibinden sapmalar meydana gelmektedir. Bu değerinin ne gibi sapmalar meydana getirdiğini gözlemlemek için Resim 4.5'e, r değeri ile hedef takibinin doğruluğu arasındaki ilişkiyi daha iyi anlayabilmek için ise Çizelge 4.6'ya bakınız. Yön sayısının değişmesi tanımlanan büyüklüğü toplam kilit nokta sayısını değiştirmemekle birlikte hedef takibinin doğruluğunu etkilemektedir.

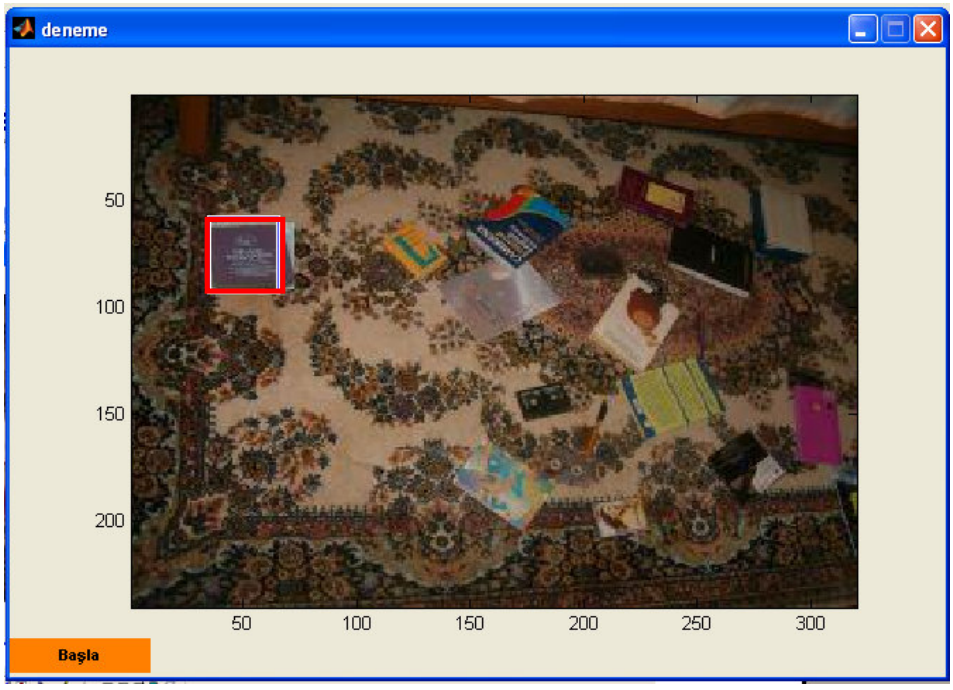
Çizelge 4.6. Yön sayısı ile hedef takibinin doğruluğu arasındaki ilişki

Yön Sayısı (r)	Doğruluk (%)
1	0
2	87.5
5	87.5
8 (Optimum Değer)	100
10	87.5
12	75



(a)

Resim 4.5. a) r=2 b) r=8 c) r=10 için için gözlemlenen sapmalar



(b)

(c)

Resim 4.5. (Devam) a) r=2 b) r=8 c) r=10 için için gözlemlenen sapmalar

4.3.2. İşlemci hızının etkisi

1.6 G Hz.lik işlemci hızı ile 320x240 piksellik resimde 1372 adet SIFT kilit noktası 11.8 s'de bulunurken, 3 G Hz.lik işlemci hızı ile 320x240 piksellik resimde 1312 adet SIFT kilit noktası 6.9 s.de tespit edilmiştir. Ayrıca SIFT kilit noktaları tespit edilen iki resim karesindeki bu noktaların birbiriyle eşleştirilme sonucu hedefi bulma süresi, 1.6 G Hz.lik işlemci kullanıldığında 6.08 s. iken, 3 G Hz.lik işlemci kullanıldığında bu süre 2.65 s.ye inmiştir.

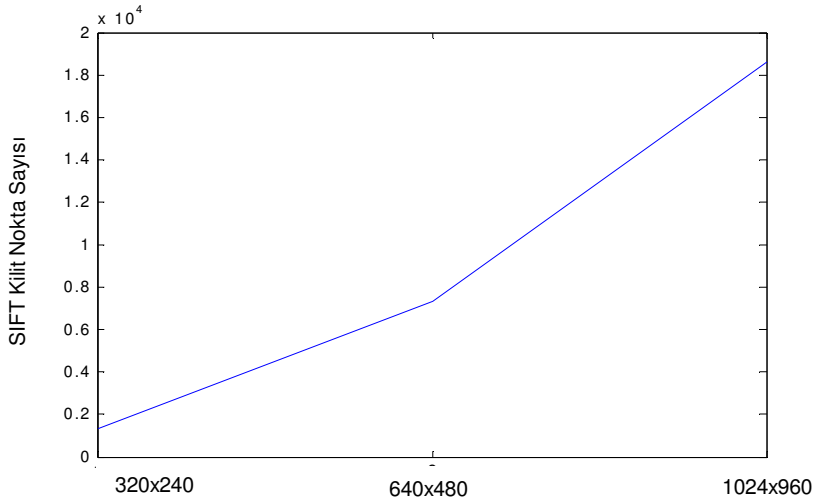
4.3.3. Resim çözünürlüğünün etkisi

Yapılan tesler düşük çözünürlüğe sahip 320x240 piksellik resim dizisi ile gerçekleştirilmiştir. Bu kadar düşük çözünürlükteki resim karelerinde bile hedef takibi başarı ile gerçekleştirilmiştir. Ancak resmin çözünürlüğü arttıkça, istenilen hedefi takip etme hassasiyeti ve doğruluğu da aynı oranda artmaktadır. Optimum değerler (s=3, D(x)=0.02, R=10, nxn=4x4, r=8) ile 640x480 ve 1024x960 piksellik çözünürlüğe sahip resim kareleri, 3 G Hz.lik işlemci kullanılarak test edilmiştir. Yapılan tesler sonucunda çözünürlüğün artması işlem zamanını arttırmasına karşın, istenilen hedef takibindeki doğruluk ve hassasiyet değeri önemli ölçüde artmıştır. Resim çözünürlüğü ile tespit edilen toplam klit nokta sayısı, SIFT noktalarının tespit süresi ve eşleşme süresi arasındaki ilişki Çizelge 4.6.da verilmiştir.

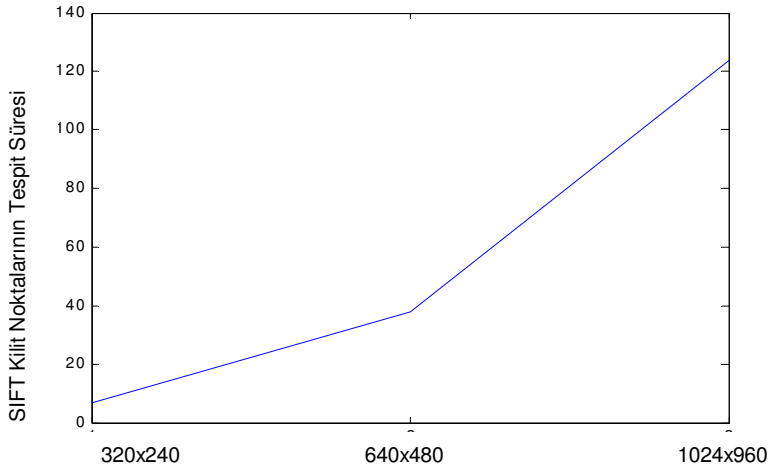
Çizelge 4.6. Resim çözünürlüğünün etkisi

Resim Çözünürlüğü	Toplam Kilit Nokta Sayısı	SIFT Kilit Noktalarının Tespit Süresi	Eşleşme Süresi
320x240	1312	6.90 s	2.65 s
640x480	7353	38.00 s	80.70 s
1024x960	18646	123.70 s	530.00 s

Şekil 4.5, 4.6 ve 4.7 incelendiğinde, resim çözünürlüğü ile kilit nokta sayısının, SIFT kilit noktalarının tespit süresinin ve hedefi bulma süresinin exponansiyal olarak arttığı görülmektedir.

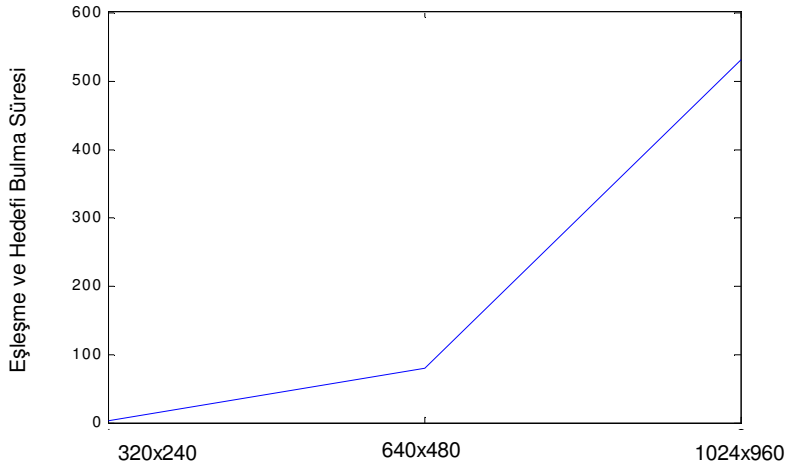


(a)



(b)

Şekil 4.7. Resim çözünürlüğünün etkisi
a) Kilit nokta sayısına etkisi
b) SIFT kilit noktalarının tespit süresi
c) Eşleşme ve hedefi bulma süresi



(c)

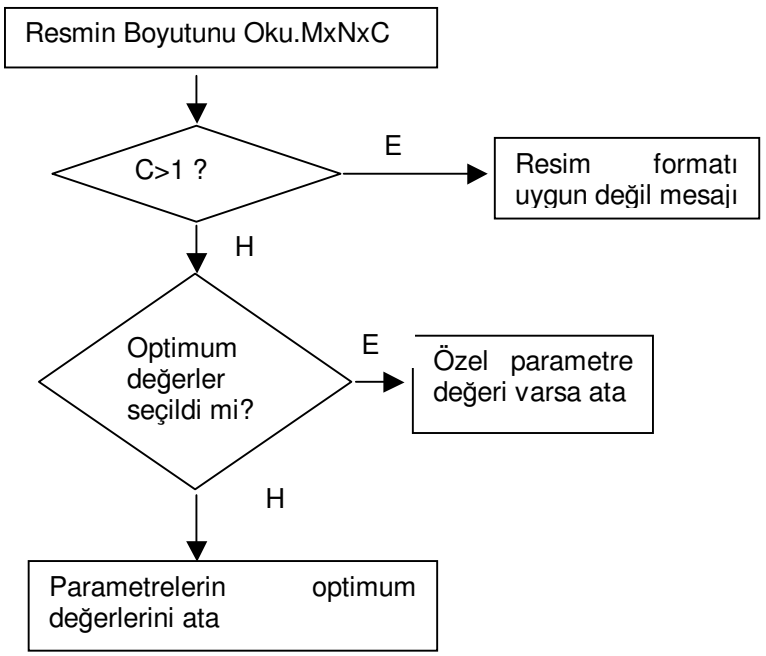
Şekil 4.7. (Devam) Resim çözünürlüğünün etkisi
a) Kilit nokta sayısına etkisi
b) SIFT kilit noktalarının tespit süresi
c) Eşleşme ve hedefi bulma süresi

4.3.4. Sift fonksiyonunun oluşturulması

Şekil 4.5'teki SIFT fonksiyonlarının işleyişindeki aşamalar aşağıda açıklanmıştır.

Fonksiyon argümanlarının kontrol edilmesi ve atanması

Bu kısımda girdi olarak alınan resmin istenilen formda olup, olmadığı değilse uyarı verilmesi sağlanır. Dış ortamdan fonksiyon parametreleri için giriş olup olmadığı kontrol edilerek; eğer istenilen parametre girişi yoksa varsayılan SIFT parametreleri atanır.



Şekil 4.7. Fonksiyon argüman kontrolü ve ataması

MxN : Resim Boyutu
C : Resmin renk bilgisi

Ölçeksel uzayın oluşturularak, uç noktaların belirlenmesi

İlk olarak fonksiyon giriş parametrelerinin uygun olup olmadığı kontrol edilir ve k=2^1/s olarak seçilir.

Ölçeksel uzay hesaplanarak, resmin Gauss filtrelerinden geçirilmesi işlemi ve ardışık olarak çıkarma işlemi bölüm 3.1.1'deki gibi yapılır. Oluşturulan ölçeksel uzaydaki uç noktalar bulunur. Burada her nokta kendisinin, bir alt ve bir üst komşu seviyesindeki 8 komşu nokta (toplam 26 nokta) ile karşılaştırılır ve bu noktanın, noktalar içindeki en düşük veya en büyük içeriğe sahip nokta olması durumunda uç nokta olarak seçilir.

Kilit noktalardan kararlı olmayanların elenmesi

Bölüm 3.1.2'de açıklandığı gibi Heissan matrisi ve D(x) fonksiyonunun Taylor serisi açılımı yapılır. $|D(\hat{x})| < 0.01$ kriteri ve R eşik değeri için de 10 değeri seçilmiştir.

Döngüsel değişime karşı dayanıklılık kazanılması

Bölüm 3.1.3'te açıklandığı üzere, her kilit nokta için kilit noktanın ölçeğine en yakın olan L(x,y) fonksiyonu kullanılarak gradyentler hesaplanır. Her gradyent için etrafındaki alanda bulunan noktalar kullanılarak döngüsel histogram oluşturulur.

Histogramda en büyük değere sahip noktanın % 80'i içinde kalan noktalar için döngüsel atama yapılır.

Kilit nokta tanımlayıcılarının oluşturulması

Bölüm 3.1.4'te belirtildiği gibi, burada kilit noktaların ölçeksel, döngüsel, konumsal bilgilerini içeren tanımlayıcıları oluşturulur. Burada nxnxr'lik büyüklüğe sahip tanımlayıcılar tanımlanır. n değeri 4, r değeri ise 8 olarak seçilmiştir. Dolayısıyla kilit nokta tanımlayıcıları 4x4x8'lik büyüklükte elde edilmiştir.

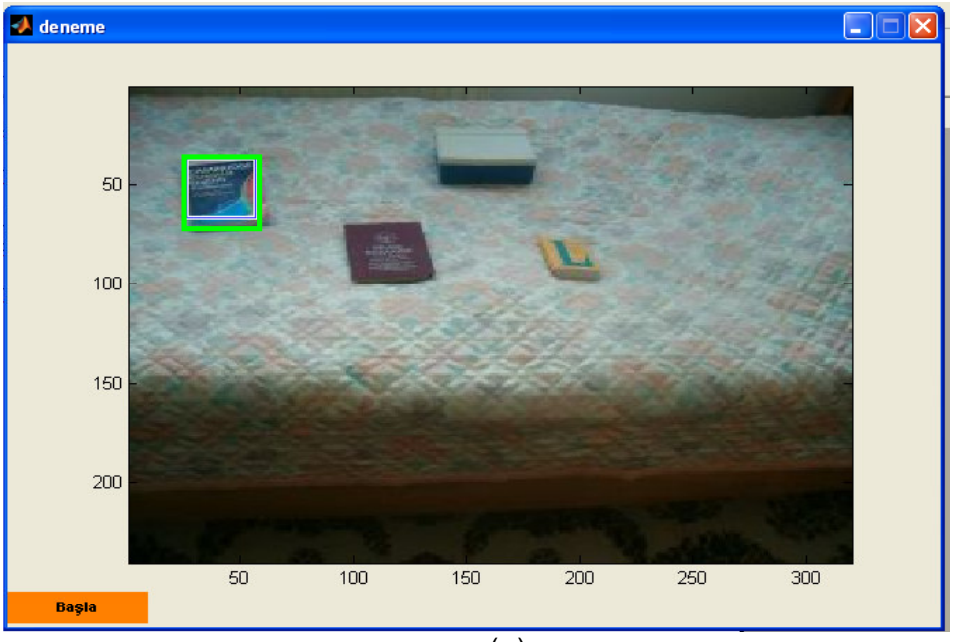
5.TESTLER

Optimum SIFT parametreleri kullanılarak oluşturulan program, video ve resim görüntüleri üzerinde aşağıdaki koşullar altında hedef takibi yapılmıştır.

1. Test : Hedef sabit, kamera hareketli, döngüsel değişim yok
2. Test : Hedef sabit, kamera hareketli, döngüsel değişim kameranın ekseni etrafında
3. Test : Hedef hareketli, kamera hareketli, döngüsel değişim yok
4. Test : Hedef sabit, kamera hareketli, döngüsel değişim nesnenin etrafında
5. Test : Helikopterden çekilen video görüntüsündeki rasgele seçilen hedefin takibi.
6. Test : Değişken arka plan ortamında hareketli hedef takibi.

