Matlab İle Görüntü İşleme ve Kenar Belirlemede Yeni Filtreleme Yöntemi

Gürkan Kaplan

YÜKSEK LİSANS TEZİ

Matematik-Bilgisayar Anabilim Dalı

Haziran 2017

Image Processing With Matlab and New Filtering Method For Edge Detection

Gürkan Kaplan

MASTER OF SCIENCE THESIS

Department of Mathematics-Computer

June 2017

Matlab İle Görüntü İşleme ve Kenar Belirlemede Yeni Filtreleme Yöntemi

Gürkan Kaplan

Eskişehir Osmangazi Üniversitesi

Fen Bilimleri Enstitüsü

Lisansüstü Yönetmeliği Uyarınca

Matematik-Bilgisayar Anabilim Dalı

Bilgisayar Bilimleri Bilim Dalında

YÜKSEK LİSANS TEZİ

Olarak Hazırlanmıştır

Danışman: Doç. Dr. Alper Odabaş

Haziran 2017

ONAY

Matematik-Bilgisayar Anabilim Dalı Yüksek Lisans öğrencisi Gürkan Kaplan' ın YÜKSEK LİSANS TEZİ olarak hazırladığı "**Matlab İle Görüntü İşleme ve Kenar Belirlemede Yeni Filtreleme Yöntemi**" başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek oybirliği ile kabul edilmiştir.

Danışman : Doç. Dr. Alper Odabaş

İkinci Danışman : -

Yüksek Lisans Tez Savunma Jürisi:

Üye : Prof. Dr. Zekeriya Arvasi

Üye : Doç. Dr. İlker Akça

Üye : Doç. Dr. Alper Odabaş

Üye : Yrd. Doç. Dr. Mustafa Saltan

Üye : Yrd. Doç. Dr. Ahmet Faruk Aslan

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... tarih ve ..... sayılı kararıyla onaylanmıştır.

Prof. Dr. Hürriyet Erşahan

Enstitü Müdürü

ETİK BEYAN

Eskişehir Osmangazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kılavuzuna göre, Doç. Dr. Alper Odabaş danışmanlığında hazırlamış olduğum ”Matlab İle Görüntü İşleme ve Kenar Belirlemede Yeni Filtreleme Yöntemi” başlıklı YÜKSEK LİSANS tezimin özgün bir çalışma olduğunu; tez çalışmamın tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı; tezimde verdiğim bilgileri, verileri akademik ve bilimsel ilke ve kurallara uygun olarak elde ettiğimi; tez çalışmamda yararlandığım eserlerin tümüne atıf yaptığımı ve kaynak gösterdiğimi ve bilgi, belge ve sonuçları bilimsel etik ilke ve kurallara göre sunduğumu beyan ederim.

06/06/2017

Gürkan Kaplan

ÖZET

Bu tez çalışmasının amacı, görüntü işleme alanını Matlab kodlarıyla birlikte detaylı inceledikten sonra kenar belirlemede yeni bir filtreleme yöntemi elde edip diğer filtrelerle retinal görüntü üzerinde görsel karşılaştırmasının yapılmasıdır.

Yedi bölümden oluşan çalışmamızın birinci bölümünde görüntü işlemenin temeli olan bilgiler aktarılmıştır. İkinci bölümde Matlab programı ile görüntü işleme konusu detaylandırılmıştır. Üçüncü bölümde çalışmamızın temelini oluşturan kenar belirleme filtreleri yazmış olduğumuz kodlarla detaylandırılarak verilmiştir. Dördüncü bölümde görüntü işlemenin tıp alanındaki bazı uygulamaları ve yardımcı programlar aktarılmıştır. Beşinci bölümde özvektörler aracılığıyla yapılmış literatür çalışmalarına değinilmiştir. Altıncı bölümde kenar belirlemede elde edilen yeni filtreleme yöntemi anlatıldıktan sonra retinal görüntüde diğer filtrelerle görsel karşılaştırması yapılmıştır. Son bölümde ise elde ettiğimiz sonuçlar ve bu sonuçlara istinaden bundan sonra yapılmasını önerdiğimiz çalışmalara değinilmiştir.

Anahtar Kelimeler: Resim işleme, medikal görüntü işleme, filtreler, retina.

SUMMARY

The aim of this thesis is to obtain a new filtering method in edge detection after detailed analysis of the image processing field with Matlab codes and visual comparison of the retinal image with other filters.

The study consists of five chapters. In the first part, basic information on image processing is transferred. In the second part, Image processing with Matlab program is detailed. In the third part, the edge detection filters, which form the basis of our work, are given in detail with the codes we have written. In the fourth part, Some applications and utilities in the medical field of the image processor have been transferred. In the fifth part, The literature studies made through eigenvectors are mentioned. In the sixth part, After describing the new filtering method obtained in edge detection, visual comparison was made with other filters in the retinal image. In the last part, the results we obtained and the studies we propose to make these results in the future are given.

Keywords: Image processing, medical image processing, filters, retina.

TEŞEKKÜR

Matlab İle Görüntü İşleme ve Kenar Belirlemede Yeni Filtreleme Yöntemi adlı tez çalışmam süresince ihtiyaç duyduğum her konuda bana yardımcı olan, engin bilgilerini ve deneyimlerini benimle paylaşan, çalışmamın yürütülmesi sırasında desteğini esirgemeyen ve bana inanan değerli danışmanım Sayın Doç. Dr. Alper Odabaş hocama ve burada teşekkür ederken bile yetersiz kalacağından dolayı üzüntü duyduğum, hayatımın her aşamasında geldiğim her noktada emekleri olan, sevgili babam, annem başta olmak üzere kardeşlerime teşekkürlerimi iletmeyi bir borç bilirim.

Gürkan Kaplan

Haziran 2017

İÇİNDEKİLER

Sayfa

ÖZET **vi**

SUMMARY **vii**

TEŞEKKÜR **viii**

İÇİNDEKİLER **ix**

ŞEKİLLER DİZİNİ **xiii**

1. GİRİŞ **1**

1.1 Görüntü Nedir 1

1.2 Piksel Nedir 1

1.3 Görüntüde Aritmetik İşlemler 1

1.3.1 Piksel ekleme 2

1.3.2 Piksel çıkarma 2

1.3.3 Piksel çarpma 3

1.3.4 Piksel bölme 3

1.3.5 Karıştırma 3

1.4 Görüntüde Mantık Operatörleri 4

1.4.1 Not operatörü 4

1.4.2 Or, xor operatörleri 4

1.4.3 And operatörü 5

1.5 Görüntülerde Nokta Tabanlı İşlemler 5

1.5.1 Logaritmik dönüşüm 5

İÇİNDEKİLER (devam)

1.6 Görüntü İşleme ve Tarihçesi 6

1.7 Görüntü İşlemede Kullanılan Programlar 7

2. GÖRÜNTÜ İŞLEMEDE TEMEL KAVRAMLAR 8

2.1 Matlab'ta Bazı Temel Görüntü İşleme Fonksiyonları 9

2.2 Matlab Görüntü İşleme Araç Kutusu (Matlab Image Processing Toolbox) . . 9

2.3 Görüntü Türleri 11

2.3.1 Binary görüntü 12

2.3.2 Grayscale görüntü 12

2.3.3 RGB görüntü 13

2.3.4 Çok spektrumlu görüntü 14

2.4 Görüntü Formatları 14

2.5 Matlab'ta Veri Sınıfları 14

2.6 Görüntü Tür Dönüşümleri 15

2.7 Temel Görüntü İşleme Fonksiyonları 17

2.7.1 imread fonksiyonu 17

2.7.2 imshow fonksiyonu 17

2.7.3 imtool fonksiyonu 18

2.7.4 imwrite fonksiyonu 18

2.7.5 imfinfo fonksiyonu 18

2.7.6 imhist fonksiyonu 20

2.7.7 Histogram eşitleme 22

2.7.8 imresize fonksiyonu 25

2.7.9 imrotate fonksiyonu 25

2.7.10 imnoise fonksiyonu 27

İÇİNDEKİLER (devam)

2.7.11 Görüntüyü çevirme ve odaklanma 27

3. FİLTRELER **30**

3.1 Bozukluk Giderme Filtreleri 30

3.1.1 Ortalama(Mean) filtre 30

3.1.2 Medyan filtresi 32

3.1.3 Gauss filtresi 36

3.2 Kenar Belirleme Filtreleri 38

3.2.1 Sobel kenar filtresi 39

3.2.2 Prewitt kenar filtresi 41

3.2.3 Roberts kenar filtresi 43

3.2.4 Canny kenar filtresi 44

3.2.5 Laplace filtresi 45

3.2.6 Log filtresi 46

4. TIP ALANINDA GÖRÜNTÜ İŞLEME UYGULAMALARI **48**

4.1 Sarı Nokta Hastalığı (Age-related Macular Degeneration (AMD)) 48

4.2 Özellik Çıkarımı 52

4.2.1 İyi-kötü huylu ben tespiti 52

4.2.2 Retinadan kan damar özelliklerinin çıkarılması 53

4.2.2.1 Imagej 54

4.2.2.2 Aria 57

5. LİTERATÜR ARAŞTIRMASI **60**

6. KENAR BELİRLEMEDE YENİ FİLTRELEME YÖNTEMİ **61**

7. SONUÇ VE ÖNERİLER **72**

İÇİNDEKİLER (devam)

KAYNAKLAR DİZİNİ **73**

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

1.1 Logaritmik dönüşüm uygulanmış görüntü 6

2.1 HSV renk uzayı örneği 11
2.2 RGB renk uzayı 11
2.3 Binary görüntü örneği 12
2.4 Grayscale görüntü örneği 13
2.5 Matlab inspect pixel values penceresinden her pikseldeki renk karışımı. 13
2.6 Rgb-grayscale dönüşüm örneği 16
2.7 Grayscale-binary dönüşüm örneği 17
2.8 Piksel değerleri 20-100 arasındaki görüntü 18
2.9 Yüklenen görüntünün histogramı 21
2.10 Görüntünün ayrı ayrı renk histogramları 22
2.11 Eşitlenmiş histogram 23
2.12 LBP uygulanmış görüntünün histogramı 25
2.13 Döndürülmüş görüntü 26
2.14 Ters çevrilmiş grayscale görüntü 28
2.15 Ters çevrilmiş rgb görüntü 28
2.16 Yatay eksende ters çevrilmiş görüntü 29

3.1 Ortalama filtreden geçirilmiş görüntü 32
3.2 Medyan filtreden geçirilmiş görüntü 34
3.3 Bozulmuş görüntüye medyan uygulanması 35
3.4 Bozulmuş görüntüye renkli medyan uygulanması 36
3.5 Gauss filtreli görüntü 38

3.6 Sobel filtreli görüntü 40
3.7 Renkli sobel filtreli görüntü 41
3.8 Prewitt filtreli görüntü 43
3.9 Roberts filtreli görüntü 44
3.10 Canny filtreli görüntü 45
3.11 Laplace filtreli görüntü 46
3.12 Log filtreli Lena görüntüsü 47

4.1 Detaylı göz altı görüntüsü 49
4.2 Filtrelenmiş sarı noktalı ve normal göz görüntüleri 50
4.3 Vücutta ben görüntüsü 53
4.4 Canny filtresinden geçirilmiş ben görüntüsü 53
4.5 imagej programı kullanıcı arayüzü 54
4.6 imagej programı uygulama görüntüsü 55
4.7 Aria programı kullanıcı arayüzü 57
4.8 Aria programı damar görüntüsü 58
4.9 Aria programı detaylı damar görüntüsü 58
4.10 Aria programı damar özellikleri 59

5.1 Binary formatta retinal görüntü 60

6.1 Retinal görüntü 61
6.2 Çeşitli filtrelerle filtrelenmiş retinal görüntü 62
6.3 Uygun threshold değerleriyle filtrelenmiş retinal görüntü 63
6.4 Filtremiz ile filtrelenen retinal görüntü 67
6.5 3×3'lük ortancası alınmış filtremiz ile filtrelenen retinal görüntü 70
6.6 Filtrenin diğer filtrelerle görsel olarak karşılaştırılması 71

7.1 Korneal görüntü 72

1. GİRİŞ

Bu bölümde, tezde kullanılan bazı temel kavramlar açıklanarak ele alınan konuların daha anlaşılır hale getiririlmesi amaçlanmaktadır.

1.1 Görüntü Nedir

Belirli özelliği ve renk yoğunluğu olan bir resmin dijital ortama belirli boyutlarda aktarılarak oluşturulan haline görüntü denir. Gerçek yaşamda herhangi bir görüntü $f(x,y)$ şeklinde iki değişkenin fonksiyonu olarak tanımlanır. Burada x,y görüntünün üzerindeki gerçek koordinatlar olup f ise parlaklık gibi görüntünün maruz kaldığı veya isteyerek maruz bırakıldığı bir şiddet birimidir (Qidwai, 2009), (Solomon ve Breckon, 2008).

1.2 Piksel Nedir

Piksel (picture element), görüntüyü oluşturan parçalardan herbirine verilen isimdir. Anlamsal olarak ise bir görüntüyü oluşturan parçalardan herbirine verilen isme denir. Görüntünün üzerindeki (x,y) koordinatlarının her birine verilen isim olarak da açıklanabilir. Yapılan bütün işlemler pikseldeki değerin yorumlanması ve istenen amaca yönelik işlemler yapılması üzerine kuruludur. Örneğin; tıp alanında çalışan biri medical imaging yaparken tomografi cihazından çıkan görüntünün veya manyetik rezonans cihazından çıkan görüntünün yoğunluğuna bakarken, diğeri örneğin, göz doktoru ise gözdeki kan damarlarının uzunluğu, büyüklüğü veya yoğunluğuyla ilgilenmektedir. Bu sebeple pikseldeki verinin yorumlanması veya yorumlamaya uygun hale getirilmesi görüntü işlemenin ana konusudur.

