GERÇEK ZAMAN GÖRÜNTÜLERDE HAREKETLİ NESNELERİN BULUNMASI VE TAKİP EDİLMESİ

Murat SÜRÜCÜ

Zonguldak Karaelmas Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalında
Yüksek Lisans Tezi
Olarak Hazırlanmıştır

ZONGULDAK
Şubat 2010

KABUL:

Murat Sürücü tarafından hazırlanan "GERÇEK ZAMAN GÖRÜNTÜLERDE HAREKETLİ NESNELERİN BULUNMASI VE TAKİP EDİLMESİ" başlıklı bu çalışma jürimiz tarafından değerlendirilerek, Zonguldak Karaelmas Üniversitesi, Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalında Yüksek Lisans Tezi olarak oybirliğiyle kabul edilmiştir. 03/02/2010

Başkan: Yrd. Doç. Dr. Rıfat HACIOĞLU (ZKÜ) .....

Üye: Yrd. Doç. Dr. Zehra SARAÇ (ZKÜ) .....

Üye: Doç. Dr. Şenol Hakan KUTOĞLU (ZKÜ) .....

ONAY:

Yukarıdaki imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım. 9../.3/2010

Prof. Dr. Kemal BÜYÜKGÜZEL
Fen Bilimleri Enstitüsü Müdürü

*"**Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim.**"*

Murat SÜRÜCÜ

ÖZET

Yüksek Lisans Tezi

GERÇEK ZAMAN GÖRÜNTÜLERDE HAREKETLİ NESNELERİN BULUNMASI VE TAKİP EDİLMESİ

Murat SÜRÜCÜ

Zonguldak Karaelmas Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı

Tez Danışmanı: Yrd. Doç. Dr. Rıfat HACIOĞLU
Şubat 2010, 63 sayfa

Bu çalışmada gerçek zamanlı görüntülerde hareketli nesne takip metotları incelenmiş, hareketli kamera donanımı ve yazılımı gerçeklenmiştir. Görüntü iyileştirme ve gürültü yok etme metotlarını da içeren hareketli nesne tespit ve takip işlemleriyle pan/tilt motorlara sahip aktif kamera içeren donanım birleştirilerek entegre bir sistem oluşturulmuştur. Görüntü işleme algoritmaları C# dilinde doğrudan hafızaya erişim metotlarıyla yazılmıştır. Kameranın hareketinden sorumlu servo motorların hareketinin kontrolü, USB arayüzlü PIC 18F2550 mikrodenetleyicisi ile sağlanmıştır. Bilgisayarda çalışan görüntü işleme programları, hareketli nesneyi tespit edip, USB veri yolu üzerinden mikrodenetleyici ile haberleşerek kameranın konumunu belirlemektedir. Hareketli nesne takibi işlemi, bu amaç için kullanılan çerçeve farkı yönteminin iki yeni yaklaşıma uyarlanmasıyla yapılmıştır. Bu yaklaşımlar gecikmeli çerçeve farkı yöntemi ve köşe tespitli çerçeve farkı yöntemidir. Hareketli nesne tespiti için yapılan çerçeve farkı alma işlemi, birinci yaklaşımda belirli bir gecikme sonrası, sonraki yaklaşımda ise köşe tespit ve eşleştirme işlemleri sonrası yapılmaktadır. Son olarak gecikmeli

çerçeve farkı yöntemi ve köşe tespitli çerçeve farkı yöntemi ile yapılan hareketli nesne takibi performansı karşılaştırılmıştır.

Anahtar Sözcükler: Görüntü İşleme, Gürültü Azaltma, PIC, USB, Pan/Tilt, Aktif Kamera, Hareketli Nesne Takibi, Çerçeve Farkı, Köşe Belirleme, Köşe Uyumlama, Köşe Takibi, Gecikmeli Çerçeve Farkı, Köşe Tespitli Çerçeve Farkı

Bilim Kodu: 609.03.02

ABSTRACT

M.Sc. Thesis

DETECTING AND TRACKING MOVING OBJECTS IN REAL TIME IMAGES

Murat SÜRÜCÜ

Zonguldak Karaelmas University
Graduate School of Natural and Applied Sciences
Department of Electrical and Electronics Engineering

Thesis Advisor: Asst. Prof. Rıfat HACIOĞLU
February 2010, 63 pages

In this study, the possible methodologies for target tracking techniques using real-time images have been investigated, and the required hardware for the motional video camera system including its software has been realized. Herein, an integrated system relied on image enhancement and noise reduction is built combining the moving object detection and tracking processes together with a hardware composed of an active camera with pan/tilt motors. Image processing algorithms are written in C# language using direct memory access technique. The control of servo motors, which are responsible for video camera movements are provided by PIC18F2550 microcontroller with an USB interface. The image processing software running on pc detects the moving object, communicate with microprocessor through USB data bus and define the position of the video camera. Moving object tracking process is achieved by adaptation of frame difference technique on two new approaches. These approaches are namely: delayed frame difference and edge detected frame difference approaches. Frame difference processes for moving object detection are accomplished after a specific delay for the first approach and after the edge detection and matching processes for the second approach. Lastly a performance analysis has been performed on moving object tracking

considering the delayed frame difference and the corner detected frame difference approaches, and the results are compared.

Key Words: Image Processing, Noise Reduction, PIC, USB, Pan/Tilt, Active Camera, Moving Object Tracking, Frame Difference, Corner Detection, Corner Matching, Corner Tracking, Delayed Frame Difference, Corner Detected Frame Difference

Science Code: 609.03.02

TEŞEKKÜR

Yüksek lisans çalışmamın oluşumu aşamasında destek ve yardımlarını esirgemeyen değerli hocam Yrd. Doç. Dr. Rıfat HACIOĞLU'na (ZKÜ), tezimin değerlendirilme aşamasında katkıda bulunan jüri üyesi hocalarım Yrd. Doç. Dr. Zehra SARAÇ (ZKÜ) ve Doç. Dr. Şenol Hakan KUTOĞLU'na (ZKÜ), tezin her safhasında eleştirileriyle, önerileriyle, fikirleriyle her zaman ve her konuda manevi desteğini esirgemeyen, eşim Okutman Dilek SÜRÜCÜ'ye (ZKÜ), emeği geçen tüm bölüm hocalarıma, her daim yanımda olan değerli dostlarıma, anneme ve babama teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

KABUL ii
ÖZET iii
ABSTRACT v
TEŞEKKÜR vii
İÇİNDEKİLER ix
ŞEKİLLER DİZİNİ xiii
ÇİZELGELER DİZİNİ xv
KISALTMALAR DİZİNİ xvii

BÖLÜM 1 GİRİŞ 1

BÖLÜM 2 GÖRÜNTÜ İŞLEMEDE KULLANILAN YÖNTEMLER 5

2.1 GÖRÜNTÜ İŞLEMEYE GİRİŞ 5
2.1.1 Sayısal Görüntü 6
2.1.2 Sayısal Görüntüde Gürültü 7
2.1.3 Hareket Algılama ve Arkaplan Belirleme 10
2.2 GÜRÜLTÜ YOK ETME METOTLARI 12
2.2.1 Doğrusal Filtreleme 13
2.2.1.1 Ortalama Filtre (Average Filter) 14
2.2.1.2 Gauss Filtre 15
2.2.2 Doğrusal Olmayan Filtreleme 16
2.2.3 Gürültü Filtresi Örnek Çıktıları 17
2.3 TEMEL GÖRÜNTÜ İŞLEME TEKNİKLERİ 18
2.3.1 Aşındırma (Erosion) 18
2.3.2 Genleştirme (Dilation) 19
2.3.3 Görüntü Keskinleştirme 20
2.4 NESNE TANIMLAMA İŞLEMLERİ 21

İÇİNDEKİLER (devam ediyor)

Sayfa

2.4.1 Kenar Tanımlama 21
2.4.2 Köşe Tanımlama 23
2.5 ARKAPLAN BELİRLEME YÖNTEMLERİ 24
2.5.1 Sabit Arkaplan 25
2.5.2 Çerçeve Farkı Yöntemi 26
2.5.3 Akan Ortalama Yöntemi 27
2.5.4 Akan Gauss Ortalama Yöntemi 29
2.5.5 Gauss Dağılımlarının Birleşimi Yöntemi 30
2.5.6 Arkaplan Belirleme Yöntemlerinin Başarımları 31
2.6 AKTİF KAMERA KALİBRASYONU 32
2.7 ÇERÇEVE FARKI TEMELLİ HAREKETLİ NESNE TAKİP YAKLAŞIMLARI 35
2.7.1 Gecikmeli Çerçeve Farkı Yöntemi 35
2.7.2 Köşe Tespitli Çerçeve Farkı Yöntemi 35

BÖLÜM 3 DONANIM 37

3.1 DONANIMIN SEÇİMİ 37
3.2 MİKRODENETLEYİCİ 37
3.2.1 Mikrodenetleyicide Kullanılan Programlama Dili 40
3.2.2 Mikrodenetleyici USB Arayüzü 40
3.3 SERVO MOTOR 41
3.4 KULLANILAN DONANIM 42

BÖLÜM 4 YAZILIM 45

4.1 SERVO MOTOR SÜRÜCÜ DEVRESİ İLE HABERLEŞME 45
4.2 USB KAMERADAN GÖRÜNTÜ ALMA 47
4.3 GÖRÜNTÜ İŞLEME YAZILIMI 48
4.3.1 Gecikmeli Çerçeve Farkı Yöntemine Dayalı Algoritma 51
4.3.2 Köşe Tespitli Çerçeve Farkına Dayalı Algoritma 52

BÖLÜM 5 HAREKETLİ NESNE TAKİP ALGORİTMALARI ANALİZİ 55

İÇİNDEKİLER (devam ediyor)

Sayfa

5.1 TEST ORTAMI 55
5.2 ANALİZ SONUÇLARI 55

BÖLÜM 6 SONUÇLAR VE ÖNERİLER 59

KAYNAKLAR 61
ÖZGEÇMİŞ 63

ŞEKİLLER DİZİNİ

No		Sayfa
2.1	Piksel komşuluklarının gösterimi.	6
2.2	Lena fotoğrafından alınan kesitler ve fotoğrafın orijinal hali.	9
2.3	Lena fotoğrafından alınan gürültülü ve gürültüsüz kesitlerin yüzeysel görünümü	10
2.4	Hareket algılama.	11
2.5	Gürültü filtreleme.	12
2.6	mxm boyutundaki örnek h matrisi.	13
2.7	Örnek konvolüsyon işlemi.	14
2.8	Sabit katsayılı ortalama filtrenin etkisi.	14
2.9	Gauss dağılımlı filtrenin katsayılarının temsili.	15
2.10	Gauss fonksiyonunun frekans bölgesindeki karşılığı.	16
2.11	Median (orta değer) filtre örneği.	16
2.12	Gauss gürültü için filtre çıktıları.	17
2.13	Tuz-Biber gürültü için filtre çıktıları	18
2.14	Aşındırma örneği.	19
2.15	Genleştirme örneği.	19
2.16	Görüntü keskinleştirme algoritması.	20
2.17	Kenar tanımlama algoritması.	21
2.18	Köşe tanımlama metodu ile örnek uygulama çıktısı.	24
2.19	Sabit arkaplan yöntemi ile hareketli nesne tespiti.	25
2.20	Çerçeve farkı yöntemi ile hareketli nesne tespiti.	27
2.21	Akan ortalama yöntemi ile hareketli nesne tespiti.	28
2.22	Akan Gauss ortalama yöntemi ile hareketli nesne tespiti.	29
2.23	Gauss dağılımlarının birleşimi yöntemi ile hareketli nesne tespiti.	31
2.24	Lens bozulmaları ihmal edilmiş kamera modeli.	33
2.25	Kamera kalibrasyonu ile hareketli kamerada hareketli nesne tespiti.	35
3.1	PIC 18F2550'ye ait bacak bağlantısı.	39
3.2	Servo motorlara ait düzenek.	41
3.3	Kullanılan donanımın blok şeması.	43

ŞEKİLLER DİZİNİ (devam ediyor)

No		Sayfa
3.4	Aktif kamera donanımı.	43
4.1	Servo motor sürücü kartı ile haberleşme algoritması.	46
4.2	USB kameradan görüntü alan programın algoritması.	47
4.3	Bazı basit görüntü işleme filtre çıktıları.	50
4.4	Gecikmeli çerçeve farkı yöntemine dayalı algoritma.	51
4.5	Köşe izleme uygulaması.	52
4.6	Köşe tespitli çerçeve farkı yöntemine dayalı algoritma.	53
5.1	Normal hızda yürüme için uygulama çıktıları.	56
5.2	Yüksek hızda yürüme için uygulama çıktıları.	57

ÇİZELGELER DİZİNİ

No		Sayfa
3.1	PIC 18F2550'ye ait teknik özellikler.	39
3.2	Proton Plus tarafından desteklenen bazı komutlar.	40
3.3	Kullanılan Servo motorların teknik özellikleri.	42
5.1	Hareketli nesne takip sonuçları.	58

KISALTMALAR DİZİNİ

ADC	:	Analog Sayısal Dönüştürücü (Analog Digital Converter)
API	:	Yazılım Programlama Arayüzü (Application Programming Interface)
CDC	:	İletişim Aygıtları Sınıfı (Communications Device Class)
CISC	:	Karmaşık Komut Setli Bilgisayar (Complex Instruction Set Computer)
CMYK	:	Turkuaz Kırmızı Sarı Siyah (Cyan Magenta Yellow Key (black))
DMA	:	Doğrudan Hafıza Erişimi (Direct Memory Access)
GÇFY	:	Gecikmeli Çerçeve Farkı Yöntemi
GUID	:	Evrensel Eşsiz Tanımlama Bilgisi (Globally Unique ID)
HID	:	İnsan Arayüz Aygıtları (Human Interface Devices)
KTÇFY	:	Köşe Tespitli Çerçeve Farkı Yöntemi
MIPS	:	Saniyedeki Milyon İşlem (Million Instructions Per Second)
PDIP	:	Plastik Çift Sıralı Paket (Plastic Dual-In-line Package)
PIC	:	Çevresel Arabirim Denetleyicisi (Peripheral Interface Controller)
PID	:	Ürün Tanımlama Bilgisi (Product ID)
PWM	:	Darbe Genişliği Modülasyonu (Pulse Width Modulation)
RGB	:	Kırmızı Yeşil Mavi (Red Green Blue)
RISC	:	İndirgenmiş Komut Setli Bilgisayar (Reduced Instruction Set Computer)
USB	:	Evrensel Seri Yol (Universal Serial Bus)
VID	:	Sağlayıcı Tanımlama Bilgisi (Vendor ID)

BÖLÜM 1

GİRİŞ

Görüntü işleme, teknolojinin ilerlemesine paralel şekilde hayatımızda daha önemli hale gelmekte ve gün geçtikçe daha çok uygulama şansı bulmaktadır. İnsanların hizmetine sunulan hemen her cihazda görüntü işleme teknolojisi kullanımı mümkün gözükmektedir. İnsanlığa hizmet eden teknolojik her ürünün insanoğluyla ve hatta birbiriyle görsel olarak da haberleşebilmesi görüntü işleme algoritmaları sayesinde mümkün olmaktadır. Bu çalışmada, görüntü işleme yöntemleri kullanılarak gerçek zaman görüntülerde hareketli nesne tespit ve takibi üzerinde durulmaktadır.

Günümüz görüntü işleme algoritmaları birçok uygulama alanı bulmaktadır. Örneğin marketten alınan aynı boy ve aynı tip gözüken elmaların tasnifinde, elmaları büyüklüğüne, rengine ve desenine göre ayıran temel görüntü işleme algoritmaları yıllardır kullanılmaktadır. Savaşlarda sıklıkla kullanılan akıllı füze olarak bildiğimiz silahlarda da GPS (Küresel Konumlama Sistemi – Global Positioning System) verisinin yanı sıra bombanın patlayacağı yerin görüntüsü işlenerek yanlış hedeflerin vurulması da önlenmektedir. Trafik denetleme birimlerinde ise geçiş noktalarındaki trafik yoğunluğundan, hatalı araç kullanımında plaka tespitine kadar pek çok uygulama alanında görüntü işleme kullanılmaktadır. Bu uygulamalarda hareketli nesnelerin tespit edilmesi ve takip edilmesi problemi üzerinde çalışılmaktadır.

Hareketli nesne takibi üzerine yapılan görüntü işleme çalışmalarında sabit ve hareketli kameralar ile farklı metotlar kullanılmaktadır. Özellikle hareketli görüntülerdeki arkaplan tespitine yönelik çalışmalar üzerinde durulmaktadır. Sabit kameraya sahip uygulamalarda arkaplan çok fazla değişime uğramadığı için sabit arkaplan tespitine yönelik metotlar, hareketli kamerada ise arkaplan sürekli değiştiği için daha karmaşık olabilecek hareketli arkaplan tespitine yönelik metotlar kullanmak gerekmektedir.

Piccardi'nin (2004) bahsettiği gibi sabit kameralardaki hareketli nesne tespitindeki problemler genellikle ışık değişimleri, kamera titreşimleri ve ağaç yaprakları gibi hızlı değişen arkaplan nesnelerinden kaynaklanmaktadır. Bu çalışmada tanımlanan sabit arkaplan, çerçeve farkı, akan ortalama, akan Gauss ortalama ve Gauss dağılımlarının birleşimi arkaplan belirleme metotları bu tezde incelenmiştir. Sabit kamera ile en başarılı sonuç Stauffer and Grimson (1999) tarafından da uygulanan Gauss Dağılımlarının Birleşimi metodunda elde edilmiştir. Çünkü Gauss dağılımlarının birleşimi metodunda, her piksele ait birden çok arkaplan modeli oluşmaktadır. Örneğin sahnede yer alan bir ağacın yapraklarının izlediği her "yaprak var"/ "yaprak yok" hareketi için ayrı bir arkaplan modeli seçildiği için yaprak hareketleri hareketli nesne olarak algılanmamaktadır. Kamera hareketli olmadığı için arkaplan seçilecek çerçeve belirleme işlemi için ayrıca bir işlem gücü gereksinimi oluşmamaktadır. Bu sebeple tüm işlem gücü, hareketli nesne tespiti işleminde kullanılabilmektedir.

Hareketli kameralarda ise sabit kameralarda karşılaşılan problemlere ilave olarak arkaplandaki tüm objelerin de takip etmek istediğimiz nesne gibi hareket ediyor oluşudur. Kameranın hareketinden doğan arkaplan kaymasını düzeltmek için çeşitli çalışmalar yapılmıştır. Bunlardan Basu'nun (1995) önerdiği metot kameranın ne kadar hareket ettiğini kesin olarak bilmeye dayanmaktadır. Eğer kameranın taradığı açının büyüklüğünü bilirsek, geometrik olarak arkaplanın bir sonraki konumda oluşturacağı kompozisyonuda tahmin edebiliriz. Kameranın fiziksel pozisyonunu öğrenme şansının olmadığı, örneğin robot üzerinde bulunan kamera gibi uygulamalarda, görüntüdeki nesnelere ait bazı özelliklerin takip edilmesi gerekmektedir. Bu bağlamda nesnelerin izdüşüm şekilleri (Loy and Barnes 2004), nesnelerin kenarları (Kim et al 2005), nesnelerin renkleri (Zuo and de With 2005) veya nesnelerin köşe noktalarına göre (Yu et al 2008) çeşitli çalışmalar yapılmıştır.

Bu tez çalışmasında, sabit kameralarda kullanılan çerçeve farkı yöntemi hareketli kameraya gecikmeli çerçeve farkı yöntemi (GÇFY) ve köşe tespitli çerçeve farkı yöntemi (KTÇFY) algoritmalarıyla uyarlanması ve bu iki algoritmanın başarımları arasında karşılaştırma yapılması üzerinde durulmaktadır. Bu amaçla hem görüntü işleme yazılımlarının hazırlanması hem de aktif kamera (Pan – yatay hareketli/Tilt – düşey hareketli motora sahip) donanımı tasarımı hedeflenmektedir.