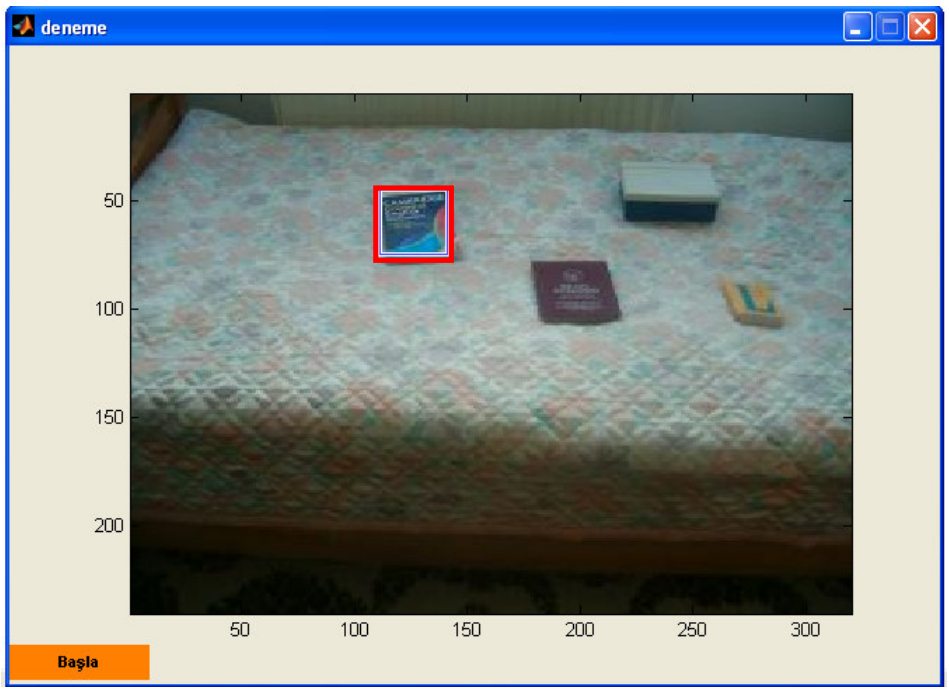
Yeşil pencere takip edilmesi istenilen hedef, kırmızı pencere olması gereken yer, beyaz pencere ise programın takip ettiği yerdir.

1. Test : Hedef sabit, kamera hareketli, döngüsel değişim yok

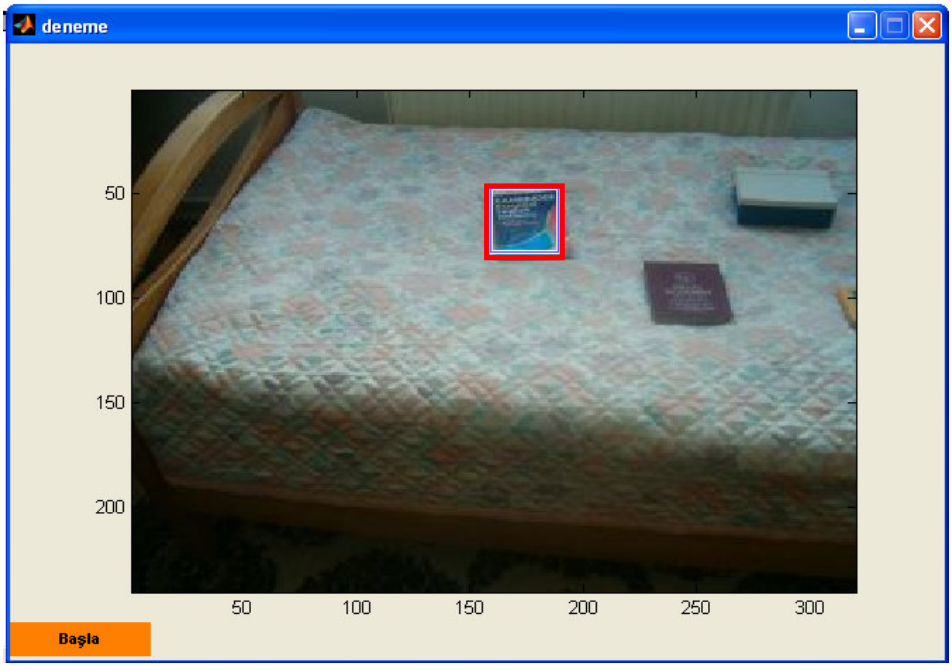


(a)

Resim 5.1. Optimum SIFT parametreleri ile birinci test sonuçları

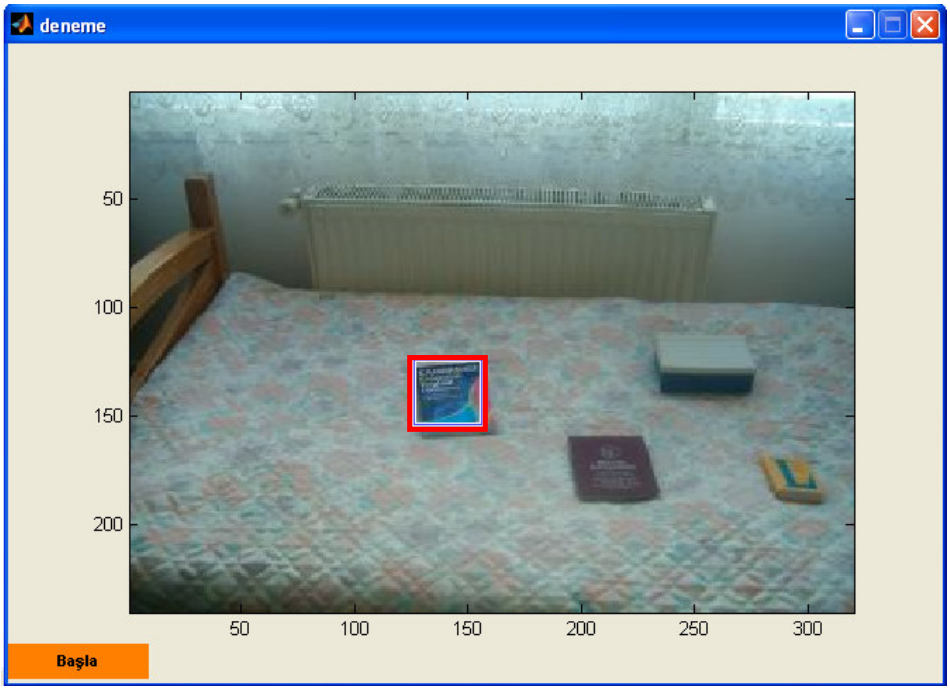


(b)

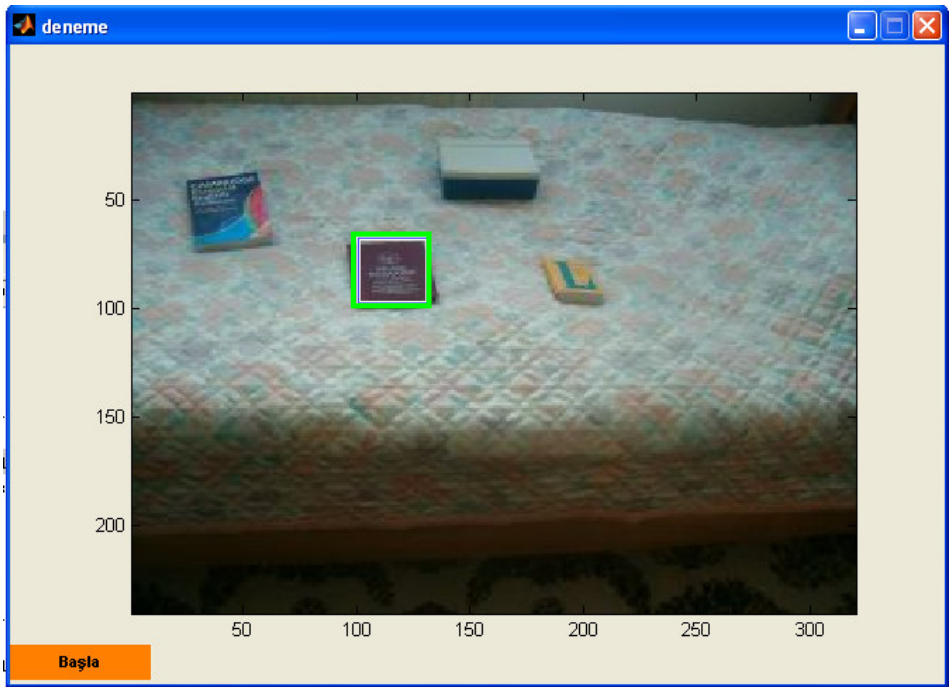


(c)

Resim 5.1. (Devam) Optimum SIFT parametreleri ile birinci test sonuçları



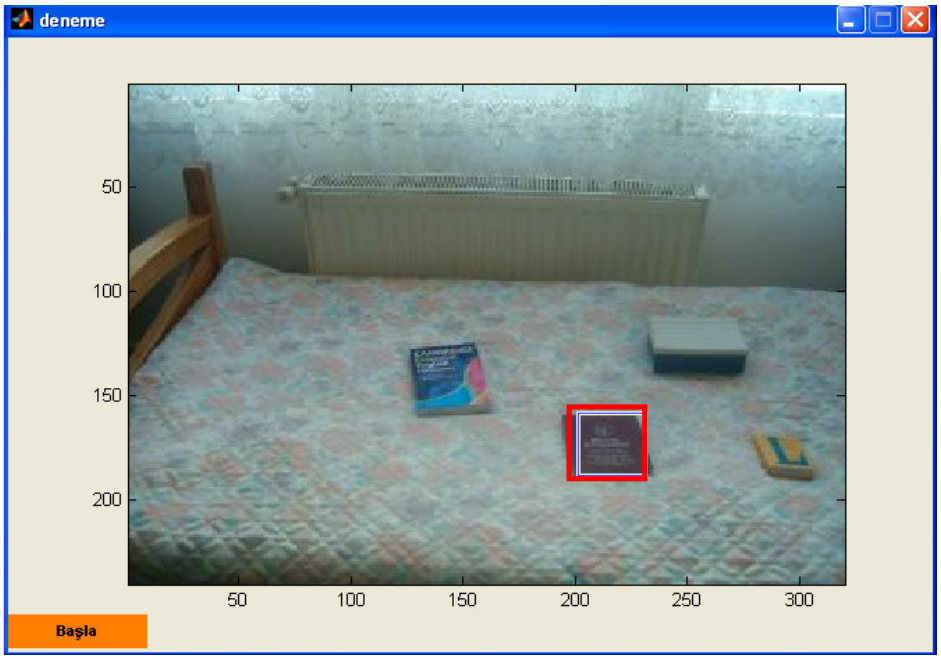
(d)



(e)

Resim 5.1. (Devam) Optimum SIFT parametreleri ile birinci test sonuçları

(f)



(g)

Resim 5.1. (Devam) Optimum SIFT parametreleri ile birinci test sonuçları

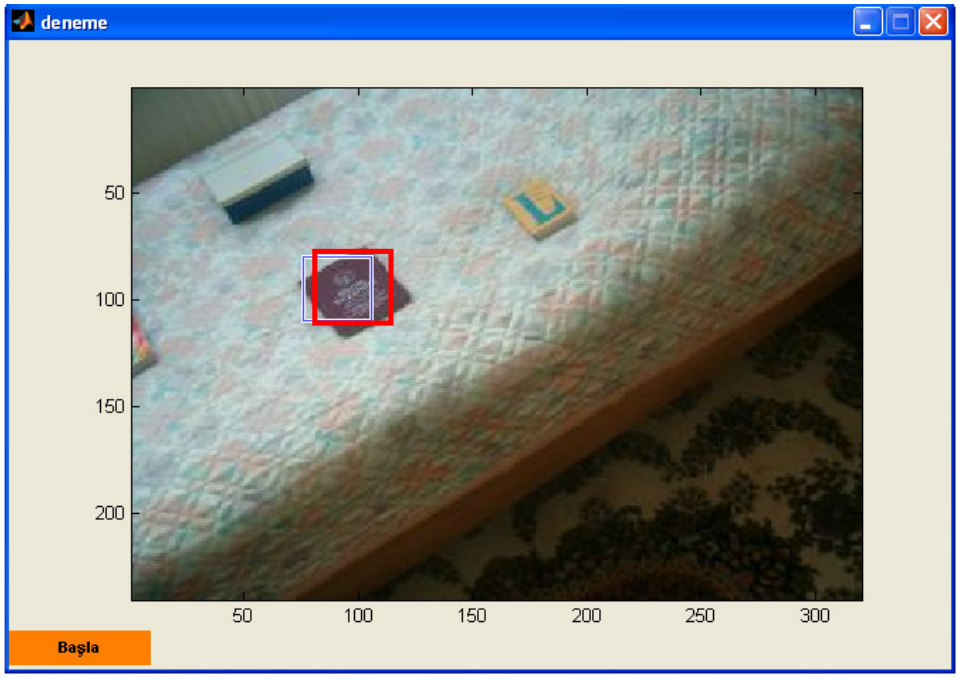
(h)

Resim 5.1. (Devam) Optimum SIFT parametreleri ile birinci test sonuçları

Yukarıda görülen resimlerde hedef takibi hedef sabit kamera hareketli döngüel değişimin olmadığı durumlarda başarı ile gerçekleştirilmiştir. (a)-(c) arası bir hedef takibi, (d)-(h) arasında ise başka bir hedef takibi gerçekleştirilmiştir. Hedef olarak seçilen nesneler yeşil pencere ile işaretlenmiştir. (a)-(c) arasındaki resimlerde seçilen hedef renkli bir kitap, (d)-(h) arasındaki resimlerde seçilen hedef ise üzerinde yazıları olan başka bir kitaptır.

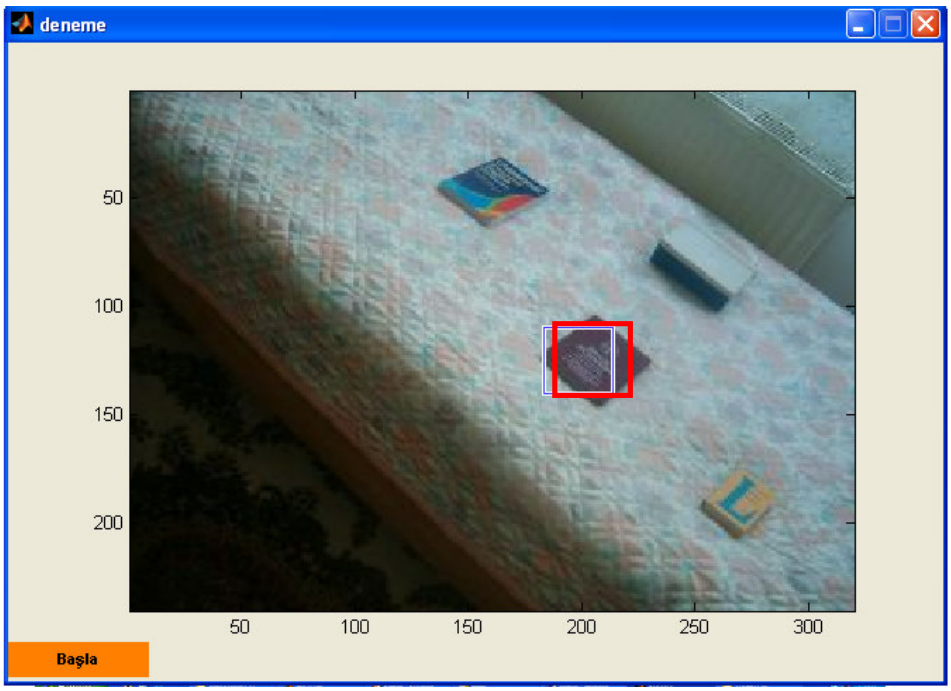
2.Test : Hedef sabit, kamera hareketli, döngüsel değişim kameranın ekseni etrafında

(a)

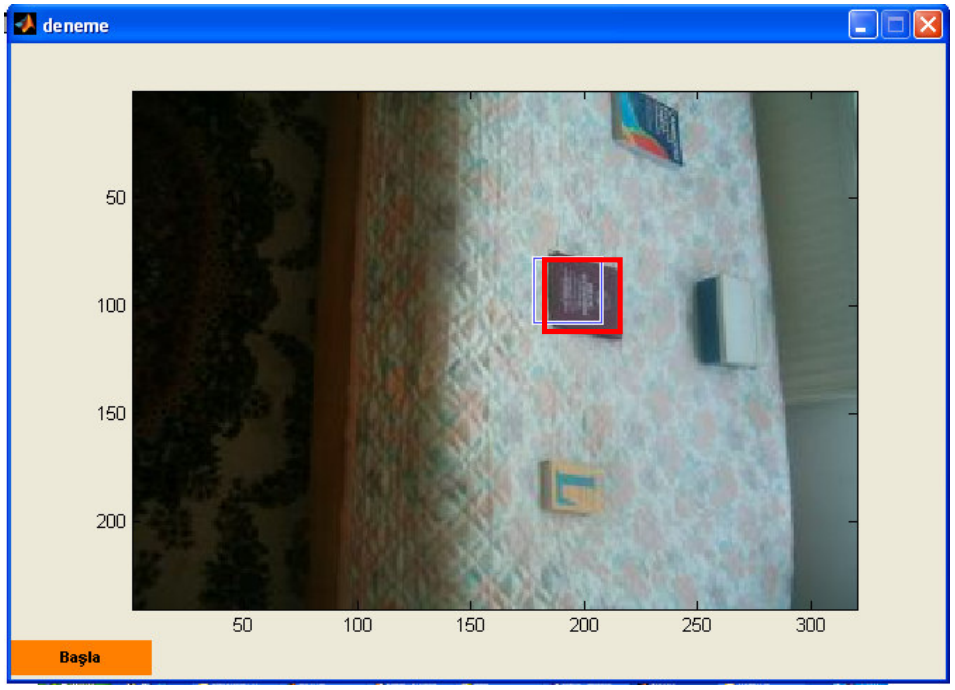


(b)