Pikseller üzerinde yapılabilecek işlemler, tezin bu kısmında anlatılacaktır.

1.3 Görüntüde Aritmetik İşlemler

Aynı boyutlu veya uygun bir skaler ile işleme tabi tutulan görüntüler üzerinde aritmetik işlemler uygunabilir. İlk durumda iki aynı boyuttaki görüntüyü oluşturan matrisler üst üste oturtulup yeni görüntü matrisi elde edilmiş olur. Diğer durumda ise görüntüyü oluşturan matrisin

her elemanı belirli bir skaler ile toplanarak yeni görüntü elde edilir (Qidwai ve Chen, 2009).

1.3.1 Piksel ekleme

Mevcut olan aynı boyuttaki iki veya daha fazla görüntü kullanılarak aynı boyutta başka bir görüntü elde etme işlemidir. Yeni elde edilen görüntüdeki her piksel kullanılan diğer görüntülerdeki aynı piksellerin toplamı şeklinde elde edilir. Yani, K, L, M, T görüntü matrisi olmak üzere,

$$K(x,y) = L(x,y) + M(x,y) + ... + T(x,y)$$

şeklinde ifade edilir. Burada x, y matris üzerindeki satır ve sütun indisini ifade etmektedir. Bu işlem eldeki bir görüntüyü belirli bir skaler ile toplayıp yeni görüntü elde etme şeklinde de yapılır. s sabit olmak üzere,

$$K(x,y) = L(x,y) + s$$

şeklinde ifade edilir. Bu durumda ise piksellerde boyut aşma sorunu ile karşılaşılması mümkündür. Yani gri görüntüyü düşünecek olursak her piksel 8 bitlik bir veri ile temsil edilirken yani 0-255 arasında değer alırken piksel toplama işlemlerinden sonra yeni görüntünün piksel değerleri bu değeri aşabilmektedir. Bu durumda normal değeri aşılan pikseller bitsel olarak izin verilen maksimum değerini alır. Bu durum literatürde doygunluk (saturation) olarak adlandırılır.

1.3.2 Piksel çıkarma

Piksel çıkarma işlemi eldeki aynı boyuttaki iki görüntünün aynı piksellerinin birbirinden çıkarılarak yeni görüntü elde etme işlemidir. Bu işlem aynı zamanda eldeki görüntüden belirlenen herhangi skalerin çıkarılmasıyla da elde edilir. Piksel çıkarma işleminde bazı piksellerde negatif çıkma durumu olabileceği için mutlak değer fonksiyonuyla işlemi gerçekleştirmek gereklidir.

$$K(x,y) = |L(x,y) - M(x,y)|$$

şeklindedir. Skaler ile piksel çıkarma işlemi ise,

$$K(x,y) = |L(x,y) - s|$$

şeklindedir. Eğer her piksel değeri RGB görüntülerde olduğu gibi birkaç farklı değer almakta ise bu durumda iki görüntü arasındaki çıkarma işlemi her pikseldeki her değerin kendi arasında çıkarılması ile hesaplanır.

1.3.3 Piksel çarpma

Görüntü işlemede tüm işlemler matrislerle olduğu için burada bir noktaya dikkat edilmesi gerekir. Piksel toplama ve piksel çıkarma işlemlerinde normal matrissel toplama ve çıkarma işlemi yapıldı. Fakat matris çarpma işlemi bilindiği gibi matrissel çarpım değil noktasal çarpım yani her matrisin aynı indisindeki değerlerin çarpımı şeklindedir. Yani,

$$K(x,y) = L(x,y) \ .\times \ M(x,y)$$

şeklindedir. Sabit ile çarpım ise,

$$K(x,y) = L(x,y) \ .\times \ s$$

şeklindedir. Burada da her piksel RGB gibi birkaç farklı değer almakta ise aynı pikseldeki aynı renk değerleri birbiriyle noktasal çarpılmaktadır. Aynı şekilde çarpım sonucu oluşan yeni matrisin oluşturduğu piksel değerleri maksimum değerini aşarsa alabileceği maksimum değere eşitlenir.

1.3.4 Piksel bölme

Piksel bölme işleminde iki görüntünün aynı piksel değerleri bölünerek yeni görüntü elde edilir. Piksel bölme işlemlerinde genellikle iki görüntünün piksellerinin birbirine bölünmesi şeklinde değil de bir görüntünün skaler ile bölünmesi kullanılmaktadır. Yani,

$$K(x,y) = L(x,y) \div M(x,y)$$

şeklindedir. Skaler ile bölme ise,

$$K(x,y) = L(x,y) \div s$$

şeklindedir. Bölme işlemi sonucunda oluşan değer için integer kullanılıyorsa tamsayıya dönüştürülür.

1.3.5 Karıştırma

Karıştırma işlemi aynı boyuttaki iki görüntünün ya başka bir görüntü ile ya da skaler ile işleme sokulmasıdır. Elde edilen görüntü ilk iki görüntünün her piksel değerinin bir lineer kombinasyonu şeklindedir. Bu kullanılan lineer kombinasyon katsayılarını ya kullanıcı belirler ya da belirli oranlar kullanılır. Yani, $K,\ L,\ M$ görüntü matrisi olmak üzere

$$K(x,y) = s \times L(x,y) + (1-s) \times M(x,y)$$

şeklinde formülize edilir. Burada s, karıştırma oranıdır.

1.4 Görüntüde Mantık Operatörleri

Bu kısımda görüntüler arasındaki mantık operatörlerine değinilecektir. Buradaki her işlem, görüntülerin birbirine karşılık gelen pikselleri arasında yapılmaktadır.

1.4.1 Not operatörü

Bu operatör pikselde bulunan değerin görüntü türündeki tersinin alınması işlemini yapar. Binary görüntüde 0 lar 1, 1 ler 0 olur. Gri görüntüde ise piksel değerinin 8 bit olması nedeniyle 255'ten çıkarılmış hali olurken, RGB görüntüde ise pikseldeki 3 ayrı renk değerinden dolayı 24 bit yani 3*8 bit olması nedeniyle ayrı ayrı 255'ten çıkarılmış halidir. EB görüntü türündeki alabileceği en büyük değer, R görüntü matrisi, x,y ise matris üzerindeki satır ve sütun indisini ifade etmek üzere,

$$R(x,y) = EB - R(x,y)$$

şeklinde gösterilir.

1.4.2 Or, xor operatörleri

Mantıksal or, xor operatörleri özellikle hareketli görüntüler arasında objelerin tespiti için binary görüntülerden obje çıkarımı konusunda önemli bir araçtır. Bu işlemler yapılırken renkli veya gri görüntülere threshold uygulanarak binary görüntüye dönüştürülür. Threshold, renkli veya gri görüntülerin threshold değerinin altında veya üstünde kalma durumuna göre istenilen değere atanma işlemidir.

```
>> thresh=0.5;
>> if x(i,j)<thresh
>> x(i,j)=0;
>> else
>> x(i,j)=1;
>> end
```

şeklinde ifade edilebilir. Threshold değeri, kullanılan görüntü türünün alabildiği değerler arasında olmalıdır. Örneğin, istenilen binary görüntüde dönüşüm ise $[0,1]$ arasında bir değer seçilmelidir.

1.4.3 And operatörü

Mantıksal and operatörü özellikle görüntüler arasındaki farklılıkların tespitinde kullanılır.

```
>> x=imread('D:\Matlab\manzara1.jpg');
>> y=imread('D:\Matlab\manzara2.jpg');
>> xb=im2bw(x);
>> yb=im2bw(y);
>> subplot(131),imshow(xb);
>> subplot(132),imshow(yb);
>> sonuc=xor(xb,yb);
>> sonuc2=and(xb,yb);
>> subplot(133),imshow(sonuc);
>> subplot(134),imshow(sonuc2);
```

şeklinde yazılabilir. Burada görüntülerin boyutlarının aynı olmasına dikkat edilmelidir.

1.5 Görüntülerde Nokta Tabanlı İşlemler

Görüntüdeki en düşük ve en yüksek piksel değerleri arasındaki fark, görüntünün dinamik aralığı olarak tanımlanır. Nokta tabanlı işlemler, görüntünün bu dinamik aralığını ihtiyaca yönelik olarak çeşitli dönüşümler ve gösterimlerle değiştirerek görüntünün amaca uygun hale gelmesine yardımcı olur. Bu işlemler görüntünün kontrastının artırılıp azaltılmasını sağlayarak görüntüyü uygun hale getirir. Bu kısımda nokta tabanlı işlemlerden logaritmik dönüşüme değinilecektir.

1.5.1 Logaritmik dönüşüm

Görüntünün dinamik aralığı belirli bölgesindeki piksel değerinin logaritma değeriyle yer değiştirmesiyle artırılıp azaltılır (Solomon ve Breckon, 2008). S (source) giriş

görüntü matrisi, T (target) çıkış görüntü matrisi olmak üzere $T(x,y) = \ln(S(x,y))$ şeklinde ifade edilir. Burada piksel değerleri $0-255$ arasında olacağından ve doğal logaritmada 0 tanımsız olduğu için formül şu şekilde kullanıma uygun hale getirilir. $c,\ \sigma \in \mathbb{R}$ olmak üzere

$$T(x,y) = c \times \ln(1 + (e^{\sigma} - 1) \times S(x,y)$$

burada σ değeri ne kadar yüksekse piksel değeri 255'e o kadar yaklaşır. Bu sayede görüntünün kontrastının artırılması veya tersi durumunda da azaltılması sağlanır. Gösterimde yer alan c, skaler bir sayıyı ifade ederken doğal logaritma içindeki 1 ile toplama ise tanımsızlık durumundan kurtulmak için yazılmıştır. Gösterimdeki görüntünün dinamik aralığını değiştirme işlemi σ parametresi ile yapılmaktadır.

Logaritmik dönüşümün kullanılmasının temel sebebi görüntüdeki düşük yoğunluklu veya karanlık olan bölgedeki piksel değerlerini arttırarak veya azaltarak görüntünün yorumlanmaya uygun hale getirilmesine yardımcı olmaktır.

Şekil 1.1: Logaritmik dönüşüm uygulanmış görüntü

1.6 Görüntü İşleme ve Tarihçesi

Dijital ortama alınarak elde edilen görüntü matrisi üzerinde, istenilen amaca yönelik olarak işlem yapıp yeni bir görüntü matrisi elde etme işlemine görüntü işleme denir.

Görüntü işlemede ilk uygulama 1920'lerin başlarında gerçekleştirildi. Görüntü belirli aşamalardan geçirilerek nihai kullanıcıya ulaştırıldı (Namee, 2012), (Anonim, 2006), (Anonim, 2010). Bu aşamalar,

- Bartlane kablo görüntü transfer servisi aracılığıyla işlemler yapıldı.
- Görüntüler Londra'dan New York'a denizaltı kabloları vasıtasıyla aktarıldı.
- Görüntüler kablo aktarımı için kodlanarak gönderilir, diğer taraftan okumak için telegraf yazıcı ucunda kod bloğu çözülürdü.

1920'lerin sonunda Bartlane kablo görüntü transfer servisinin gelişimi o dönem için yüksek kalitede görüntülerin oluşmasına neden oldu.

- Fotoğrafik tekniklerle yeni süreçler geliştirildi.
- Yansıtılan görüntülerde ton sayısının artırılması sağlandı.

1960'larda bilgi işlem teknolojisinin gelişimiyle beraber uzay araştırmaları için teknikler geliştirildi. 1964 yılında Ranger 7 probu tarafından aydan alınan görüntülerin kalitelerinin artırılması için bilgisayarlar geliştirildi. Bu geliştirilen teknikler Apollo dahil diğer uzay görevlerinde kullanıldı. 1970'lerde görüntü işleme tıp alanında da kullanılmaya başlandı. 1979 yılında bilgisayarlı tomografi taramasının arkasındaki teknolojinin sahibi Godfrey N.Hounsfield ve Prof. Allan M.Cormack tıp alanında Nobel ödülünü aldı. Bu aşamalardan sonra 1980'lerden itibaren kullanılan teknikler yazılımsal olarak geliştirilerek günümüze kadar getirildi.

1.7 Görüntü İşlemede Kullanılan Programlar

Görüntü işlemenin geniş bir konu olması nedeniyle kullanıcılar ihtiyaç duydukları alana özgü kullanım kolaylığı sunan çeşitli programlama dillerini kullanırlar. C++, Opencv kütüphanesi ile birlikte görüntü işlemede kullanılır. Benzer şekilde c# için de Emgucv ve Aforge kütüphaneleri kullanılarak görüntü işleme yapılabilir. Diğer taraftan matris işlemleri bakımından kolay bir dil olmasa da Java'da görüntü işlemede kullanılan programlama dillerindendir. Temelinde matris işlemleri için yapılmış olan Matlab (Matriz Laboratory), görüntü işleme için çok kullanışlı bir dildir. Matlab'ın diğer kullanışlı olan tarafı da Matlab ile oluşturulan herhangi bir .m file dosyası eğer c# ve Java gibi programlama dilleriyle çalışılıyorsa o dillerden çağırılıp işlem yapılabilmesidir.

2. GÖRÜNTÜ İŞLEMEDE TEMEL KAVRAMLAR

Bu bölümde görüntü işlemenin kapsamı ve görüntü işlemede kullanılan temel Matlab fonksiyonları tanıtılacaktır.