Hedeflenen yazılım ve donanımla elde edilen tümleşik sistemin birçok uygulamada

kullanılması mümkündür. Hedef uygulamalar konferans ve seminerlerin uzak mesafelere gerçek zamanlı aktarılması, farklı kampuslardaki öğrencilerin tek bir merkezden dersleri izlemesi gibi uygulamalarda aktif sunum takibinde eğitim amaçlı kullanımı da içermekle birlikte robot görüşü, güvenlik sistemleri, askeri hedef takibi, sportif aktivitelerin izlenmesi uygulamalarını da kapsamaktadır.

Tezin ikinci kısmında hareketli nesne tespiti ve takibi uygulamalarında kullanılan yöntemler anlatılarak görüntü iyileştirme yöntemleri incelenmektedir. Görüntüdeki gürültü çeşitlerine göre değişik metotların başarımları gösterilip sabit kameralar için kullanılan arkaplan belirleme yöntemleri incelenmiş, çerçeve farkı yönteminin hareketli kameralar için uyarlaması olan GÇFY ve KTÇFY algoritmaları anlatılmaktadır. Üçüncü bölümde, uygulamanın yapıldığı donanım üzerinde durularak, donanımı oluşturan bileşenler hakkında bilgi verilmektedir. Dördüncü bölümde ise yazılan görüntü işleme uygulaması hakkında bilgi verilmektedir ki burada hazırlanan program anlatılmakta, kullanılan metotlar açıklanmaktadır. Beşinci bölümde incelenen iki farklı metodun analiz sonuçları üzerinde durularak, metotların başarımları ve karmaşıklığına göre performansları değerlendirilmektedir. Altıncı bölümde çalışmanın sonuçları hakkında değerlendirme yapılarak gelecek çalışmalar ve öneriler vurgulanmaktadır.

BÖLÜM 2

GÖRÜNTÜ İŞLEMEDE KULLANILAN YÖNTEMLER

Bu bölümde, görüntü işlemede kullanılan temel işlemler, görüntüde bulunan gürültüyü yok etme metotları, hareket tespiti ve nesne belirleme metotları üzerinde durulacaktır. Temel görüntü işleme metotları, her türlü görüntü işleme uygulamasında kullanılan en basit işlemlerden oluşmaktadır ki bu metotlar, işlenilen görüntü üzerinde kullanılan iyileştirme ve kesitleme işlemleri gibi ön hazırlık aşaması metotlarıdır. Gürültü yok etme metotları ise işlenecek görüntülerin çoğu zaman saf görüntü olmaması sebebi ile neredeyse tüm görüntü işleme uygulamalarında, görüntü işleme başarımını arttırmak için kullanılmaktadır. Hareket tespiti ve nesne belirleme işlemleri ise uygulamaya özel işlemler olup, bu çalışmanın temel konusunu oluşturmaktadır.

2.1 GÖRÜNTÜ İŞLEMEYE GİRİŞ

Görüntü işleme, verinin görüntüden oluştuğu sayısal işaret işleme yaklaşımlarıdır. Çoğu görüntü işleme tekniği, görüntüyü iki boyutlu sinyal kabul ederek standart işaret işleme tekniklerinin bu işarete uygulanmasıyla geliştirilmiştir (Kang 2007).

Günümüzde görüntü işleme, sabit bir görüntünün ayrıntılı analizinden (bilgisayarlı tomografi) hareketli görüntüdeki istenilen nesneleri takip etmeye (akıllı füzeler) kadar birçok amaca hizmet etmektedir. Gün geçtikçe daha performanslı ve daha tutarlı algoritmalar geliştirilmektedir. İşlemci hızlarının da artmasıyla görüntü işleme uygulamaları hayatımıza daha çok etki etmektedir.

Bütün bu gelişmelere ve ilerlemelere rağmen insan algılaması halen en karmaşık görüntü işleme becerisine sahiptir. Görüntü işleme ve robotik alanındaki iyileştirmelerin çoğu, insan algılamasına yetişebilme amacıyla yapılmaktadır. Ancak halen insan algılamasının karmaşıklığına yaklaşılamamıştır.

Sabit veya hareketli bir görüntüdeki bir nesneyi takip etmek için bilgisayar aracılığıyla işlerken çeşitli adımları uygulamamız gerekmektedir. Bu adımlar en genel anlamda;

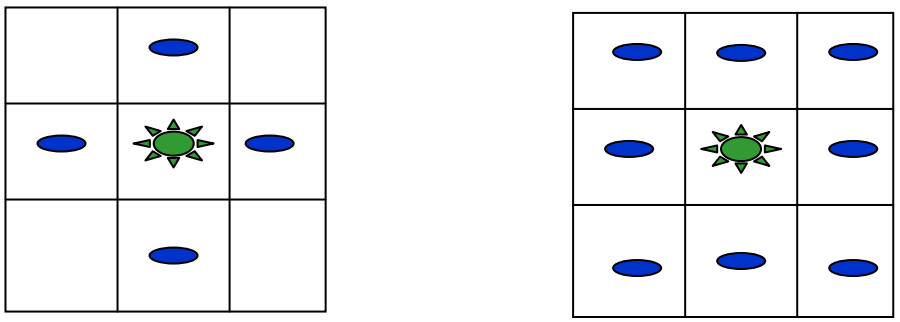
- Görüntüyü sayısala çevirme ve bilgisayara aktarma,
- Gürültüyü azaltma,
- Görüntüyü iyileştirme,
- Görüntüdeki anlamlı nesneyi işleme,
- İşlenmiş görüntü sonucu gerekli işlemleri gerçekleştirme

işlemlerini içerir.

2.1.1 Sayısal Görüntü

Sayısal görüntüler aslında minimum ve maksimum değerleri sınırlandırılmış matrissel sayılar kümesidir. Matristeki her satır ve sütunun kesiştiği noktadaki sayıya/sayılara karşılık, görüntümüzde bir nokta denk gelmektedir (bu noktalara piksel diyeceğiz). Tüm bu pikseller komşuluklara sahiptir.

Hesaplamalarımızda referans alacağımız bazı noktalar olacaktır. Bu referanslarımız Şekil 2.1'de gösterildiği gibi ilgilendiğimiz piksel etrafında olan 4'lü veya 8'li komşulukları olabileceği gibi, görüntünün tümü ya da görüntü üzerindeki bir nesne de olabilir. Bu referans seçimi, yapacağımız işleme göre değişiklik göstermektedir.



a. 4'lü komşuluk b. 8'li komşuluk

Şekil 2.1 Piksel komşuluklarının gösterimi.

Oldukça fazla sayıda sayısal görüntü standardı vardır. En çok Gri Tonlama, RGB (Kırmızı

Yeşil Mavi – Red Green Blue) ve CMYK (Turkuaz Kırmızı Sarı Siyah - Cyan Magenta Yellow Key (black)) standartları kullanılmaktadır (Bovik 1999). Biz işlemlerimizde hem Gri Tonlama, hem de RGB standardını kullanacağız. Gri Tonlamada pikseller grinin tonları olarak ifade edilirken RGB gösterimde ise her piksel 3 farklı rengin, kırmızı, yeşil ve mavinin, birleşiminden oluşmaktadır.

RGB'den Gri Tonlama'ya çeşitli dönüşüm şekilleri olsa da en doğru sonuçları aşağıdaki eşitlik vermektedir.

$$GRI = (11 * B + 59 * G + 30 * R)/100 \quad (2.1a)$$

$$GRI = (28 * B + 151 * G + 77 * R)/256 \quad (2.1b)$$

(2.1a)'daki eşitlik daha doğru sonuç verse de (2.1b)'deki eşitlik bilgisayarla hesaplama yaparken ikili sayı sistemi kullanıldığı için daha hızlı sonuç vermektedir. Daha hızlı sonuç için 256'ya bölme işlemi yerine toplam sonucu 8 bit sağa kaydırma işlemi kullanılmalıdır.

2.1.2 Sayısal Görüntüde Gürültü

Gürültü, işarette ilgilendiğimiz veri dışında kalan tüm verileri kapsasa da biz sadece ilgilendiğimiz veriyi bozan işaretleri gürültü olarak tanımlayacağız.

Görüntüde gürültü pek çok sebeple oluşabilmektedir ve birçok türü vardır. Örneklendirmek gerekirse;

- Sayısal görüntü sensörleri, görüntüyü üzerlerine düşen fotonları sayarak oluştururlar. Bu yüzden sayısal görüntülerde sıklıkla foton sayma gürültüsü vardır.
- Grain gürültüsü, ışığa duyarlı filmlerin (negatiflerin) oluşturduğu gürültü türüdür. Kimi zaman Gauss dağılımlıdır, kimi zaman ise Poisson dağılımlıdır.
- Sayısal sensör duyarlılığının büyüklüğü ve sensörün ısınmasının etki ettiği gürültüye "Kumlanma" denir. Sensörün uçlarına verilen polarma arttıkça sensörün duyarlılığı ve hızı artar. Ancak sensörün oluşturduğu sayma hataları

ve ısıdan dolayı oluşan termal gürültülerde artar. Filmli cihazlardaki "grain" gürültüsü gibidir. Ancak buradaki gürültü Gauss dağılımlıdır.

- Çoğu görüntüde az da olsa "tuz-biber" gürültüsü mevcuttur. Bu gürültü türünde, görüntümüzün üzerine serpilmiş siyah (minimum) ve beyaz (maksimum) noktacıklar vardır.
- Sayısal sistemlerin tümünde mevcut olan bir diğer gürültü türü ise kuantalama gürültüsüdür.
- Optik arabirimlerin oluşturduğu gürültülerde çok çeşitlidir. Renk sapıncı, küresel sapınç, koma, astigmat, alan eğriliği, alan bozulması ve non-lineerite başlıca gürültü sebepleridir.

Gürültüyü formülüze edebilmek için ilk olarak bozucu, bozulmuş görüntü ve saf görüntümüz (bozucu içermediğini varsaydığımız) arasında bir bağıntı kurmamız (modelleme yapmamız) gerekmektedir. Bu amaçla bağıntımızda kullanacağımız büyüklükleri tanımlayalım.

Bozulmuş görüntüyü *f(.)*, saf görüntüyü *g(.)* ve gürültüyü *q(.)* ile gösterirsek iki farklı model oluşturabiliriz.

$$f(.) = g(.) + q(.) \tag{2.2}$$

$$f(.) = g(.) \times q(.) \tag{2.3}$$

$$f(.) = g(.) + q(.) \quad \Rightarrow \quad e^{f(.)} = e^{g(.)+q(.)} = e^{g(.)} \times e^{q(.)} \quad \Rightarrow \quad f'(.) = g'(.) \times q'(.) \tag{2.4}$$

$$f(.) = g(.) \times q(.) \quad \Rightarrow \quad \log f(.) = \log g(.) \times q(.) = \log g(.) + \log q(.) \quad \Rightarrow \quad f'(.) = g'(.) + q'(.) \tag{2.5}$$

Eşitlik (2.2) gürültüyü toplamsal gürültü olarak modellemede kullanılırken, Eşitlik (2.3) ise gürültüyü çarpımsal gürültü olarak tanımlamada kullanılır. Bu iki eşitlik birbiri cinsinden yazılabilse de (Eşitlik (2.4) ve (2.5)) sistemimizi tanımlarken "en basit" modeli bulmaya çalıştığımız için uygun bir gürültü modeli seçilmelidir.

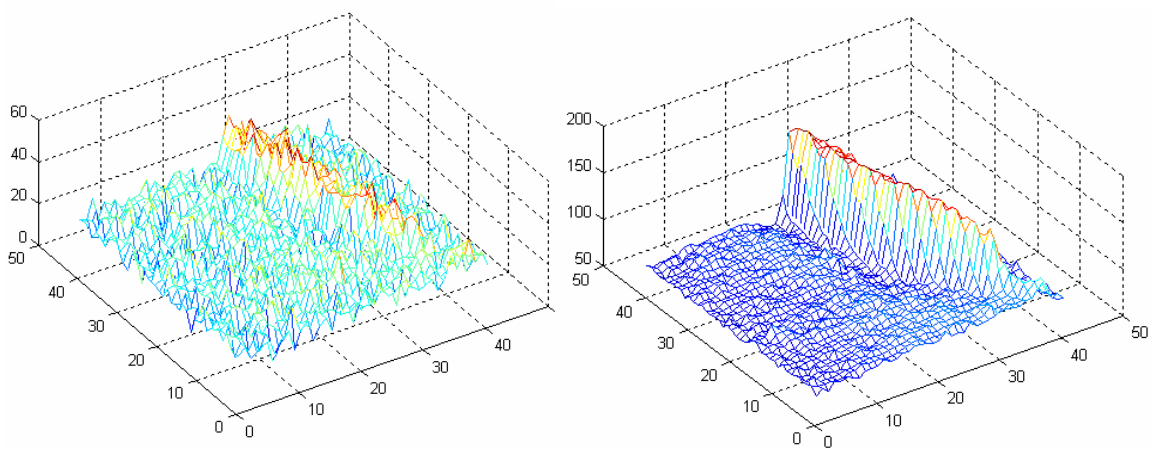
Genellikle gürültü ile görüntü birbirinden bağımsız iki fonksiyon ile tanımlanabiliyor ise

modelimizi toplamsal gürültü modeli seçeriz. Termal gürültü, kuantalama gürültüsü, kumlanma gibi gürültüler bu tür gürültülerdir. Görüntü ile gürültü arasında bir bağıntı var ise, bu sefer modelimizi çarpımsal gürültü modeli seçmemiz daha mantıklıdır. Bu tarz gürültülere örnek olarak, belirli tayflarda oluşan ve görüntüye göre oranı değişen karıncalanma gürültüsünü verebiliriz. Bazı gürültüler ise iki modelle de tam olarak modellenememektedir. Çünkü bazı gürültüler ne toplamsal modellemeye girer, ne de çarpımsal modellemeye. Örneğin "tuz-biber" gürültüsü ve Poisson dağılımlı sayma gürültüsü bu tarz gürültülerdir.

Gürültüyü yok etmeye çalışırken de ilk bakmamız gereken olgu, gürültü ile saf görüntümüz arasında bir bağımlılığın olup olmadığıdır. Çünkü gürültümüz ile saf görüntümüz (g ile q) bağımlı ise gürültüyü asla tamamen yok edemeyiz.

Şekil 2.2 Lena fotoğrafından alınan kesitler ve fotoğrafın orijinal hali.

Şekil 2.2'de Orijinal Lena fotoğrafı ve gürültü eklenmiş kesiti görülmektedir. Bu 2 boyutlu gösterimi gri renk bilgisini derinlik bilgisi olarak ele alıp 3 boyutlu modellersek Şekil 2.3'de görüldüğü gibi gürültünün etkisini daha iyi algılayabiliriz.



Şekil 2.3 Lena fotoğrafından alınan gürültülü ve gürültüsüz kesitlerin yüzeysel görünümü

2.1.3 Hareket Algılama ve Arkaplan Belirleme

Arkaplan ya da diğer kullanımıyla artalan, bir görüntüde ilgilenilen nesne veya nesnelerin dışında kalan alanı tanımlar. Hareketli nesne veya görüntünün ilgilenilen kısımları ise önalan olarak adlandırılır.

Hareket algılama, görüntü işleme alanında sıklıkla başvurulan yöntemlerden biridir. Hareketi algılama ve algılanan hareketli nesneyi takip etme birlikte kullanılabildiği gibi bazı uygulamalarda sadece hareketi algılama üzerine kurulmuştur.

Hareket algılama, sahnedeki belirgin değişimleri takip ederek yapılır. Bu değişimler nesne veya piksel tabanlı olabilir. Hareketli nesne tespiti için hareketsiz olduğu düşünülen veya kabul edilen arkaplanı seçmek gerekmektedir. Çünkü arkaplanı biliyorsak oluşan değişimler ya gürültü ya da hareketli nesnemiz olmak zorundadır.

Sabit bir arkaplanımız var ise hareketi ve hareketli nesneleri takip etmek oldukça kolay olacaktır. Örneğin Şekil 2.4a'daki gibi arkaplana sahip bir kameradan görüntüleri aldığımızı düşündüğümüzde Şekil 2.4b'de görüldüğü gibi hareketli bir nesne kameranın önünden geçtiğinde kolaylıkla hareketi algılayabiliriz (Şekil 2.4c). Ancak hareketli nesnenin tamamını algılayabilmek için ileride anlatılacak olan metotları kullanmak gereklidir.

a. Sabit arkaplan görüntüsü.

b. Hareketli nesnenin görüntüsü.

c. Tespit edilen hareketli nesne.

Şekil 2.4 Hareket algılama.

Uygulamaların çoğunda ne yazık ki sabit bir arkaplan bulunmamaktadır. Arkaplana sonradan dahil olan nesneler olabildiği, arkaplandaki nesnelerin yerlerinin değişebildiği ve bütün bunlara ilave olarak kameranın pozisyonunun da değişebildiği düşünülürse genellikle bu basit metot, görüntü işleme uygulamalarında kullanılmamaktadır. Burada örneklendirilen basit yapı ve daha karmaşık olan arkaplan belirleme metotları ileriki bölümlerde ayrıntılı anlatılacaktır.

2.2 GÜRÜLTÜ YOK ETME METOTLARI

Gürültüyü görüntüden ayırmak, orijinal görüntümüzü tahmin etmek anlamına gelmektedir, ayrıca bu işleme gürültüyü filtrelemek de diyebiliriz. Bu yüzden uygulamalarda hem "tahmin edici", hem de "filtre" tanımı kullanılabilmektedir. Bu çalışmada yaptığımız gürültü yok etme işleminden bahsederken "filtre" tanımını kullanacağız. Şekil 2.5'deki gürültü yok etme filtresini göz önüne alırsak, çıkıştaki *o(.)* fonksiyonu ile girişteki *f(.)* fonksiyonunun içinde bulunan gürültüsüz *g(.)* fonksiyonu birbirine olabildiğince yakın olmalıdır.

Gürültüyü filtrelemek için iki farklı filtreleme tipi vardır;

- Doğrusal Filtreleme
- Doğrusal Olmayan Filtreleme

Buradaki doğrusal kelimesi, filtreleme işleminde komşu piksellerin sahip olduğu değerlerin doğrusal birleşimlerinin kullanıldığını tanımlamaktadır.

Şekil 2.5 Gürültü filtreleme.

2.2.1 Doğrusal Filtreleme

Doğrusal filtrelemede çıkışın her pikseli, giriş görüntüsünün karşılığı olan pikselin ve komşuluklarının belirli ve sabit bir katsayı matrisi ile ağırlıklı çarpımlarının toplamlarından oluşmaktadır. Çıkış görüntüsü, giriş görüntüsünün lineer bir fonksiyonudur. Süperpozisyon ve kaydırma işlemlerinde lineer tepki verir.

Doğrusal filtreleme için Şekil 2.5'de yapılan işlemi;

- $o(x,y) = (g(x,y) + q(x,y)) * h(x,y)$ zaman (konum) bölgesinde
- $O(u,v) = (G(u,v) + Q(u,v)) \, . \, H(u,v)$ frekans bölgesinde

şeklinde yazabiliriz.

Burada $h(x,y)$ ile tanımladığımız filtre, aslında mxm boyutunda (m tek sayı) bir matristir (Şekil 2.6).