Resim 5.2. (Devam) Optimum SIFT parametreleri ile ikinci test sonuçları

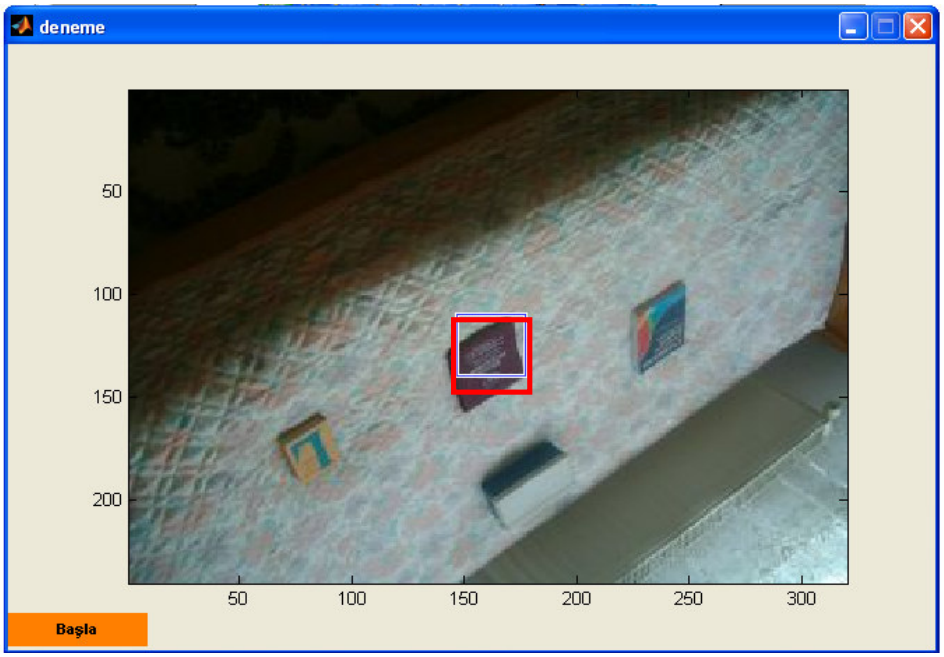


(c)



(d)

Resim 5.2. (Devam) Optimum SIFT parametreleri ile ikinci test sonuçları



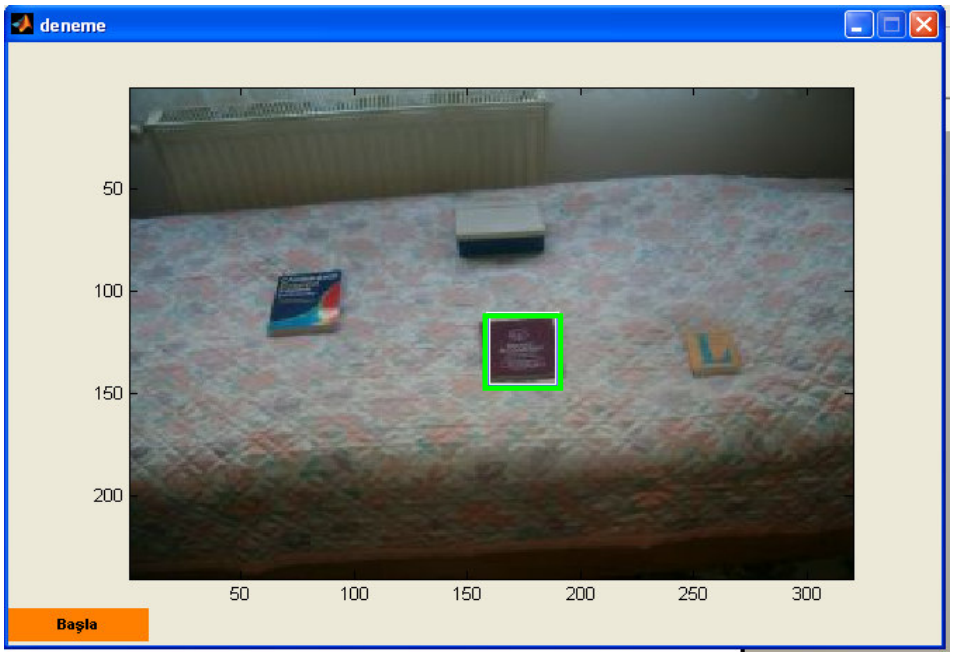
(e)

(f)

Resim 5.2. (Devam) Optimum SIFT parametreleri ile ikinci test sonuçları

Resim 5.2 (a)-(f)'deki gibi, hedef sabit, kamera hareketli, döngüsel değişim kameranın ekseni etrafında iken yapılan hedef takibi başarı ile gerçekleştirilmiştir.

3. Test : Hedef ve kamera hareketli, döngüsel değişimin olmadığı durumda hedef takibi



(a)

(b)

Resim 5.3. Optimum SIFT parametreleri ile üçüncü test sonuçları

(c)

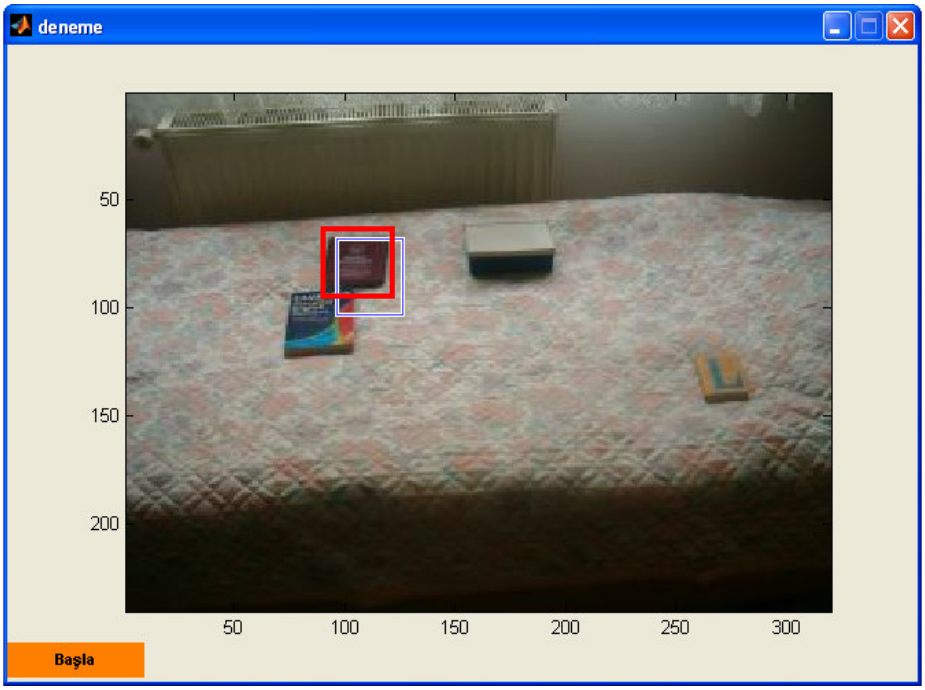
(d)

Resim 5.3. (Devam) Optimum SIFT parametreleri ile üçüncü test sonuçları

(e)

(f)

Resim 5.3. (Devam) Optimum SIFT parametreleri ile üçüncü test sonuçları

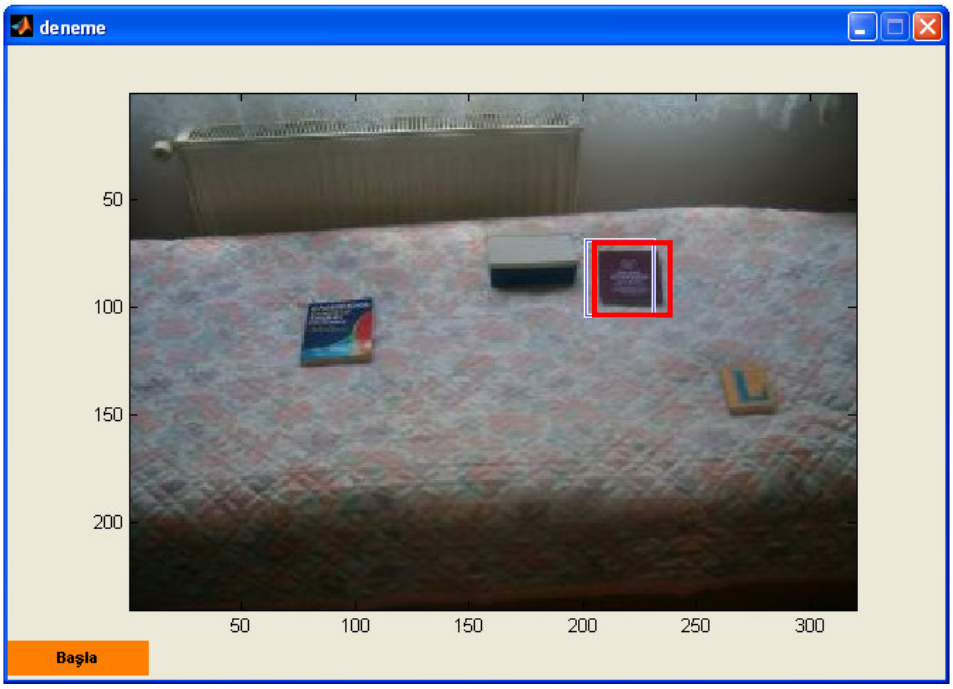


(g)

(h)

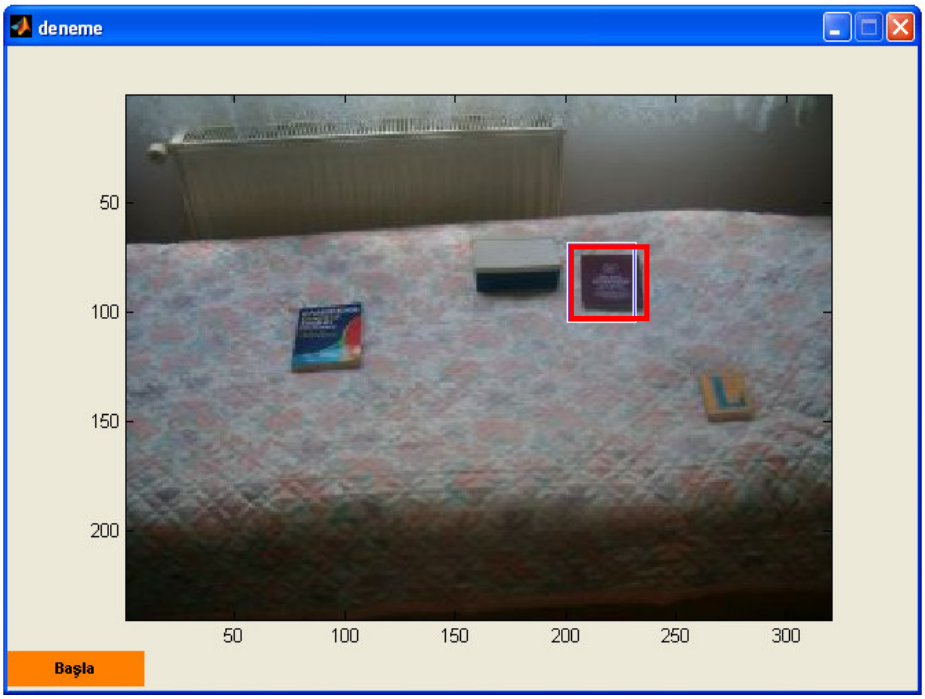
Resim 5.3. (Devam) Optimum SIFT parametreleri ile üçüncü test sonuçları

(i)



(j)

Resim 5.3. (Devam) Optimum SIFT parametreleri ile üçüncü test sonuçları

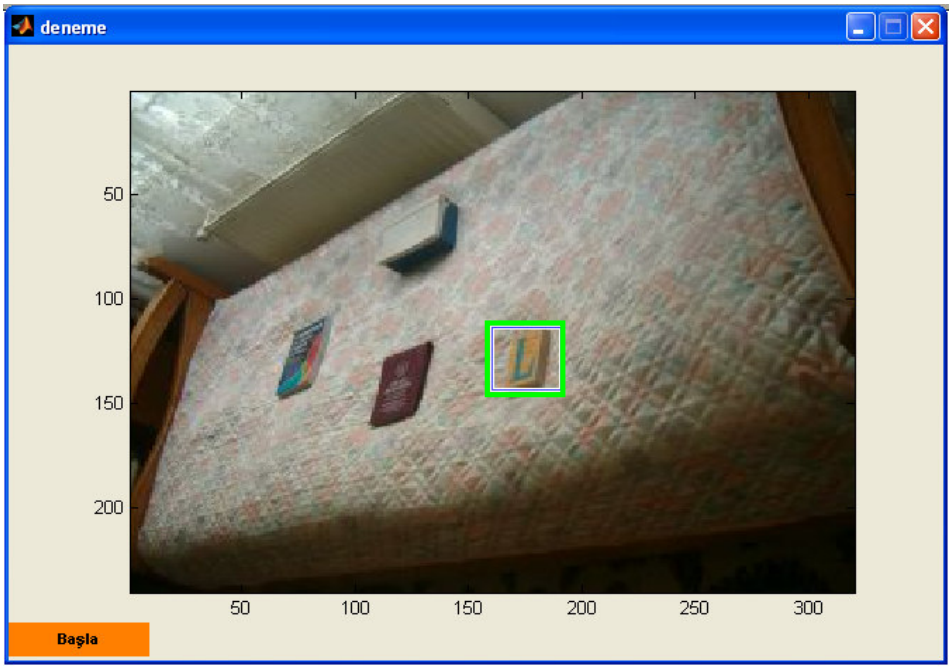


(k)

Resim 5.3. (Devam) Optimum SIFT parametreleri ile üçüncü test sonuçları

(a)-(k) arası resimlerde görüldüğü gibi, hedef ve kamera hareketli ve döngüsel değişimin olmadığı şartlarda hedef takibi başarı ile gerçekleştirilmiştir. Hedef başka bir cisme çok fazla yaklaşmasına rağmen; hedef takibi sırasında seçtiğimiz hedef, oluşturulan program tarafından başarı ile izlenebilmiştir.

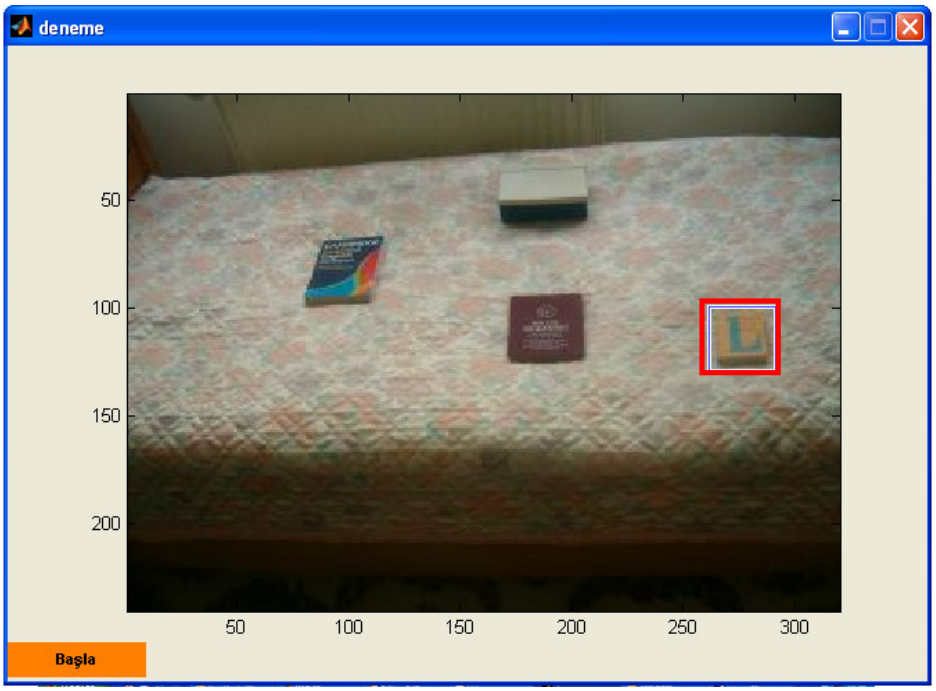
4. Test : Hedef sabit, kamera hareketli, döngüsel değişim nesnenin etrafında iken hedef takibi



(a)

(b)

Resim 5.4. (Devam) Optimum SIFT parametreleri ile dördüncü test sonuçları

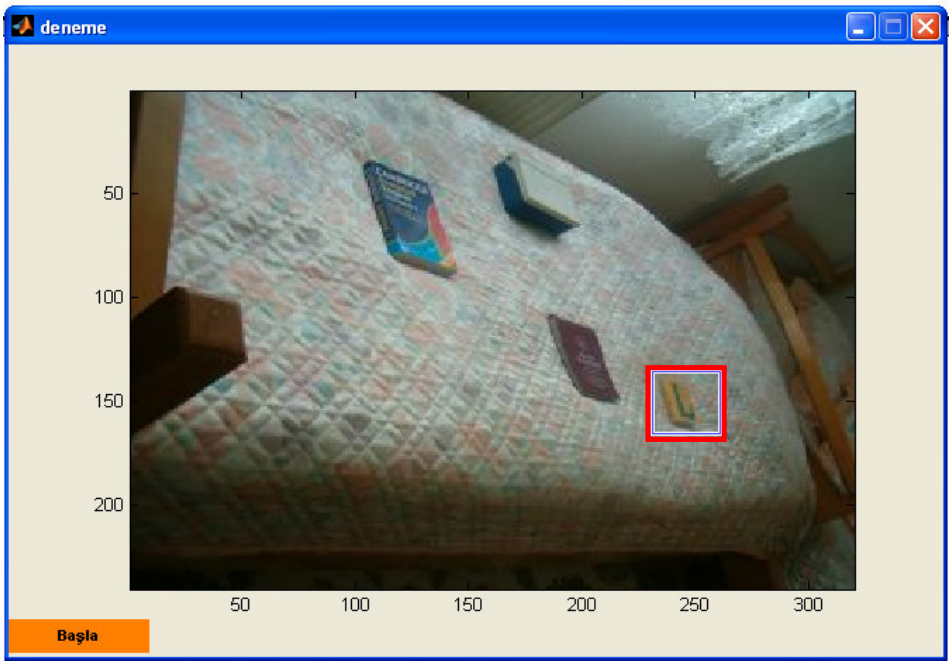


(c)

(d)

Resim 5.4. (Devam) Optimum SIFT parametreleri ile dördüncü test sonuçları

(e)