Günümüzde basit cihazlardan çok gelişmiş bilgisayarlara kadar çoğu makine, dijital ortamda aldığı görüntüyü tasarlandığı ihtiyaca yönelik olarak işleyip sonuca ulaştırmaktadır. Bu durum, yani görüntünün ihtiyaca yönelik olarak işlenmesi işlemi tam olarak görüntü işlemenin konusudur.

Matlab, bu konu ile ilgilenen birçok araştırmacının kullandığı bir dil olması, görüntü işleme araç kutusuna sahip olması, bu konuda özellikli ve kapsamlı bir dil olması bakımından çok tercih edilir.

Görüntü işleme günümüzde çoğu alanda kendisine uygulama alanı bulabilmektedir. Yüz tanıma, iris tanıma, parmak izi tespiti, glokom teşhisi, sarı nokta hastalığı teşhisi, korneal neovasculer rahatsızlıklarda, kornea epitelinin bozulma büyüklüğünün ölçümünde, iyi-kötü huylu ben tespitinde, röntgen, ultrason, tomografi gibi cihazlardan çıkan görüntünün iyileştirilmesi safhasında, bitki hastalıklarının tespitinde, bitkinin maruz kaldığı azotun bitkide oluşturduğu zararın derecesine kadar geniş bir perspektifte kullanılan önemli bir alandır. Günlük yaşamda da plaka tanıma sistemlerinde, trafik uyarı sistemlerinde, yapısal görüntüleme sistemleri gibi çoğu cihazın yorumlama safhasında kullanılır. *x* satır *y* sütunlu görüntü, koordinat sisteminde şu şekilde ifade edilir;

$$\begin{bmatrix} \mathbf{r(1,1)} & \mathbf{r(1,2)} & \mathbf{r(1,3)} & \mathbf{\ldots} & \mathbf{r(1,y)} \\ \mathbf{r(2,1)} & \mathbf{r(2,2)} & \mathbf{r(2,3)} & \mathbf{\ldots} & \mathbf{r(2,y)} \\ \mathbf{r(3,1)} & \mathbf{r(3,2)} & \mathbf{r(3,3)} & \mathbf{\ldots} & \mathbf{r(3,y)} \\ \mathbf{\ldots} & \mathbf{\ldots} & \mathbf{\ldots} & \mathbf{\ldots} & \mathbf{\ldots} \\ \mathbf{r(x,1)} & \mathbf{r(x,2)} & \mathbf{r(x,3)} & \mathbf{\ldots} & \mathbf{r(x,y)} \end{bmatrix}$$

sütun ve satırdaki piksel sayısının çarpımı çözünürlük değerini verir. Çözünürlük ne kadar yüksekse görüntüdeki ayrıntılar netleşir ve daha düzgün bir görüntü ortaya çıkar. Burada dikkat edilmesi gereken önemli bir husus farklı programlama dillerinde matris başlangıç değerlerinin değişiklik göstermesidir. Mesela Java, C gibi dillerde matris başlangıç değeri $(0,0)$ iken Matlab'ta $(1,1)$'dir.

Resimler genellikle analog ortamdan dijital ortama geçirilirken renk kayıpları, boyut uyumsuzluğu nedeniyle görüntü kayıpları v.s. gibi birçok nedenle bozukluklar(noise) oluşur.

Bu bozuklukların minimize edilmesi ve daha da önemlisi ihtiyaca yönelik olarak yazılacak programlar yardımıyla görüntüsel analiz yapılarak bilime hizmet edilmesi bakımından görüntü işleme önemli bir alandır.

2.1 Matlab'ta Bazı Temel Görüntü İşleme Fonksiyonları

Fonksiyon	**Kullanım Şekli**	**Açıklama**
imread	a=imread('bellek adresi');	resmin bilgileri değişkene atanır.
imshow	imshow(a);	resim figure penceresinde görüntülenir.
imtool	imtool(a);	resim imtool penceresinde görüntülenir.
size	[x,y]=size(a);	resmin boyutları değişkene alınır.
imfinfo	imfinfo('bellek adresi');	resmin bilgilerinin öğrenilmesini sağlar.

bunlar gibi çok sayıda komut mevcut olup detaylı şekilde ileriki bölümlerde anlatılacaktır. Daha detaylı incelenmek istenirse image processing toolbox incelenerek bilgi sahibi olunabilir.

2.2 Matlab Görüntü İşleme Araç Kutusu (Matlab Image Processing Toolbox)

Matlab görüntü işleme araç kutusu birçok görüntü işleme uygulaması için kodların nasıl kullanılacağı hakkında detaylı bilgiler verir. Ancak kodların açık kaynak kodlu olmaması sebebiyle bu alanda detaylı çalışacak kişilerin Matlab diline iyi seviyede hakim olmaları gerekir. Görüntü işleme araç kutusu,

- Uzaysal dönüşümler,
- Görüntü kaydetme,
- Görüntü verileri için 2 boyutlu doğrusal filtre tasarlama ve uygulama,
- Dönüşümler,
- Morfolojik işlemler,
- Görüntü analizi ve geliştirme,
- ROI tabanlı görüntü işleme,
- Görüntü netleştirme,
- Renkler,
- Komşuluk ve blok işlemleri,
- Görüntü işleme araç kutusu için kod üretimi,
- Resim görüntüleme ve farketme,
- GUI araçları,

- Uzaysal dönüşümler ve görüntü kaydetme,
- Görüntü analizi ve istatistikler,
- Görüntü aritmetiği,
- Görüntü geliştirme ve iyileştirme,
- Doğrusal filtreler ve dönüşümler,
- Renk haritaları ve renk uzayları

gibi ana alanlarda çeşitli fonksiyonlar sunar. İşlemler görüntü üzerinde yapılacağından tezin bu kısmında renk uzaylarına değinilecektir.

Renk uzayları, renkleri tanımlamak için kullanılan matematiksel modellerdir. Renk uzayları 3 boyutlu olarak tasarlanır nedeni ise bütün renkleri oluşturma gerekliliğidir. Çünkü renkmetri biliminin temelinin oluşturan Grassman'ın ilk kanununa göre bir rengi belirlemek için birbirinden bağımsız üç değişkene ihtiyaç vardır. Grassman'ın kanunu şu şekilde sembolize edilir;

$$R = \int_0^\infty I(\lambda) r(\lambda) d\lambda$$

$$G = \int_0^\infty I(\lambda) g(\lambda) d\lambda$$

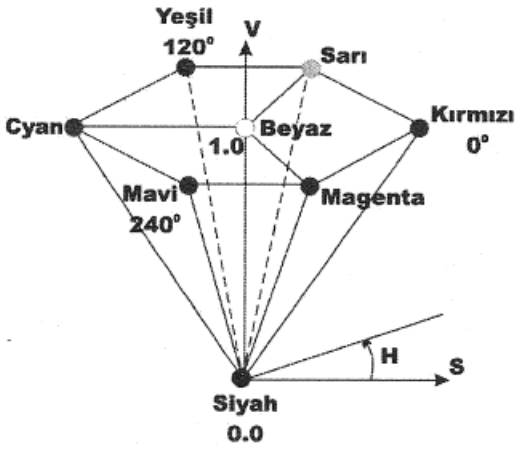
$$B = \int_0^\infty I(\lambda) b(\lambda) d\lambda$$

burada $r(\lambda), g(\lambda), b(\lambda)$ belirlenen şartlara bağlı renk karışım fonksiyonları, $I(\lambda)$ spektral güç dağılımını ifade etmektedir. Renklerin renk uzayındaki yerleri bu değişkenlere göre belirlenir. Farklı renkli görüntüleme cihazları farklı renk uzaylarını kullanırlar. Başlıca renk uzayları;

- RGB, Koordinat eksenleri kırmızı, yeşil, mavidir.
- CMY, Koordinat eksenleri cyan, magenta ve sarıdır.
- HLS, Koordinat eksenleri renk, parlaklık değeri ve doygunluktur.
- HSI, Koordinat eksenleri renk, doygunluk ve yoğunluktur.
- HSV, Koordinat eksenleri renk, doygunluk ve değerdir.
- YES,
- YCC,
- CIE XYZ, Kırmızı, yeşil, mavi renklerin beyindeki sinirler tarafından belirli oranlarla algılanmasıdır.
- CIE Lab,

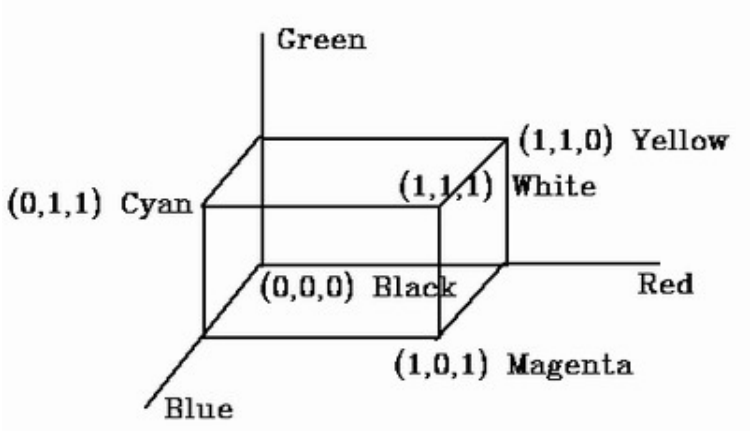
- CIE Luv,

dir (Gonzalez vd., 2009). Renk uzaylarının birbiri arasında dönüşümleri de yapılabilmektedir (Yılmaz vd., 2010). HSV renk uzayı örneği şekil 2.1'dedir.



Şekil 2.1: HSV renk uzayı örneği

RGB renk uzayı, kırmızı, yeşil, mavi renkleri koordinat eksenleri olacak şekilde bu üç renk kullanılarak toplamalı renk karışımı yöntemiyle bir birim küpün içinde renkleri tanımlayacak şekilde belirlenir. Bu tez çalışmasında RGB renk uzayı kullanılacaktır. RGB renk uzayı şekil 2.2'dedir.



Şekil 2.2: RGB renk uzayı

2.3 Görüntü Türleri

Genel olarak görüntü işlemede 4 farklı görüntü türü mevcuttur. Bunlar,

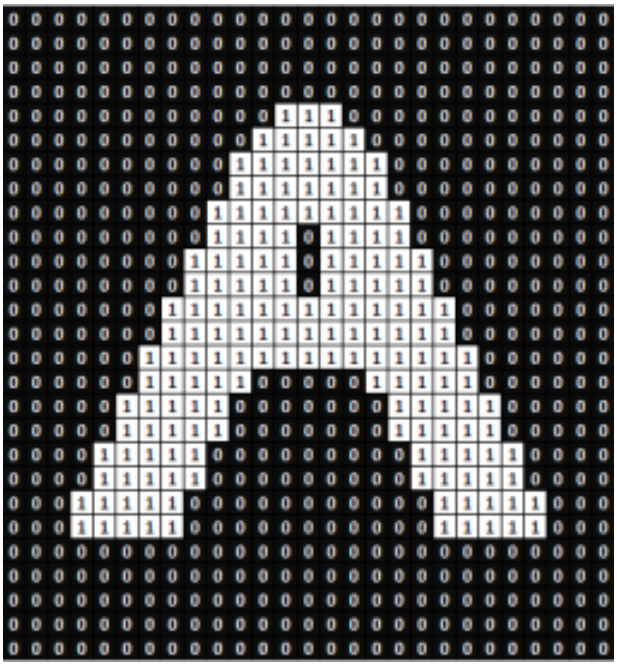
- Binary Görüntü,
- Gray Scale Görüntü,

- RGB(True Color) Görüntü,
- Çok Spektrumlu Görüntü,

dür.

2.3.1 Binary görüntü

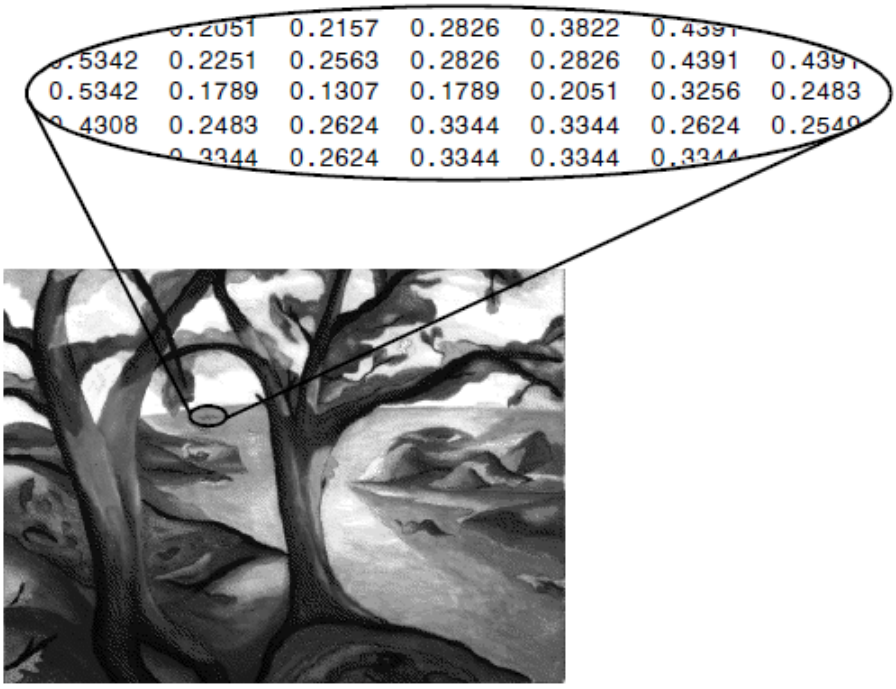
Binary görüntülerde her piksel 0 (siyah) veya 1 (beyazdır)'dir. Görüntünün bilgisayar diline dönüştürülmüş biçimi olarak düşünülebilir.