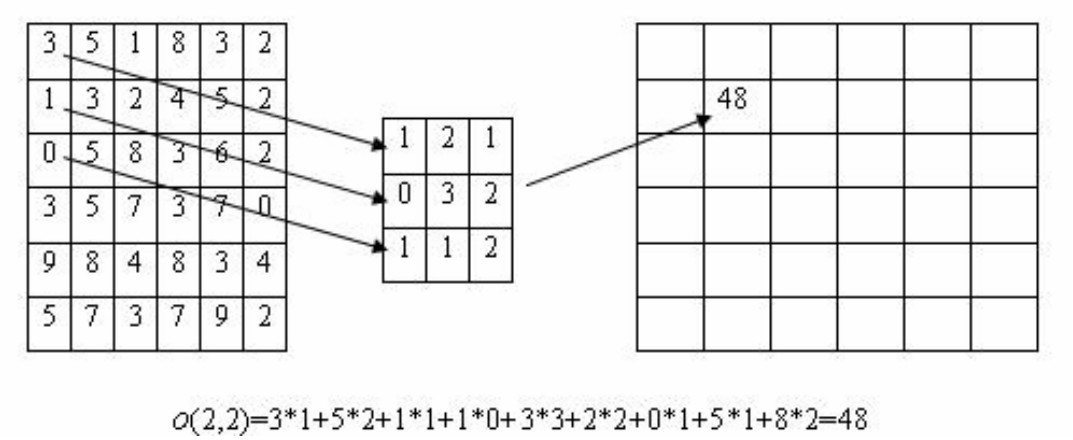
$$\begin{bmatrix} h(1,1) & h(1,2) & \dots & h(1,m) \\ h(2,1) & h(2,2) & \dots & h(2,m) \\ \dots & \dots & \dots & \dots \\ h(m,1) & h(m,2) & \dots & h(m,m) \end{bmatrix}$$

Şekil 2.6 mxm boyutundaki örnek **h** matrisi.

Zaman (konum) bölgesinde yapılan konvolüsyon işlemi Eşitlik (2.6)'da verilmiştir. Bu konvolüsyon işlemi, bütün (x,y) değerleri için tekrarlanmaktadır.

$$o(x, y) = \sum_{k=1}^{m} \sum_{l=1}^{m} h(k, l) f(x - k, y - l) \tag{2.6}$$

Şekil 2.7'de örnek bir konvolüsyon işlem aşaması gözükmektedir. Her piksel için **h** matrisinin boyutu kadar çarpma ve toplama işlemi yapılması gerekmektedir.

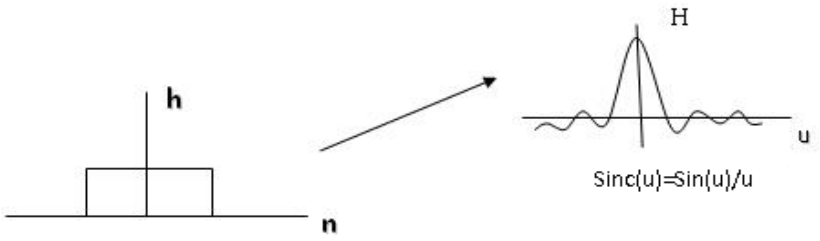


Şekil 2.7 Örnek konvolüsyon işlemi.

2.2.1.1 Ortalama Filtre (Average Filter)

Ortalama filtrede *h(x,y)* filtremizin katsayıları pozitif ve katsayıları toplamı 1 olmalıdır. Eğer tüm katsayılar eşit olursa bu kez filtremiz özel olarak "BOX" filtre adını alır. Bu filtrede yeni oluşan her piksel, eski pikselin komşuluklarının ortalamasıdır.

Ortalama filtremizin boyutunu ne kadar büyütürsek, gürültünün etkisi o derece azalacaktır. Ancak burada unutulmaması gereken filtre matrisimiz büyüdükçe işlem sayısının oldukça fazla artacağıdır. Buna ek olarak, filtre matrisimiz büyüdükçe oluşan görüntü de bulanıklaşmaya başlayacaktır.



Şekil 2.8 Sabit katsayılı ortalama filtrenin etkisi.

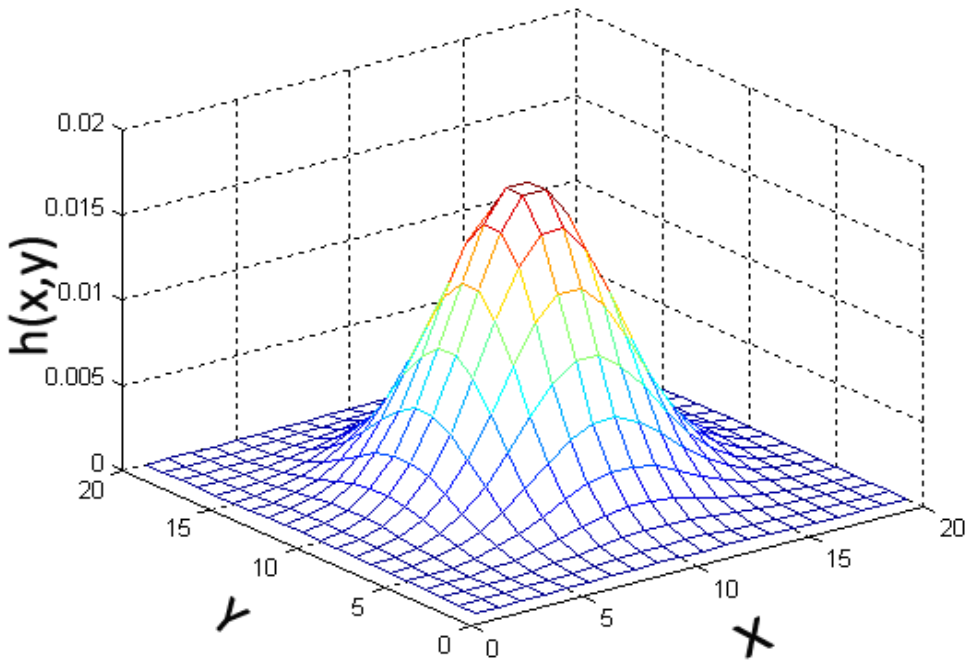
Şekil 2.8'de ortalama filtrenin frekans bölgesindeki karşılığı olan *Sinc* fonksiyonu gözükmektedir. *Sinc* fonksiyonunun dalgalanması yüzünden işlediğimiz görüntüde bozulmalar olacaktır.

2.2.1.2 Gauss Filtre

Ortalama filtrenin frekans bölgesinde gösterdiği bozulmanın önüne geçebilmek ve merkezdeki pikselimizin önem derecesini arttırarak bulanıklaşmayı bir nebze olsun azaltmak için filtre katsayılarımızı Şekil 2.9'da görüldüğü gibi Gauss dağılımlı yapılabilir ve bu filtre Gauss filtre olarak bilinir (Seul et al 2000).

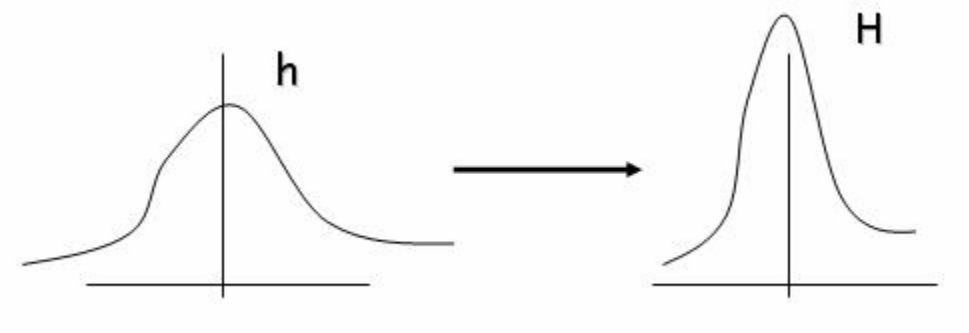
Gauss dağılımlı filtre katsayıları için Eşitlik (2.7) kullanılabilir. Buradaki *x* ve *y* değişkenleri, matrisin orta konumunda sıfır değerini alacak şekilde ötelenmelidir. *K* katsayısı, **h** matrisinin tüm elemanlarının toplamının bire eşit olmasını sağlar, σ katsayısı ise sönümleme hızı ile ilgilidir.

$$h(x, y) = Ke^{\frac{-(x+y)^2}{2\sigma^2}} \tag{2.7}$$



Şekil 2.9 Gauss dağılımlı filtrenin katsayılarının temsili.

Şekil 2.10'da Gauss fonksiyonunun frekans bölgesindeki karşılığı görülmektedir. Gauss fonksiyonunun frekans bölgesinde de, zaman bölgesinde de dalgalanması olmadığı için, işlenen görüntü üzerinde dalgalanma kaynaklı herhangi bir bozulma oluşturmamaktadır.

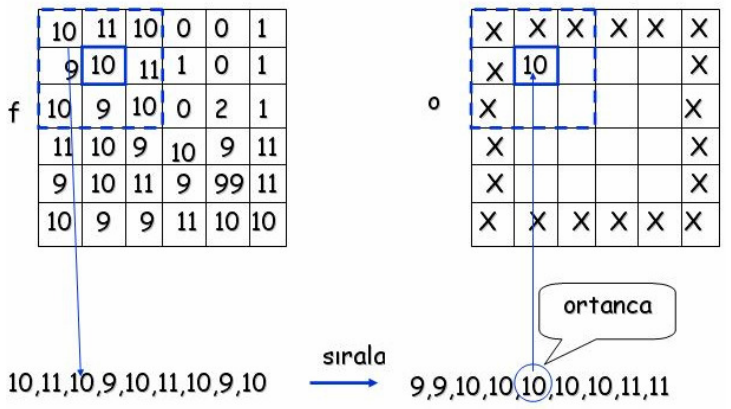


Şekil 2.10 Gauss fonksiyonunun frekans bölgesindeki karşılığı.

2.2.2 Doğrusal Olmayan Filtreleme

Gürültünün, giren görüntüyü doğrusal olmayan yapıyla bozduğu durumlarda doğrusal olmayan filtrelere ihtiyaç duyulur. Burada Orta Değer Filtre sıklıkla kullanılan doğrusal olmayan filtrelerdendir.

Orta değer filtre, özellikle Gauss dağılımlı olmayan gürültüleri yok etmede başarılı olan, uygun değerler seçildiğinde görüntünün ayrıntılarını yok etmeden gürültüyü yok eden, bulanıklaştırma sorunu olmayan bir filtredir. Filtrenin yaptığı işlem, Şekil 2.11'de de görüldüğü üzere, bütün piksel değerlerini, komşuluklarının ortanca değerleriyle değiştirmektir.



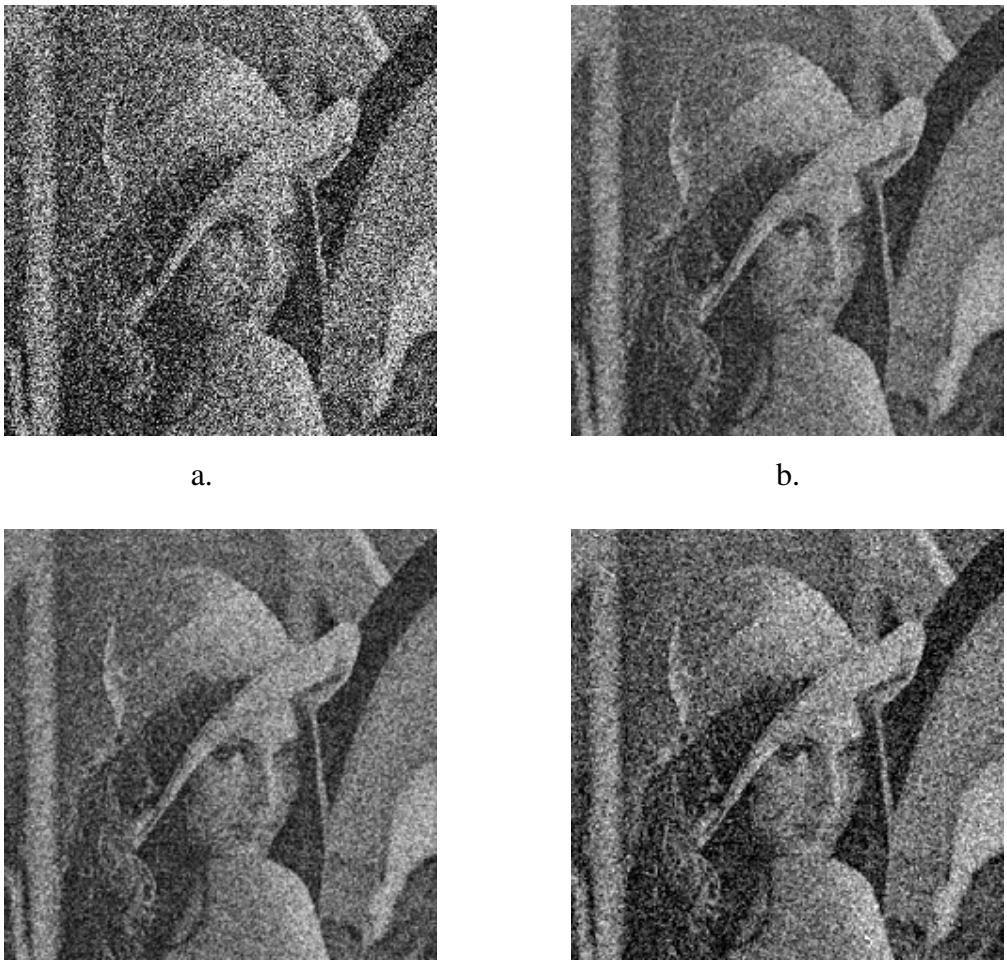
Şekil 2.11 Median (orta değer) filtre örneği.

Her piksel için tüm komşulukların tekrar sıralanması gerektiği için bu filtrede yapılan işlem sayısı fazladır. Tuz-biber gürültüsü gibi gürültülerde oldukça başarılıdır. Doğrusal filtrelerde

olduğu gibi gürültüyü tüm görüntü içine dağıtmaz.

2.2.3 Gürültü Filtresi Örnek Çıktıları

Şekil 2.12a'daki fotoğraf, orijinal Lena fotoğrafının Gauss dağılımlı gürültü eklenmiş halidir. Şekil 2.12b ortalama filtre çıktısını, Şekil 2.12c Gauss filtre çıktısını ve Şekil 2.12d ise orta değer filtre çıktısını göstermektedir. Ortalama filtre çıktısında gürültü tüm fotoğrafa yayıldığı için fotoğraftaki zıtlık (kontrast) değeri düşmüştür. Gauss filtre ise ortalama filtre ile benzer gürültü yok etme başarımı göstermesinin yanında fotoğrafa ait zıtlık değerini de büyük ölçüde koruyabilmiştir. Orta değer filtre ise Gauss dağılımlı gürültüyü yok etmede çok başarılı olamamıştır.



a. b.

c. d.

Şekil 2.12 Gauss gürültü için filtre çıktıları a. Gauss dağılımlı gürültü eklenmiş fotoğraf b. Ortalama filtre çıkışı c. Gauss filtre çıkışı d. Orta Değer filtre çıkışı

Şekil 2.13a'da ise Lena fotoğrafının tuz-biber gürültü eklenmiş hali görülmektedir. Şekil 2.13b ortalama filtre çıktısını, Şekil 2.13c Gauss filtre çıktısını ve Şekil 2.13d ise orta değer filtre çıktısını göstermektedir. Tuz-biber gürültüsünde ortalama filtre ve Gauss filtre çok başarılı olamamışlardır. Ayrıca ortalama filtre, zıtlık değerini Gauss filtreye nazaran daha fazla düşürmüştür. Orta değer filtresi ise tuz-biber gürültüsünde oldukça başarılı olmuştur.

a. b.

c. d.

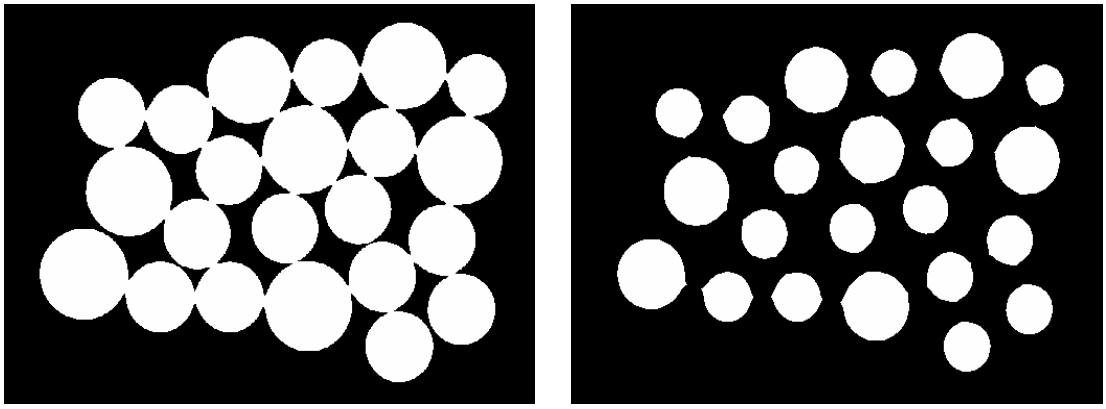
Şekil 2.13 Tuz-Biber gürültü için filtre çıktıları a. Tuz-Biber gürültü eklenmiş fotoğraf b. Ortalama filtre çıkışı c. Gauss filtre çıkışı d. Orta Değer filtre çıkışı

2.3 TEMEL GÖRÜNTÜ İŞLEME TEKNİKLERİ

2.3.1 Aşındırma (Erosion)

Aşındırma işlemi, genellikle ikili görüntülerde nesnelerin küçültülmesi işlemidir. Tek başına kullanıldığında ilgilenilmeyen noktalardan kurtulmak veya nesneleri parçalara ayırmak gibi

işlere yarar (Şekil 2.14). En basit aşındırma işlemi, ilgilenilen noktanın komşuluklarından herhangi birinde arkaplan rengi var ise, o noktanın arkaplan rengine atanmasıdır (Bovik 1999).



a. Orijinal ikili görüntü. b. Aşındırılmış ikili görüntü.

Şekil 2.14 Aşındırma örneği.

2.3.2 Genleştirme (Dilation)

Genleştirme işlemi, aşındırma işleminin tersidir. İkili resimdeki nesnelerin büyütülmesi işlemidir. Tek başına kullanıldığında küçük ayrıntıları bütüne katmak veya parçalara bölünmüş nesneleri birleştirmek gibi işlere yarar (Şekil 2.15). En basit genleştirme işlemi, ilgilenilen noktanın komşuluklarından herhangi birinde önplan rengi var ise, o noktanın önplan rengine atanmasıdır (Bovik 1999).

a. Orijinal ikili görüntü b. Genleştirilmiş ikili görüntü

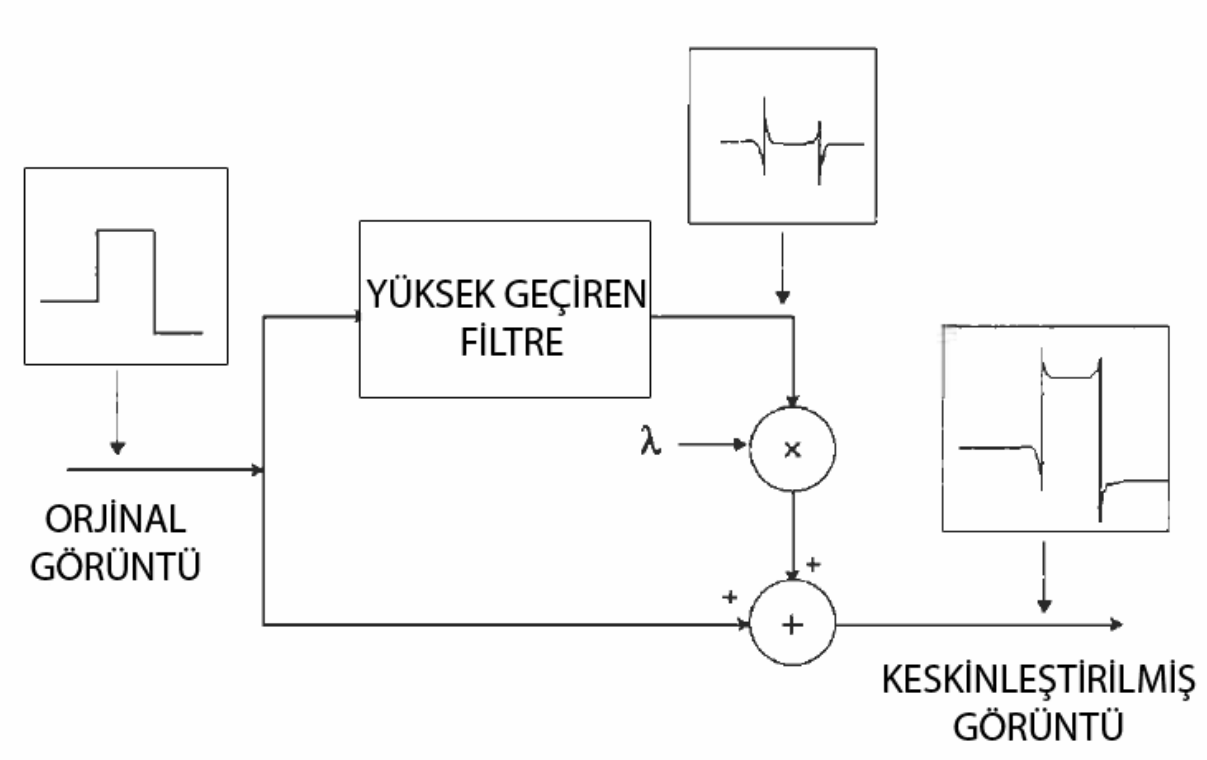
Şekil 2.15 Genleştirme örneği.