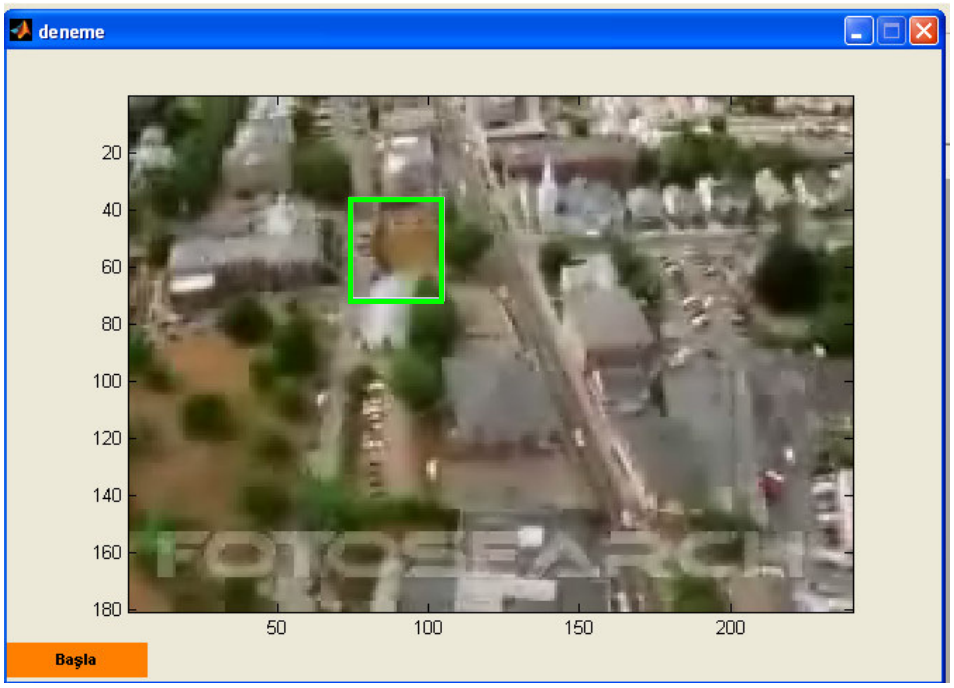


(f)

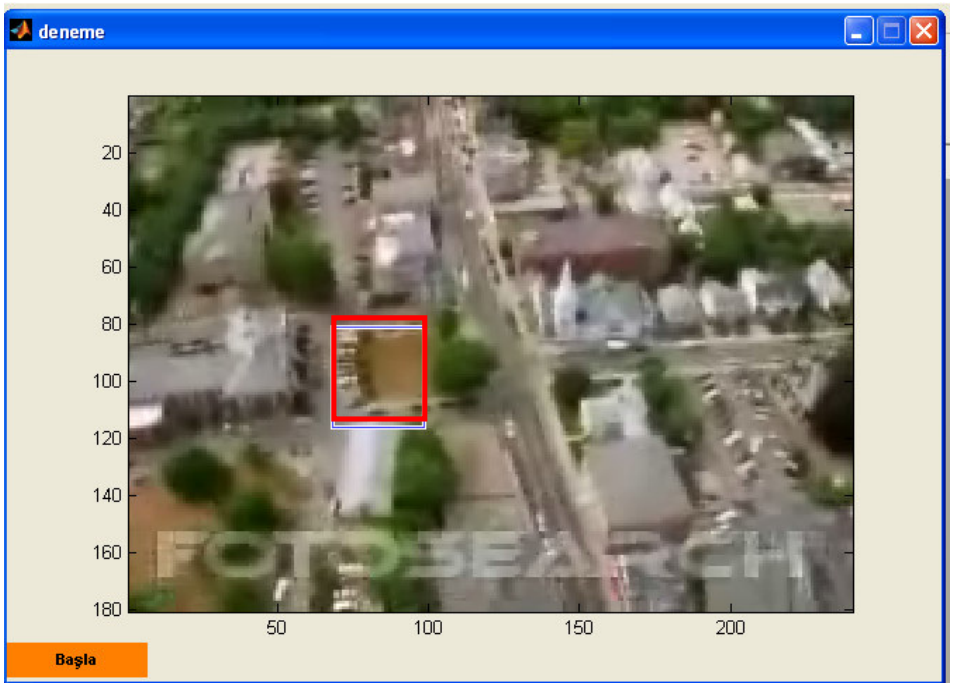
Resim 5.4. (Devam) Optimum SIFT parametreleri ile dördüncü test sonuçları

Resim 5.4 (a)-(f)'de görüldüğü gibi, hedef ve kamera hareketli ve döngüsel değişimin hedefin ekseni etrafında olduğu durumda hedef takibi başarı ile gerçekleştirilmiştir.

Test 5 : Helikopterden çekilen video görüntüsündeki rasgele seçilen hedefin takibi.

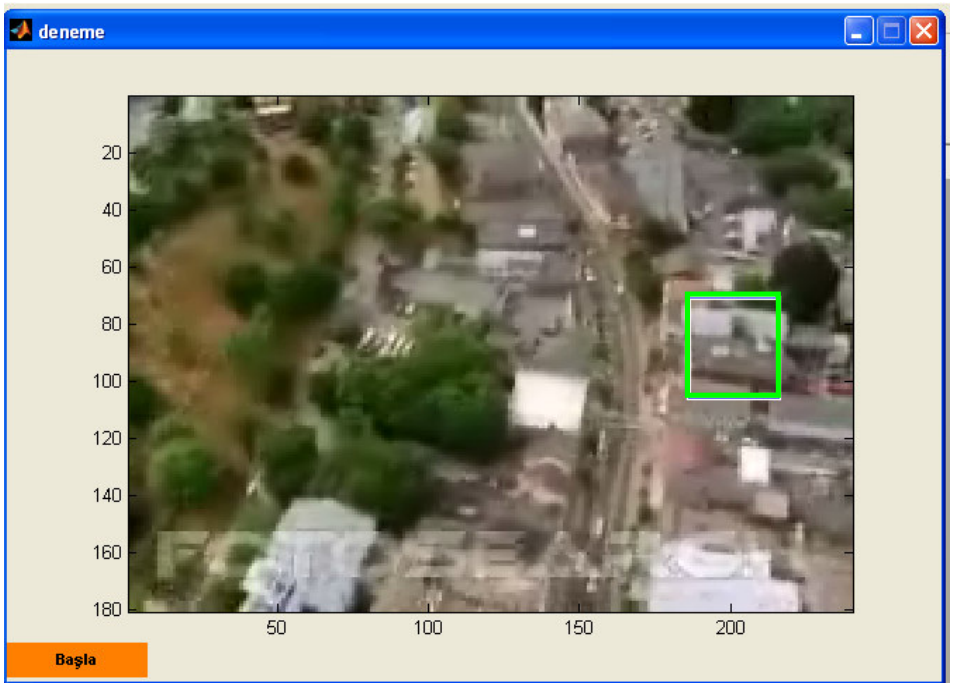


(a)

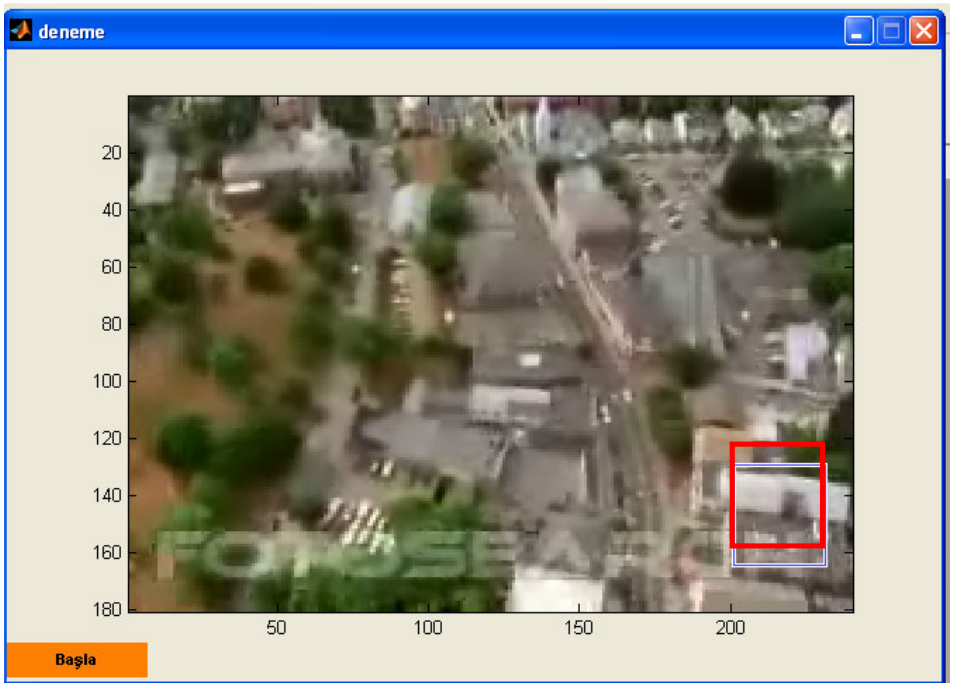


(b)

Resim 5.5. Optimum SIFT parametreleri ile beşinci test sonuçları

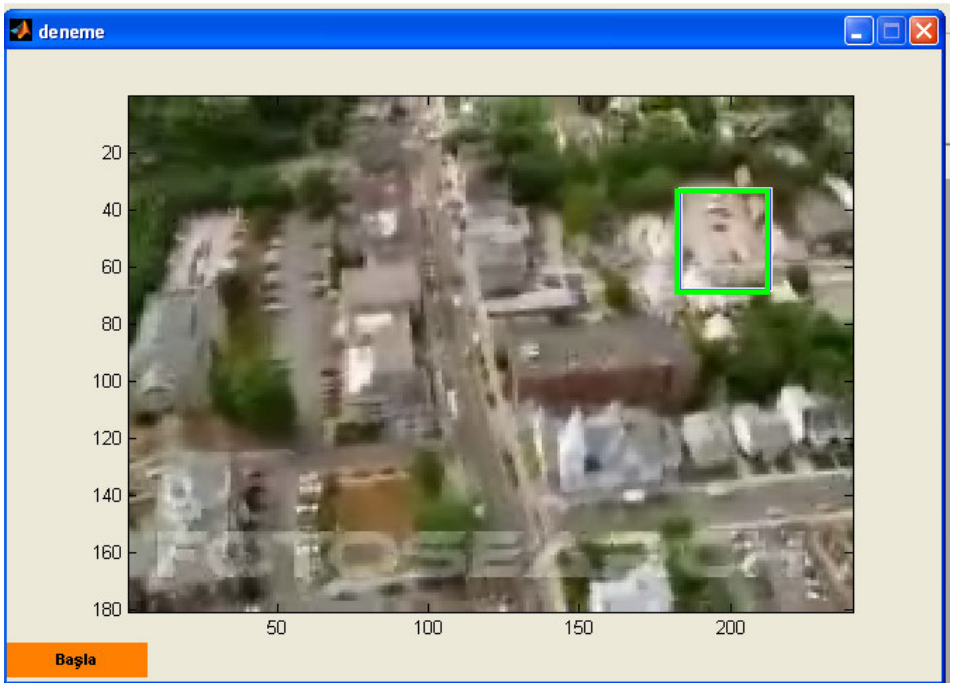


(c)

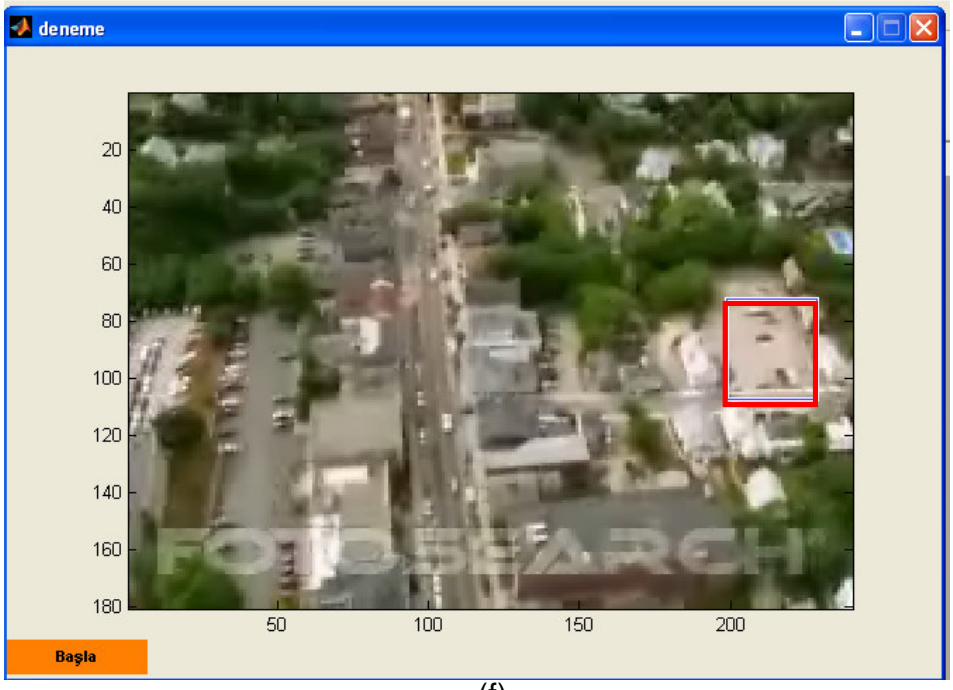


(d)

Resim 5.5. (Devam) Optimum SIFT parametreleri ile beşinci test sonuçları

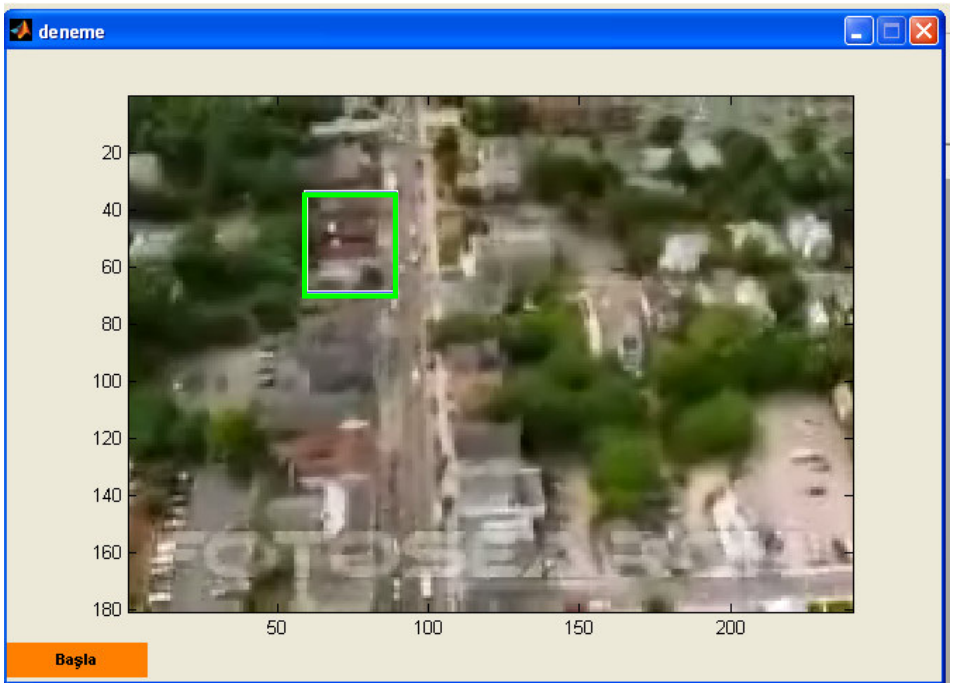


(e)

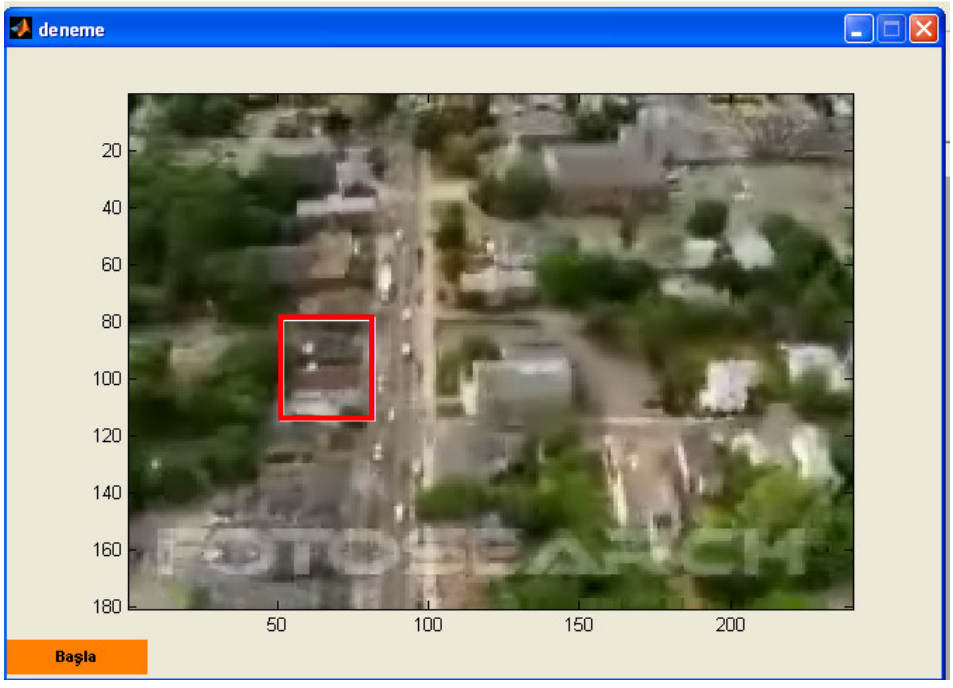


(f)