Şekil 2.3: Binary görüntü örneği

2.3.2 Grayscale görüntü

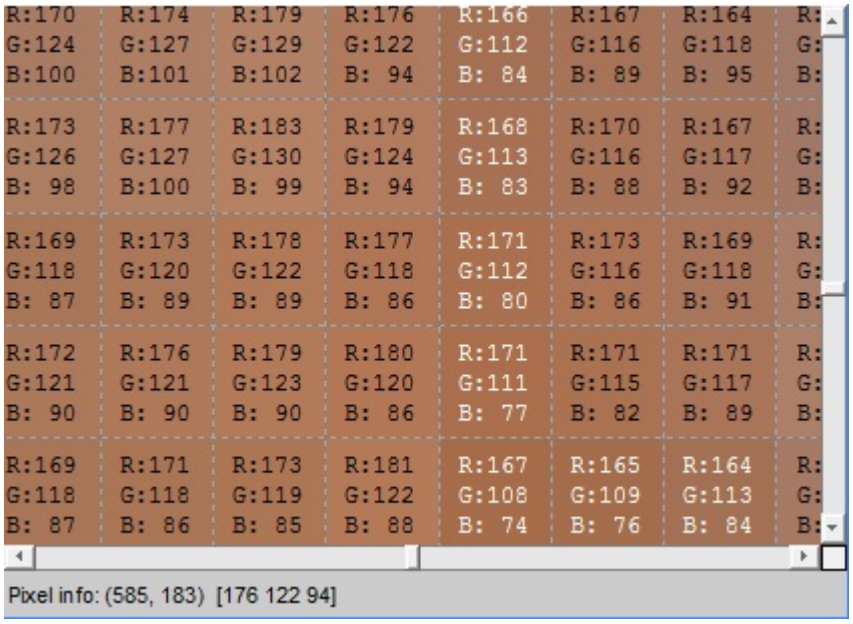
Grayscale görüntülerde binary görüntülerde olduğu gibi siyah ve beyazlar vardır. Fakat farklı olarak tonların karışımı söz konusudur. Her piksel $0-255$ arasında bir değer alır. 0 siyah, 255 beyaz olup aradaki sayılar karışımları şeklindedir. Görüntüdeki değerlerin $0-1$ aralığına dönüştürülmesiyle elde edilmiş görüntü şekil 2.4'te verilmiştir.



Şekil 2.4: Grayscale görüntü örneği

2.3.3 RGB görüntü

RGB (red, green, blue) görüntülerde her piksel 3 rengin karışımından elde edilir. Yani her piksel $0-255$ arasında bir kırmızı, $0-255$ arasında bir yeşil ve $0-255$ arasında bir mavinin karışımından oluşur. Her pikseldeki renk için 256^3 farklı ihtimal söz konusudur. Günlük hayatta da çevremizde gördüğümüz her görüntü bu 3 rengin belirli oranlarla karışımından oluşmaktadır.



Şekil 2.5: Matlab inspect pixel values penceresinden her pikseldeki renk karışımı.

2.3.4 Çok spektrumlu görüntü

Görüş alanı dışındaki spektrumdan alınan ve yanlış renkli görüntü olarak da bilinen görüntülerdir. Çok spektrumlu görüntü, aynı görüntünün çeşitli tek renkli görüntülerinden oluşan bir görüntü koleksiyonudur. Bu görüntülerin her biri farklı sensörle çekilmektedir.

2.4 Görüntü Formatları

Matematiksel açıdan bakılırsa her anlamlı iki boyutlu sayı dizisi bir görüntü olarak düşünülebilir. Bu görüntünün uygun biçimde belleğe alınıp saklanabilmesi için çeşitli görüntü formatları kullanılmaktadır. Kullanılan bazı görüntü formatları ve özellikleri şunlardır;

Format	**Özellik**
GIF	256 renk ile sınırlıdır. Kayıpsız sıkıştırma yapar.
JPEG	Günümüzde en yaygın olarak kullanılan formattır. Kayıpsız varyantları mevcuttur.
BMP	Temel resim formatıdır. Sınırlı kayıpsız sıkıştırma ve kayıplı varyantları mevcuttur.
PNG	GIF'i değiştirmek için tasarlanmış olup kayıpsız sıkıştırma yapar.
TIF/TIFF	Esnek, detaylı ve uyarlanabilir formattır. Çeşitli varyantları mevcuttur.

Windows Meta File (.wmf), Postscript (.ps), Encapsulated Postscript (.eps), Extended Meta File (.emf) v.s. gibi başka görüntü formatlarıda mevcuttur. Fakat en yaygın olarak yukarıdaki 5 görüntü formatı kullanılır.

2.5 Matlab'ta Veri Sınıfları

Matlab'ta her değişkene kullanacağı veri sınıfına göre bellekte bir alan ayrılır. Görüntü işlemeyle uğraşırken kullanılan görüntü türüne göre uygun bir veri sınıfı seçilmelidir. Örneğin, RGB görüntü ile ilgilenirken, görüntü verisini oluşturan matriste veri kaybı olmaması için uint8 veya daha büyük bir bellek alanına ihtiyaç duyulmalıdır. Eğer, int8 veri sınıfı kullanılırsa veri kaybı söz konusu olacaktır.

Kullanım şekli	**Açıklama**
double	Her eleman 8 byte ile ifade edilir. $[-10^{308}, 10^{308}]$ aralığındadır.
uint8	Her eleman 1 byte ile ifade edilir. $[0, 255]$ aralığındadır.
uint16	Her eleman 2 byte ile ifade edilir. $[0, 65535]$ aralığındadır.
uint32	Her eleman 4 byte ile ifade edilir. $[0, 4294967295]$ aralığındadır.
int8	Her eleman 1 byte ile ifade edilir. $[-128, 127]$ aralığındadır.
int16	Her eleman 2 byte ile ifade edilir. $[-32768, 32767]$ aralığındadır.
int32	Her eleman 4 byte ile ifade edilir. $[-2147483648, 2147483647]$ aralığındadır.
char	Her eleman 2 byte ile ifade edilir. Karakterden oluşur.
logical	Her eleman 1 byte olup 0 veya 1 mantıksal değerini alır.

Yukarıda belirtilen int8 kullanılmaması durumu rgb görüntünün 0 ile 255 arasında değer almasından kaynaklanmaktadır.

2.6 Görüntü Tür Dönüşümleri

Görüntü işlemede genellikle belirli bir görüntü türünde çalışma yapılır. Örneğin; bir plaka tanıma sistemi yapılacaksa o arabanın renginin çok önemi yoktur. Bu durumda renkli görüntüyle uğraşmak son derece gereksiz olup doğrudan binary görüntü üzerinde çalışma yapmak gerekir veya tıp alanında bir kornea üzerinde çalışılıcaksa gözün yapısı ve damarlar önemlidir, rengi değildir. Bu gibi sebeplerle ihtiyaca göre görüntü türü kullanılmalı veya istenilen türe dönüşüm yapılmalıdır.

Görüntü işlemede birçok işlem gri (grayscale) görüntü üzerinden yapılmaktadır. Renkli görüntülerin gri görüntüye dönüştürülmesi için Matlab'ta *rgb2gray* fonksiyonu kullanılır.

```
>>x=imread('bellek adresi');
>>y=rgb2gray(x);
```

RGB görüntünün her pikselinde 3 ayrı değer mevcuttur. Bunlar (x,y,1), (x,y,2), (x,y,3) olup (x,y,1), x,y koordinatında bulunan pikseldeki kırmızı rengi, (x,y,2), x,y koordinatında bulunan pikseldeki yeşil rengi ve (x,y,3) ise x,y koordinatında bulunan pikseldeki mavi rengi ifade eder. *rgb2gray(x)* fonksiyonuyla yapılan işlem,

```
>>gri=uint8(zeros(size(x,1),size(x,2)));
>>for i=1:size(x,1)
>>for j=1:size(x,2)
>>gri(i,j)=0.2989*x(i,j,1)+0.5870*x(i,j,2)+0.1140*x(i,j,3);
>>end
>>end
```

dönüşümün sağlanmasıdır. Yani her pikseldeki kırmızı, yeşil ve mavinin belirli oranda karıştırılarak gri rengin elde edilmesidir. Buradaki karışım oranı için,

$$R_{gri}(x,y) = \alpha \times R_{rgb}(x,y,r) + \beta \times R_{rgb}(x,y,g) + \gamma \times R_{rgb}(x,y,b)$$

formülü kullanılır. NTSC (National Television System Committee) tarafından kabul edilen renk karışım değerleri olarak $\alpha = 0.2989, \beta = 0.5870, \gamma = 0.1140$ kullanılır.

Şekil 2.6: Rgb-grayscale dönüşüm örneği

Grayscale görüntünün binary görüntüye dönüşümü *im2bw* fonksiyonuyla yapılır. Kullanımı,

```
>>x=imread('bellek adresi');
>>t=rgb2gray(x);
>>y=im2bw(t);
```

şeklindedir. *im2bw* fonksiyonuyla yapılan işlem,

```
>>s=zeros(size(x,1),size(x,2));
>>for i=1:size(x,1)
>>for j=1:size(x,2)
>>if (sum(x(i,j,:))>0)
>>s(i,j)=1;
>>end
>>end
>>end
```

dönüşümün sağlanmasıdır.

Şekil 2.7: Grayscale-binary dönüşüm örneği

2.7 Temel Görüntü İşleme Fonksiyonları

2.7.1 imread fonksiyonu

Bilgisayar ortamında tutulan görüntünün piksel piksel matris olarak okunmasını sağlar. Matris işlemleri yapılarak görüntü istenilen forma taşınır. Kullanımı;

```
>> k=imread('bellek adresi');
```

şeklindedir.

2.7.2 imshow fonksiyonu

imread fonksiyonu ile belleğe alınan görüntünün ekranda görüntülenmesi sağlanır. Kullanımı,

```
>> x=imread('bellek adresi');
>> imshow(x);
```

şeklindedir. Matris üzerinde yapılan değişikliklerle yeni görüntüler elde edilebilir.

```
>> x=imread('bellek adresi');
>> y=rgb2gray(x);
>> subplot(211),imshow(y);
>> subplot(212),imshow(y,[20,100]);
```

şeklinde de kullanılabilir.

Şekil 2.8: Piksel değerleri 20-100 arasındaki görüntü

2.7.3 imtool fonksiyonu

Kullanımı *imshow* komutuyla aynı olup görüntünün image tool penceresinde açılmasını sağlar. Burada görüntünün her pikselindeki renk yoğunluğu görülüp renk karışımlarına bakılarak görüntü detaylı incelenebilir.

2.7.4 imwrite fonksiyonu

Üzerinde işlemler yapılmış görüntünün istenilen formatta kaydedilmesini sağlar. Kullanım şekli;

```
imwrite(resmin matrissel ifadesi,'resmin yuklenecegi bellek adresi','format');
```

şeklindedir.

2.7.5 imfinfo fonksiyonu

Görüntünün birçok özelliğini elde etmek için kullanılır.

```
>> x=imfinfo('D:\Matlab\manzara2.jpg')
x =
           Filename: 'D:\Matlab\manzara2.jpg'
        FileModDate: '25-Feb-2016 09:00:23'
```

```
            FileSize: 185416
              Format: 'jpg'
       FormatVersion: ''
               Width: 900
              Height: 598
            BitDepth: 24
           ColorType: 'truecolor'
    FormatSignature: ''
    NumberOfSamples: 3
       CodingMethod: 'Huffman'
      CodingProcess: 'Sequential'
            Comment: {'CREATOR: gd-jpeg v1.0 (using IJG JPEG v62), quality = 90
'}
>> x.Format
ans = jpg
```

buradaki Filesize özelliği, görüntünün sıkıştırılmış halde kapladığı hafızanın bayt cinsinden ifadesidir. Sıkıştırılmış görüntünün gerçek boyutu ise şu şekilde hesaplanır (McAndrew, 2004):

$$\text{Width} \times \text{Height} \times \text{BitDepth} = 900 \times 598 \times 24 = 12.916.800 \text{ bayt}$$

Elde edilen iki değerin oranı,

$$\frac{12916800}{185416} = 69.6639$$

sıkıştırma oranı olarak adlandırılır.

Her görüntü formatının sıkıştırma için kullanmış olduğu özel şifreleme yöntemi vardır. Bazen aynı şifreleme yönteminin eklemeler yapılarak kullanımı patent sorunlarını da meydana getirmektedir. CodingMethod özelliği görüntüde hangi sıkıştırma metodunun kullanıldığını belirtir.

Huffman algoritması (McAndrew, 2004), bir veri grubundaki her veriyi kullanıldığı tekrar sayısına göre sıralayıp ağacın dallarını en çok kullanılandan en az kullanılana doğru 0-1 kodlarıyla numaralandırmaktadır. Burada en çok tekrar edilen veriler bellekte daha az

yer kaplayacak şekilde numaralandırıldıklarından dolayı Huffman algoritması yüksek bir veri sıkıştırmasına olanak sağlamaktadır. Burada da görüldüğü gibi her algoritmadan geçirilen veri farklı olduğu için sıkıştırma oranının da farklı olacağı açıktır.

BitDepth, her pikselde yer alan renk değeri için bellekte ayrılan alanı gösterir. Şöyleki örnekte verilen görüntü 'true color' yani rgb olup 3 rengin karışımından oluşur. Her renk 0-255 arasında değer alabileceği için her renk için 8 bitlik yer ayrılır toplamda rgb görüntü için bu değer 24 bit, gri görüntü için bu değer 8 bit, siyah beyaz görüntü için ise 1 bit'tir.