2.3.3 Görüntü Keskinleştirme

İnsan algısı, görüntüdeki geçiş noktalarına ve ince detaylara karşı çok duyarlıdır. Bu detaylar ve geçişler görüntüde yüksek frekans bölgesinde yer almaktadır. Dolayısıyla yüksek frekans bölgesi kısmen veya tamamen bastırılmış görüntülerin kalitesi oldukça düşük algılanacaktır. Bir yüksek geçiren filtre uygulaması ilgilenilen görüntünün kaliteli görünmesine sebep olur.

$$o(x,y) = f(x,y) + \lambda(f(x,y) * h(k,l)) \tag{2.8}$$

Eşitlik (2.8), görüntü keskinleştirme için kullanılan fonksiyon olup burada λ filtre katsayısıdır. Sıfır veya sıfırdan büyük bir değere sahiptir. $h(k,l)$ ise yüksek geçiren filtredir. Keskinleştirme işleminin algoritması Şekil 2.16'da görülmektedir. Orijinal görüntünün üzerine yüksek geçiren filtreden geçen görüntü eklenerek kullanıcıya daha detaylı bir görüntü izlenimi verilmektedir. Bu filtreleme tekniği fotoğrafçılıkta sıklıkla kullanılmaktadır (Bovik 1999).



Şekil 2.16 Görüntü keskinleştirme algoritması.

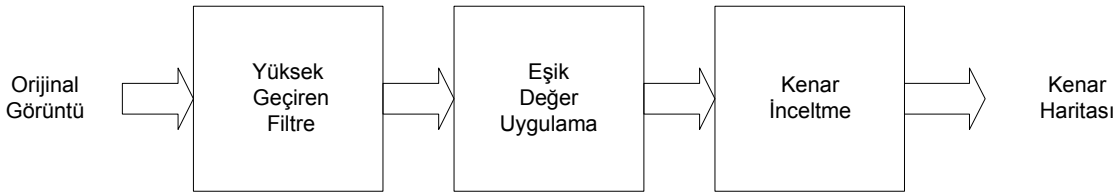
2.4 NESNE TANIMLAMA İŞLEMLERİ

Nesne tanımlama, işlenen görüntüdeki nesnelerin takip edilebilir özelliklerinin belirlenmesine dayalı yöntemlerdir. Bu yöntemlerin en çok kullanılanları kenar tanımlama ve köşe tanımlama yöntemleridir (Castleman 1995).

2.4.1 Kenar Tanımlama

Görüntü işlemede kenar tanımlama işlemi önemli bir yer tutmaktadır. Bunun sebebi kenar tanımlamanın görüntüdeki nesnelerin ayrımıyla ilgili önemli ipuçları vermesidir (URL-1).

Bir kenar tanımlama algoritması görüntüdeki ana kenarları ortaya çıkarırken gürültünün sebep olduğu sahte kenarları göz ardı edebilmelidir. Genellikle uygulanan yaklaşım Şekil 2.17'de görülmektedir. Yüksek geçiren filtre ile görüntüdeki kenarlar belirginleştirildikten sonra eşik değerinin altında kalan bölümler görüntüden çıkarılmakta ve elde edilen görüntüde kenar inceltilmesi yapılmaktadır. Sonuçta ortaya çıkan görüntü, gerçek görüntümüzdeki nesnelerin bir nevi haritasıdır.



Şekil 2.17 Kenar tanımlama algoritması.

Kenar tanımlamada kullanılan yüksek geçiren filtrede genellikle Sobel maskeleri kullanılır (Gonzales and Woods 1992). Sobel maskesi, 1968 yılında Sobel tarafından önerilmiş operatördür. Bu operatör sayesinde gri skala görüntümüzde bulunan ton farkı geçişleri tespit edilebilmektedir. 3x3 boyutundaki Sobel maskesi olarak adlandırdığımız bu matrisin gri skala görüntümüzle konvolüsyonu sonucu, gri skala görüntümüzdeki ani geçiş noktaları belirginleşir, aynı tonlamada olan veya yumuşak geçişli tonlama farkları bulunan kısımlar ise sönükleşir. Bu işlemi ayrık zamanlı türev alma olarak da tanımlayabiliriz (Balta C. vd. 2009). Eşitlik (2.9)'da görülen bu iki matrisin birisi yatay kenarları belirginleştirirken diğeri ise

dikey kenarları belirginleştirmektedir.

$$h^{y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.9a}$$

$$h^{d} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.9b}$$

Bu iki matrisin doğrusal uygulanması sonucu elde edilen iki farklı sonucun birleşimi,

$$o^{toplam}(x, y) = \sqrt{\left(o^{y}(x, y)\right)^2 + \left(o^{d}(x, y)\right)^2} \tag{2.10}$$

ile kullanılır. Burada $o^{y}(x, y)$ ve $o^{d}(x, y)$ sırasıyla yatay ve dikey Sobel maskelerinin uygulandığı görüntülerdir.

Kenar tanımlamada, yukarıda açıklanan yöntem dışında doğrusal olmayan filtreleme yöntemi de kullanılabilir, örneğin orta değer (median) filtre. Orta değer filtreleme yapılırken de Sobel maskeleri kullanılabilir.

Eğer yatay ve dikey kenarlar dışında çapraz kenarların da belirginleşmesi istenirse bu sefer Eşitlik (2.11a) ve (2.11b)'de ki maskeler kullanılmalıdır.

$$h^{çapraz1} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \tag{2.11a}$$

$$h^{çapraz2} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \tag{2.11b}$$

2.4.2 Köşe Tanımlama

Görüntü işlemede köşe tanımlama, hareketli nesne takibinde olduğu gibi pek çok uygulama için de önem taşımaktadır (Baştanlar ve Yardımcı 2007). Köşelerin tespiti sayesinde nesneler tanımlanabilmekte ve takip edilebilmektedir. Uzun yıllar boyunca çeşitli köşe bulma algoritmaları geliştirilmiştir. Moravec'in (1977) önerdiği yaklaşımda ana fikir, köşe noktalarının etrafında ışık yoğunluğu farkı oluşmasına dayanmaktadır. Harris and Stephens (1988) köşe noktalarına ait yatay ve dikey türevlerinin oluşturduğu özilinti matrisinin (Eşitlik (2.12)) determinantı ve izini kullanarak (Eşitlik (2.13)) köşelik ölçüsü oluşturmuşlardır.

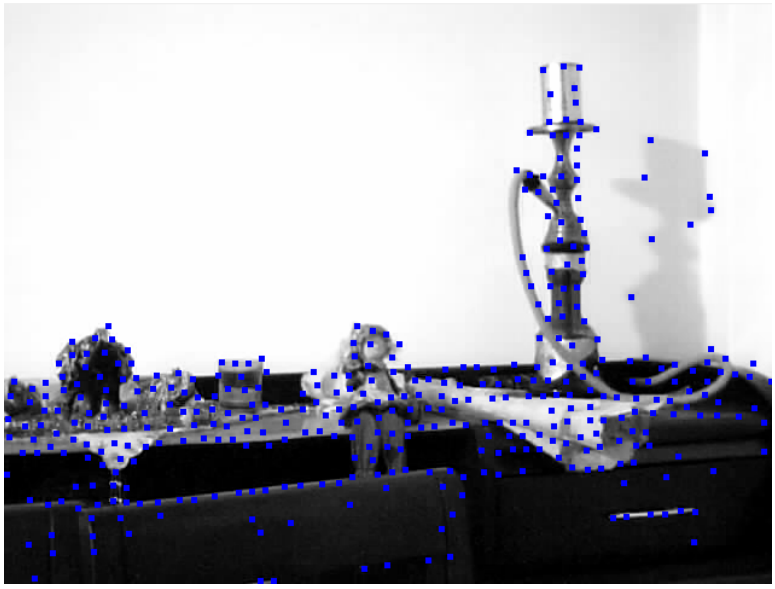
$$C = \sum_{x,y \in R} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2.12}$$

$$M_{(x,y)} = Det(C) - k * Trace^2(C) \tag{2.13}$$

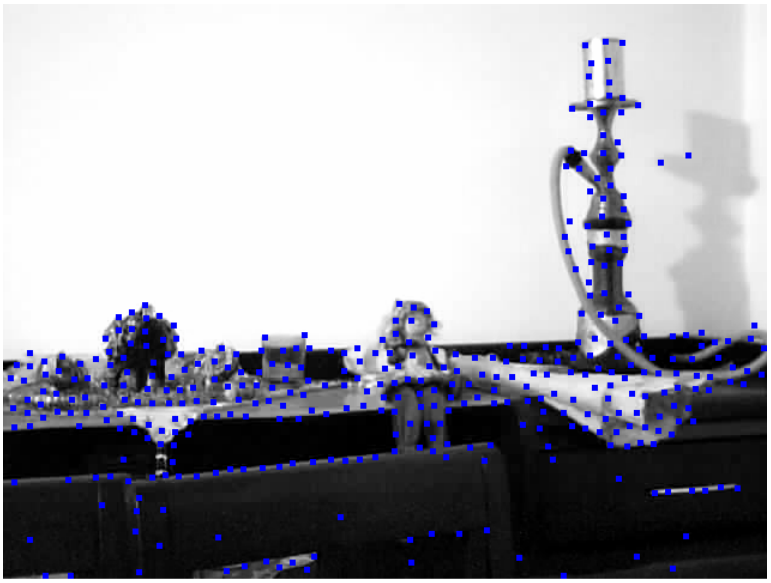
Burada I_x, görüntünün ışık yoğunluğunu temsil eden I matrisinin x koordinatındaki türevi, I_y ise aynı I matrisinin y koordinatındaki türevidir. k katsayısı ise Harris and Stephens (1988) tarafından 0.14 olarak önerilen katsayıdır.

Eşitlik (2.13)'deki **M** matrisinin aldığı değerler sıfıra yakın ise ilgilenilen noktanın köşe ve kenar harici nokta olduğunu, sıfırdan küçük ise ilgilenilen noktanın kenar olduğunu, sıfırdan büyük ise de ilgilenilen noktanın köşe olduğunu anlarız.

Bu durumu daha yakından gözlemleyebilmek amacıyla Şekil 2.18'de farklı açılardan alınmış fotoğraflara Harris and Stephens'in (1988) önerdiği yöntem uygulanmış ve bu yöntemle belirlenen 400 köşe işaretlenmiştir. Bu 400 noktadan çoğu bir önceki fotoğrafla ortak köşeleri tanımlasa da, farklı noktalar da köşe olarak seçilebilmektedir. İki farklı çerçevenin eşleştirilmesinde bu büyük bir problem oluşturmaktadır.



a. İlk çerçeve



b. İkinci çerçeve

Şekil 2.18 Köşe tanımlama metodu ile örnek uygulama çıktısı.

2.5 ARKAPLAN BELİRLEME YÖNTEMLERİ

Hareket tespitinde ve hareketli nesne takibinde arkaplan seçimi en önemli adımlardan biridir. Arkaplan seçiminde amaç, hareketli nesneyi barındıran çerçevede bulunan hareketsiz alan ile ortak bir görüntüye sahip çerçeveyi seçebilmektir. Bu sayede hareketsiz olan ortak kısımlar haricinde kalan bölüm hareketli nesneyi bulmada veya hareket takibinde kullanılabilir.

2.5.1 Sabit Arkaplan

Sahneye ait çerçevelerden hareketli nesne barındırmayan biri arkaplan seçilir. Hareketli nesne, o anki çerçeve ile sabit arkaplan arasındaki fark ile bulunur,

$$\left|F_i - B_0\right| > Th \tag{2.14}$$

ki burada Th eşik değeridir ve uygun seçilmez ise gürültüden etkilenme ve/veya hareketi tespit edememe gibi sonuçlar doğurabilir. B_0 ilk çerçeveden seçilen sabit arkaplanı, F_i ise ilgilenilen çerçeveyi tanımlamaktadır.

a. İlk çerçeve

b. 100. çerçeve

c. Mutlak fark

d. Eşik değeri sonrası

Şekil 2.19 Sabit arkaplan yöntemi ile hareketli nesne tespiti.

Sabit arkaplan yöntemi ile oldukça hızlı işlemler yapılabilmektedir, ayrıca bu yöntemin hafıza

gereksinimi de oldukça düşüktür. Ancak ortamdaki ışık değişimleri ve arkaplana sonradan dahil olmuş nesneler hareketli nesne olarak algılanır. Bu sebeple bu yöntem sadece sabit ışıklandırmaya sahip ve denetim altında tutulabilen ortamlar için uygundur.

Şekil 2.19'da örnek bir sabit arkaplan uygulaması görülmektedir. Şekil 2.19a'da sabit arkaplan olarak seçilen ilk çerçeve, Şekil 2.19b'de ise hareketli nesnenin bulunduğu 100. çerçeve seçilmiştir. Şekil 2.19c'de, seçilen bu iki çerçevenin mutlak farkı ve Şekil 2.19d'de ise eşik değeri sonrası sonuç görülmektedir. Işık yoğunluğunun değişimi ve kameranın titremesinden kaynaklı hatalar, ilgili görüntüdeki ekranın yüzeyinin de hareketli algılanmasına sebep olmuştur.

2.5.2 Çerçeve Farkı Yöntemi

Bu yöntemde sahneye ait çerçevelerden hep bir önceki çerçeve arkaplan olarak seçilir (Eşitlik (2.15a)). Bu iki çerçeve arasındaki farka bakılarak belirli bir eşik değeri üstündeki değişim hareketli nesne olarak algılanır.

$$B_i = F_{i-1} \tag{2.15a}$$

$$\left|F_i - B_i\right| > Th \tag{2.15b}$$

Sabit arkaplan yönteminde olduğu gibi bu yöntemde de eşik değeri seçimi önemlidir (Eşitlik (2.15b)) ve eşik değeri uygun seçilmez ise yine gürültüden etkilenme görülebilir.

Çerçeve farkı yönteminde de yüksek hızla işlem yapılabilir. Sabit arkaplan yönteminin aksine çerçeve farkı yönteminde ışık değişimleri çok ani olmadığı sürece sistemin başarımını etkilemez. Hareketli nesnenin tespiti başarımı ise nesnenin hızına ve çerçeve işleme hızına bağlıdır. Çok yavaş ilerleyen nesnelerin bu yöntemle tespiti çok başarılı değildir.

Şekil 2.20'de örnek bir çerçeve farkı uygulaması görülmektedir. Şekil 2.20a'da arkaplan olarak seçilen 99. çerçeve, Şekil 2.20b'de ise ilgilenilen 100. çerçeve bulunmaktadır. Şekil 2.20c'de, seçilen bu iki çerçevenin mutlak farkı ve Şekil 2.20d'de ise eşik değeri sonrası sonuç görülmektedir. Sabit arkaplan yönteminde görülen ışık değişimi ve ufak titreşimlerden

oluşan hatalar çerçeve farkı yönteminde görülmemiştir.

a. 99. çerçeve

b. 100. çerçeve

c. Mutlak fark

d. Eşik değeri sonrası

Şekil 2.20 Çerçeve farkı yöntemi ile hareketli nesne tespiti.

2.5.3 Akan Ortalama Yöntemi

Bu yöntemde, bir önceki çerçeve ile bir önceki arkaplanın birleşimi yeni arkaplanı belirlemektedir. Sabit arkaplan ile çerçeve farkı yöntemi arasında bir başarım sergiler. Arkaplan seçimi dinamik olduğu için değişken arkaplanlar için uygundur.

$$B_i = \alpha * F_{i-1} + (1-\alpha) * B_{i-1} \qquad (2.16a)$$

$$\left|F_i - B_i\right| > Th \qquad (2.16b)$$

Eşitlik (2.16)'daki α, öğrenme katsayısıdır, sıfıra yaklaştıkça sabit arkaplan yöntemine, bire yaklaştıkça çerçeve farkı yöntemine benzemektedir. Genellikle 0.05 civarı seçilir. F_{i-1} bir önceki sahneye ait çerçeveyi, B_{i-1} bir önceki sahneye ait arkaplanı ve B_i ise o anki sahneye ait hesaplanmış arkaplanı tanımlamaktadır. Bu yöntemin hafıza gereksinimi düşük, hesaplama hızı ise yüksektir. Değişiklik olmakla birlikte bu değişimin çok fazla olmadığı sahnelerde tercih edilir. Şekil 2.21a'da görülen arkaplan, daha önceki arkaplanlar ile işlenen çerçevelerin birleşiminden oluşmuştur. Eşik değeri ve öğrenme katsayısının seçimi ve yöntemin arkaplan tespitinden kaynaklanan hareketli nesnenin izi Şekil 2.21d'de görülmektedir.

a. 99. arkaplan

b. 100. çerçeve

c. Mutlak fark

d. Eşik değeri sonrası

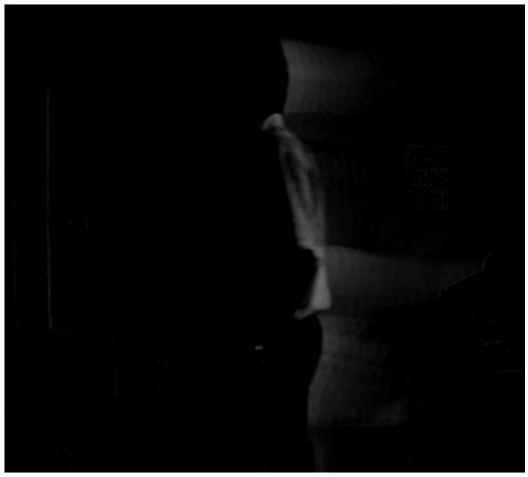
Şekil 2.21 Akan ortalama yöntemi ile hareketli nesne tespiti.

2.5.4 Akan Gauss Ortalama Yöntemi

Akan ortalama yöntemindeki eşik değeri seçiminin arkaplan histogramına uyan Gauss dağılımı parametreleriyle belirlendiği yöntemdir (Eşitlik (2.17)). Burada eşik seviyesi olan Th Gauss dağılım parametresi olan σ_i'ye eşit seçilmektedir. Eşik değeri dinamik olarak seçildiği için akan ortalama yöntemine göre daha başarılı sonuçlar elde edilir. Çok sayıda dokunun yer aldığı ve/veya sürekli olarak sahnenin değiştiği sistemlerde bu yöntem de başarısızdır.

a. 99. arkaplan

b. 100. çerçeve



c. Mutlak fark

d. Eşik değeri sonrası

Şekil 2.22 Akan Gauss ortalama yöntemi ile hareketli nesne tespiti.

$$B_i = \alpha F_{i-1} + (1-\alpha) B_{i-1} \tag{2.17a}$$

$$\sigma_i^2 = \alpha (F_{i-1} - B_{i-1})^2 + (1-\alpha)\sigma_{i-1}^2 \tag{2.17b}$$

$$\left| F_i - B_i \right| > Th = \sigma_i \tag{2.17c}$$

Şekil 2.22a'da daha önce seçilen 99 arkaplanın ve işlenen 99 çerçevenin birleşiminden oluşmuştur. Eşik değerinin dinamik olarak seçilmesinden dolayı Şekil2.22d'deki sonuçta hareketli nesneye ait iz daha küçük olmakla birlikte tamamen yok olmamıştır.