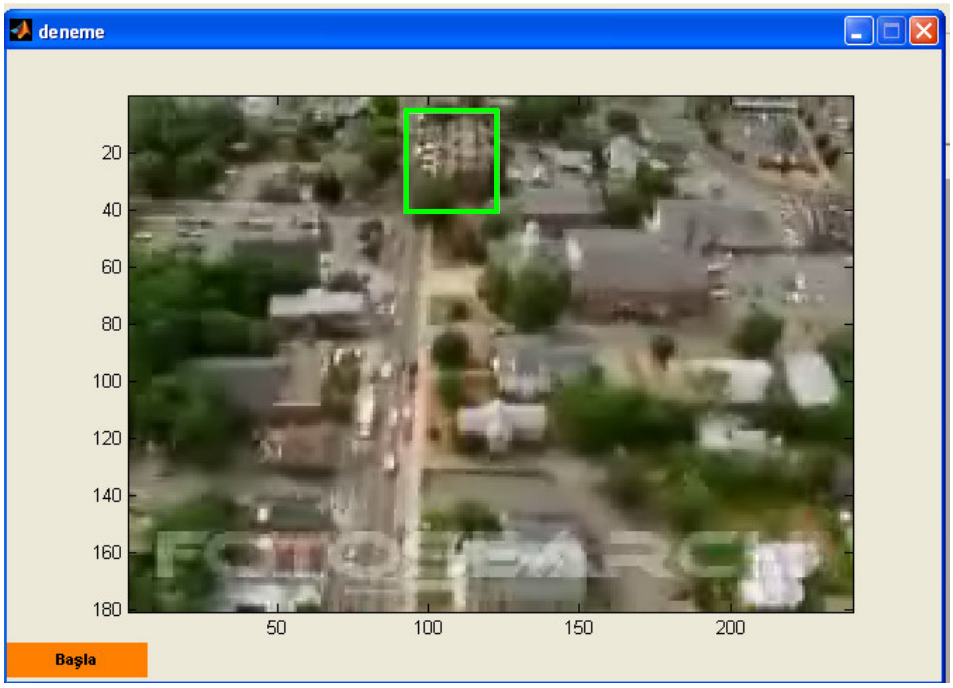
Resim 5.5. (Devam) Optimum SIFT parametreleri ile beşinci test sonuçları


deneme
20
40
60
80
100
120
140
160
180
50
100
150
200
Başla

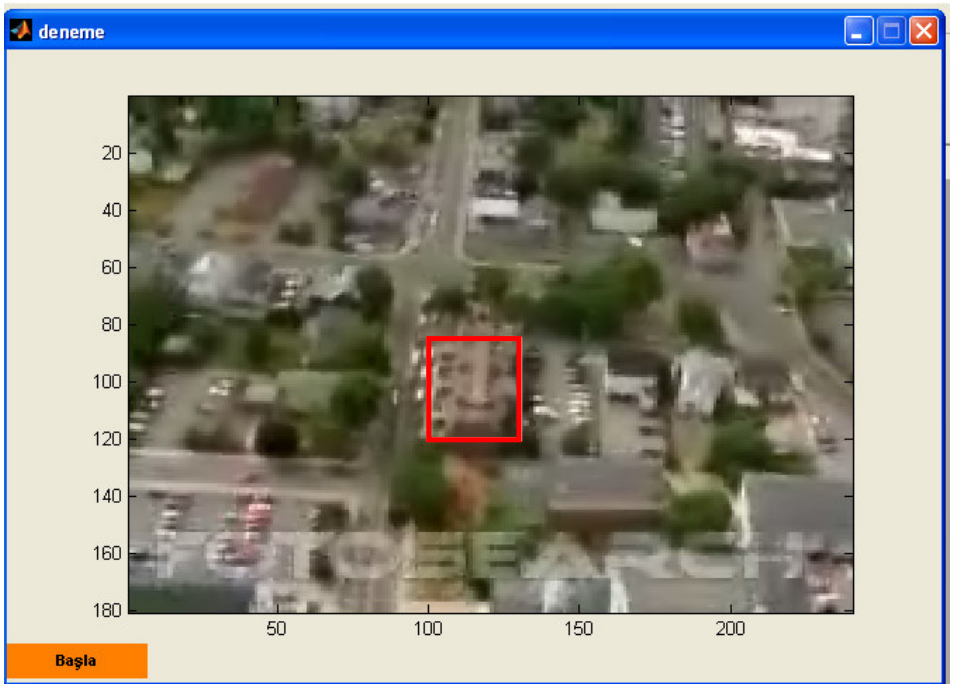
(g)


deneme
20
40
60
80
100
120
140
160
180
50
100
150
200
Başla

(h)



(i)

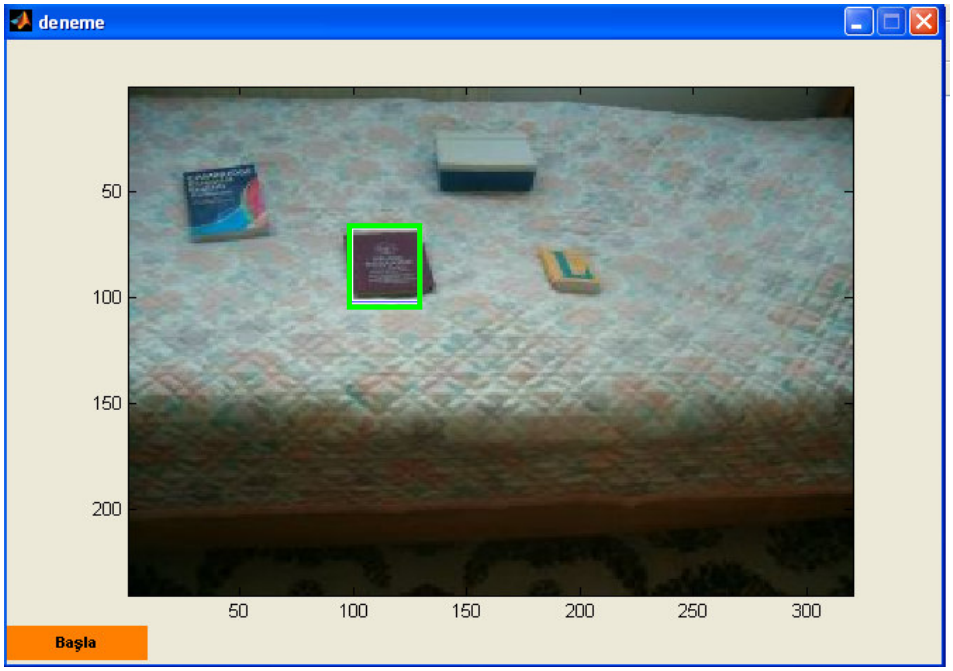


(j)

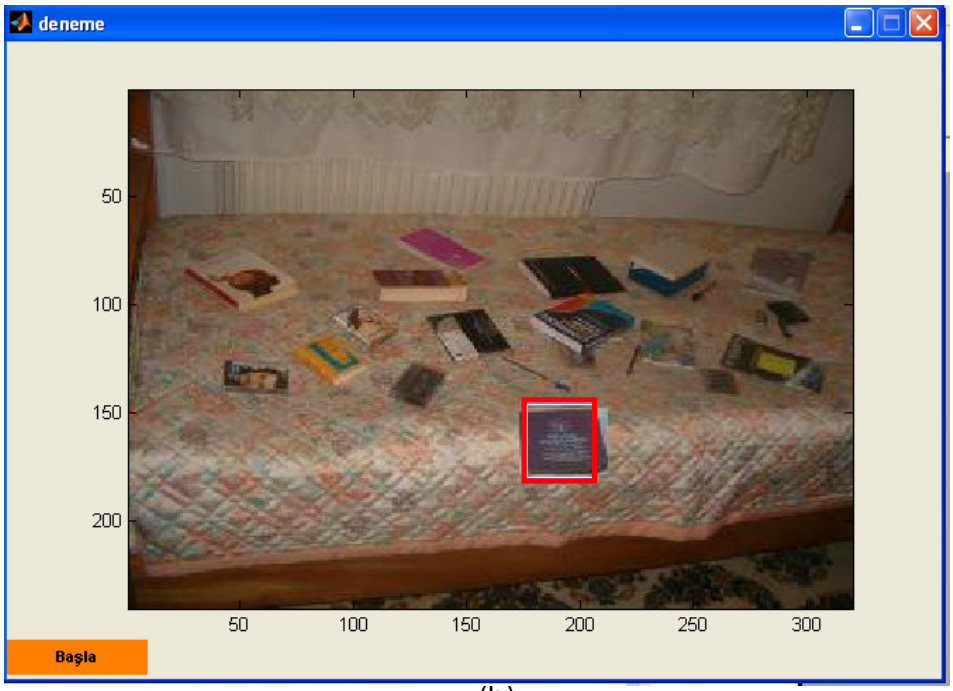
Resim 5.5. (Devam) Optimum SIFT parametreleri ile beşinci test sonuçları

Resim 5.5 (a)-(j)'da görüldüğü gibi, video görüntülerinde rasgele seçilen bir hedefin takibi başarı ile gerçekleştirilmiştir.

6. Test : Hareketli arka plan ve nesne ortamında takip

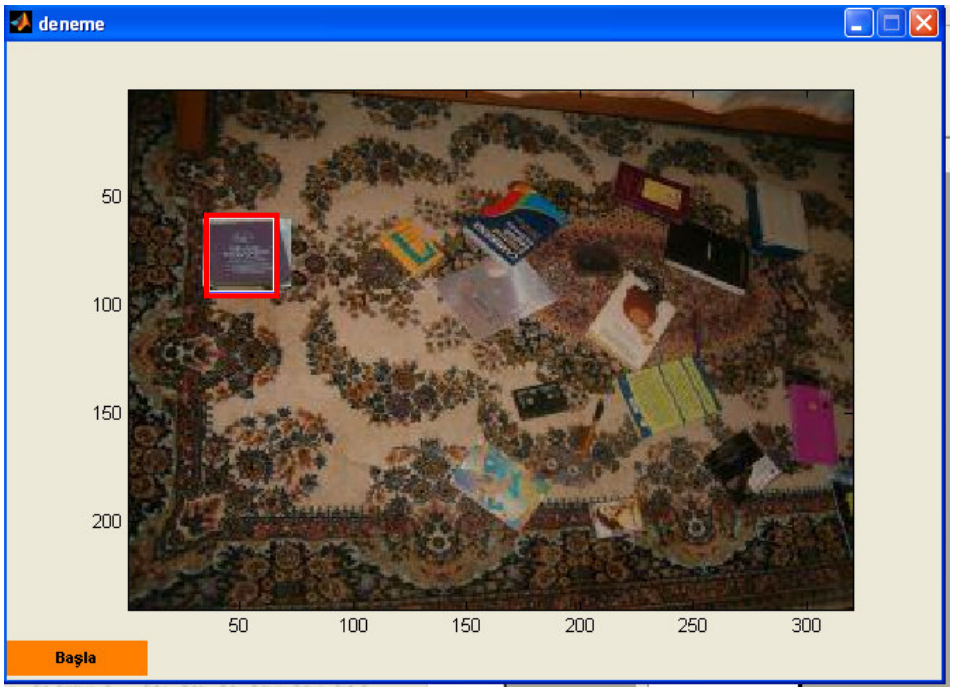


(a)

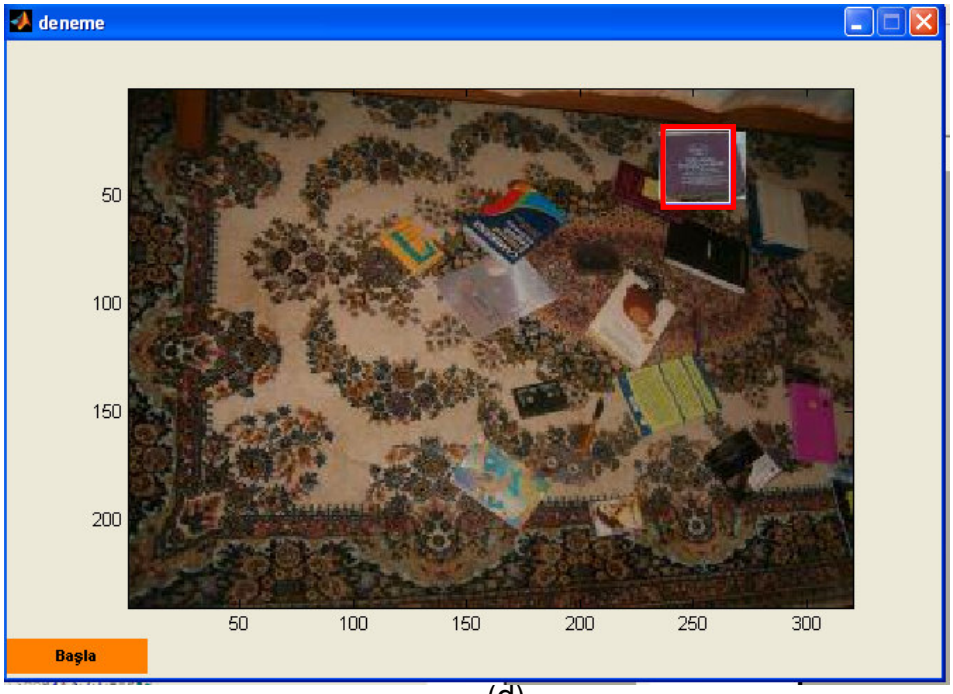


(b)

Resim 5.6. Optimum SIFT parametreleri ile altıncı test sonuçları

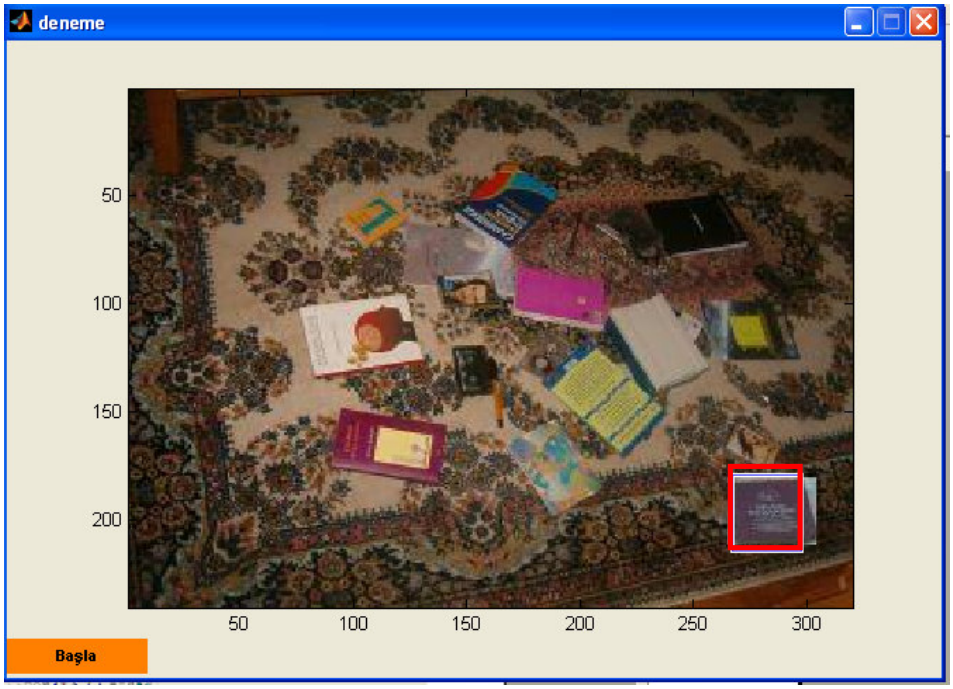


(c)

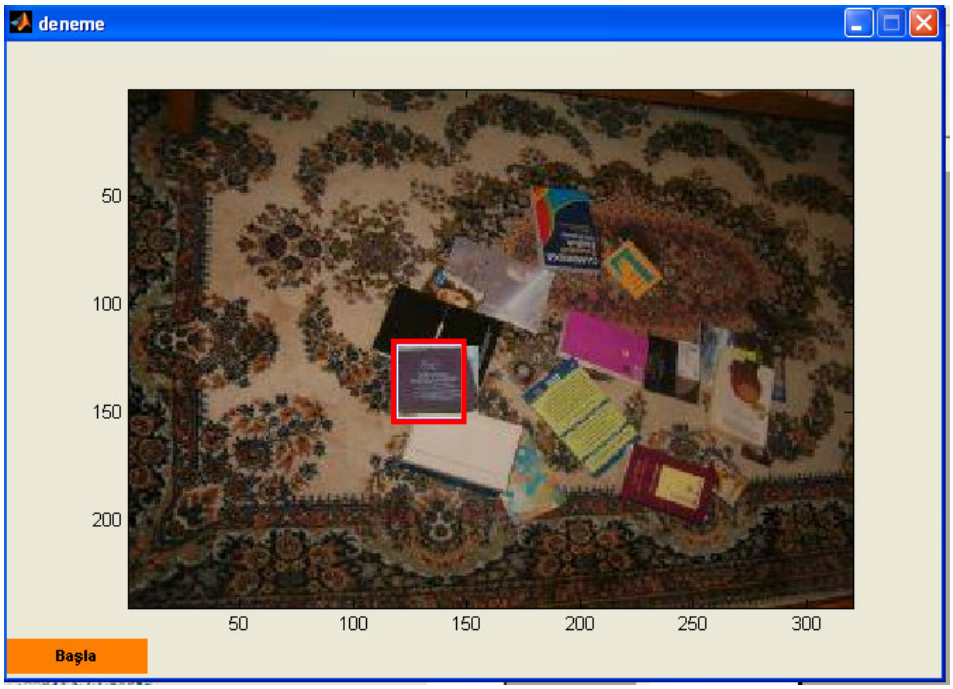


(d)

Resim 5.6. (Devam) optimum SIFT parametreleri ile altıncı test sonuçları

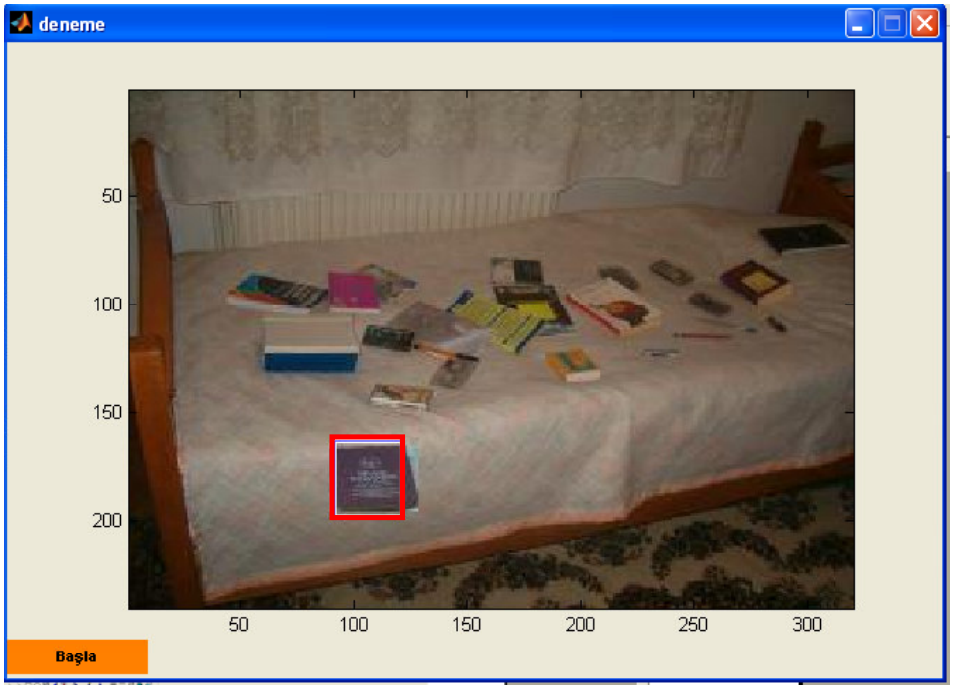


(e)



(f)

Resim 5.6. (Devam) optimum SIFT parametreleri ile altıncı test sonuçları



(g)

Resim 5.6. (Devam) optimum SIFT parametreleri ile altıncı test sonuçları

Resim 5.6 (a)-(g)'da görüldüğü gibi, değişken arka plan ortamında hareketli hedef takibi başarı ile gerçekleştirilmiştir.