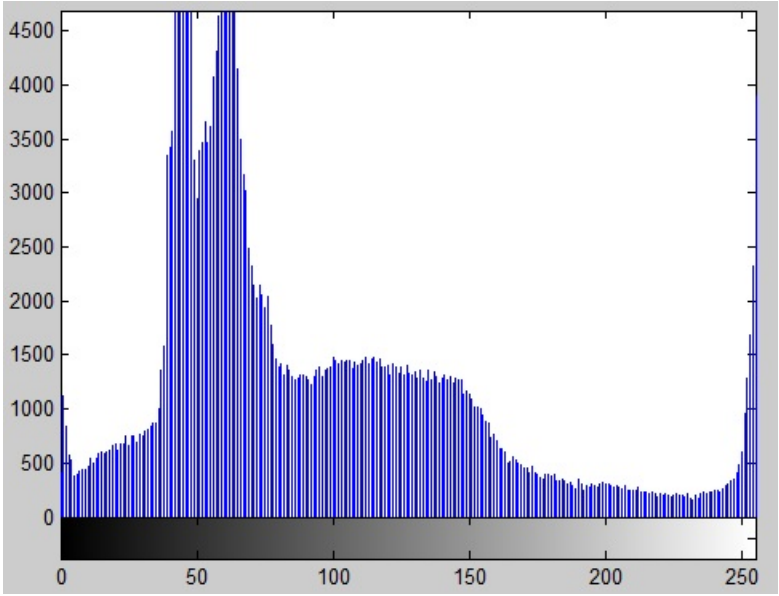
CodingProcess özelliği ise kodlamanın nasıl yapıldığı hakkında bilgi verir. Örnekte bu özellik 'sequential' yani ardışıl kodlama olarak verilmiştir. Buradaki diğer yöntemler ise arithmetic, progression, special selection, lossless, losy, LZW, Delta encoding, RLE, v.s. şeklindedir.

2.7.6 imhist fonksiyonu

Histogram, bir görüntü içerisindeki renk değerlerinin tekrar sıklığını gösteren grafiktir. Histogram ile görüntünün kontrast değeri hakkında bilgi sahibi olunur. Görüntü karanlık bir görüntü ise histogramda sıfıra yakın değerlerde yığılma, aydınlık bir görüntü ise yüksek değerlerde bir yığılma gözlenir. *imhist* fonksiyonu görüntünün renk yoğunluk histogramının görüntülenmesini sağlar.

```
>> x=imread('D:\Matlab\manzara3.jpg');
>> imhist(rgb2gray(x));
```

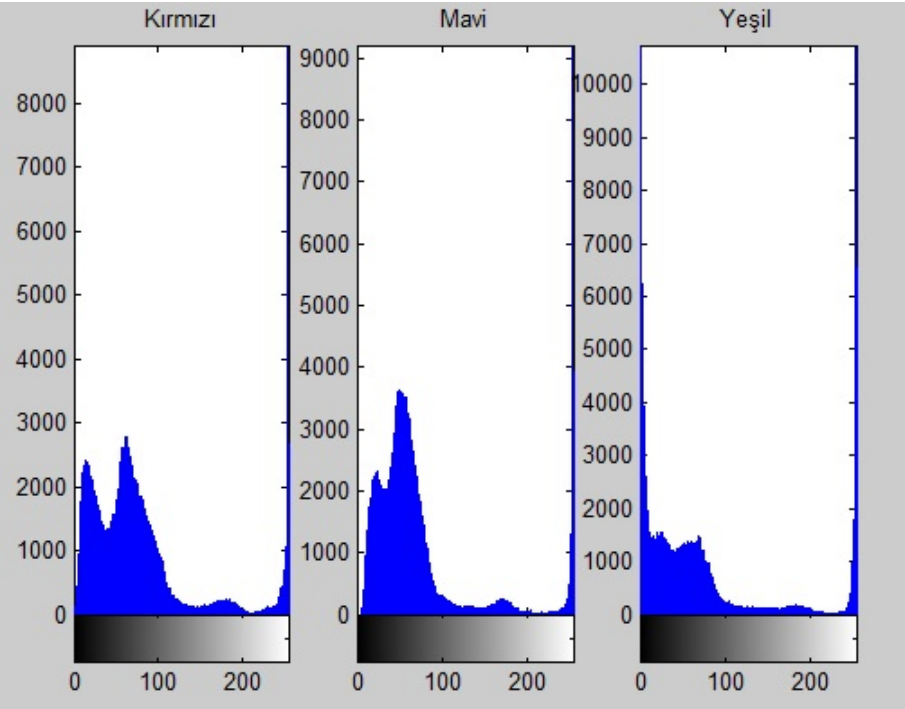
bu kod ile figure penceresinde açılan histogram şekil 2.9'dadır.



Şekil 2.9: Yüklenen görüntünün histogramı

Burada RGB'den grayscale'e dönüştürülen görüntüde 45-70 aralığında renk oranından yoğun bir şekilde kullanıldığı anlaşılır. Matlab gri görüntünün histogramını vermekte ve RGB görüntüyü oluşturan renklerin ayrı ayrı histogramları ile ilgili bir komut sunmamaktadır. Eğer RGB görüntü üzerinde çalışılıyorsa, her pikseldeki renk histogramlarını ayrı ayrı görmek için aşağıdaki gibi bir fonksiyon oluşturulabilir.

```
function [ ] = histogram( adres )
matris=imread(adres);
k=rgb2gray(matris);
[x,y,z]=size(matris);
a=k;b=k;c=k;
for i=1:x
    for j=1:y
        a(i,j)=matris(i,j,1);
        b(i,j)=matris(i,j,2);
        c(i,j)=matris(i,j,3);
    end
end
subplot(1,3,1),imhist(a),title('Kirmizi ');
subplot(1,3,2),imhist(b),title('Mavi');
subplot(1,3,3),imhist(c),title('Yesil');
end
>> histogram('D:\Matlab\manzara3.jpg');
```



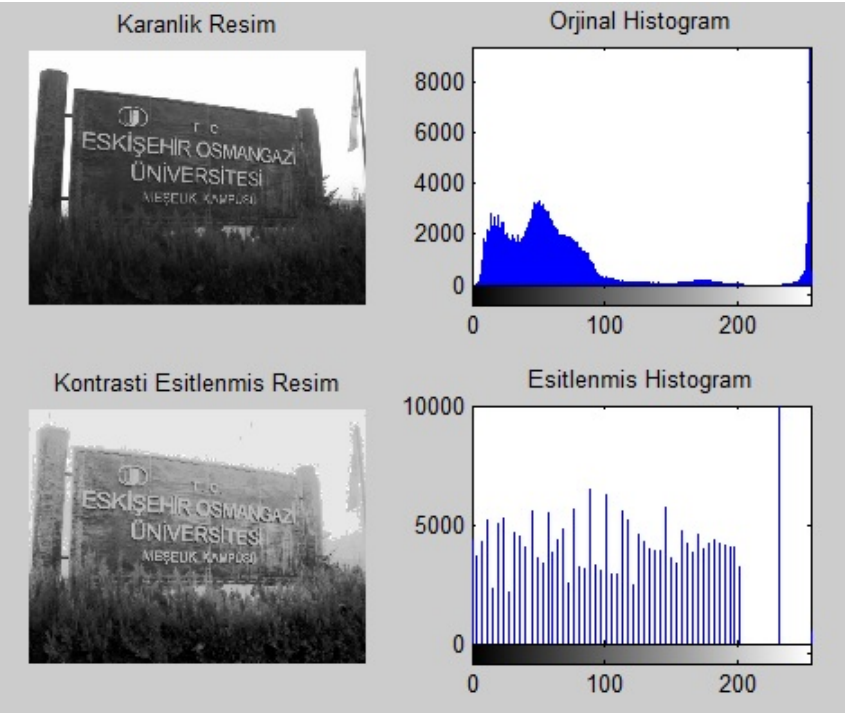
Şekil 2.10: Görüntünün ayrı ayrı renk histogramları

2.7.7 Histogram eşitleme

Histeq fonksiyonu kullanılarak görüntüler üzerinde daha net fikir sahibi olunması sağlanır. Matlab kodları,

```
resim=imread('D:\Matlab\manzara4.jpg');
resim=rgb2gray(resim);
h=histeq(resim);
subplot(221);imshow(resim);title('Karanlik Resim');
subplot(222);imhist(resim);title('Orjinal Histogram');
subplot(223);imshow(h);title('Kontrasti Esitlenmis Resim');
subplot(224);imhist(h);title('Esitlenmis Histogram');
```

şeklindedir. Kod bloğu sonunda oluşan figure penceresindeki görüntü ise şekil 2.11'dedir.



Şekil 2.11: Eşitlenmiş histogram

Gözde kan hücreleri çok sıklıkla bulunduğu için yüksek kontrastta net bilgi alınabilmektedir. 4 renk uzayı kontrast seviyesine göre daha uygun görülmekle birlikte her renk uzayındaki uygun boyut şu şekildedir;

RGB - Renkli görüntü olarakta bildiğimiz boyutları kırmızı, yeşil, mavi renkler olan uzayın yeşil(G) olan boyutu.

YCbCr - Dijital videolarda kullanılan renk uzayıdır. Burada boyutlar Y parlaklık(luminance), Cb Chroma (mavi luma), Cr Chroma (kırmızı luma)'dır. En uygun boyutu ise Y olarak belirtilmektedir.

Lab - CIE tarafından oluşturulan modellerden biridir. Lab renk modeli dikey sarı-mavi ve yeşil-kırmızı eksenlerine dayanan dörtgensel koordinatları kullanır. CIE Lab renk modelinin en önemli özelliği algılama yönünden düzgün değişim göstermesidir. Genelde günümüzde bilgisayarlar ve programlar Lab modelini kullanırlar. Burada L(Lightness) açıklık-koyuluk değerini, a kırmızı-yeşil değerini, b ise sarı-mavi değerini belirtir. En uygun boyut L olarak belirtilir. Diğer renk modeli ise Gaussian yoğunluk modellerinden G1 olarak ifade edilir.

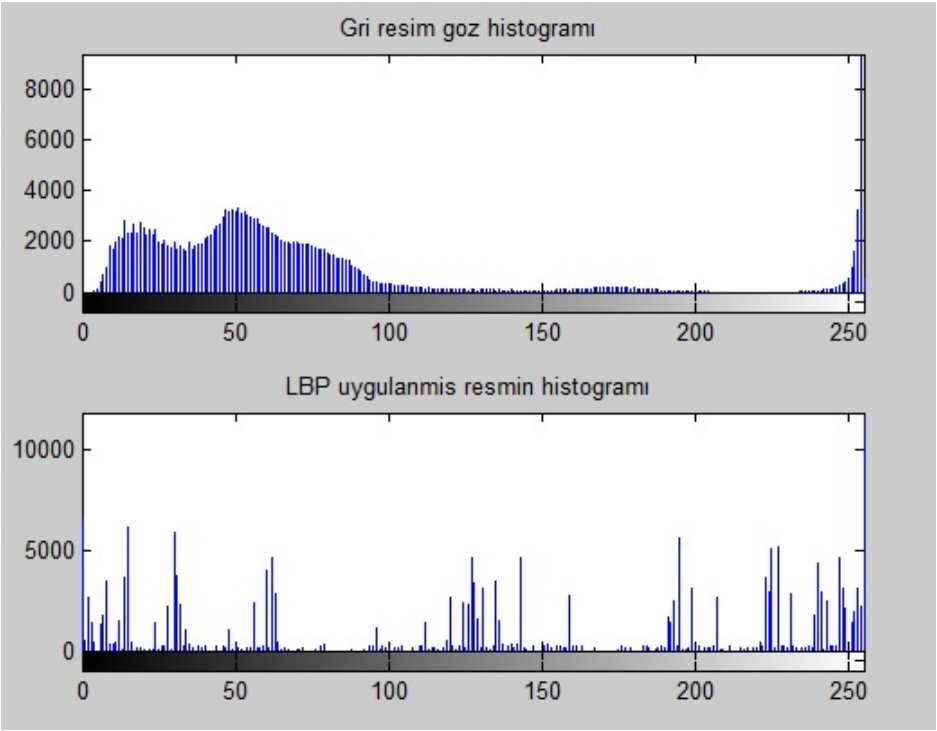
Yavuz vd. (2013) yayınlamış oldukları makalede ele aldıkları konuda değindikleri LBP yaklaşımı doku belirlemede güçlü bir özelliktir. Ayrıca bu özellik yüz tanıma alanında da kullanılır. Bu sistem 3×3 matrislerde uygulanmakta olup gri resimlerde merkezdeki piksel değeri ile çevresindeki piksel değerleri arasında belirli bir işlem ile belirlenir. Çevre piksellerin

değeri merkez pikselin değerinden büyük yada eşit ise 1, değilse 0 değerini almaktadır. Oluşan çevre piksellerdeki binary değerler ise matrisin $(2,1)$'deki binary değerinden başlayarak saat yönü tersine ikilik sistemde yazıldıktan sonra onluk sistemdeki değeri merkez piksel değerine atanır. Bu yöntemle yüz veya gözün özellik çıkarımı yapılmakta olup ayrıca bu özellik sayesinde resmin histogramı da eşitlenmektedir.