2.5.5 Gauss Dağılımlarının Birleşimi Yöntemi

Piksellerin geçmişindeki bölgesel dağılımlara göre farklı arkaplan modellerinin seçimine dayanır. Her piksele ait geçmiş verilerin dağılımına bakılır, şuan ki değeri belli alanlardaki kümeleşmelere dahil ise arkaplan kabul edilir, değilse hareketli nesne kabul edilir. Her piksel için ayrı birçok hesaplama (Eşitlik 2.18) yapılması gerektiği için anlatılan diğer yöntemlere kıyasla en yavaş yöntemdir. Ancak hareketli nesne takibinde oldukça başarılıdır.

$$f_{X|k}(X \mid k, \Theta_k) = \frac{1}{2\pi^{n/2} \left| \Sigma_k \right|^{1/2}} e^{-\frac{1}{2}(X-\mu_k)^T \Sigma_k^{-1} (X-\mu_k)} \tag{2.18a}$$

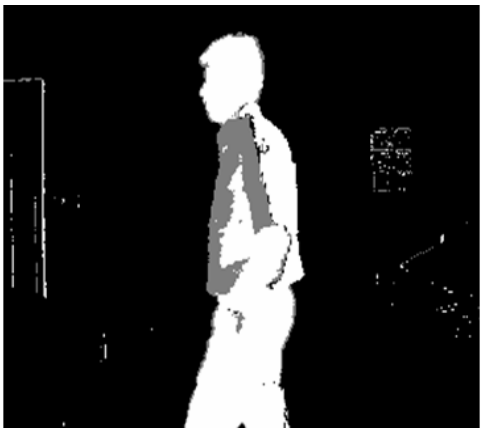
$$f_X(X \mid \Phi) = \sum_{k=1}^{K} P(k) * f_{X|k}(X \mid k, \Theta_k) \tag{2.18b}$$

$$\text{Kovaryans Matris} = \Sigma_i \approx \sigma_i^2 I \tag{2.18c}$$

Şekil 2.23'te görüldüğü gibi bu yöntem, hareketli nesnenin tespitinde oldukça başarılı olmuştur. Bunun sebebi ise bu yöntemin, her piksele ait birden fazla sayıda arkaplan belirleyebilmesidir. Örneğin bir ağaç dalının sürekli olarak arkaplanda hareket ettiğini düşünelim, bu hareket eden ağaç dalının her pozisyonu arkaplana dahil edilebildiği için sistem

tarafından hareketli nesne olarak algılanmamaktadır.

a. 100. çerçeve



b. Gauss dağılımı hesabı sonrası

c. Eşik değeri sonrası

Şekil 2.23 Gauss dağılımlarının birleşimi yöntemi ile hareketli nesne tespiti.

2.5.6 Arkaplan Belirleme Yöntemlerinin Başarımları

Arkaplan belirleme yöntemleri uygulamaya göre seçilmelidir. Kimi uygulamalarda hız gerekli iken kimi uygulamalarda ise doğruluk daha önemlidir. Ayrıca her yöntemin başarısız olabildiği farklı durumlar da mevcuttur.

- Hız
    - Hızlı: Statik arkaplan, çerçeve farkı, akan ortalama
    - Orta: Gauss ortalama
    - Yavaş: Gauss dağılımları birleşimi
- Hafıza gereksinimi

- Yüksek: Orta/ortalama değer
- Orta:Gauss ortalama
- Az: Statik arkaplan, çerçeve farkı, akan ortalama

- Doğruluk
  - Şartlara bağlı
  - Genellikle Gauss ortalama daha iyi

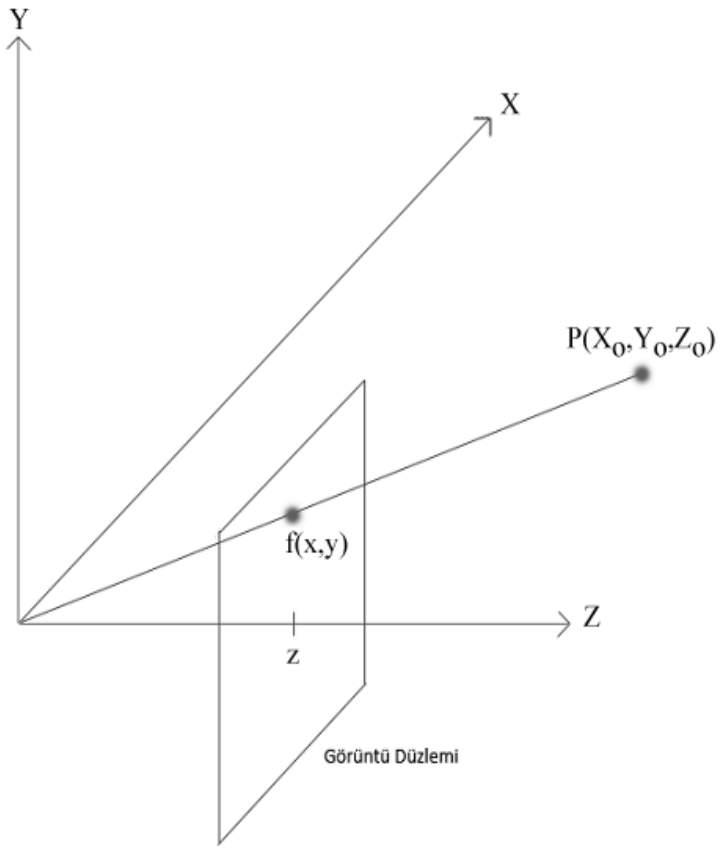
2.6 AKTİF KAMERA KALİBRASYONU

Uzun yıllar boyunca araştırmacılar çeşitli kamera kalibrasyon teknikleri geliştirmişlerdir. Bu teknikler genellikle doğrusal ve doğrusal olmayan metotlardan oluşmuştur (Basu 1995). Doğrusal olmayan tekniklerle çok ayrıntılı modellemeler yapılabilmektedir. Tekrarlanan iterasyonlarla doğrusal olmayan bu karmaşık modellemelerin parametreleri tespit edilebilmektedir. Ancak bu iterasyonların karmaşıklığının yüksek oluşu ve yüksek işlem gücü gerektirmesi, ayrıca iterasyon başlangıç değerlerinin sonuca ulaşmadaki belirleyiciliği sebepleriyle yöntem sorunludur. Doğrusal teknikler ise basit ve yüksek işlem gücü gerektirmemesi sebebiyle avantajlı gibi gözükseler de çoğu doğrusal teknik kamera lenslerini modellemede başarısızdır. Ancak doğrusal tekniklerin kamera lenslerini modellemelerindeki eksikliklere rağmen doğrusal olmayan tekniklerinin performans sorunları araştırmacıları doğrusal tekniklere yönlendirmişlerdir (Basu 1995).

Şekil 2.24'teki kamera modelinde z odak uzaklığı, *f(x,y)* ise $P(X_0,Y_0,Z_0)$ noktasının görüntü düzlemindeki iz düşümü olmak üzere Eşitlik (2.19)'daki gibi *f(x,y)* koordinatlarını hesaplayabiliriz. Bu eşitlikteki z_x ve z_y, fotoğrafın en/boy oranının genellikle bire bir olmaması sebebiyle konulmuş sabitlerdir.

$$x = z_x \frac{X_0}{Z_0} \tag{2.19a}$$

$$y = z_y \frac{Y_0}{Z_0} \tag{2.19b}$$



Şekil 2.24 Lens bozulmaları ihmal edilmiş kamera modeli.

Kamera *y* ekseni etrafında dönüş yaptığında yatay dönüş (pan hareketi), *x* ekseni etrafında döndüğünde ise dikey dönüş (tilt hareketi) yapmaktadır. Bu dönüşler sonucunda aynı $P(X_0,Y_0,Z_0)$ noktasının iz düşümü artık başka $f(x,y)$ noktasına karşılık gelecektir. Dikey dönüş yapıldığında, dikey dönüş açısı θ_t olmak üzere Eşitlik (2.20), yatay dönüş yapıldığında ise, yatay dönüş açısı θ_p olmak üzere Eşitlik (2.21) elde edilir. Bu eşitliklerde Z_0 mesafesinin *z* odak uzaklığına kıyasla yeteri kadar büyük olduğu ve θ_t ile θ_p açılarının küçük olduğu düşünülmüştür.

Eşitlik (2.20) ve Eşitlik (2.21)'in uygulaması Şekil 2.25'te görülmektedir. Sahnede hareketli nesne yok iken sadece bir derecelik yatay dönüş hareketinin kalibrasyon uygulanmış ve uygulanmamış görüntüleri Şekil 2.25a ve Şekil 2.25b'de verilmiştir. Görüldüğü üzere kameranın dönüş açısına dayalı kalibrasyon küçük dereceli dönüşlerde gayet başarılıdır. Aynı işlemleri bu kez hareketli nesne sahnemizdeyken gerçekleştirince kalibrasyon sayesinde nesne doğru olarak tespit edilebilmiştir.

$$x_t \cong x\left(1+\theta_t \frac{y}{z_y}\right) \tag{2.20a}$$

$$y_t \cong \left(y+\theta_t z_y\right)\left(1+\theta_t \frac{y}{z_y}\right) \tag{2.20b}$$

$$x_p \cong \left(x+\theta_p z_x\right)\left(1+\theta_p \frac{x}{z_x}\right) \tag{2.21a}$$

$$y_p \cong y\left(1+\theta_p \frac{x}{z_x}\right) \tag{2.21b}$$

a. Kalibrasyonsuz ve hareketsiz sonuç.

b. Kalibrasyonlu ve hareketsiz sonuç.

c. Kalibrasyonsuz ve hareketli sonuç.

d. Kalibrasyonlu ve hareketli sonuç

Şekil 2.25 Kamera kalibrasyonu ile hareketli kamerada hareketli nesne tespiti.

2.7 ÇERÇEVE FARKI TEMELLİ HAREKETLİ NESNE TAKİP YAKLAŞIMLARI

Hareketli kamera ile hareketli nesne takibindeki en büyük sorun, değişen her görüntünün hareketli nesne olarak tanımlanamamasıdır. Bu sebeple sabit kameralarda uygulanan çerçeve farkı yöntemi, hareketli kameralarda doğrudan uygulanamamaktadır. Bu çalışmada hareketli kamera aracılığıyla hareketli nesnelerin tespit edilmesi için çerçeve farkı yönteminin hareketli kameralara uyarlanmasıyla oluşmuş iki farklı algoritma kullanılmıştır. Bu algoritmalardan ilki gecikmeli çerçeve farkı yöntemine (GÇFY), ikincisi ise köşe tespitli çerçeve farkı yöntemine (KTÇFY) dayanmaktadır.

2.7.1 Gecikmeli Çerçeve Farkı Yöntemi

Gecikmeli çerçeve farkı yöntemi, kameranın hareket ettiği varsayılan süre boyunca arkaplan seçiminin yapılmadığı, süre sonunda seçilen arkaplan ile çerçeve farkı yönteminin uygulanmaya başlandığı, kamera hareketi başlayana kadar çerçeve farkı yönteminin uygulanmaya devam edildiği yöntemdir. Eşitlik (2.22)'de tanımlanan, arkaplan seçiminin yapılmadığı bu k aralığı boyunca görüntü işleme algoritmaları çalıştırılmamaktadır. GÇFY algoritmasında, kameranın belirli konum değişikliklerine karşılık gelen bekleme süreleri önceden belirlenmiştir. Seçilen bu bekleme sürelerinin uygunluğu, algoritmanın başarımını belirlemektedir. Önceden tanımlanan k bekleme değerleri uygun seçilmez ise hareketli nesne tespiti başarımı düşecektir.

$$\left|F_{i+k} - B_{i+k}\right| > Th \tag{2.22}$$

2.7.2 Köşe Tespitli Çerçeve Farkı Yöntemi

Köşe tespitli çerçeve farkı yöntemi, köşe tespitine dayalı bir bekleme sonunda çerçeve farkı yönteminin uygulandığı metottur. Bu yöntemdeki n bekleme süresi sabit değildir (Eşitlik (2.23)). İşlenen çerçevelerin tespit edilen köşeleri eşleştirilerek eşleşen köşeler arası mesafelerin histogramı belli bir değerin altında kaldığında kamera hareketinin sonlandığı düşünülmektedir. Hareketin sonlandığı tahmin edilen süre sonunda yeni arkaplan seçimi

yapılarak çerçeve farkı yöntemi ile hareket tespiti yöntemi uygulanmaktadır.

$$\left|F_{i+n} - B_{i+n}\right| > Th \tag{2.23}$$

Kamera hareketi devam ederken köşe tespiti ve eşleştirilmesinin yanı sıra yanlış köşe eşlemelerini azaltmak için kamera kalibrasyonundan faydalanılmaktadır. Buna rağmen tespit edilen köşelere ait takip edilebilir değerler, ışık, kamera açısı ve gürültü gibi birçok parametre ile sürekli değiştiğinden hatalı eşlemelerden tamamen kaçınmak mümkün olmamaktadır. Ayrıca köşe tespiti ve eşleştirmesi işlemi çok fazla işlemci gücü gerektirmektedir. Diğer algoritmanın aksine, hareket tespiti yapılmayan aralıkta çok fazla hesaplama yapılması gerekmekte ve dolayısıyla işlenemeyen çerçeve sayıları da artmaktadır.

İşlenemeyen çerçeveler ve hatalı eşlemeler sebebiyle KTÇFY algoritmasının başarımı, GÇFY algoritmasının başarımına göre düşüktür. Ancak gerekli uygunlaştırmalar sonrası adaptif olan bu yöntemin daha başarılı sonuçlar vereceği düşünülmektedir.

BÖLÜM 3

DONANIM

3.1 DONANIMIN SEÇİMİ

Bu çalışmada amaçlanan hareketli nesne takibini gerçekleştirmek için yazılımla hareket ettirilebilen bir kamera düzeneğine ihtiyaç duyulmuştur. Hazır pan/tilt donanımlı kameraların maliyetinin çok yüksek olması ve ihtiyaca uygun donanım uygunlaştırmanın mümkün kılınması sebebiyle donanımı üretme yoluna gidilmiştir. Bu amaçla, elimizde mevcut olan USB kamerayı taşıyabilecek ve uygun hızda hareket ettirebilecek donanım arayışına gidilmiştir. Kamerayı zorlanmadan hareket ettirebilmesi için kullanılan motorlar yüksek torka sahip Servo Motor seçilmiştir. Ayrıca motor kontrol devresinin kamera ile aynı veri yolunu kullanması ve harici güç kaynağı gerektirmemesi için USB destekli olması tercih edilmiştir. Bütün bunlar göz önüne alınınca iki servo motordan oluşan pan/tilt motor aksamı ve USB destekli bir mikrodenetleyiciden oluşan donanım üretilmiştir.

3.2 MİKRODENETLEYİCİ

Mikrodenetleyiciler çalışması için gerekli olan minimum donanımı üzerinde barındıran mikroişlemcilerdir. Mikroişlemcilerde sadece merkezi işlem birimi bulunmasına karşın bir mikrodenetleyicide hafıza, giriş/çıkış çeviricileri/portları, özel haberleşme arabirimleri gibi bileşenler de bulunmaktadır. Yani kısaca mikrodenetleyiciler uygulama geliştirmek için çoğu zaman kendi başlarına yeterli olabilmektedir. Günümüzde elektronik eşyaların ve otomasyonların hemen hepsinde, çamaşır makinesinden televizyona, otomobilden asansörlere, fabrikalardan iletişim cihazlarına kadar, mikrodenetleyiciler kullanılmaktadır.

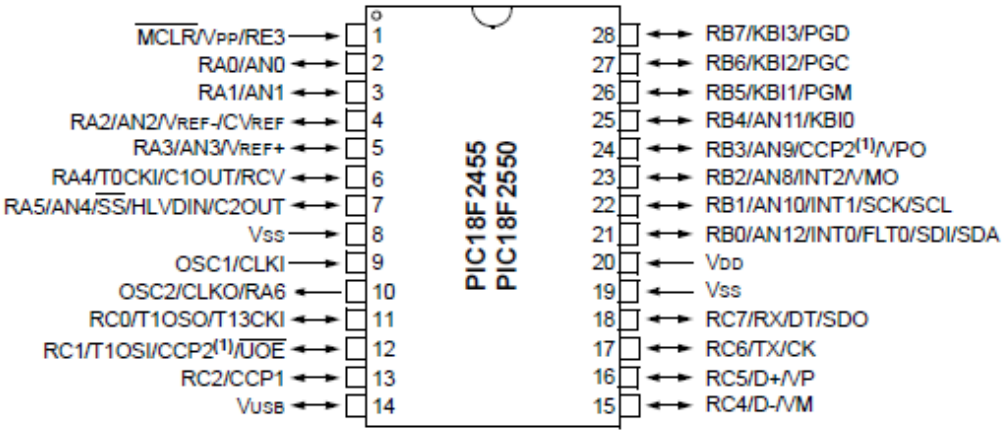
Piyasada USB desteği veren çeşitli mikrodenetleyiciler bulunmaktadır (URL-2 2010, URL-3 2010). Mevcut uygulamaya yeterli kapasitede ve bulunabilirliği yüksek olduğu için bu uygulamada PIC 18F2550 denetleyicisi kullanıldı. Bu mikrodenetleyici, bilgisayarla USB2.0

tam hız destekli haberleşebilen uygulamalar geliştirmekte kullanılabilmektedir.

PIC mikrodenetleyiciler RISC mimarisine sahiptirler. RISC mimarisi kabaca komut setinin sınırlandırılmış olduğu mimaridir (İndirgenmiş Komut Setli Bilgisayar - Reduced Instruction Set Computer). Komut seti, denetleyicinin yapacağı işlemlerin öntanımlı olduğu kodların bulunduğu tablodur. Denetleyiciler sadece bu kodlarla karşılaştığında hangi işlemlerin yapılacağını bildikleri için denetleyici veya işlemci ile hangi işlemler yapılması gerekiyor ise o komutların, mimaride denetleyici veya işlemciye yüklü olması gerekmektedir. Ama bazı komutlar, başka komutların birleştirilmesiyle de yapılabildiği için RISC mimarisinde sadece temel komutlar denetleticiye yüklenmiştir (URL-4 2010). CISC (Karmaşık Komut Setli Bilgisayar - Complex Instruction Set Computer) mimaride ise çok kullanılacağı düşünülen her işlem gurubu da ayrı bir komut olarak denetleyiciye yüklenir. CISC mimarideki denetleyicide makine dilinde program yazmak daha kolayken, örneğin bir dosyaya erişebilmek için sadece bir komut girmek yeterli olabiliyorken, günümüzde yüksek seviyeli dillerle programlama yapıldığından bu avantaj pek fazla işe yaramamaktadır. Çünkü yüksek seviyeli diller, gerektiğinde karmaşık bir işlemi alt işlemlere programcıyı yormadan bölebilmekte, programcı yine dosyaya erişmek için sadece tek komut yazmasına rağmen, derleyici, makine diline indirgenirken bu tek komutu, komut setinde bulunan çeşitli basit komutlar cinsinden tekrar yazar ve dolayısıyla CISC mimarisine ihtiyaç duyulmamaktadır. Bunun da ötesinde RISC mimarisinde hemen bütün komutlar çok basit işlemleri yapmakta olduğundan her komutu işleme süreci birbirine yakın ve kısadır ve de aynı saat hızında olunmasına rağmen daha yüksek işlem hızlarına ulaşılabilmektedir. Günümüzde bilgisayarlarımızda kullanılan işlemciler, geriye uyumlu olması açısından, CISC mimarisinde üretiliyor olsalar da aslında işlemcinin içinde tüm komutlar temel komutlara indirgenip yine RISC mimarisine benzer yapıda işlenmektedir.