6.SONUÇ VE ÖNERİLER

Hedef takibi için literatürde birçok çalışma yapılmıştır. Bu kapsamda önce hedef takibi için literatürdeki geliştirilen yöntemler üç grup halinde (arka plan çıkartımı, bölümleme ve özellikli nokta tanımlayıcıları) sınıflandırılmış ve tarihsel gelişimleri ile sunulmuştur. Daha sonra özellikli nokta tanımlayıcılarına, diğer yaklaşımlara oranla daha geniş yer verilmiştir. Özellikli nokta tanımlayıcıları içinde ise, SIFT algoritması seçildiği için bu algoritma ayrıntılı olarak açıklanmıştır.

Bu çalışmada SIFT algoritmasında belirtilen parametrelerin (Kademe sayısı s, düşük kontrasta sahip noktaların ayıklanması için gerekli eşik değer D(x), kenarlarda tespit edilen noktaların ayıklanması için gerekli eşik değer R, örnekleme alanı (nxn) ve kilit nokta tanımlayıcıları için gerekli her örnek noktanın temsil edileceği yön sayısı r) hedef takibi üzerindeki etkileri incelenmiştir. Yapılan analizler sonucunda bu parametrelere ait optimum değerler tespit edilmiştir. Bu değerlerin literatürdeki değerler ile uyumlu olduğu gözlenmiştir. Buna göre;

- Kademe sayısının kilit nokta sayısı ile olan ilişkisi literatürdeki ile aynı bulunmuş ve seçilmesi gereken optimum değerin hedef takip doğruluğu değerlendirildiğinde 3 olması gerektiği sonucuna varılmıştır.
- Düşük kontrasta sahip noktaların ayıklanması için gerekli eşik değer D(x)'in optimum değerinin etrafındaki küçük değişimlerin hedef takibindeki doğruluğu ve bulunan kilit nokta sayısını (işlem süresini) çok büyük oranda etkilemediği, ancak optimum değerin 10 kat altında veya üstünde alınmasının başarı oranını düşürdüğü gözlenmiştir.
- Kenarlarda tespit edilen noktaların ayıklanması için gerekli eşik değer R değerinin optimum değerinin 10 olduğu tespit edilmiş olup, R eşik değeri kilit nokta sayısı ile doğru orantılı olarak değişmektedir.
- Örnekleme alanı (nxn) optimum değeri, 4x4 olarak tespit edilmiştir. Bu değerin artması veya azalması hedef takibinin doğruluğunu etkilemektedir. Örnekleme alanı kilit nokta sayısı ile ters, işlem zamanı ile doğru orantılıdır.

- Kilit nokta tanımlayıcıları için gerekli her örnek noktanın temsil edileceği yön sayısı r, 8 olarak seçilmiştir. Bu değerin değişmesi kilit nokta sayısını etkilememekle birlikte başarı oranını düşürmektedir.

Ayrıca SIFT algoritması sonucunda oluşabilecek hatalı eşleşmeleri yok etmek için yeni bir düzenleme yapılmış olup, bu düzenleme ile hedef takip doğruluğu önemli ölçüde arttırılmıştır. Yine bu düzenleme ile, SIFT parametre değerlerinin belirlenen optimum değerlerden farklı olarak alındığında da meydana gelebilecek sapmaları önlemek için düzeltmeler yaptığı gözlenmiştir.

Daha sonra hedef takibinde etkili olacağı düşünülen takip edilecek nesnenin bulunduğu ortam (ortamdaki gürültü ve parlaklık değişimleri), nesnenin alabileceği konum (açısal, boyutsal, üç boyutlu değişimleri) ve görüntünün alındığı kameranın durumu gibi faktörler de göz önüne alınarak çeşitli testler başarı ile gerçekleştirilmiştir. Yapılan testlerde 320x240 (5.test 240x180) çözünürlüğe sahip resim kareleri ile 1.6 GHz.lik işlemci hızı kullanılmıştır. Bu testlerle ilgili görüntüler bölüm 5'te görülmektedir.

Yapılan testler sonucunda hedef takibinin doğruluğunun alınan resmin çözünürlüğünden ve açısal değişimlerden ile çok büyük oranda etkilendiği gözlenmiştir.

Ayrıca resim çözünürlüğünün ve işlemci hızının etkisi de bu çalışma kapsamında incelenmiştir. Resim çözünürlüğü ne kadar artarsa, hedef takibinin doğruluğu ve hassasiyeti aynı oranda artmakla birlikte işlem hızı da o oranda azalmaktadır. Çünkü tanımlanan özellikli nokta sayısı artmaktadır. Resim çözünürlüğü ile kilit nokta sayısının, SIFT kilit noktalarının tespit süresinin ve hedefi bulma süresinin exponansiyal olarak arttığı gözlenmiştir. Bu çalışmada 320x240 gibi düşük çözünürlüğe sahip resim kareleri kullanılmıştır. Bu çözünürlükten daha düşük resim karelerinde ise SIFT kilit noktaları başarılı bir şekilde tanımlanamadığından başarılı sonuçlar elde

edilememiştir. Oluşturulan program bölüm 5'te de görüldüğü gibi başarılı sonuçlar vermiştir. İşlemci hızının iki katına çıkması işlem süresini yaklaşık olarak yarıya düşürmüştür.

Açısal değişimler iki sebepten ileri gelebilmektedir. Bunlardan biri hedefteki hareket, diğeri ise kameradaki harekettir. Bu iki etki sebebiyle hedefin kameradaki görüntüsü önceki görüntülerine göre açısal olarak değişir. Bu değişim kameranın kendi ekseni etrafında ve hedefin etrafında da olabilir. Bu iki etki de oluşturularak, bölüm 5'te başarılı olarak test edilmiştir. Başarılı olmasının en önemli sebebi SIFT algoritması ile dönmeye karşı dayanıklı noktaların tanımlanabilmesidir.

İleriye Yönelik Çalışmalar

Tezin hayata geçirilmesi ve uygulanabilir olması için, işlem zamanını verimli kılacak önlemlerin alınması gerekmektedir. Yazılan programların basitleştirilerek, FPGA gibi paralel işlem yapabilme kapasitesine sahip gömülü sistemlere entegre edilmesi ve hareketli hedeflere uygulanması tez sonucunda yapılabilecek çalışmalar olacaktır.

KAYNAKLAR

1. Wong S., "Advanced Correlation Tracking of Objects in Cluttered Imagery", ***The Proceedings of SPIE: Acquisition, Tracking and Pointing XIX***,158-169 (2005).

2. Murray D. and Basu A., "Motion tracking with an active camera", ***IEEE Trans. Pattern Anal. & Mach. Intell.***, 16 (5): 449-459 (1995).

3. Araki S., Matsuoaka T., Yokoya N. and Takemura H., "Real time tracking of multiple moving object contours in a moving camera image sequence", ***IEICE Trans. Inf. & Syst.***, (7): 83 (2001).

4. *Hsieh F., Han C., Wu N. and Fan K.,* "A Novel Approach To The Noise Removal And Detection Of Small Objects With Low Contrast", ***Institute of Computer Science and Information Engineering National Central University***, 71-83 (2004)

5. Watson G.A., Blair W.D., "IMM Algorithm for tracking targets that maneuver through coordinated turns", ***Proc. of SPIE Signal And Data Processing of Small Targets,*** 236-247 (1992).

6. Murase H. and Nayar S.K., "Visual learning and recognition of 3-D objects from appearance", ***International Journal of Computer Vision***, 14 (1): 5-24 (1995).

7. Ramanan D. and Forsyth D.A. "Finding and tracking people from the bottom up", ***In IEEE Conference on Computer Vision and Pattern Recognition***, 467-474 (2003).

8. Stepanyan V. and Hovakimyan N., "An Adaptive Disturbance Rejection Controller for Visual Tracking of a Maneuvering Target", ***AIAA Journal of Guidance, Control and Dynamics***, 30 (4): 1090-1106 (2007).

9. Tao H., Sawhney H.S. and Kumar R., "Object Tracking with Bayesian Estimation of Dynamic Layer Representations", ***IEEE Transactıons On Pattern Analysıs And Machıne Intellıgence***, 24 (1): 75-89 (2002).

10. Burt, P.J., Bergen, J.R., Hingorani, R., Kolczynski, R., Lee, W.A., Leung, A., Lubin, J., Shvayster, H., "Object Tracking With a Moving Camera - An Application of Dynamic Motion Analysis", ***IEEE,*** 27: 9-16 (1989).

11. Chatterji G.B., Menon P.K., and Sridhar B., "Vision-Based Position and Attitude Determination for Aircraft Night Landing", ***Journal of Guidance, Control, and Dynamics***, 21 (1): 84-92 (1998).

12. DeWagter, C., Proctor, A., and Johnson, E., “Vision-Only Aircraft Flight Control”, ***Proceedings of the 22nd Digital Avionics Systems Conference***, 12-15 (2002).

13. Yao, Y., Chellappa, R., “Dynamic Feature Point Tracking in an Image Sequence”, **IEEE**, 654-657 (1994).

14. Kumar R., “Target Tracking”, Yüksek Lisans Tezi, ***Department of Computer Science and Engineering Indian Institute of Technology***, 23-27 (2006).

15. Harville M., “Stereo person tracking with adaptive plan-view statistical templates”, ***Proc. ECCV Workshop on Statistical Methods in Video Processing****,* 67–72 (2002).

16. Darrell T., Gordon G., Harville M. and Woodfill J., “Integrated person tracking using stereo, color, and pattern detection”, ***International Journal of Computer Vision***, 37 (2): 175–185 (2000).

17. Shao J., Zhou S.K. and Zheng Q., “Robust Appearance-Based Tracking of Moving Object from Moving Platform”, ***Proceedings of the 17th International Conference on Pattern Recognition (ICPR’04),***1051-4651 (2004)

18. Zhou S., Chellappa R. and Moghaddam B., “Visual tracking and recognition using appearance-based modeling in particle filters”, ***Proc. Intl. Conf. on Multimedia and Expo.,IEEE Trans. Image Processing***, 45-52 (2003).

19. Yue Z., Guarino D. and Chellappa R., “Moving Object Verification from Airborne”, ***Video Proceedings of the Fourth IEEE International Conference on Computer Vision Systems (ICVS 2006),*** 2506-2512 (2006).

20. Collett, T., "Visual neurons for tracking moving targets." ***Nature (Lond),*** 232: 127-130 (1971).

21. Bruce V., Henderson Z., Greenwood K., Hancock P. J. B., Burton A. M., and Miller P. I., Verification of face identities from images captured on video, ***J. Experimental Psychol.: Applied***, 5 (4): 339–360 (1999).

22. Gavrilla, D.M. And Davis, L.S., “3D Model Based Tracking of Humans in Action: Amulti-View Approach”, ***CVPRR***, 234-240 (1996).

23. O’Toole A. J., Roark D. A., and Abdi H., Recognizing moving faces: A psychological and neural synthesis, ***Trends Cogn. Sci***., 6: 261–266, (2002).

24. Lowe, D.G., "Object Recognition from Local Scale-Invariant Features", ***Proc. of the International Conference on Computer Vision***, (2): 1150-1157 (1999).

25. Lowe, D.G., "Local Feature View Clustering for 3D Object Recognition", ***Proc. of the IEEE Conference on Computer Vision and Pattern Recognition***, (1): 682-688 (2001).

26. Brown, M. and Lowe, D.G., "Invariant Features From Interest Points", ***British Machine Vision Conference,*** 656-665 (2003).

27. Se S., Lowe, D.G., Little, J., "Vision-based Mapping with Backward Correction", ***Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems EPFL***, 245-254 (2002).

28. Brown, M. And Lowe, D.G., "Recognising Panoramas", ***University of British Colombia***, 24-35 (2006).

29. Sivic, J. And Zisserman, A., "Video Google: A Text Retrieval Approach to Object Matching in Videos", ***Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003)***, 2 (0): 56-61 (2003).

30. İnternet : Wikipedia, "Scale invariant feature transform" http://en.wikipedia.org/wiki/Scale-invariant_feature_transform, (2008)

31. Lee, L., Romanio, R., Stein, G., "Monitoring activities from multiple video streams: Establishing acommon coordinate frame", ***IEEE Trans. Patt. Recogn. Mach. Intell. 22***, 8: 758–768 (2000).

32. Lindeberg T., "Detecting salient blob-like image structures and their scales with a scale-space primal sketch : A method for focus-of-attention," ***International Journal of Computer Vision***, 11 (3): 283–318 (1993).

33. Jain, R. And Nagel, H., "On the analysis of accumulative difference pictures from image sequences of real world scenes", ***IEEE Trans. Patt. Analy. Mach. Intell.***, 1 (2): 206–214 (1979).

34. Wren, C., Azarbayejani, A., And Pentland, A., "Pfinder: Real-time tracking of the human body", ***IEEE Trans. Patt. Analy. Mach. Intell.***, 19 (7): 780–785 (1997).

35. Gao, X., Boult, T., Coetzee, F., And Ramesh, V., "Error analysis of background adaption", ***IEEE Conference on Computer Vision and Pattern Recognition (CVPR)***, 503–510 (2000).

36. Stauffer, C. And Grimson, W., “Learning patterns of activity using real time tracking”, ***IEEE Trans.Patt. Analy. Mach. Intell****., 22*, 8, 747–767 (2000).

37. Elgalmmal, A., Harwood, D., And Davis, L.,”Non-parametric model for background subtraction”, ***European Conference on Computer Vision (ECCV),*** 751–767 (2000).

38. LEE, L., ROMANO, R., AND STEIN, G. 2000. Monitoring activities from multiple video streams: Establishing a common coordinate frame. ***IEEE Trans. Patt. Recogn. Mach. Intell. 22***, 8 (Aug.), 758–768.

39. Toyama, K., J. Krumm, B. B., And Meyers, B. “Wallflower: Principles and practices of background maintenance”, ***IEEE International Conference on Computer Vision (ICCV)***, 255–261 (2000).

40. Monnet, A., Mittal, A., Paragios, N., And Ramesh, V. “Background modeling and subtraction of dynamic scenes”, ***IEEE International Conference on Computer Vision (ICCV)***, 1305–1312 (2003).

41. Zhong , J. And Sclaroff, S. “Segmenting foreground objects from a dynamic textured background via a robust kalman filter”, ***IEEE International Conference on Computer Vision (ICCV)*** , 44–50 (2003).

42. Haralick, R., Shanmugam, B., And Dinstein, I., “Textural features for image classification”, ***IEEE Trans.Syst. Man Cybern****.,* 33 (3): 610–622 (1973).

43. Harville M., “A framework for high-level feedback to adaptive,per-pixel, mixture-of-gaussian background models”, ***European Conference on Computer Vision***, 543–560 (2002).

44. Haritaoğlu, İ., Harwood, D., And Davis, L. “Real-time surveillance of people and their activities”, ***IEEE Trans. Patt. Analy. Mach. Intell****. 22* (8): 809–830 (2000).

45. Collins, R., Lipton, A., Fujiyoshi, H., And Kanade, T., “Algorithms for cooperative multisensor surveillance” ***Proceedings of IEEE****, 89* (10): 1456–1477 (2001).

46. Kanade, T., Collins, R., Lipton, A., Burt, P., And Wixson, L. “Advances in cooperative multi-sensor video surveillance”, ***Darpa IU Workshop***., 3–24 (1998).

47. Rowe, S. And Blake, A., “Statistical mosaics for tracking”, ***Israel Verj. Cap. J.,*** 14: 549–564 (1996).

48. Shi, J. And Malik, J., “Normalized cuts and image segmentation”, ***IEEE Trans. Patt. Analy. Mach. Intell.,*** 22 (8): 888–905 (2000).

49. Comaniciu, D. And Meer, P., “Mean shift: A robust approach toward feature space analysis”, ***IEEE Trans. Patt. Analy. Mach. Intell.***, 24 (5): 603–619 (2002).