LBP için şöyle bir fonksiyon yazılabilir,

```
function [ ] = Lbp( adres )
resim=imread(adres);
resim=rgb2gray(resim);
[x,y]=size(resim);
yeni=resim;
for i=2:x-1
    for j=2:y-1
        dizi=zeros(3,3);
        for k=i-1:i+1
            for l=j-1:j+1
            if resim(i,j) > resim(k,l)
                dizi(k-i+2,l-j+2)=0;
            else
                dizi(k-i+2,l-j+2)=1;
            end
            end
        end
        yeni(i,j)=dizi(1,1)*2^0+dizi(1,2)*2^1+dizi(1,3)*2^2+...
            dizi(2,3)*2^3+dizi(3,3)*2^4+dizi(3,2)*2^5+...
            dizi(3,1)*2^6+dizi(2,1)*2^7;
    end
end
subplot(211);imhist(resim);title('Gri resim goz histogrami ');
subplot(212);imhist(yeni);title('LBP uygulanmis resmin histogrami ');
end
```

Yazılan kod ile oluşan figure penceresindeki görüntü ise şekil 2.12'dedir.



Şekil 2.12: LBP uygulanmış görüntünün histogramı

2.7.8 imresize fonksiyonu

Görüntünün boyutunu oransal olarak arttırmak veya azaltmak için kullanılır.

```
>> x=imread(''D:\tez\bolum1\resimler1\esogu.jpg'');
>> y=imresize(x,0.5);
>> imshow(y);
```

burada 0.5 ile resmin boyutları %50 küçültülmektedir.

2.7.9 imrotate fonksiyonu

Görüntünün istenilen açıda döndürülmesi için kullanılır.

```
>> x=imread(''D:\tez\bolum1\resimler1\esogu.jpg'');
>> y=imrotate(x,30);
>> imshow(y);
```

Şekil 2.13: Döndürülmüş görüntü

görüntü diğer tarafa döndürülmek istenirse;

```
>> x=imread('D:\Matlab\manzara6.jpg');
>> y=imrotate(x,-30);
```

şeklindedir. 3 parametreli kullanımı ise,

```
>> imrotate(x,30,'bilinear');
>> imrotate(x,30,'bicubic');
```

biçimindedir. 2 parametreli olarak verilen döndürme fonksiyonu nearest neighbor interpolation dönüşümünü kullanırken 3 parametreli olanlar da kendine özgü dönüşüm yöntemlerini kullanırlar. Kısaca değinilecek olunursa dönüşümlerde kullanılan 3 algoritma da 3×3 matrisler kullanılarak yapılmaktadır. Nearest neighbor interpolationda döndürme işlemi kendi algoritması ile 1.7 piksel piksel yapılırken, bilinear ve bicubic'te kendi algoritmaları ile 0.5 piksel piksel yapılmaktadır. Bilinear ve bicubic daha küçük adımlarla döndürme işlemini yaptıkları için daha net sonuç vermektedir. Burada dikkat edilmesi gereken husus ise görüntünün, piksel değeri ile döndürme sonrası oluşan yeni değerler döndürmede kullanılan yöntem sebebiyle farklılık gösterir. Örneğin, görüntü matrisinin $\begin{bmatrix} 3 & 5 & 7 \\ 1 & 2 & 7 \\ 6 & 0 & 3 \end{bmatrix}$ gibi bir matris

olduğunu düşünelim. imrotate(x,30) komutu ile görüntünün son matrisi, $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 7 & 7 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ şeklinde değişir.

2.7.10 imnoise fonksiyonu

Üzerinde işlem yapılacak görüntüye filtre uygulanması sağlanır. Filtreler sonraki bölümde detaylı olarak açıklanacaktır.

```
>> x=imread('D:\Matlab\manzara6.jpg');
>> y=imnoise(x,'gaussian',0.1);
```

yukarıda imnoise komutu ile x değişkenine alınan görüntü matrisine belirli oranda gauss filtresi uygulanmıştır.

```
>> x=imread('D:\Matlab\manzara6.jpg');
>> y=imnoise(x,'salt & pepper',0.05);
```

burada ise belirli oranda tuz ve biber filtresi uygulanmıştır.

2.7.11 Görüntüyü çevirme ve odaklanma

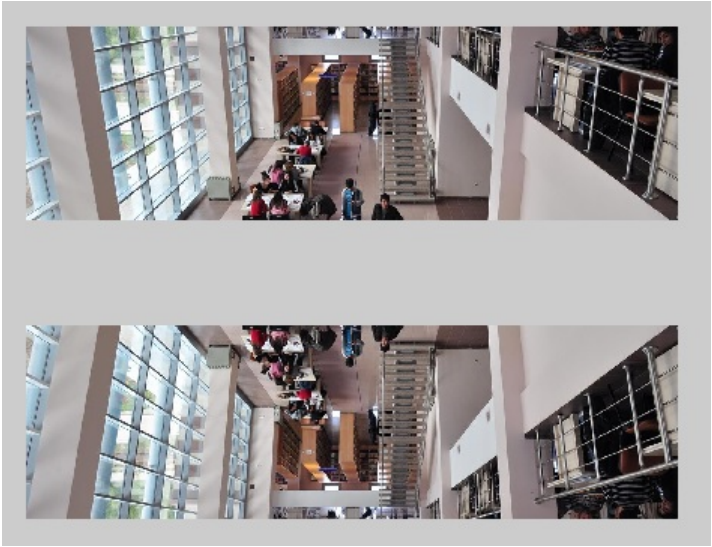
Aşağıda verilen kodlar yardımıyla görüntü dikey eksende ters çevrilmektedir.

```
>> x=imread('D:\Matlab\manzara6.jpg');
>> y=rgb2gray(x);
>> z=y(end:-1:1,:);
>> subplot(2,1,1),imshow(y);subplot(2,1,2),imshow(z);
```

Şekil 2.14: Ters çevrilmiş grayscale görüntü

benzer şekilde işlem RGB görüntü üzerinde de yapılır.

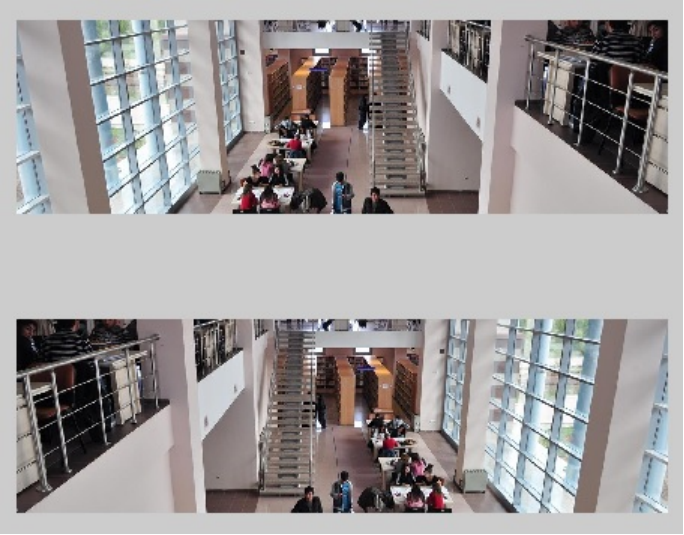
```
>> x=imread('D:\Matlab\manzara6.jpg');
>> z=x(end:-1:1,:,:);
>> subplot(2,1,1),imshow(x);subplot(2,1,2),imshow(z);
```



Şekil 2.15: Ters çevrilmiş rgb görüntü

Görüntüyü yatay eksende ters çevirmek için aşağıdaki kodlar kullanılabilir.

```
>> x=imread('D:\Matlab\manzara6.jpg');
>> z=x(:,end:-1:1,:);
>> subplot(2,1,1),imshow(x);subplot(2,1,2),imshow(z);
```



Şekil 2.16: Yatay eksende ters çevrilmiş görüntü

Görüntünün belirli bir alanına odaklanmak için aşağıdaki kodlar kullanılabilir.

```
>> x=imread('D:\Matlab\manzara6.jpg');
>> y=x(300:600,300:600,:);
>> subplot(2,1,1),imshow(x);subplot(2,1,2),imshow(y);
```

Görüntüyü belirli oranda küçültmek için aşağıdaki kodlar kullanılabilir.

```
>> x=imread('D:\Matlab\manzara6.jpg');
>> y=x(1:10:end,1:10:end,:);
>> subplot(2,1,1),imshow(x);subplot(2,1,2),imshow(y);
```

3. FİLTRELER

Bu bölümde görüntü işleme tekniklerinde kullanılan filtrelerden bahsedilecektir.

Görüntü üzerinde görüntü formatından, ışığın durumundan veya başka bir sebeple birçok bozukluk oluşmuş olabilir. Bu bozukluklar görüntü işlemede gürültü (noise) olarak adlandırılır. Bu bozuklukların giderilmesi veya minimum seviyeye indirilmesi için çok sayıda filtre kullanılır. Bu filtreleri incelerken hem Matlab'ın hazır kodlarına hem de filtreler için yazmış olduğumuz kodlara değinilecektir.

Filtreleme için kullanılacak filtre matrisi ile görüntüyü oluşturan matrisin $(2,2)$'deki değerinden başlayarak bütün görüntü taranıp yeni piksel değerli matris elde edilir. Burada görüldüğü gibi görüntünün kenar pikselleri filtrelemeye dahil edilmez. Eğer 3×3'lük bir filtre matrisi ile çalışılıyorsa işlem yapılacak pikselin 8 komşusu sonuç matrisini etkiler. Görüldüğü gibi filtre işlemlerinde komşu piksel değerleri önemlidir.

3.1 Bozukluk Giderme Filtreleri

Bozukluk giderme filtrelerinin temel amacı görüntü üzerindeki bozukluğun minimum seviyeye indirilmesidir.

3.1.1 Ortalama(Mean) filtre

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Ortalama filtre kısmen bozulmuş görüntüye uygulandığında bozukluğun bulunduğu piksel değeri, çevresindeki piksel değerlerinin ortalaması alınarak hesaplanır. Böylelikle bozuk piksel ile çevresindeki pikseller arasındaki fark azalacaktır. Görüntü üzerindeki bozukluk yüksek seviyede değil ise bu filtre bozukluğu minimum seviyeye indiren filtrelerden biridir (Solomon ve Breckon, 2008). Fakat, ortalama filtre resmin bulanıklaşmasına sebep olur. Bu sebeple ortalama filtre için daha büyük kare matrisler kullanmaktansa 3×3'lük matris

kullanmak bulanıklaşmayı minimum düzeyde tutacağı için daha çok tercih edilir. Matlab'da *imfilter* fonksiyonuyla ortalama filtre uygulanır.

```
x=imread(resmin bellek adresi);
k=ones(3,3)/9;
m=imfilter(x,k);
imshow(m);
```

burada k ortalama filtre matrisi, x ise görüntüdeki piksel değerlerinin oluşturduğu matristir. Ortalama filtre için aşağıdaki gibi bir fonksiyon yazılabilir.

```
function [ ] = Ortalama( adres )
r=imread(adres);
g=r;
t=double(r);
[x,y,z]=size(t);
for i=2:x-1
    for j=2:y-1
        for k=1:3
    g(i,j,k)=(1/9)*(t(i,j,k)+t(i-1,j-1,k)+t(i-1,j,k)+...
t(i-1,j+1,k)+t(i,j-1,k)+t(i,j+1,k)+...
t(i+1,j-1,k)+t(i+1,j,k)+t(i+1,j+1,k));
    end
    end
end
imshow(g);
end
```

fonksiyon yazıldıktan sonra

```
x=imread('D:\Matlab\resim4.jpg');
subplot(121),imshow(x);
subplot(122),Ortalama('D:\Matlab\resim4.jpg');
```

şeklinde kullanılırsa oluşan resmin son hali şekil 3.1'dedir.

Şekil 3.1: Ortalama filtreden geçirilmiş görüntü

3.1.2 Medyan filtresi

Medyan filtresi genellikle salt & pepper bozukluğu denilen tuz ve biber bozukluğunun giderilmesi için kullanılan filtredir. Matematiksel olarak da bilindiği gibi medyan ortanca demek olup çalışılacak matrisin değerlerinin küçükten büyüğe sıralanıp ortadaki sayının, görüntünün bulunduğu matrisin ilgili pikseline atanması işlemidir (Blackledge, 2005). Bu işlemde görüntünün köşelerindeki bozukluk yok edilemez. Filtre matrisi farklı boyutlarda da seçilebilir.

$\begin{bmatrix} 37 & 23 & 42 \\ 98 & \mathbf{14} & 22 \\ 55 & 43 & 87 \end{bmatrix}$, 14'ün bulunduğu piksel değerine medyan filtresi uygulanırsa oluşan yeni piksel değeri 42 olacaktır.

```
x=imread('D:\Matlab\manzara6.jpg');
y=rgb2gray(x);
k=medfilt2(y,[3 3]);
imshow(k);
```

medyan filtresi için aşağıdaki gibi bir fonksiyon yazılabilir.

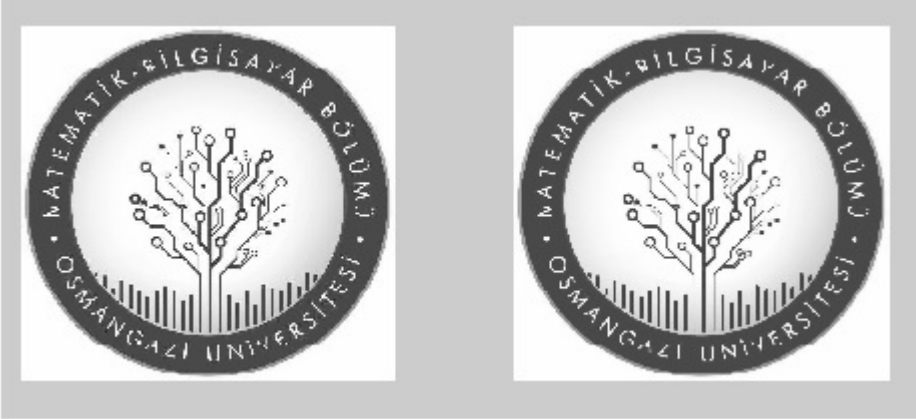
```
function [ ] = Medyan( adres )
r=imread(adres);
g=rgb2gray(r);
t=double(g);
[x,y]=size(t);
```

```
for i=2:x-1
    for j=2:y-1
    b=[];z=0;
for k=i-1:i+1
    for l=j-1:j+1
        z=z+1;
        b(z)=t(k,l);
    end
end
for d=1:8
    eb=d;
    for e=d+1:9
    if (b(e)>eb)
        eb=e;
    end
    end
    if i~=eb
        gecici=b(d);
        b(d)=b(eb);
        b(eb)=gecici;
    end
end
g(i,j)=b(5);
    end
end
imshow(g);
end
```

fonksiyon aşağıdaki gibi kullanılırsa,

```
x=imread('D:\Matlab\resim4.jpg');
subplot(121),imshow(rgb2gray(x));...
subplot(122),Medyan('D:\Matlab\resim4.jpg');
```

oluşan görüntünün son hali şekil 3.2'dedir.