Bu çalışmada kullanılan PIC18F2550, 28 bacaklı olup PDIP yapıdadır (Şekil 3.1). Üzerinde dahili USB 2.0 Tam Hızlı arayüz, 10 bit ADC, bir tane 8 bit, 3 tane 16 bit olmak üzere 4 adet zamanlayıcı barındırmaktadır (Çizelge 3.1).

28-Pin PDIP, SOIC



Şekil 3.1 PIC 18F2550'ye ait bacak bağlantısı.

Çizelge 3.1 PIC 18F2550'ye ait teknik özellikler.

Parametre Adı	**Değer**
Program Hafıza Türü	Flash
Program Hafızası (KB)	32
CPU Hızı (MIPS)	12
RAM (Byte)	2,048
Veri Hafızası (EEPROM - byte)	256
Sayısal Haberleşme Çevrebirimleri	1-A/E/USART, 1-MSSP(SPI/I2C)
Yakalama/Karşılaştırma/PWM Çevrebirimleri	2 CCP
Zamanlayıcı (Timer)	1 x 8-bit, 3 x 16-bit
Analog Sayısal Çevirici (ADC)	10 kanal, 10-bit
Karşılaştırıcı	2
USB	1 adet, Full Speed(Tam Hızlı), USB 2.0
Sıcaklık Aralığı (C)	-40 - 85
Çalışma Gerilim Aralığı (V)	2 - 5.5
Bacak Sayısı	28

3.2.1 Mikrodenetleyicide Kullanılan Programlama Dili

Bu tezde kullanılacak olan PIC 18F2550 denetleyicisini programlamak için kullanılan Proton Plus (URL-5 2010) olarak anılan PIC Basic Pro'nun gelişmiş halidir. Bu sayede yüksek seviyeli programlama yapılarak denetleyicinin özel mimari farklılıklarını öğrenme zorunluluğu ve anlaşılması güç programlar yazma gerekliliği ortadan kalkmıştır. Ayrıca bu sayede servoları kontrol etmek ve USB haberleşmesi gibi karmaşık işlemler için Çizelge 3.2'de ki gibi basit komutlar yeterlidir.

Çizelge 3.2 Proton Plus tarafından desteklenen bazı komutlar.

Komut	İşlev
Servo *Pin*,*Deger*	Belirtilen *Pin*'de bulunan servo motora belirtilen *Deger* aktarılır
UsbIn *EndPoint*,*Degisken*,AUTO	Belirtilen *EndPoint* değerinden sonraki USB arayüzünden gelen verileri *Degisken*'in içine atar
UsbOut *EndPoint*,*Degisken*,AUTO	Belirtilen *EndPoint* değerinden sonra *Degisken* içindeki veri USB arayüze gönderilir

3.2.2 Mikrodenetleyici USB Arayüzü

USB üzerinden haberleşebilmek için birçok metot ve sınıf vardır. Ama genellikle bu sınıflardan ikisi çok kullanılmaktadır. İlki HID (İnsan Arayüz Aygıtları – Human Interface Devices) sınıfı, diğeri ise CD (veya CDC) (İletişim Aygıtları - Communications Device Class) sınıfıdır (URL-6 2010). CD sınıfında haberleşme yapabilmek için bilgisayar tarafında işletim sistemine sürücü kurulması ve/veya sanal seri port ayarlanması gerekmektedir. Buna karşın HID sınıfındaki aygıtlar temel fonksiyonlarıyla sürücü veya ek yazılıma ihtiyaç olmadan işletim sistemleri tarafından otomatik tanınmaktadır. Bu sebeple mikrodenetleyici HID sınıfında kullanılmıştır.

Bir cihazı USB sınıflarından birinde tanımlamak için oldukça karmaşık bir yapıda veri

göndermek gerekmektedir. Her sınıfın ayrı tanımlaması ve her kullanım şeklinin de ayrı bir tanımlaması vardır (URL-6 2010). Bu karmaşık tanımlamaları yapmak için çeşitli yardımcı programlar mevcuttur. Bu projede EasyHID (URL-7 2010) adı verilen yardımcı program ile mikrodenetleyicinin sınıf tanımlaması yapılmıştır. Mikrodenetleyiciye girilen tanımlama bilgisi içindeki sağlayıcı firma numarası (VendorID) ile ürün numarası (ProductID) bilgileri bilgisayardaki programda da belirtilmek zorundadır.

3.3 SERVO MOTOR

Servo motorlar, üzerindeki dahili işlemcisi sayesinde sayısal olarak haberleşebilen, açısal hatalarının sürekli olarak düzeltildiği, düşük hızlarına rağmen yüksek torka sahip açısal konumlama motorlarıdır. Kullanılan servolarda darbe genişliği ile veri aktarımı yapılmaktadır. Robotik uygulamalarda bu tarz motorlar sıklıkla kullanılmaktadır.

Kullanılan motorun marka ve modeline göre açısal konuma karşılık gelen darbe genişliği değişse de genellikle 1.5 ms süreli darbe 90 derecelik başlangıç konumu kabul edilir. Servo motora konum bilgisi belirli aralıklarla tekrar yollanmalıdır. Bu sebeple mikrodenetleyici döngüde çalıştırılarak USB arayüzünden gelen veriler kısa aralıklarla tekrar motorlara gönderilmiştir. Servo motorlarda veri akışı tek taraflı olduğu, yani motorun konum bilgisi geri dönmediği ve konum bilgisini aktaran bir alıcımız da olmadığı için devreye ilk enerji verildiğinde motorların önceden belirlenen başlangıç konumlarını alması sağlanmıştır.

Şekil 3.2 Servo motorlara ait düzenek.

Kameranın yatay ve dikey eksende çevrilmesini sağlamak amacıyla iki servoya ihtiyaç duyulmaktadır. Servoları yatay ve dikey eksende hareket ettirmek için kullanılan düzenek Şekil 3.2'de gösterilmiştir. Altta bulunan servo pan (yatay dönme) hareketinden sorumlu iken, üstte bulunan servo ise tilt (dikey dönme) hareketinden sorumludur. Bu düzenekte kullanılan motorların teknik özellikleri Çizelge 3.3'de belirtilmiştir.

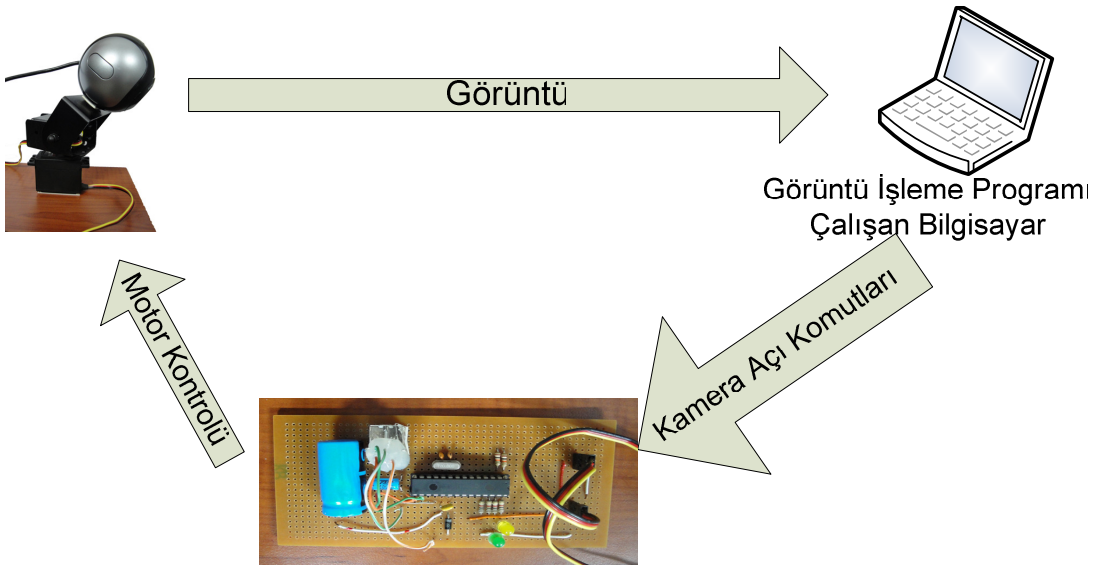
Çizelge 3.3 Kullanılan Servo motorların teknik özellikleri.

Kontrol Sistemi	PWM Kontrol (1.5ms ilk pozisyon)
Çalışma Gerilimi	4,8V – 6V
Çalışma Sıcaklığı	-20 - 60
Çalışma Hızı	0.21 sn/60 derece (4.8V) – 0.16sn/60 derece (6V)
Tork	3.3kg cm (4.8V) – 4.1kg cm (6V)

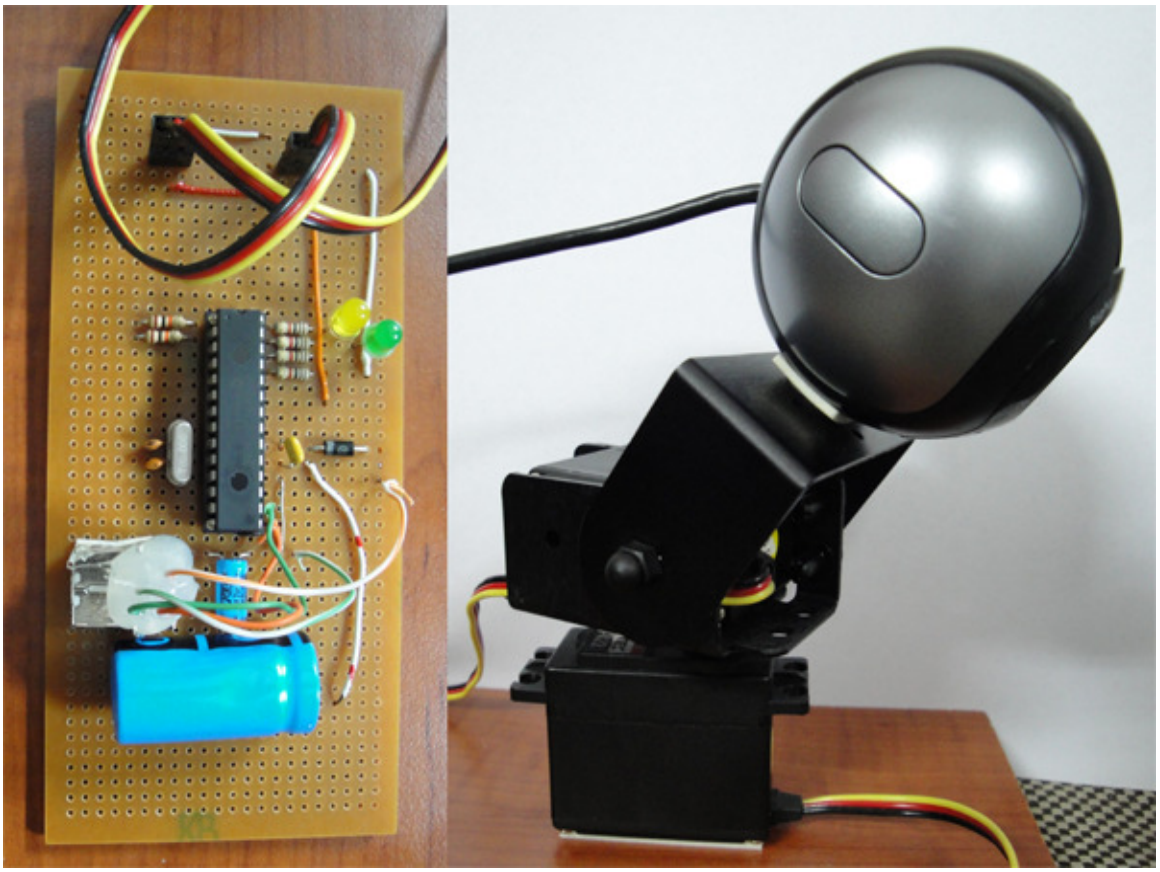
3.4 KULLANILAN DONANIM

Bu çalışmada Şekil 3.3'te blok şeması görülen donanım yapısı kullanılmıştır. Bilgisayar ile mikrodenetleyici arasında ve bilgisayar ile kamera arasında USB veri yolu üzerinden haberleşme sağlanmıştır. Veri gönderimi, iki veri yolunda da tek yönlü gerçekleşmektedir. Kameradan bilgisayara USB arayüzden akan görüntü bilgileri işlenip gerekli Pan / Tilt açı bilgileri mikrodenetleyici devresine yine USB arayüzden gönderilmektedir. Mikrodenetleyici devresi ise gerekli darbe genişliklerini hesaplayarak, portlarına doğrudan bağlı olan servo motorları darbe genişliği yöntemiyle kontrol etmektedir. Bu yapıda motorların geri beslemesi, sadece görüntü aracılığıyla bilgisayara ulaşmakta ve görüntü işleme programı bu geri beslemeyle motorları kontrol etmektedir.

Şekil 3.4'te ise uygulama geliştirilen test devresi ve motor – kamera bağlantı düzeneği görülmektedir. Kamera ve sürücü devresi aynı düzleme monte edilmiştir. Devre kartı üzerine konulan soketler sayesinde kamerayı hareket ettiren motorlar devreden kolaylıkla sökülüp takılabilmektedir. Beslemenin varlığını gösteren yeşil LED (Light Emitting Diode – Işık Yayan Diyot) ve USB haberleşmesini sağlayan sarı LED ile devrenin doğru şekilde bilgisayar ile haberleştiğinden emin olunması sağlanmıştır.



Şekil 3.3 Kullanılan donanımın blok şeması.



Şekil 3.4 Aktif kamera donanımı.

BÖLÜM 4

YAZILIM

Görüntü işleme ve USB donanımlarla haberleşme için gerekli olan yazılımlarda C#.NET dili kullanılmıştır. C# dilinin kullanılmasının amacı, görüntü işleme fonksiyonlarını güvensiz kodlama olan doğrudan belleğe erişim metotlarıyla yazma gerekliliğidir. Bu sayede görüntü üzerindeki işlemler daha hızlı yapılmış ve gerçek zamanlı nesne takibi mümkün olmuştur.

Bilgisayarda C# dilindeki yazılımı üç ayrı başlıkta inceleyebiliriz. Bunlar;

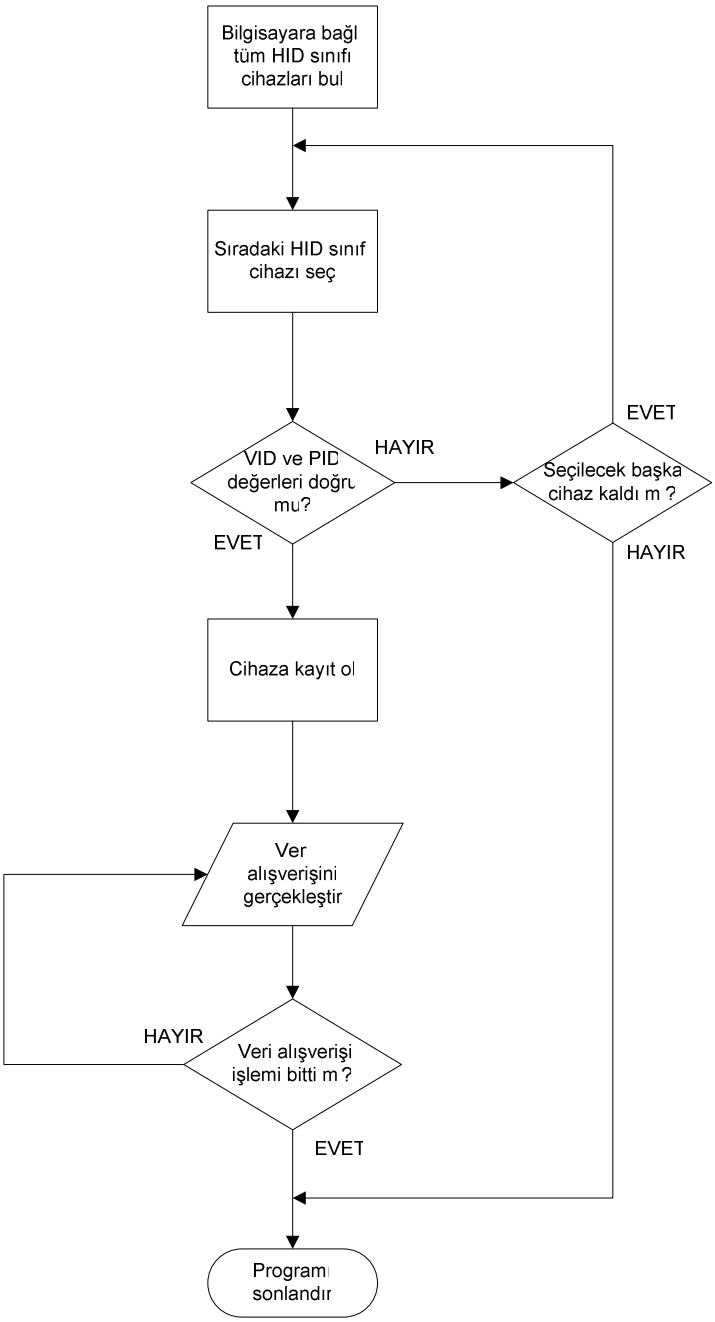
- Servo motorlarla haberleşen yazılım,
- Kameradan görüntüleri alan yazılım,
- Görüntü işleme yazılımıdır.

4.1 SERVO MOTOR SÜRÜCÜ DEVRESİ İLE HABERLEŞME

Tasarlanan servo motor sürücü devresinde USB destekli PIC 18F2550 kullanmıştır. PIC denetleyicisi HID sınıfında programlandığı için, HID sınıfı aygıtlarla haberleşmekte kullanılan Microsoft kütüphaneleri uygulamaya dahil edilmişlerdir. Bu projede gerekli olan haberleşme işlemi için hid.dll, user32.dll, setupapi.dll ve kernel32.dll kütüphanelerine ait fonksiyonlar kullanılmıştır. Daha önce mikrodenetleyiciyi programlarken atanan VendorID (VID) ve ProductID (PID) değerleri, sürücü devresi ile haberleşmek için C# ile yazılan kısımda da kullanılarak servo motor sürücüsüne kayıt olunmuştur.

Şekil 4.1'de akış diyagramı görülen program kabaca bilgisayara bağlı olan tüm HID sınıfındaki donanımları tarayıp, VendorID ve ProductID değerleri önceden belirlenen değerlerle aynı olan donanıma ait, bilgisayarın her defasında rasgele verdiği evrensel eşsiz tanımlama bilgisini (GUID – Globally Unique ID) bulur. Daha sonra bu tanımlama bilgisine kayıt olarak ilgili donanımı bir dosya gibi görülmesini sağlar. Ayrıca mikrodenetleyicide daha önce EasyHID programı yardımıyla tanımlamış olduğumuz gönderilen ve alınan parametre

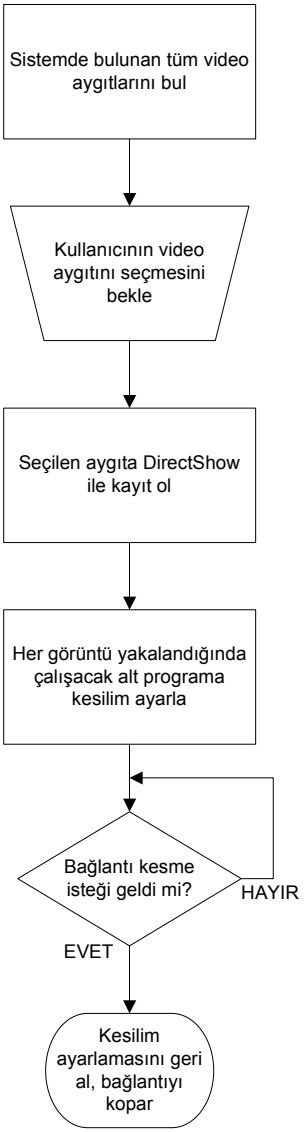
değerleri tespit edilir. Bu sayede mikrodenetleyiciye ne büyüklükte veriler yazıp ne büyüklükte veriler okunabileceği öğrenilmiş olur. Mikrodenetleyiciye veri göndermek istendiğinde, dosyaya veri yazma işlemine benzer işlemler; mikrodenetleyiciden veri almak istendiğinde de, dosyadan veri okuma işlemine benzer işlemler yaparak USB aygıt ile haberleşme sağlanır.