50. Wu, Z. And Leahy, R., “An optimal graph theoretic approach to data clustering: Theory and its applications to image segmentation”, ***IEEE Trans. Patt. Analy. Mach. Intell.*** ,11: 1101–1113 (2002).

51. Shi, J. And Malik, J., “Normalized cuts and image segmentation”, ***IEEE Trans. Patt. Analy. Mach.Intell***., *22* (8): 888–905 (2000).

52. Kass, M., Witkin, A., And Terzopoulos, D., “Snakes: active contour models”, ***Int. J. Comput. Vision 1***, 321–332 (1988).

53. Caselles, V., Kimmel R., And Sapiro, G., “Geodesic active contours”, ***IEEE International Conference on Computer Vision (ICCV),*** 694–699 (1995).

54. Paragios, N. And Deriche, R., “Geodesic active contours and level sets for the detection and tracking of moving objects”, ***IEEE Trans. Patt. Analy. Mach. Intell. 22***, 3: 266–280 (2000).

55. Ronfard, R. “Region based strategies for active contour models”, ***Int. J. Comput. Vision*** 13 (2): 229–251 (1994).

56. Zhu, S. And Yuille, A. “Region competition: unifying snakes, region growing, and bayes/mdl for multiband image segmentation” ***IEEE Trans. Patt. Analy. Mach. Intell.*** 18 (9): 884–900 (1996).

57. Yilmaz, A., Li, X., And Shah, M. “Contour based object tracking with occlusion handling in video acquired using mobile cameras”, ***IEEE Trans. Patt. Analy. Mach. Intell.*** 26 (11): 1531–1536 (2004).

58. Paragios, N. And Deriche, R., “Geodesic active regions and level set methods for supervised texture segmentation”, ***Int. J. Comput. Vision*** 46 (3): 223–247(2002).

59. Moravec, H., “Visual mapping by a robot rover”, ***Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)***. 598–600 (1979).

60. Harris, C. And Stephens, M., “A combined corner and edge detector”,***4th Alvey Vision Conference***, 147–151(1988).

61. Kanade, T., Collins, R., Lipton, A., Burt, P., And Wixson, L., “Advances in cooperative multi-sensor video surveillance” ***Darpa IU Workshop***, 3–24 (1998).

62. Lowe, D., “Distinctive image features from scale-invariant keypoints”, ***Int. J. Comput. Vision****,* 60 (2): 91–110(2004).

63. Moravec. H. P., “Towards Automatic Visual Obstacle Avoidance”, ***Proc. 5th International Joint Conference on Artificial Intelligence***, 584 (1977).

64. Moravec. H. P., “Visual Mapping by a Robot Rover”, ***International Joint Conference on Artificial Intelligence***, 598-600 (1979).

65. Harris C. and Stephens M., “A Combined Corner and Edge Detector”, ***Proc. Alvey Vision Conf.*****, Univ. Manchester**, 147-151 (1988).

66. Tomasi C. ve Kanade T., “Detection and Tracking of Point Features”, ***Carnegie Mellon University,*** 34-40 (1991).

67. Mikolajczyk, K. Schmid, C. , “A performance evaluation of local descriptors”, ***Dept. of Eng., Oxford Univ.***, 78-82 (2005).

68. Koenderink, J.J., “The structure of images”, ***Biological Cybernetics***, 50: 363-396 (1984).

69. Lindeberg, T., “Scale-space theory: A basic tool for analysing structures at different scales”, ***Journal of Applied Statistics***, 21(2): 224-270 (1994).

70. Mikolajczyk, K., “Detection of local features invariant to affine transformations”, Doktora, ***Institut National Polytechnique de Grenoble***, Fransa, (2002).

71. Brown, M. and Lowe, D.G., “Invariant features from interest point groups”, ***British Machine Vision Conference***, 656-665 (2002).

72. Harris, C. and Stephens, M., “A combined corner and edge detector”, ***Fourth Alvey Vision Conference***, 147-151 (1988).

73. Edelman, S., Intrator, N. and Poggio, T. 1997. “Complex cells and object recognition” http://kybele.psych.cornell.edu/~edelman/archive.html (1988)

EKLER

EK-1 Hedef_takibi Fonksiyonu

```
function varargout = Hedef_takibi(varargin)

% Arayüz Başlatımı
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Hedef_Takibi_OpeningFcn, ...
                   'gui_OutputFcn',  @Hedef_Takibi_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% Arayüz Başlatım Sonu

% --- Executes just before deneme is made visible.
function deneme_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to deneme (see VARARGIN)

% Choose default command line output for deneme
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% Arayüze program girişi resmi yüklenir.
myImg=imread('test.bmp');
imagesc(myImg);

% --- Başla tuşuna basıldığında aşağıdaki kod işletilir.
function Load_push_button_Callback(hObject, eventdata, handles)
% hObject    handle to Load_push_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

EK-2 Takip Fonksiyonu

```
function [position2,framex,descriptorx] = takip
(pre_image,curr_image,frames1,descriptors1,flag,position);

I1=pre_image;

if (flag==1)
   imagesc(I1);
   h=imrect(gca,position);
   pause(7);
   api = iptgetapi(h)
   position=api.getPosition();

end;

I2=curr_image;
imagesc(I2);
I2 = double(rgb2gray(I2)) ;
[frames2,descriptors2] = sift(I2, 'Verbosity', 1) ;

matches=siftmatch(descriptors1,descriptors2);


% Hedef Penceresinin ve eşleşmelerin konumları karşılaştırılarak hedef
penceresi içinde kalan eşleşmelerin bir sonraki resim karesindeki
karşılıklarının indeksleri bulunur.


int a;a=0;
int b;b=0;
int deil_x;
int deil_y;
int s;s=0;
int r;r=0;

% match.lerin frame1.deki karşılıkları takip penceresi içinde olanlarını al.
for i=1:size(matches,2)
   if (position(1) <= frames1(1,matches(1,i))) && (frames1(1,matches(1,i))<=
(position(1)+position(3)))
       a=a+1;
       match_index(1,a) = matches(1,i); % x yönünde takip pennceresi içinde
kalan match.lerin frame1 indexleri
       match_index(2,a) = matches(2,i); % x yönünde takip pennceresi içinde
kalan match.lerin frame2 indexleri
   end;
```

EK-3 Sift Fonksiyonu

```
function [frames,descriptors,gss,dogss]=sift(I,varargin)
% I: Gri ton formunda ve noktadan sonra 4 basamak hassasiyetinde olmalıdır.
```

Resim bölümleri Frames: Resimdeki kareler. 4xK'lık bir matristir. Her sütun bir resim bölümüne ait bilgiyi açıklar. Dolayısıyla K adet resim bölümü vardır. Her bölüm 4 adet özelik ile açıklanır. Bunlar;

- Matrisin ilk iki satırındaki karenin x ve y konumları Frames(1:2,k)
- Matrisin üçüncü satırındaki karenin ölçeksel SIGMA değeri Frames(3,k)
- Matrisin dördüncü satırındaki karenin açısal değeri Frames(4,k)

Özellik noktaları (Descr): SIFT algoritması uygulanarak tespit edilen kilit noktalara ait tanımlayıcılardır. DxK'lık bir matristir. (D genellikle 128'dir.)

Fonksiyon aşağıdaki parametrelerin değiştirilmesine de izin vermektedir.
Bu parametreler SIFT algoritmasında kullanılan ve kilit noktaların tanımlanmasında büyük öneme sahip parametrelerdir. Bu parametrelere göz atarsak;

NumOctaves : Ölçeksel uzayda bulunacak kademe sayısı. Atanmamış ise olası tüm özelliklerin büyüklüğünü kapsayacak şekilde hesaplanır.

FirstOctave : Birinci ölçeksel uzay kademesinin indeksi. '-1' olarak ayarlanırsa ölçeksel uzay hesaplanmadan önce resimdeki değerlerin hassasiyeti noktadan sonra 4 basamak hassasiyetine sahip olacak şekilde düzenlenir.

NumLevels : Her ölçeksel uzay kademesindeki ölçek seviyesi miktarı.

Sigma0 : Gauss fonksiyonunun temel standart sapma değeri

EK-3 (Devam) Sift Fonksiyonu

Ölçeksel uzay kademesi 0'daki 0'ıncı ölçek seviyesine ait değer. Varsayılan değeri 1.6'dır.

SigmaN : Gauss fonksiyonunun nominal standart sapma değeri

Threshold : DoG ölçeksel uzayındaki uç noktaları bulmak için kullanılan eşik değeri. Bu değerin altında bulunan noktalar elenir. Bu değer 0.01 olarak alınıştır. Daha küçük alınması daha fazla noktanın elde edilmesine sebep olur.

EdgeThreshold : Kenar Eşik değeri.
Kenarlarda bulunan gürültüden çok fazla etkilenen noktalardan güçsüz olan noktaların elenmesinde kullanılan eşik değeridir. Bu değer Lowe'un makalesinde belirttiği üzere 10 olarak alınmıştır.

NumSpatialBins : Kilit nokta tanımlayıcılarının açı yönlerinin sayısı.

Fonksiyonun içine alabileceği değerler olup olmadığını kontrol eder
yazılan değerler yoksa default olarak SIFT parametrelerine değer atar.

```
if(nargin < 1)
   error('At least one argument is required.') ;
end

[M,N,C] = size(I) ;  % I resminin boyutları MxN piksele sahip

%------ SIFT Parametrelerinin girişi------------------
S      =  3 ;   % Kademelerdeki seviye sayısı
omin   = -1 ;   % Birinci kademe
O      = floor(log2(min(M,N)))-omin-3 ; % Kademe Sayısı
sigma0 = 1.6*2^(1/S) ;                  % smooth lev. -1 at 1.6
sigman = 0.5 ;
thresh = 0.04 / S / 2 ;
r      = 10 ;
NBP    = 4 ;
NBO    = 8 ;
magnif = 3.0 ;
```

EK-4 Gaussians Fonksiyonu

```
function SS = gaussianss(I,sigman,O,S,omin,smin,smax,sigma0)
% Resmin Gauss filtresinden geçirilmesi

% ------------------------------------------------------------Fonksiyon Argümanlarının
Kontrol edilmesi
if(nargin < 6)
   error('Six arguments are required.') ;
end

if(~isreal(I) || ndims(I) > 2)
% I resmi içindeki sayılar real değilse veya 2 boyuttan çok ise hata mesajı ver
   error('I must be a real two dimensional matrix') ;
end

if(smin >= smax)                                    %
   error('smin must be greather or equal to smax') ;
end

% ----------------------------------------------------------------
% Filtreden geçirme işlemi

k = 2^(1/S) ;

% Kademe o ve seviye s'e (o,s)=(0,-1) ait standart sapma değeri 1.6
if(nargin < 7)
Fonksiyonun içindeki argüman sayısı 7'den küçük ise sigma0'a değer
atanması
  sigma0 = 1.6 * k ;
end

dsigma0 = sigma0 * sqrt(1 - 1/k^2) ;
sigman  = 0.5 ;

% Ölçeksel uzay yapısı
SS.O         = O ;
SS.S         = S ;
SS.sigma0    = sigma0 ;
SS.omin      = omin ;
SS.smin      = smin ;
SS.smax      = smax ;

if omin < 0
   for o=1:-omin
      I = doubleSize(I) ;
   end
```

EK-5 Diffss Fonksiyonu

```
function dss = diffss(ss)
% DOG resimlerinin oluşturulması

dss.smin = ss.smin ;
dss.smax = ss.smax-1 ;
dss.omin = ss.omin ;
dss.O = ss.O ;
dss.S = ss.S ;
dss.sigma0 = ss.sigma0 ;

for o=1:dss.O
  [M,N,S] = size(ss.octave{o}) ;
  dss.octave{o} = zeros(M,N,S-1) ;
  for s=1:S-1
    dss.octave{o}(:,:,s) = ...
        ss.octave{o}(:,:,s+1) - ss.octave{o}(:,:,s) ;
  end
end
```

EK-6 Siftdescriptor Fonksiyonu

Siftdescriptor.c

```
% SIFT tanımlayıcıları hesaplanır.

#define LOWE_COMPATIBLE

#include"mexutils.c"
#include<stdlib.h>
#include<math.h>

#ifdef WINDOWS
#include<string.h>
#ifndef __cplusplus
#define sqrtf(x)    ((float)sqrt((double)(x)))
#define powf(x,y)   ((float)pow((double)(x),(double)(y)))
#define fabsf(x)    ((float)fabs((double)(x)))
#define sinf(x)     ((float)sin((double)(x)))
#define cosf(x)     ((float)cos((double)(x)))
#define expf(x)     ((float)exp((double)(x)))
#define atan2f(x,y) ((float)atan2((double)(x),(double)(y)))
#endif
#else
#include<strings.h>
#endif

#if defined( MACOSX ) && defined( __ALTIVEC__ )
#include<Accelerate/Accelerate.h>
typedef union
{
  float x[4] ;
  vFloat vec ;
} float4 ;
#endif

#define greater(a,b) a > b
#define min(a,b) (((a)<(b))?(a):(b))
#define max(a,b) (((a)>(b))?(a):(b))


enum {SCALESPACE, NOSCALESPACE} ;

enum {PROP_MAGNIF=0,
      PROP_NBP,
      PROP_NBO,
      PROP_UNKNOWN} ;

char const * properties [4] =
```

EK-6 (Devam) Siftdescriptor Fonksiyonu

```
   /Gradyentlerin hesaplanması

   for(s = 0 ; s < num_levels ; ++s) {
     const double* pt = G_pt + s*so ;
     for(x = 1 ; x < N-1 ; ++x) {
       for(y = 1 ; y < M-1 ; ++y) {
         float Dx = 0.5 * ( at(x+1,y) - at(x-1,y) ) ;
         float Dy = 0.5 * ( at(x,y+1) - at(x,y-1) ) ;
         buffer_pt[(x*xo+y*yo+s*so) + 0          ] = Dx ;
         buffer_pt[(x*xo+y*yo+s*so) + buffer_size] = Dy ;
       }
     }
   }

   /* Açıların ve modüllerin hesaplanması */
   {
     float* pt = buffer_pt ;
     int j = 0 ;
     while (j < N*M*num_levels) {

#if defined( MACOSX ) && defined( __ALTIVEC__ )
       if( ((unsigned int)pt & 0x7) == 0 && j+3 < N*M*num_levels ) {
         /* 128 bite ayarlandıysa en az 4 piksel kalmıştır.
         float4 a, b, c, d ;
         a.vec = vec_ld(0,(vector float*)(pt              )) ;
         b.vec = vec_ld(0,(vector float*)(pt + buffer_size)) ;
         c.vec = vatan2f(b.vec,a.vec) ;
         a.x[0] = a.x[0]*a.x[0]+b.x[0]*b.x[0] ;
```

EK-7 Siftlocalmax Fonksiyonu

```
siftlocalmax.c
Ölçeksel uzaydaki uç noktaların bulunması
#include"mex.h"
#include<mexutils.c>
#include<stdlib.h>

#define greater(a,b) ((a) > (b)+threshold)

void
mexFunction(int nout, mxArray *out[],
            int nin, const mxArray *in[])
{
  int M, N ;
  const double* F_pt ;
  int ndims ;
  int pdims = -1 ;
  int* offsets ;
  int* midx ;
  int* neighbors ;
  int nneighbors ;
  int* dims ;
  enum {F=0,THRESHOLD,P} ;
  enum {MAXIMA=0} ;
  double threshold = - mxGetInf() ;

  /* ----------------------------------------------------------------
Argümanların kontrol edilmesi

  if (nin < 1) {
    mexErrMsgTxt("At least one input argument is required.");
  } else if (nin > 3) {
    mexErrMsgTxt("At most three arguments are allowed.") ;
  } else if (nout > 1) {
    mexErrMsgTxt("Too many output arguments");
  }

  /* Giriş reel olmalıdır */
  if (!mxIsDouble(in[F]) || mxIsComplex(in[F])) {
    mexErrMsgTxt("Input must be real matrix.");
  }

  if(nin > 1) {
    if(!uIsRealScalar(in[THRESHOLD])) {
      mexErrMsgTxt("THRESHOLD must be a real scalar.") ;
    }
```

EK-7 (Devam) Siftlocalmax Fonksiyonu

```
    threshold = *mxGetPr(in[THRESHOLD]) ;
  }

  if(nin > 2) {
    if(!mxIsRealScalar(in[P]))
      mexErrMsgTxt("P must be a non-negative integer") ;
    pdims = (int) *mxGetPr(in[P])  ;
    if(pdims < 0)
      mexErrMsgTxt("P must be a non-negative integer") ;
  }

  ndims = mxGetNumberOfDimensions(in[F]) ;
  {
    int d ;
    const int* const_dims = (int*) mxGetDimensions(in[F]) ;
    dims = mxMalloc(sizeof(int)*ndims) ;
    for(d=0 ; d < ndims ; ++d) dims[d] = const_dims[d] ;
  }
  M = dims[0] ;
  N = dims[1] ;
  F_pt = mxGetPr(in[F]) ;

  /*
     Matlab 1xN or Mx1 matrisi ayırt etmez.
  if((ndims == 2) && (pdims < 0) && (M == 1 || N == 1)) {
    pdims = 1 ;
    M = (M>N)?M:N ;
    N = 1 ;
    dims[0]=M ;
    dims[1]=N ;
  }

  /* p boyutu boyunca maksimum noktaları araştır.*/
  if(pdims < 0)
    pdims = ndims ;

  if(pdims > ndims) {
    mxFree(dims) ;
    mexErrMsgTxt("P must not be greater than the number of dimensions") ;
  }

  /* ---------------------------------------------------------------
İşlemin yapılması
  {
    int maxima_size = M*N ;
```

EK-7 (Devam) Siftlocalmax Fonksiyonu

```
  int* maxima_start = (int*) mxMalloc(sizeof(int) * maxima_size) ;
  int* maxima_iterator = maxima_start ;
  int* maxima_end = maxima_start + maxima_size ;
  int i,h,o ;
  const double* pt = F_pt ;