Şekil 3.2: Medyan filtreden geçirilmiş görüntü

Salt & Pepper ise görüntü üzerine istenilen kadar bozuk piksel eklenmesi için kullanılır. Genellikle görüntülerde salt & pepper'da olduğu gibi bir çok nedenle bozukluk oluşmaktadır. Salt & pepper, Matlab'da *imnoise* fonksiyonu ile gerçekleştirilir.

```
x=imread('D:\Matlab\resim4.jpg');
k=imnoise(x,'salt & pepper');
imshow(k);
```

imnoise fonksiyonu 3 parametreli olarakta kullanılabilir.

```
x=imread('D:\Matlab\resim4.jpg');
k=imnoise(x,'salt & pepper',0.05);
imshow(k);
```

buradaki 3. parametre ile görüntüyü oluşturan piksellerin %5'ine salt & pepper uygulanmıştır. 3 paremetreli kullanılmaz ise varsayılan değer Matlab tarafından 0.05 olarak kabul edilir. Salt & pepper için aşağıdaki gibi bir fonksiyon yazılabilir.

```
function [] = tuzbiber( adres,n )
r=imread(adres);
g=r;
t=double(g);
[x,y,z]=size(t);
```

```
top=0;
while (top<=n)
    g(1+round((x-1)*(rand(1))),1+round((y-1)*(rand(1))),:)=...
round(255*(rand(1)));
top=top+1;
end
imshow(g);
imwrite(g,'D:\Matlab\bozukresim.jpg');
end
```

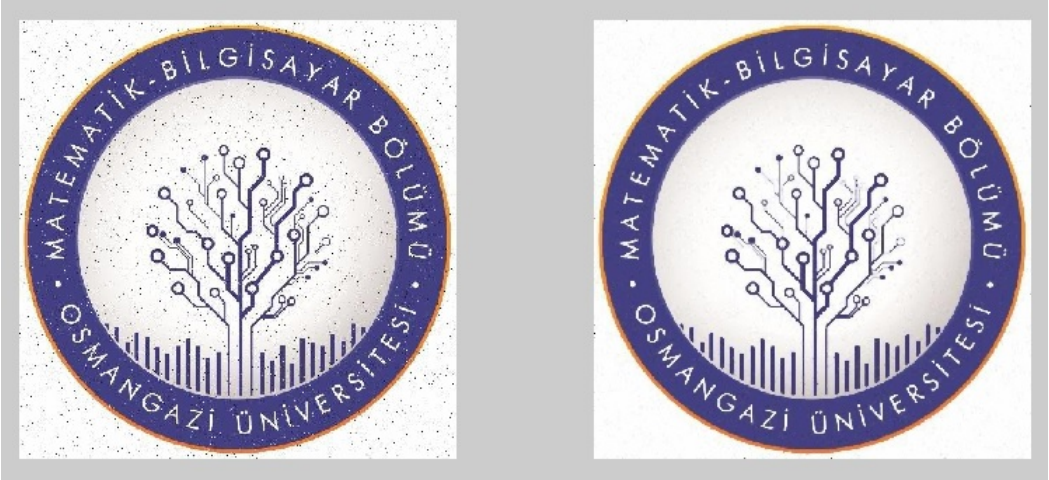
iki giriş parametreli olarak yazılan fonksiyonda n, kaç adet salt & pepper noktası istendiğini, adres ise görüntünün bilgisayardaki konumunu belirtir. İşlem sonrası figure perceresindeki görüntü kaydedilip yukarıda yazılan Medyan fonksiyonu ile çalıştırıldığında bozukluğun yok edildiği görülür. Medyan filtresi grayscale görüntüye uygulandığı için Şekil 3.3'deki gibi sonuç grayscale görüntü olacaktır.

Şekil 3.3: Bozulmuş görüntüye medyan uygulanması

RGB görüntüde çalışacak bir medyan fonksiyonu aşağıdaki gibi yazılabilir.

```
function [  ] = renklimedyan( adres )
r=imread(adres);
[a,b,c]=size(r);
son=r;bos1=[];bos2=[];bos3=[];
for i=2:a-1
    for j=2:b-1
    z=0;
```

```
for k=i-1:i+1
    for l=j-1:j+1
        z=z+1;
        bos1(z)=r(k,l,1);
        bos2(z)=r(k,l,2);
        bos3(z)=r(k,l,3);
    end
end
bos1=sort(bos1);son(i,j,1)=bos1(5);
bos2=sort(bos2);son(i,j,2)=bos2(5);
bos3=sort(bos3);son(i,j,3)=bos3(5);
    end
end
imtool(son);
end
```



Şekil 3.4: Bozulmuş görüntüye renkli medyan uygulanması

3.1.3 Gauss filtresi

Gauss filtresi, Gauss dağılımını kullanarak ortalama filtrenin değiştirilmiş halidir. Gauss filtresi, görüntüdeki bulanıklaşmayı iyileştirmek için kullanılır (Solomon ve Breckon, 2008), (Zhou vd., 2010).

Gauss filtresi 2 boyutta;

$$T(x,y) = \frac{1}{2\pi\sigma^2} \times e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Gauss filtresi *n* boyutta;

$$T(x,y) = \frac{1}{(2\pi\sigma^2)^{n/2}} \times e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

şeklindedir. Burada σ Gauss dağılımının standart sapma değeri, $\frac{1}{2\pi\sigma^2}$ ise normalleştirme katsayısıdır. Kullanımı,

```
>> a=imread('D:\Matlab\manzara2.jpg');
>> b=fspecial('gaussian',[3 3]);
>> c=imfilter(a,b);
>> imshow(c);
```

şeklindedir. Gauss filtresi için

$$x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \; y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

matrisleri kullanılır. $x^2 + y^2$ ifadesi ile ilgili matrislerin noktasal kareler toplamı ifade edilmektedir. Gauss filtresi için aşağıdaki gibi bir fonksiyon yazılabilir.

```
function [ ] = gauss( adres )
r=imread(adres);
g=rgb2gray(r);
t=double(g);
pi=3.14;
s=2;
x=[-1 0 1;-1 0 1;-1 0 1];
y=[-1 -1 -1;0 0 0;1 1 1];
[k,l]=size(t);
```

```
filtre=(1/(2*pi*s^2))*exp(-(x.^2+y.^2)/(2*s^2));
for i=2:k-1
    for j=2:l-1
        matris=[t(i-1,j-1) t(i-1,j) t(i-1,j+1);t(i,j-1) t(i,j) t(i,j+1);
t(i+1,j-1) t(i+1,j) t(i+1,j+1)];
        toplam=sum(sum(filtre*matris));
        g(i,j)=toplam;
    end
end
imshow(g);
end
```

oluşan gauss filtreli görüntü şekil 3.5'tedir.

Şekil 3.5: Gauss filtreli görüntü

3.2 Kenar Belirleme Filtreleri

Görüntüdeki kenarların belirlenmesi, görüntü işleme ile yapılacak işlemlerin çoğunda gerekli olan objelerin tespiti aşamasında önemlidir. Kenar, görüntüdeki renk geçişlerinin olduğu bölgedir. Renk geçişi piksel değerleri arasında farkın olduğu bölge anlamına geldiği için pikseller arasında fark alındığında diğer yerler 0'a yaklaşırken kenarların olduğu yerler yüksek değerler alır (Qureshi, 2005). Bu işleme görüntü üzerinde türev alma işlemi denir.

3.2.1 Sobel kenar filtresi

Sobel kenar filtresi işlem yapılacak görüntüdeki kenarların belirlenmesini sağlar. Türev almaya dayalı filtrelerden biridir. Görüntüye uygulanacak filtre 3×3'lük olup x ve y koordinatlarına ayrı ayrı uygulanır (Qidwai ve Chen, 2009).

$$R_{x,y} = \begin{bmatrix} x,y & x,y+1 & x,y+2 \\ x+1,y & x+1,y+1 & x+1,y+2 \\ x+2,y & x+2,y+1 & x+2,y+2 \end{bmatrix}$$

görüntü üzerindeki koordinatlar olmak üzere,

$$R_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, R_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

x ve y koordinatlarına uygulanacak Sobel filtreleri olup x,y deki piksel değeri şu şekilde ifade edilir:

$$R_{x,y} = \sqrt{R_x^2 + R_y^2}$$

```
x=imread('D:\Matlab\resim4.jpg');
y=rgb2gray(x);
z=edge(y,'sobel');
imshow(z);
```

Sobel filtresi için aşağıdaki gibi bir fonksiyon yazılabilir.

```
function [ ] = Sobel( adres )
r=imread(adres);
g=rgb2gray(r);
t=double(g);
[x,y]=size(t);
for i=2:x-1
    for j=2:y-1
        k1=t(i-1,j+1)+2*t(i,j+1)+t(i+1,j+1)...
            -(t(i-1,j-1)+2*t(i,j-1)+t(i+1,j-1));
        k2=t(i-1,j-1)+2*t(i-1,j)+t(i-1,j+1)...
            -(t(i+1,j-1)+2*t(i+1,j)+t(i+1,j+1));
```

```
        g(i,j)=sqrt(k1.^2+k2.^2);
    end
end
imshow(g);
end
```

fonksiyon Sobel('bellek adresi'); şeklinde kullanıldığında oluşan görüntü şekil 3.6'dadır.

Şekil 3.6: Sobel filtreli görüntü

RGB görüntü için aşağıdaki gibi bir fonksiyon yazılabilir.

```
function [  ] = renklisobel(adres)
r=imread(adres);
[x,y,z]=size(r);
r=double(r);
t=uint8(zeros(x,y,3));
k1=[0 0 0];
k2=[0 0 0];
for i=2:x-1
    for j=2:y-1
        for l=1:z
        k1(l)=r(i-1,j+1)+2*r(i,j+1)+r(i+1,j+1)...
            -(r(i-1,j-1)+2*r(i,j-1)+r(i+1,j-1));
```

```
        k2(l)=r(i-1,j-1)+2*r(i-1,j)+r(i-1,j+1)...
            -(r(i+1,j-1)+2*r(i+1,j)+r(i+1,j+1));
        t(i,j,l)=sqrt(k1(l)^2+k2(l)^2);
        end
    end
end
imshow(t);
end
```

Şekil 3.7: Renkli sobel filtreli görüntü

Kenar bulma filtrelerindeki asıl amaç bit sayısını minimuma indirerek kenarlara odaklanmak olduğu için genellikle RGB görüntü ile çalışılmaz.

3.2.2 Prewitt kenar filtresi

Prewitt kenar filtresi işlem yapılacak görüntüdeki kenarların belirlenmesini sağlayan türev almaya dayalı filtrelerden bir diğeridir. Görüntüye uygulanacak filtre 3×3'lük olup x ve y koordinatlarına ayrı ayrı uygulanır (Qidwai ve Chen, 2009).

$$R_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \; R_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

x ve y koordinatlarına uygulanacak Prewitt filtreleri olup x,y deki piksel değeri şu şekilde ifade edilir.

$$R_{x,y} = \sqrt{R_x^2 + R_y^2}$$

```
x=imread('D:\Matlab\resim4.jpg');
y=rgb2gray(x);
z=edge(y,'prewitt');
imshow(z);
```

Prewitt filtresi için aşağıdaki gibi bir fonksiyon yazılabilir.

```
function [ ] = Prewitt(adres)
r=imread(adres);
g=rgb2gray(r);
t=double(g);
[x,y]=size(t);
for i=2:x-1
    for j=2:y-1
    k1=t(i-1,j+1)+t(i,j+1)+t(i+1,j+1)...
        -(t(i-1,j-1)+t(i,j-1)+t(i+1,j-1));
    k2=t(i+1,j-1)+t(i+1,j)+t(i+1,j+1)...
        -(t(i-1,j-1)+t(i-1,j)+t(i-1,j+1));
    g(i,j)=sqrt(k1.^2+k2.^2);
    end
end
imshow(g);
end
```

fonksiyon Prewitt('bellek adresi'); kodu ile kullanıldığında figure penceresindeki görüntü şekil 3.8'dedir.