Şekil 4.1 Servo motor sürücü kartı ile haberleşme algoritması.

4.2 USB KAMERADAN GÖRÜNTÜ ALMA

USB kameradaki görüntüye gerçek zamanlı ulaşabilmek için DirectShow kütüphaneleri kullanılmıştır. DirectShow, Microsoft tarafından yazılım geliştiricilerin çoklu ortam dosyaları veya akan verilerle çalışabilmesini sağlamak için yazılmış çoklu ortam yapısı ve uygulama programlama arayüzüdür (API – Application Programming Interface). DirectShow kütüphanesi kullanılarak, kameradan her görüntü alındığında bir kesilim (interrupt) oluşması sağlanmıştır. Görüntü işleme işlemleri bu kesilimler arasında yapılmıştır.



Şekil 4.2 USB kameradan görüntü alan programın algoritması.

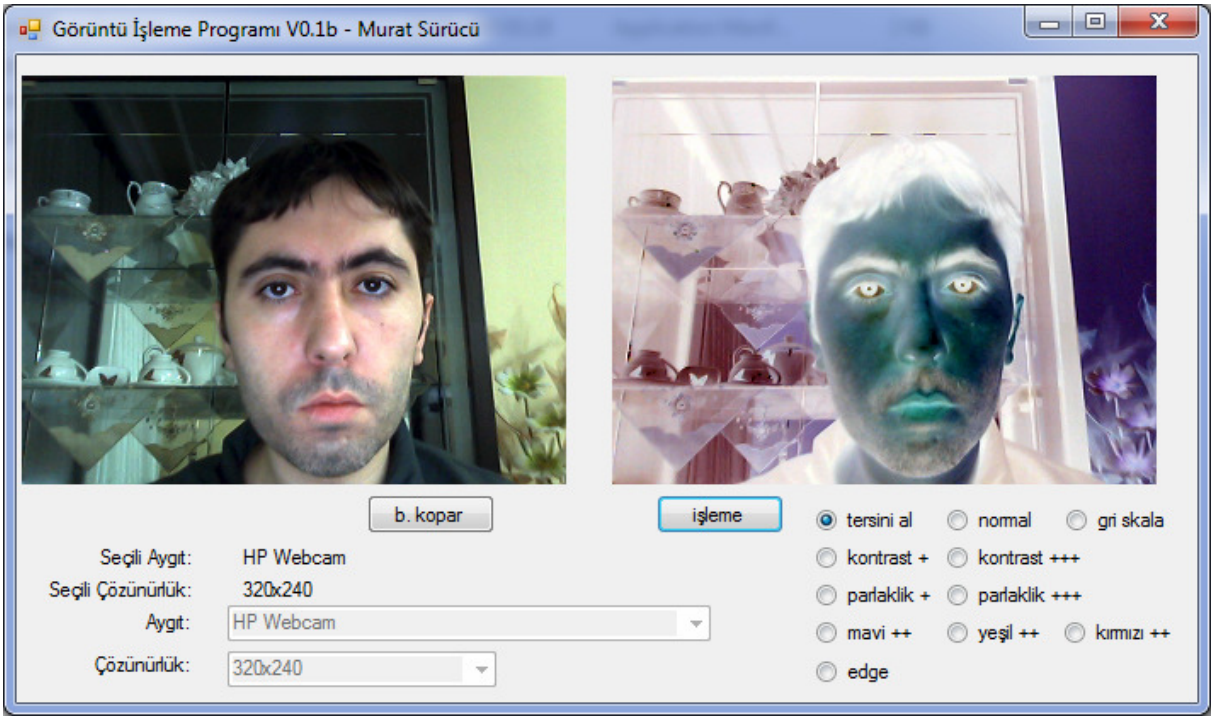
Şekil 4.2'de akış şemasında görüldüğü gibi, ilk olarak sistemde bulunan tüm görüntüleme

aygıtları tespit edilmektedir. Kullanıcının bu görüntüleme aygıtları arasından seçeceği aygıta kayıt olunmakta ve o aygıta ait görüntü aktarımları yakalanılarak işlenmektedir.

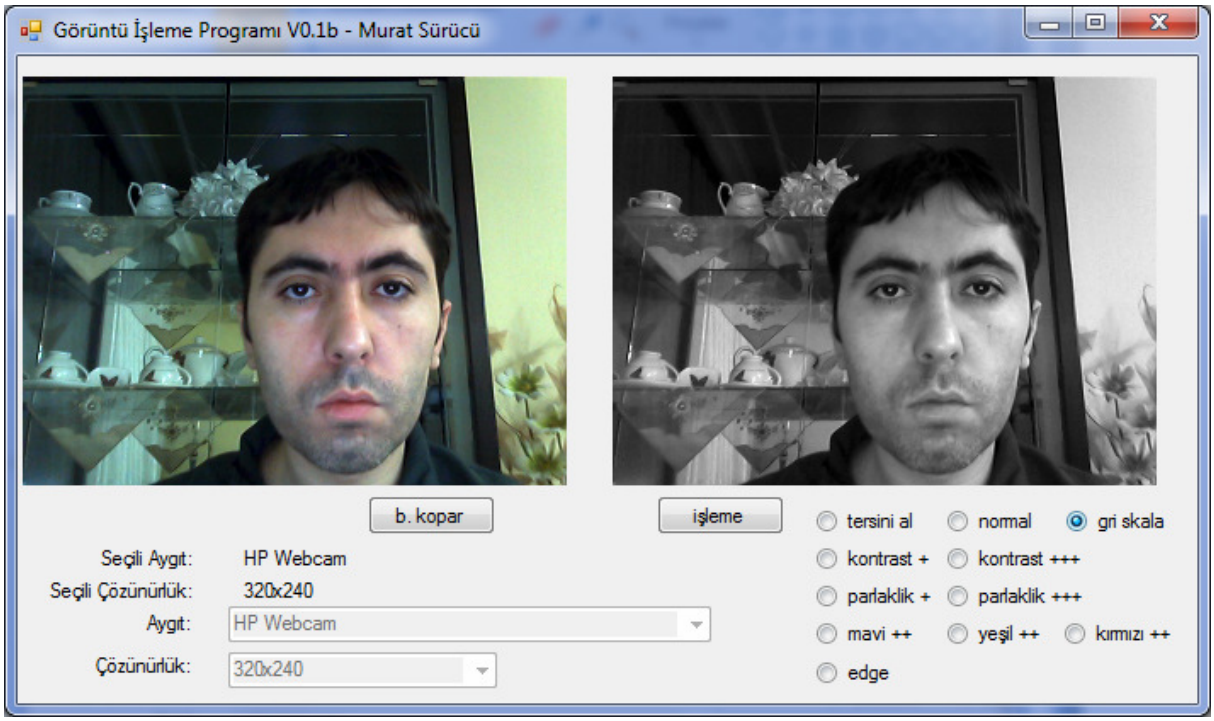
4.3 GÖRÜNTÜ İŞLEME YAZILIMI

Programın bu kısmında, DirectShow aracılığıyla hafızaya alınan görüntü çerçeveleri, hafızaya doğrudan erişim metoduyla (DMA – Direct Memory Access) işlenerek görüntü işleme filtrelerinden geçirilmektedir. Proje süresince, Şekil 4.3'te bazı ekran çıktıları görülen gri skalaya çevirme, renklerin tersini alma, zıtlık, parlaklık, renk doygunluğu ayarları ve kenar tespiti gibi pek çok filtre yazılıp denenmiştir. Soldaki çerçeveler orijinal görüntüleri, sağdaki çerçeveler ise işlemeden geçmiş görüntüleri göstermektedir. Şekil 4.3a'da sol taraftaki renkli aktif görüntünün renkleri ters çevrilerek sağ tarafa aktarılmış hali görülmektedir. Şekil 4.3b ise renkli olan soldaki orijinal çerçeveyi gri tonlamaya çeviren filtrenin çıktısını göstermektedir. Gri tonlamaya çevrimde Eşitlik 2.1'den yararlanılmıştır. Şekil 4.3c daha önce Şekil 2.18'de gösterilmiş olan Sobel maskelerinin uygulanmasıyla oluşturulmuştur. Şekil 4.3d ise orijinal çerçevedeki yeşil kanala ait bilginin baskınlaştırılmasıyla elde edilmiş örnektir.

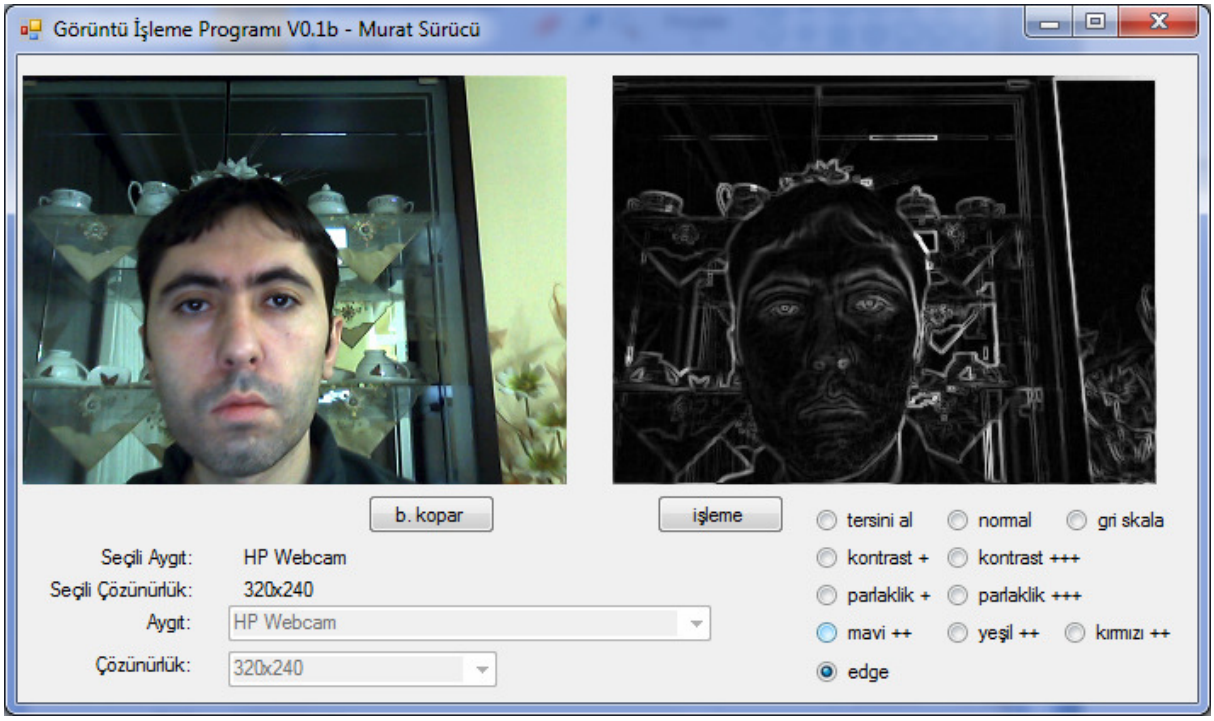
Yukarıda anlatılan yöntemler dışında sabit arkaplan, çerçeve farkı, akan ortalama, akan Gauss ortalama ve Gauss dağılımlarının birleşimi arkaplan seçme yöntemleri üzerinde de çalışılmış ve incelenen yöntemler arasında işleme hızı ve arkaplan değişikliğinden etkilenmeyişi nedeniyle, hareketli kameraya en uygun olan yöntemin değişken arkaplan yöntemi (çerçeve farkı yöntemi) olduğu görülmüştür. Bu uygunluğa rağmen çerçeve farkı yöntemi hareketli kameraya doğrudan uygulanamaz. Bu sebeple kamera hareketinin olmadığı anlarda çerçeve farkı yönteminin çalışması sağlanmıştır. Ancak hareketli kamerada hareketli nesneyi belirlemede karşılaşılan en büyük problem kameranın hareketinin ne zaman bittiğinin belli olmamasıdır. Kameraya pozisyon değişikliği komutu yollanması ile kameranın bu pozisyona ulaşması ve durağan hale gelmesi arasında geçen süre, mevcut pozisyon ile gidilecek pozisyon arasındaki konum farkına bağlı olarak değişmektedir. Ancak kameranın titreşimi, hareket bittikten bir süre sonra daha devam ettiği için, pozisyona bağlı süre hesaplaması yeterli olmamaktadır. Bu sebeple hem motor hareketinin bitmesi için gerekli süreden güvenlik payı kadar fazla bekleyen gecikmeli çerçeve farkı yöntemi (GÇFY) üzerinde, hem de görüntünün belirleyici özelliklerini kullanarak hareketi tespit eden köşe tepitli çerçeve farkı yöntemi (KTÇFY) üzerinde çalışılmıştır.



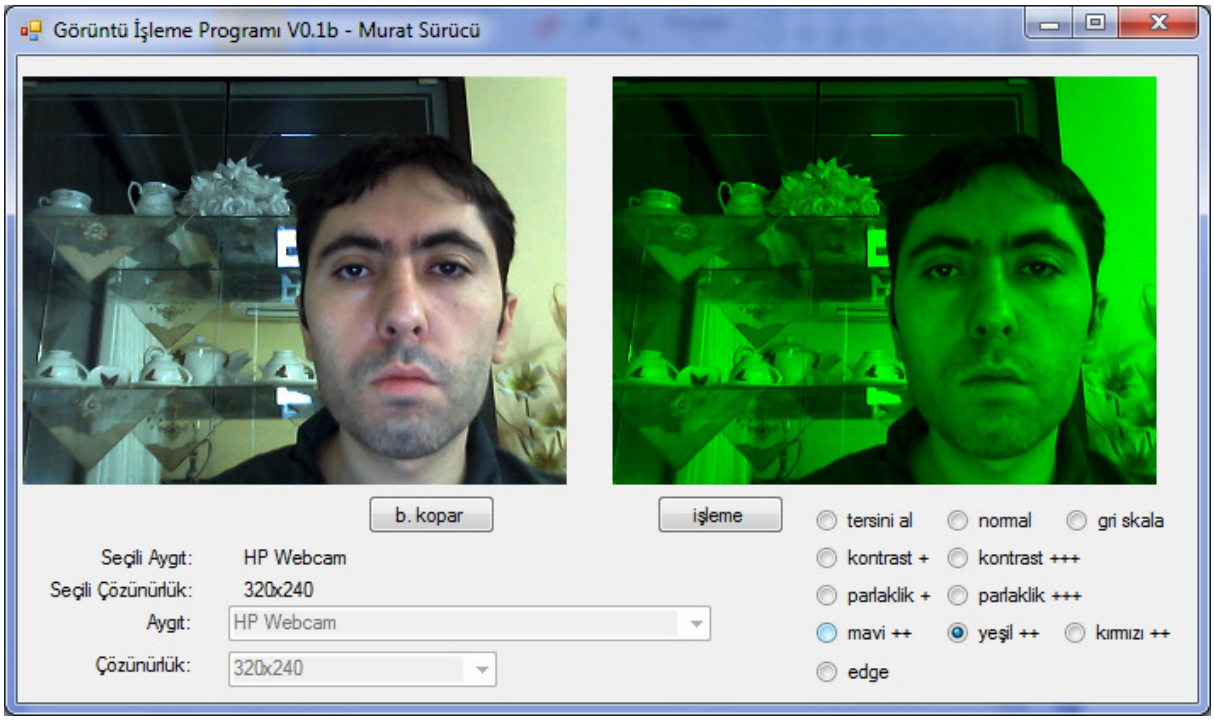
a. Tersini alma örneği.



b. Gri tonlama örneği.



c. Kenar bulma örneği.

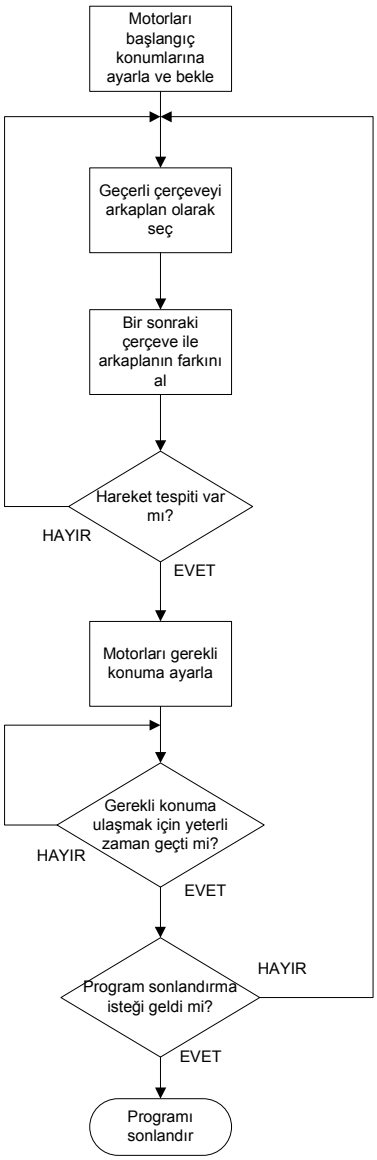


d. Renk işlemeye dair örnek.

Şekil 4.3 Bazı basit görüntü işleme filtre çıktıları.

4.3.1 Gecikmeli Çerçeve Farkı Yöntemine Dayalı Algoritma

Beklemeye dayalı algoritma, hareket tespitinin kameranın hareketsiz olduğu anlarda yapılmasına dayanmaktadır. Kamera hareket etmiyorken seçilen arkaplan ve sonraki çerçeveler karşılaştırılarak hareketli nesne tespit edilir (Şekil 4.4). Tespit edilen harekete göre kameranın gideceği pozisyon bulanık mantık kullanılarak hesaplanır. Hesaplanan konuma gidilene kadar kameradan gelen veriler görüntü işlemeye tabi tutulmazlar. Kamera hareketi ve titreşiminin biteceği tahmin edilen süre kadar sonra arkaplan seçimi ve hareket tespiti alt programları tekrar aktive edilir.

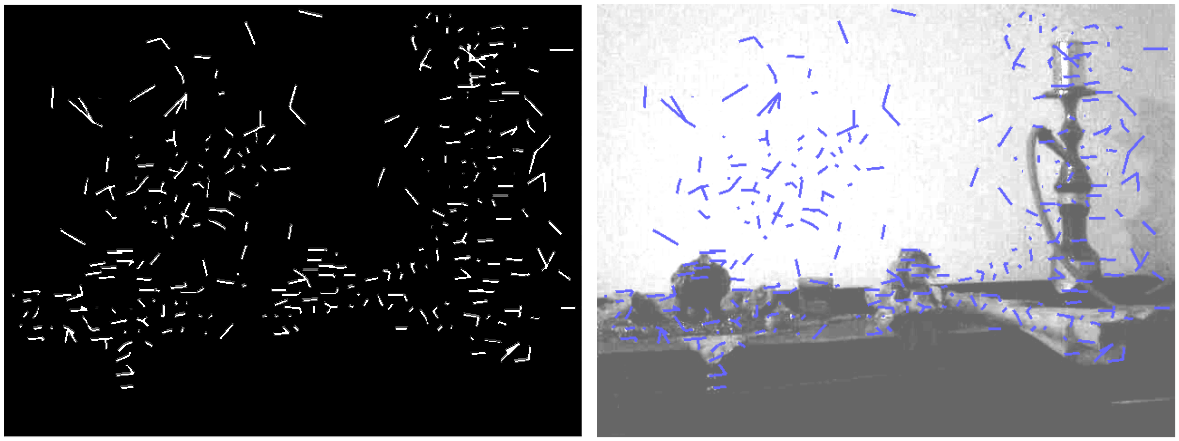


Şekil 4.4 Gecikmeli çerçeve farkı yöntemine dayalı algoritma.