  /* Boyutlar arasındaki offset değerleri hesaplanır. */
  offsets = (int*) mxMalloc(sizeof(int) * ndims) ;
  offsets[0] = 1 ;
  for(h = 1 ; h < ndims ; ++h)
    offsets[h] = offsets[h-1]*dims[h-1] ;

  /* Çoklu indeks */
  midx = (int*) mxMalloc(sizeof(int) * ndims) ;
  for(h = 0 ; h < ndims ; ++h)
    midx[h] = 1 ;

  /* Komşular */
  nneighbors = 1 ;
  o=0 ;
  for(h = 0 ; h < pdims ; ++h) {
    nneighbors *= 3 ;
    midx[h] = -1 ;
    o -= offsets[h] ;
  }
  nneighbors -= 1 ;
  neighbors = (int*) mxMalloc(sizeof(int) * nneighbors) ;


  i = 0 ;

  while(true) {
    if(o != 0)
      neighbors[i++] = o ;
    h = 0 ;
    while( o += offsets[h], (++midx[h]) > 1 ) {
      o -= 3*offsets[h] ;
      midx[h] = -1 ;
      if(++h >= pdims)
        goto stop ;
    }
  }
stop: ;

  /* Araştırmaya köşeden başlanır. */
```

EK-7 (Devam) Siftlocalmax Fonksiyonu

```
  for(h = 0 ; h < pdims ; ++h) {
    midx[h] = 1 ;
    pt += offsets[h] ;
  }
  for(h = pdims ; h < ndims ; ++h) {
    midx[h] = 0 ;
  }

  /* ------------------------------------------------------------
     Döngü
  for(h=0 ; h < pdims ; ++h)
    if(dims[h] < 3) goto end ;

  while(true) {

    h = 0 ;
    while((midx[h]) >= dims[h] - 1) {
      pt += 2*offsets[h] ; /* skip first and last el. */
      midx[h] = 1 ;
      if(++h >= pdims)
        goto next_layer ;
      ++midx[h] ;
    }


    /*  Komşuları tara */
    {
      double v = *pt ;
      bool is_greater = (v >= threshold) ;
      i = 0  ;
      while(is_greater && i < nneighbors)
        is_greater &= v > *(pt + neighbors[i++]) ;

      /* Uç nokta maksmumuna ekle */
      if(is_greater) {
        if(maxima_iterator == maxima_end) {
          maxima_size += M*N ;
          maxima_start = (int*) mxRealloc(maxima_start,
                                          maxima_size*sizeof(int)) ;
          maxima_end = maxima_start + maxima_size ;
          maxima_iterator = maxima_end - M*N ;
        }

        *maxima_iterator++ = pt - F_pt + 1 ;
```

EK-7 (Devam) Siftlocalmax Fonksiyonu

```
      /* Bir sonraki elemana geç */
      pt += 1 ;
      ++midx[0] ;
      continue ;

    next_layer: ;
      if( h >= ndims )
        goto end ;

      while((++midx[h]) >= dims[h]) {
        midx[h] = 0 ;
        if(++h >= ndims)
          goto end ;
      }
    }
  }
 end:;
  /* Geri Dön */
  {
    double* M_pt ;
    out[MAXIMA] = mxCreateDoubleMatrix
      (1, maxima_iterator-maxima_start, mxREAL) ;
    maxima_end = maxima_iterator ;
    maxima_iterator = maxima_start ;
    M_pt = mxGetPr(out[MAXIMA]) ;
    while(maxima_iterator != maxima_end) {
      *M_pt++ = *maxima_iterator++ ;
    }
  }

  /* Geçici olarak kullanılan değişkenleri serbest bırak. */
  mxFree(offsets) ;
  mxFree(neighbors) ;
  mxFree(midx) ;
  mxFree(maxima_start) ;
 }
 mxFree(dims) ;
}
```

EK-8 Imsmooth Fonksiyonu

imsmooth.c
Resimdeki düzensizliklerin atılması

```
#include"mex.h"
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<assert.h>

#define greater(a,b) ((a) > (b))
#define min(a,b) (((a)<(b))?(a):(b))
#define max(a,b) (((a)>(b))?(a):(b))

#define PAD_BY_CONTINUITY

const double win_factor = 1.5 ;
const int nbins = 36 ;

/* Resim Konvülasyona ve transpos işlemlerine tabi tutulur */
void
convolve(double*       dst_pt,
         const double* src_pt,    int M, int N,
         const double* filter_pt, int W)
{
  int i,j ;
  for(j = 0 ; j < N ; ++j) {
    for(i = 0 ; i < M ; ++i) {
      double const* start = src_pt + max(i-W,0  ) ;
      double const* stop  = src_pt + min(i+W,M-1) + 1 ;
      double const* g     = filter_pt + max(0, W-i) ;
      double acc = 0.0 ;
      while(stop != start) acc += (*g++) * (*start++) ;
      *dst_pt = acc ;
      dst_pt += N ;
    }
    src_pt += M ;
    dst_pt -= M*N - 1 ;
  }
}

/* Resim Konvülasyona ve transpos ve sürekli doldurma işlemlerine tabi
tutulur
void
econvolve(double*       dst_pt,
      const double* src_pt,    int M, int N,
      const double* filter_pt, int W)
```

EK-8 (Devam) Imsmooth Fonksiyonu

```
{
  int i,j ;
  for(j = 0 ; j < N ; ++j) {
    for(i = 0 ; i < M ; ++i) {
      double       acc = 0.0 ;
      double const* g = filter_pt ;
      double const* start = src_pt + (i-W) ;
      double const* stop  ;
      double       x ;

      /* Başlangıç */
      stop = src_pt ;
      x    = *stop ;
      while( start <= stop ) { acc += (*g++) * x ; start++ ; }
      stop =  src_pt + min(M-1, i+W) ;
      while( start <  stop ) acc += (*g++) * (*start++) ;

      /* Bitiş */
      x    = *start ;
      stop = src_pt + (i+W) ;
      while( start <= stop ) { acc += (*g++) * x ; start++ ; }

      /* Kayıt*/
      *dst_pt = acc ;
      dst_pt += N ;
    }
    /* Bir sonraki sütun */
    src_pt += M ;
    dst_pt -= M*N - 1 ;
  }
}

void
mexFunction(int nout, mxArray *out[],
            int nin, const mxArray *in[])
{
  int M,N ;
  double* I_pt ;
  double* J_pt ;
  double s ;
  enum {I=0,S} ;
  enum {J=0} ;

  /* -----------------------------------------------------------
Argümanları kontrol et
  ** ----------------------------------------------------------- */
```

EK-9 Mexutils Fonksiyonu

mexutils.c
MEX dosyalarına yazmak için yardımcı fonksiyonlar

```
#include"mex.h"
#include<math.h>

#undef M_PI
#define M_PI 3.14159265358979

/* Matrisin Reel ve Ölçeksel olduğunu kontrol eder. A matrisi reel ve ölçeksel
ise ‘Doğru’ değerini döndürür

int mxIsRealScalar(const mxArray* A)
{
  return
    mxIsDouble(A) &&
    !mxIsComplex(A) &&
    mxGetNumberOfDimensions(A) == 2 &&
    mxGetM(A) == 1 &&
    mxGetN(A) == 1 ;
}

/* Matrisin Reel ve Ölçeksel olduğunu, M (satır sayısı) nin ve N (sütun
sayısı)nın ‘0’ olmadığını kontrol eder. Eğer Evet ise ‘Doğru’ değerini
döndürür.

int
mxIsRealMatrix(const mxArray* A, int M, int N)
{
  return
    mxIsDouble(A) &&
    !mxIsComplex(A) &&
    mxGetNumberOfDimensions(A) == 2 &&
    ((M>=0)?(mxGetM(A) == M):1) &&
    ((N>=0)?(mxGetN(A) == N):1) ;
}


V matrisinin reel olduğunu ve boyutunun ‘0’ olmadığını kontrol eder. Eğer
Evet ise ‘Doğru’ değerini döndürür.

int
mxIsRealVector(const mxArray* V, int D)
{
  int M = mxGetM(V) ;
```

EK-9 (Devam) Mexutils Fonksiyonu

```
  int N = mxGetN(V) ;
  int is_vector = (N == 1) || (M == 1) ;

  return
    mxIsDouble(V) &&
    !mxIsComplex(V) &&
    mxGetNumberOfDimensions(V) == 2 &&
    is_vector &&
    ( D < 0 || N == D || M == D) ;
}
```

S matrisinin string yapıda olduğunu ve pozitif olan L uzunluğuna sahip olduğunu kontrol eder.

```
int
uIsString(const mxArray* S, int L)
{
  int M = mxGetM(S) ;
  int N = mxGetN(S) ;

  return
    mxIsChar(S) &&
    M == 1 &&
    (L < 0 || N == L) ;
}
```

EK-10 Siftormx Fonksiyonu

siftormx.c
Döngüsel histogramdaki pikleri hesaplar

```
#include"mex.h"
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<assert.h>

#include<mexutils.c>

#define greater(a,b)  ((a)>(b))
#define min(a,b)     (((a)<(b))?(a):(b))
#define max(a,b)     (((a)>(b))?(a):(b))

const double win_factor = 1.5 ;
#define NBINS 36

void
mexFunction(int nout, mxArray *out[],
            int nin, const mxArray *in[])
{
  int M,N,S,smin,K ;
  const int* dimensions ;
  const double* P_pt ;
  const double* G_pt ;
  double* TH_pt ;
  double sigma0 ;
  double H_pt [ NBINS ] ;

  enum {IN_P=0,IN_G,IN_S,IN_SMIN,IN_SIGMA0} ;
  enum {OUT_Q=0} ;

  /* Argümanları kontrol eder.  */
  if (nin != 5) {
    mexErrMsgTxt("Exactly five input arguments required.");
  } else if (nout > 1) {
    mexErrMsgTxt("Too many output arguments.");
  }

  if( !uIsRealScalar(in[IN_S]) ) {
    mexErrMsgTxt("S should be a real scalar") ;
  }

  if( !uIsRealScalar(in[IN_SMIN]) ) {
    mexErrMsgTxt("SMIN should be a real scalar") ;
```

EK-10 (Devam) Siftormx Fonksiyonu

```
if( !uIsRealScalar(in[IN_SIGMA0]) ) {
  mexErrMsgTxt("SIGMA0 should be a real scalar") ;
}

if( !uIsRealMatrix(in[IN_P],3,-1)) {
  mexErrMsgTxt("P should be a 3xK real matrix") ;
}

if(mxGetNumberOfDimensions(in[IN_G]) != 3) {
  mexErrMsgTxt("SSO must be a three dimensional array") ;
}

dimensions = mxGetDimensions(in[IN_G]) ;
M = dimensions[0] ;
N = dimensions[1] ;
S = (int)(*mxGetPr(in[IN_S])) ;
smin = (int)(*mxGetPr(in[IN_SMIN])) ;
sigma0 = *mxGetPr(in[IN_SIGMA0]) ;

K = mxGetN(in[IN_P]) ;
P_pt = mxGetPr(in[IN_P]) ;
G_pt = mxGetPr(in[IN_G]) ;


/* Giriş mtrisi boş ise, çıkış matrisi de boş olarak ata */
if(K == 0) {
  out[OUT_Q] = mxCreateDoubleMatrix(4,0,mxREAL) ;
  return ;
}

/* ----------------------------------------------------------
İşlemin yapılması
 * ----------------------------------------------------------
{
  int p ;
  const int yo = 1 ;
  const int xo = M ;
  const int so = M*N ;

  int buffer_size = K*4 ;
  double* buffer_start = (double*) mxMalloc( buffer_size *sizeof(double)) ;
  double* buffer_iterator = buffer_start ;
  double* buffer_end = buffer_start + buffer_size ;
```

EK-11 Siftrefinemx Fonksiyonu

siftrefinemx.c
Eşikleme ve kenardaki noktaların elenmesi yolu ile kararlı olmayan kilit noktaların ayıklanması

```
#include"mex.h"

#include<mexutils.c>

#include<stdlib.h>
#include<string.h>

#ifdef WINDOWS
#define DGESV dgesv
#undef min
#undef max
#else
#define DGESV dgesv_
#endif

#ifdef WINDOWS
#ifdef __cplusplus__
extern "C" {
   extern int DGESV(int *n, int *nrhs, double *a, int *lda,
                    int *ipiv, double *b, int *ldb, int *info) ;
}
#else
  extern int DGESV(int *n, int *nrhs, double *a, int *lda,
                   int *ipiv, double *b, int *ldb, int *info) ;
#define sqrtf(x) ((float)sqrt((double)x)
#define powf(x)  ((float)pow((double)x)
#define fabsf(x) ((float)fabs((double)x)
#endif
#else
extern int DGESV(int *n, int *nrhs, double *a, int *lda,
                 int *ipiv, double *b, int *ldb, int *info) ;
#endif

#define greater(a,b) ((a) > (b))
#define min(a,b) (((a)<(b))?(a):(b))
#define max(a,b) (((a)>(b))?(a):(b))

const int max_iter = 5 ;

void
mexFunction(int nout, mxArray *out[],
```

EK-11 (Devam) Siftrefinemx Fonksiyonu

```
          int nin, const mxArray *in[])
{
  int M,N,S,smin,K ;
  const int* dimensions ;
  const double* P_pt ;
  const double* D_pt ;
  double threshold = 0.01 ; /*0.02 ;*/
  double r = 10.0 ;
  double* result ;
  enum {IN_P=0,IN_D,IN_SMIN,IN_THRESHOLD,IN_R} ;
  enum {OUT_Q=0} ;

  /* -----------------------------------------------------------------Argümanlar kontrol edilir

  if (nin < 3) {
    mexErrMsgTxt("At least three input arguments required.");
  } else if (nout > 1) {
    mexErrMsgTxt("Too many output arguments.");
  }

  if( !uIsRealMatrix(in[IN_P],3,-1) ) {
    mexErrMsgTxt("P must be a 3xK real matrix") ;
  }

  if( !mxIsDouble(in[IN_D]) || mxGetNumberOfDimensions(in[IN_D]) != 3) {
    mexErrMsgTxt("G must be a three dimensional real array.") ;
  }

  if( !uIsRealScalar(in[IN_SMIN]) ) {
    mexErrMsgTxt("SMIN must be a real scalar.") ;
  }

  if(nin >= 4) {
    if(!uIsRealScalar(in[IN_THRESHOLD])) {
      mexErrMsgTxt("THRESHOLD must be a real scalar.") ;
    }
    threshold = *mxGetPr(in[IN_THRESHOLD]) ;
  }

  if(nin >= 5) {
    if(!uIsRealScalar(in[IN_R])) {
      mexErrMsgTxt("R must be a real scalar.") ;
    }
    r = *mxGetPr(in[IN_R]) ;
  }
```

EK- 12 Siftmatch Fonksiyonu

```
siftmatch.c
SIFT tanımlayıcılarının eşleştirilmesi
**/

#include"mexutils.c"

#include<stdlib.h>
#include<string.h>
#include<math.h>

#define greater(a,b) ((a) > (b))
#define min(a,b) (((a)<(b))?(a):(b))
#define max(a,b) (((a)>(b))?(a):(b))

#define TYPEOF_mxDOUBLE_CLASS double
#define TYPEOF_mxSINGLE_CLASS float
#define TYPEOF_mxINT8_CLASS   char
#define TYPEOF_mxUINT8_CLASS  unsigned char

#define PROMOTE_mxDOUBLE_CLASS double
#define PROMOTE_mxSINGLE_CLASS float
#define PROMOTE_mxINT8_CLASS   int
#define PROMOTE_mxUINT8_CLASS  int

#define MAXVAL_mxDOUBLE_CLASS mxGetInf()
#define MAXVAL_mxSINGLE_CLASS ((float)mxGetInf())
#define MAXVAL_mxINT8_CLASS   0x7fffffff
#define MAXVAL_mxUINT8_CLASS  0x7fffffff

typedef struct
{
  int k1 ;
  int k2 ;
  double score ;
} Pair ;

#define _COMPARE_TEMPLATE(MXC)                                  \
  Pair* compare_##MXC (Pair* pairs_iterator,                    \
                       const TYPEOF_##MXC * L1_pt,              \
                       const TYPEOF_##MXC * L2_pt,              \
                       int K1, int K2, int ND, float thresh)    \
  {                                                             \
    int k1, k2 ;                                                \
    const PROMOTE_##MXC maxval = MAXVAL_##MXC ;                 \
    for(k1 = 0 ; k1 < K1 ; ++k1, L1_pt += ND ) {                \
```

EK- 12 (Devam) Siftmatch Fonksiyonu

```
 if(nout > 1) {
       out[D] = mxCreateDoubleMatrix(1,
                                pairs_end-pairs_begin,
                                mxREAL) ;
       D_pt = mxGetPr(out[D]) ;
      }

      for(pairs_iterator = pairs_begin ;
          pairs_iterator < pairs_end  ;
          ++pairs_iterator) {
        *M_pt++ = pairs_iterator->k1 + 1 ;
        *M_pt++ = pairs_iterator->k2 + 1 ;
        if(nout > 1) {
          *D_pt++ = pairs_iterator->score ;
        }
      }
    }
    mxFree(pairs_begin) ;
  }
}
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ÖZGEN, Nazım
Uyruğu : T.C.
Doğum tarihi ve yeri : 22.08.1982 İstanbul
Medeni hali : Bekar
Telefon : 0 (312) 282 73 15
Faks : 0 (312) 282 73 16
e-mail : nzmzgn@gmail.com.

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Hacettepe Üniversitesi	2004
Lise	Kırklareli Anadolu Lisesi	2000

İş Deneyimi

Yıl	Yer	Görev
2005-	Hv.K.K.lığı	Subay

Yabancı Dil

İngilizce

Hobiler

Futbol, Basketbol, Satranç, Yüzme, Bilgisayar teknolojileri, Kitap, Gezi