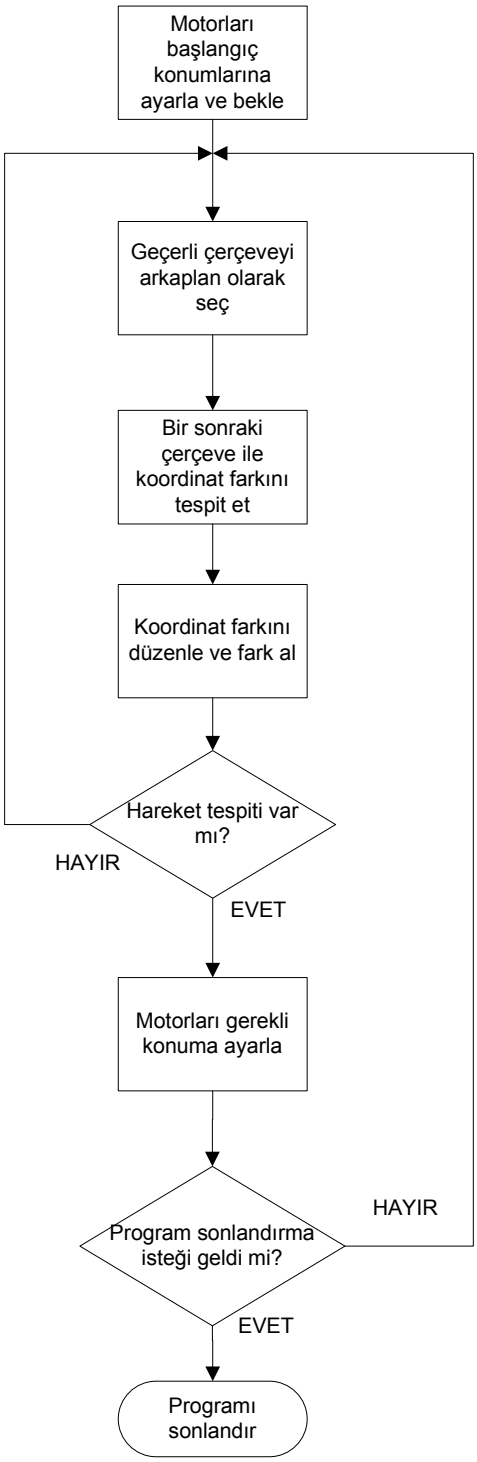
4.3.2 Köşe Tespitli Çerçeve Farkına Dayalı Algoritma

Hareket halindeyken seçilen arkaplan ile o an işlenen çerçevenin görüş açılarının farklı olması sebebiyle, iki görüntüdeki koordinatlar birbiriyle uyuşmamaktadır. Bu sebeple farklı konumlardan alınmış görüntüler arasında, dönüşüm yapılmadan bir fark alma işlemi gerçekleştirilememektedir. Hareketten kaynaklanan bu görüntü kaymasını tespit edebilmek için görüntünün takip edilebilir özelliklerine bakmak gereklidir. Bu sebeple arkaplan ve geçerli çerçeveye ait takip edilebilir köşeler seçilmiştir. Bu seçilen köşelerin eşleştirilebilmesi için Yu et al'un (2008) önerdiği düşümlerin (gradient) farkına dayalı hesaplamadan yararlanılmıştır.

Farklı açılardan alınan görüntüler için yapılan köşe belirleme işlemi sonucunda konum farkından dolayı bazı köşeler tespit edilememektedir (Şekil 2.20). Ayrıca ışık değişimlerinden veya tekrarlayan görüntülerin fazla oluşundan dolayı bazı köşelerde hatalı eşleştirilebilmektedir (Şekil 4.5). Bu sebeple bu hatalı eşleştirmeler ve eksik köşeler yüzünden hareket tespiti doğru yapılamamaktadır. Köşe tespiti ve eşleştirmesinin yanında motor açılarının konumlarına dayalı yaklaşık kayma tahmininden de yararlanılarak hatalı ve eksik eşlemelerin çoğu tespit edilebilmiştir. Son olarak tespit edilen köşe hareketlerinin histogramı çıkartılarak hareketin çoğu köşe için durduğu an tespit edilmektedir. Hareketin bittiği çerçeve sonrası çerçeve farkı yöntemi kullanılarak hareketli nesne tespit edilmektedir (Şekil 4.6).



Şekil 4.5 Köşe izleme uygulaması.



Şekil 4.6 Köşe tespitli çerçeve farkı yöntemine dayalı algoritma.

BÖLÜM 5

HAREKETLİ NESNE TAKİP ALGORİTMALARI ANALİZİ

5.1 TEST ORTAMI

Bu çalışmada kullanılan gecikmeli çerçeve farkı yöntemi (GÇFY) ve köşe tespitli çerçeve farkı yöntemi (KTÇFY) algoritmalarının test edilmesi için kapalı ve yapay ışıklandırmalı ofis ortamında değişik hızlarda yürüyen kişinin takibi esas alınmıştır. Takip edilecek kişi normal ve hızlı adımlarla kameradan 1.5 metre mesafede, kameranın bulunduğu platforma paralel olacak şekilde geçişler yapmıştır. Normal adımla yapılan yürümelerde ortalama 0,59m/sn'lik hız, hızlı adımlarla yapılan yürümelerde ise ortalama 1,22m/sn'lik hız yapılmıştır. Sonuçların başarımı hareketli kişinin tamamının ekran içinde kalması ile ölçülmüştür.

5.2 ANALİZ SONUÇLARI

Gecikmeli çerçeve farkı yöntemi ve köşe tespitli çerçeve farkı yöntemi ile normal ve hızlı adımlarla yürüyen kişinin takibinde her kategori için yirmibeş deneme yapılmıştır. Deneme başarımları Çizelge 5.1'de görülmektedir. Şekil 5.1 ve 5.2 de ise normal ve hızlı adımlarla yürüme sonucu algoritmaların başarımlarını gösteren uygulama çıktıları görülmektedir. Şekillerin sol sütununda GÇFY sonuçları, sağ sütununda ise KTÇFY sonuçları bulunmaktadır.

Gecikmeli çerçeve farkı yöntemi ile yapılan denemelerden örnek olabilecek olan görüntüler normal ve hızlı yürüme için Şekil 5.1 ve 5.2 de verilmiştir. Şekil 5.1(a,c,e) GÇFY algoritması ile normal yürüme hızındaki bir kişinin takibini göstermektedir. Başlangıç konumunda aktive edilen sistem, normal hızdaki kişiyi başarılı bir şekilde takip edip bitiş konumuna kadar şahsı görüş alanında tutabilmiştir. Şekil 5.2(a,c,e) GÇFY algoritmasının hızlı adımlarla aynı şahsın takibini göstermektedir. Hızlı adımlarla hareket eden şahıs genellikle doğru şekilde takip edilebilmekle birlikte bazı denemelerde kamera şahsı izleyememiştir.

a. GÇFY için başlangıç konumu.

b. KTÇFY için başlangıç konumu.

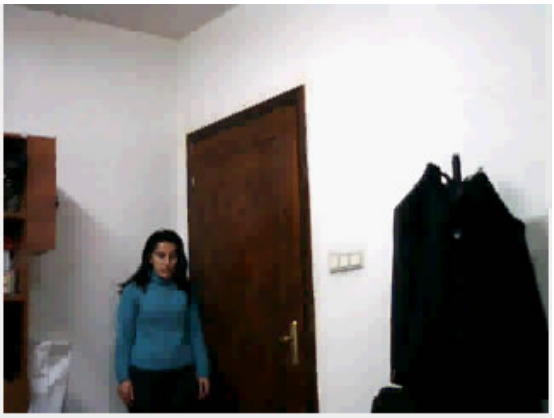
c. GÇFY için orta konum.

d. KTÇFY için orta konum.

e. GÇFY için bitiş konumu.

f. KTÇFY için bitiş konumu.

Şekil 5.1 Normal hızda yürüme için uygulama çıktıları.

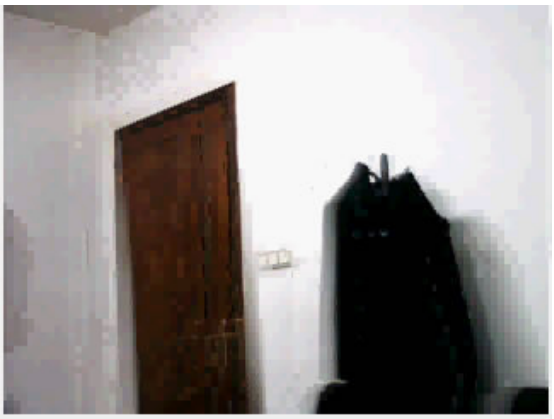


a. GÇFY için başlangıç konumu.

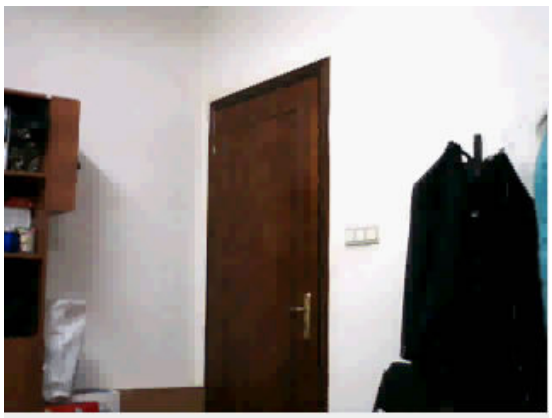
b. KTÇFY için başlangıç konumu.

c. GÇFY için orta konum.

d. KTÇFY için orta konum.



e. GÇFY için bitiş konumu.



f. KTÇFY için bitiş konumu.

Şekil 5.2 Yüksek hızda yürüme için uygulama çıktıları.

Köşe tespitli çerçeve farkı yöntemi ile de yirmibeşer deneme yapılmıştır. Bu denemelerden de örnek olabilecek görüntüler normal ve hızlı yürüme için ayrıca seçilmiş ve Şekil 5.1 ile 5.2 de gösterilmiştir. Şekil 5.1(b,d,f) KTÇFY algoritması ile normal hızda yürüyen bir kişinin takibini göstermektedir. Normal hızda yürüyen kişinin takibinde bile zaman zaman başarısız sonuçlar alınmıştır. Şekil 5.2(b,d,f) ise yine KTÇFY algoritması ile aynı şahsın hızlı adımlarla yürümesinin takibi sonucu elde edilen görüntüleri göstermektedir. Hızlı adımlarla yürüyen şahsın takibinde de KTÇFY tutarlı çalışmamıştır.

Sonuç olarak gecikmeli çerçeve farkı yöntemine dayalı olan ilk algoritma normal hızdaki hareketlerde daha başarılı sonuçlar almasına karşın (Şekil 5.1e), hızlı hareketlerin bazılarını yakalayamamıştır (Şekil 5.2e). Köşe tespitli çerçeve farkı yöntemine bağlı ikinci algoritma ise normal hızdaki hareketlerde ya nesne hareketini algılayamamış, ya da nesnenin konumunu hatalı tespit etmiştir (Şekil 5.1f). Hızlı hareketlerde de KTÇFY sonuçları başarılı değildir (Şekil 5.2f).

Çizelge 5.1 Hareketli nesne takip sonuçları.

Algoritma	**Yürüme Hızı**	**Toplam Deneme**	**Başarılı Sonuçlar**	**Başarı Yüzdesi**
GÇFY	Normal	25	25	%100
GÇFY	Hızlı	25	22	%72
KTÇFY	Normal	25	17	%68
KTÇFY	Hızlı	25	14	%56

BÖLÜM 6

SONUÇLAR VE ÖNERİLER

Bu çalışmada hareketli nesne tespiti için temel olarak çerçeve farkı yöntemi üzerinde çalışılmıştır. İki farklı algoritma ile çerçeve farkı yönteminin hareketli kameraya uyarlaması yapılmıştır. Bu algoritmalardan ilki gecikmeli çerçeve farkı yöntemi (GÇFY), ikincisi ise köşe tespitli çerçeve farkı yöntemidir (KTÇFY). Bu algoritmaların yanı sıra görüntü iyileştirmesi için gürültü yok etme metotları, hareketli nesne tespitini kolaylaştırmak içinse aşındırma ve genleştirme yaklaşımları kullanılmıştır. Aktif kamera için PIC mikrodenetleyici ile sürülen servo motorlu devre tasarlanmıştır. Tüm görüntü işleme algoritmaları C# dilinde, mikrodenetleyici programı ise Pic Basic Pro dilinde uygulamaya geçirilmiştir.

GÇFY algoritması, hareketli nesne tespiti ve takibinde daha başarılı sonuçlar vermiştir. Normal yürüme hızında %100 başarım, yüksek yürüme hızında ise %72 başarım göstermiştir. KTÇFY algoritması ise hareketli nesne tespitinde tutarsız bir başarım sergilemiştir. Normal yürüme hızında %68, yüksek yürüme hızında da %56 başarıma sahiptir. GÇFY algoritması ön tanımlı sabit bir gecikme ile arkaplan ve çerçeve seçimi yapıyor olmasına rağmen, adaptif yaklaşıma sahip olan KTÇFY algoritmasından daha başarılı sonuçlar üretmiştir. Bunun ana sebebi ise KTÇFY algoritmasının yüksek işlem gücü gereksinimi ile algoritmaya bağımlı önlenemeyen hatalı köşe eşlemelerdir. Yüksek işlem gücü gereksinimi sebebiyle çerçeve atlamaları yaşanmaktadır. Hatalı köşe eşlemeleri ise hareketli nesne tespitinde yanlış sonuçlar üretilmesine sebep olmaktadır. Ayrıca KTÇFY algoritması için gerekli olan kamera geometrisinin bütün parametrelerinin hesaplanması yüksek işlem gücü ihtiyacı dolayısıyla yapılamadığı için, geometrik dönüşümlerde hatalar oluşmakta ve dolayısıyla hareketli nesnenin yerinin tayininde sorunlar yaşanmaktadır. Bu sorunlar sebebiyle de hareketli nesnenin küçük hareketinde bile kamera açısı fazla değişim gösterebilmektedir.

KTÇFY algoritmasının iyileştirilmesiyle GÇFY algoritmasından daha iyi başarım sağlayabileceği düşünülmektedir. Bu iyileştirme, algoritmanın karmaşıklık derecesini

düşürmeyi ve hatalı köşe eşlemelerini elemeyi içermelidir. Gelecekte algoritmalar geliştirilerek daha kapsamlı uygulamalarda kullanılabilir. Özellikle akıllı sınıf, sunum, konferans gibi uygulamalar için yüz tanıma yöntemleri de algoritmalara dahil edilerek başarımı daha da arttırılabilir. Genel uygulamalara yönelik yazılmış olan algoritmalar, istenilen uygulamaya yönelik uygun önişlemlerden geçirilirse daha tutarlı ve hızlı işlem yapılabilir.

KAYNAKLAR

Balta C, Öztürk S ve Oysu C (2009) Görme Destekli Kartezyen Robot İçin Kenar Resmi Vektorizasyon Uygulaması, Bildiri 81, *EEBBM 13. Ulusal Kongresi*, ODTÜ, s. 1-4

Bastanlar Y ve Yardimci Y (2007) Selecting Image Corner Points Using Their Corner Properties, *Signal Processing and Communications Applications*, SIU IEEE 15th, Eskisehir, pp. 193-196

Basu A (1995) Active Calibration of Cameras: Theory and Implementation, *Telecommun. Res. Labs.*, Edmonton, Alta., Canada, pp. 256-265

Bovik A (1999) *Handbook of Image and Video Processing*, ISBN:0121197905, Academic Press, Inc. Orlando, FL, USA, pp. 95-99

Castleman K R (1995) *Digital Image Processing*, Prentice Hall, ISBN:0132114674, pp. 447-547

Gonzalez R C and Woods R E (1992) *Digital Image Processing*, Addison-Wesley Pub (Sd); 3rd edition, ISBN: 0201508036, pp. 416-423

Harris C G and Stephens M (1988) A Combined Corner and Edge Detector, *Proc. of Fourth Alley Vision Conference*, Manchester, pp. 147-151

Jung B and Sukhatme G S (2004) Detecting Moving Objects using a Single Camera on a Mobile Robot in an Outdoor Environment, *The 8th Conference on Intelligent Autonomous Systems*, Amsterdam, The Netherlands, pp. 980-987

Kang B (2007) A Review on Image and Video Processing, *International Journal of Multimedia and Ubiquitous Engineering*, 2: 49-63

Kim K K, Cho S H, Kim H J and Lee J Y (2005) Detecting and Tracking Moving Object Using An Active Camera, *Advanced Communication Technology, ICACT*, Phoenix Park, pp. 817-820

Loy G and Barnes N (2004) Fast Shape-Based Road Sign Detection For A Driver Assistance System, *Intelligent Robots and Systems*, pp. 70-75

Moravec H P (1977) Towards Automatic Visual Obstacle Avoidance (short version), *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, MIT, Cambridge, 584 pp.

Piccardi M (2004) Background Subtraction Techniques: A Review, Systems, *IEEE international conference on systems, man & cybernetics,* 4: 3099– 3104

KAYNAKLAR (devam ediyor)

Seul M, O'Gorman L and Sammon M J (2000) Practical Algorithms for Image Analysis – Description, Examples, and Code, Cambridge University Press, ISBN:0521660653, pp. 68-74

Stauffer C and Grimson W E L (1999) Adaptive Background Mixture Models For Real-Time Tracking, *Computer Vision and Pattern Recognition, IEEE Computer Society Conference*, pp. 246-252

URL-1 (2010) http://robot.cmpe.boun.edu.tr/593/gorusimge/I_c_cindekiler.html (11.12.2009).

URL-2 (2010) http://www.keil.com/usb/chips.asp (08.05.2009).

URL-3 (2010) http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=111 & mid=10&lang=en&pageId=74 (04.01.2010).

URL-4 (2010) http://en.wikipedia.org/wiki/Reduced_instruction_set_computer (25.12.2009).

URL-5 (2010) http://www.picbasic.org/proton_plus.php (15.01.2010).

URL-6 (2010) http://www.usb.org (03.05.2009).

URL-7 (2010) http://www.mecanique.co.uk/code-studio/ (15.01.2010).

Yu H, Ren C and Qiao X (2008) A New Corner Matching Algorithm Based On Gradient, *Signal Processing, ICSP*, pp. 1346-1349

Zuo F and de With P H N (2005) Real-Time Embedded Face Recognition For Smart Home, *Consumer Electronics, IEEE Transactions on*, Eindhoven Univ. of Technol., Netherlands, 51: 183-190

ÖZGEÇMİŞ

Murat Sürücü 1979'da İzmir'de doğdu. İlk ve orta öğrenimini İzmir'de tamamladı. İzmir Çınarlı Teknik Lisesi Bilgisayar Donanımı Bölümü'nden mezun olduktan sonra 1998 yılında Süleyman Demirel Üniversitesi Mühendislik Mimarlık Fakültesi Elektronik ve Haberleşme Bölümü'ne girdi. 2002'de "iyi" derece ile mezun olduktan sonra radyo vericileri ve kablosuz otomasyon sistemleriyle ilgili olarak Antecom Elektronik'te çalışmaya başladı. 2003'de SDÜ Bilgisayar Bilimleri Araştırma ve Uygulama Merkezinde işe başlayarak elektronik devre tasarımı ve üretimi üzerine çalıştı. 2005 yılından beri ZKÜ Bilgi İşlem Daire Başkanlığı'nda ağ sorumlusu olarak Uzman kadrosunda görev yapmaktadır. Halen ZKÜ Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda yüksek lisans programına devam etmektedir.

ADRES BİLGİLERİ

Adres : Zonguldak Karaelmas Üniversitesi Merkez Kampus Rektörlük Binası
Bilgi İşlem Daire Başkanlığı Dördüncü Kat
Merkez / ZONGULDAK

Telefon : 0 372 2574010 - 1502

Faks : 0 372 2573162

E-posta : msurucu@karaelmas.edu.tr – msurucu@gmail.com

Murat SÜRÜCÜ