Şekil 3.8: Prewitt filtreli görüntü

3.2.3 Roberts kenar filtresi

Kenar bulmak için türev almaya dayalı filtrelerden biridir. Sobel ve Prewitt filtreleri gibi kullanımı aynı olup farklı filtreye sahiptir.

$$R_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \; R_y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

x ve y koordinatlarına uygulanacak Roberts filtreleri olup x,y deki piksel değeri aşağıdaki gib ifade edilir.

$$R_{x,y} = \sqrt{R_x^2 + R_y^2}$$

```
x=imread('D:\Matlab\resim4.jpg');
y=rgb2gray(x);
z=edge(y,'roberts');
imshow(z);
```

Roberts filtresi için aşağıdaki gibi bir fonksiyon yazılabilir.

```
function [ ] = Roberts( adres )
r=imread(adres);
g=rgb2gray(r);
t=double(g);
```

```
[x,y]=size(t);
for i=2:x-1
    for j=2:y-1
    k1=t(i,j)-t(i+1,j+1);
    k2=t(i,j+1)-t(i+1,j);
    g(i,j)=sqrt(k1.^2+k2.^2);
    end
end
imshow(g);
end
```

fonksiyon, Roberts('bellek adresi'); kodu ile kullanıldığında figure penceresindeki görüntü şekil 3.9'dadır.

Şekil 3.9: Roberts filtreli görüntü

3.2.4 Canny kenar filtresi

Matlab'ta ki bir diğer kenar bulma filtresidir. Kenarları inceltip eşikleme (threshold) değeri ile doğru şekilde ayarladığı için en iyi kenar bulma filtresi olduğu kabul edilir. Canny filtresinde ilk etapta görüntüdeki bozunma Gauss çekirdeği ile konvolüsyonu alınarak minimize edilir. Sonrasında gradient büyüklüğü ve yönü hesaplanıp kenarlar inceltilir. Daha sonra ikili eşiklemeden geçirilerek istenmeyen kısımları atılır (Qidwai ve Chen, 2009).

```
x=imread('D:\Matlab\cameraman.png');
y=rgb2gray(x);
z=edge(y,'canny');
imshow(z);
```

Şekil 3.10: Canny filtreli görüntü

3.2.5 Laplace filtresi

Laplace filtresi görüntü üzerinde 2. türev kullanılarak oluşturulan bir kenar bulma filtresidir (Solomon ve Breckon, 2008). Laplace filtrenin kullandığı formül;

$$\nabla^2 r(x,y) = \frac{\partial^2 r}{\partial x^2} + \frac{\partial^2 r}{\partial y^2} = \frac{4}{(\alpha+1)} \times \begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

şeklindedir. İfadenin oluşumu,

$$\begin{aligned} \frac{\partial^2 r}{\partial x^2} &= \lim_{\nabla x \to 0} \frac{(\partial r/\partial x)(x+\nabla x, y)(\partial r/\partial x) r(x,y)}{\nabla x} \\ \frac{\partial^2 r}{\partial y^2} &= \lim_{\nabla y \to 0} \frac{(\partial r/\partial x)(x, y+\nabla y)(\partial r/\partial x) r(x,y)}{\nabla y} \end{aligned}$$

$$\nabla^2 r(x,y) = r(x+1,y) + r(x-1,y) - 4r(x,y) + r(x,y+1) + r(x,y-1)$$

şeklindedir. Bu durumdan sonra oluşan ve kullanılan laplace filtreleri ise,

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

şeklindedir.

```
>> x=imread('D:\Matlab\resim4.jpg');
>> y=fspecial('laplacian',0.5);
>> son=imfilter(x,y);
>> imshow(son);
```

filtre ile oluşan görüntü şekil 3.11'dedir.

Şekil 3.11: Laplace filtreli görüntü

3.2.6 Log filtresi

Log filtresi bir diğer kenar bulma filtresi olup Gauss filtresinin Laplacian'i olarak da bilinir. Gauss filtresinin kullandığı formül;

$$r(x,y) = \frac{1}{2\pi\sigma^2} \times e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Laplacian filtrenin kullandığı formül;

$$\nabla^2 = \frac{\partial^2 r}{\partial x^2} + \frac{\partial^2 r}{\partial y^2} = \frac{4}{(\alpha+1)} \times \begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

Gauss filtrenin Laplaciani olarak bilinen Log filtresinin formülü;

$$Log(x,y) = -\frac{1}{\pi\sigma^4} \times (1 - \frac{x^2+y^2}{2\sigma^2}) \times e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

şeklindedir.

```
>> x=imread('D:\Matlab\lena.jpg');
>> y=fspecial('log');
>> z=imfilter(x,y);
>> imshow(z);
```

şeklindedir. Filtreleme sonucu oluşan görüntü şekil 3.12'dedir.

Şekil 3.12: Log filtreli Lena görüntüsü

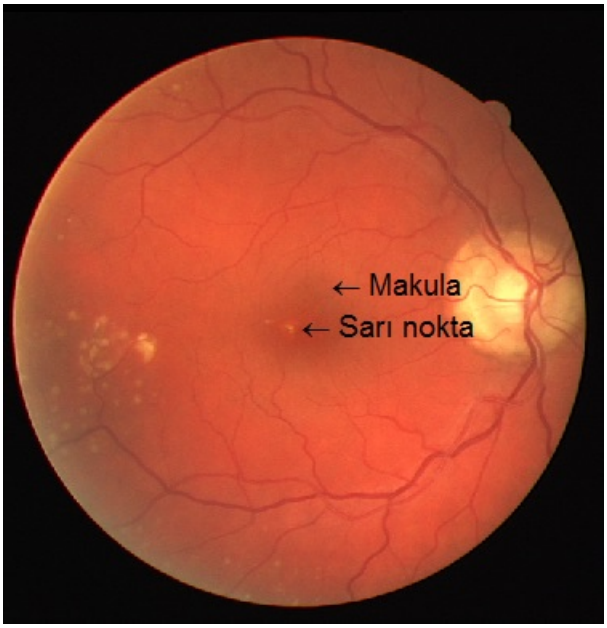
4. TIP ALANINDA GÖRÜNTÜ İŞLEME UYGULAMALARI

Göz, vücuttaki en hassas organlardan biri olması, elle muayene imkanının çok az olması ve göz doktorlarının dışarıdan göze bakarak sınırlı sayıda alabildiği bilgi üzerinden yorum yapmak zorunda kaldıkları için dışarıdan yardıma ihtiyaç duydukları önemli bir tıp alanıdır. Göz doktorları gözdeki en küçük hastalıklardan arpacık gibi, önem derecesi en yüksek hastalıklara kadar kornea epitelinin zedelenmesi hatta kornea nakline kadar birçok rahatsızlıkla ilgilenirler. Sarı nokta hastalığı, glokom, kornea epitel bozukluğunun büyüklüğü, retina veya korneadan kan damarları üzerine çalışmalar v.s. gibi birçok durum görüntü işleme teknikleriyle incelenebilir.

Bu bölümde görüntü işlemenin yoğunlaştığı göz hastalıklarından sarı nokta hastalığı üzerindeki yapılan çalışmalar anlatıldıktan sonra dermatoloji alanında iyi-kötü huylu ben tespitine değinilecektir. Son olarakta özellik çıkarımı konusu incelenecektir.

4.1 Sarı Nokta Hastalığı (Age-related Macular Degeneration (AMD))

Yaşa bağlı makula dejenerasyonu olarak adlandırılan, halk dilinde ise sarı nokta hastalığı olarak bilinen hastalık gözde druzen denilen koroidea'nın bazal tabakasında hiyalin birikmesiyle sarımsı kabartılar şeklinde gözlenir. Bu durum nadiren görmeyi etkilemekle birlikte bunun kronik hale gelmesiyle sarı nokta hastalığı oluşur. 50 yaş üzeri insanlar arasında görme kayıplarının en büyük nedenlerinden biri olarak da gösterilmektedir (De Jong, 2006). Sarı nokta hastalığındaki riski arttıran birkaç faktör yaş, sigara kullanımı, yüksek tansiyon ve genetik olarak belirtilmektedir (Chapdar vd., 2003), (Evans, 2001), (Bhuiyan vd., 2014). Dünya Sağlık Örgütü (WHO), son zamanda 8 milyon insanın sarı nokta hastalığından dolayı kör olduğunu belirtmiştir. Birleşmiş milletler bu hastalıkla mücadele eden insan sayısının 2020 yılında 196 milyona, 2040 yılında ise 288 milyona ulaşabileceğini belirtmektedir (Wong vd., 2014).



Şekil 4.1: Detaylı göz altı görüntüsü

Ortaya konulan klinik sonuçlarına göre sarı nokta seviyesi 3 sınıfa ayrılmıştır (De Jong, 2006). Kısaca özetlemek gerekirse;

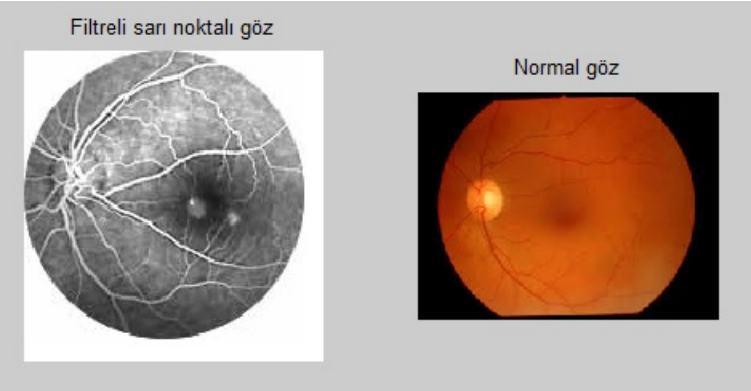
$(i)-$ Druzen, çap olarak $15\mu m$ ile $63\mu m$ arasında ise erken sarı nokta (Early AMD) olarak adlandırılmaktadır (De Jong, 2006), (Hijazi vd., 2012). Bu durumda makulada normal olmayan lezyonlar görülmektedir.

$(ii)-$ Druzen, çap olarak $63\mu m$ ile $125\mu m$ arasında ise orta seviye sarı nokta (Intermediate AMD) olarak adlandırılır (De Jong, 2006), (Hijazi vd., 2012). Lezyonlar makulanın çevresinde gözlenmektedir.

$(iii)-$ Daha üst seviyedekiler ise geç sarı nokta (Late AMD) olarak adlandırılmaktadır. Lezyonlar makulanın merkezinde toplanmıştır. Bu seviyedeki sarı nokta 2 şekilde ifade edilmektedir.

$(a)-$ Makulanın merkezinde druzen ve Geographic Athropy (GA)'de ki duruma göre kuru tip veya neovascular olmayan sarı nokta olarak adlandırılır. Sarı noktadan oluşan görme kayıplarının neredeyse %90'ı bu tipten dolayı oluşmaktadır.

$(b)-$ Choroidal Neovascularization (CNV) tarafından neovascular AMD olarak sınıflandırılan ıslak sarı noktadır. Gözde kan sızıntısı görülebilmektedir. Yaklaşık %10'u görme kaybına yol açar.



Şekil 4.2: Filtrelenmiş sarı noktalı ve normal göz görüntüleri

Sarı nokta retinanın dip görüntülerinde belirlenen druzen segmentasyonu tarafından teşhis edilebilir. Sarı nokta hastalığı teşhis sürecinde druzen segmentasyonu ve ölçümüne ihtiyaç duyulmaktadır. Birkaç bilim adamı sarı nokta hastalığı teşhisi için çalışmalar yapmıştır.

Brandon ve Hoover (2003) retinal görüntüden druzen'ı ortaya çıkarabilmek için çok seviyeli analiz ortaya çıkardılar. Bu analizlerin doğru teşhis oranı %87 olarak belirlenmiştir. Fakat bu metot optik disk ve kan damarı varyasyonlarına duyarlıdır. Bensbeh vd. (2001) belirlediği parlak noktalara matematiksel morfoloji uyguladı. Bu morfoloji druzen'dan alan, şekil ve kontrast çıkarılmasını sağlamıştır. Fakat bu metodun performans ölçümü kaydedilmemiştir. HALT operatörü druzen segmentasyonunda kullanılmaktadır. Duyarlılık %98 olarak kaydedilmiştir (Rapantzikar vd. 2007). Druzen'ın belirlenmesinde Gauss filtresi ve k-NN sınıflandırma kullanılmıştır (Niemeijer vd. 2007). Duyarlılığı sırasıyla %77 ve %88 olarak raporlanmıştır. Metot bütün druzen noktalarını belirlemede etkili olamamıştır. Soliz vd. (2009) druzen'ları belirlemek için Independent Component Analysis(ICA) kullanmıştır. Sadece 12 retina altı görüntüde çalışılmış olup %100 belirleme sağlanmıştır. Druzen belirlemede Amplitude Modulation(AM) - Frequency Modulation(FM) tabanlı çok ölçekli çıkarım kullanılmış ve kaydedililmiştir (Barraga vd. 2009). Bu metot düşük yoğunluklu retina altı görüntüleri sınıflandırabilir. Normal ve sarı noktalı görüntülerden druzen belirlemek için Mexican hat wavelet ve Support Vector Data Description(SVDD) kullanılarak sadece 7 görüntü test edilmiş ve %100 doğruluk sağlanmıştır (Freund vd. 2009). Liang vd. (2010) yoğun tabanlı druzen segmentasyon yaklaşımını ortaya koydular. Hassasiyet ve doğru belirleme %75 olarak raporlandı. Burlina vd. (2011) yaklaşımlarında makulanın yeri ve kan damarı çıkarılması üzerine yoğunlaştılar. Druzen segment noktaları için çoklu çözünürlükte yerel adaptif segmentasyon metodunu ortaya koydular. Bu metodun hassasiyeti %95, doğru belirleme ise %96 olarak kaydedilmiştir. Druzen segmentinde Sobel operatörü ve Gauss fonksiyonu kullanıldı (Mora vd. 2011). Bu metotta %60 kappa doğrusu elde edildi. Druzen ve benekleri ortaya çıkarmak için Optimal filter bank geliştirildi (Quellec vd. 2011). Metot,

Area Under receiver operator characteristics Curve(AUC)'ün %85'ini ortaya koydu. Retina altı görüntülerde druzen teşhisi için BIF ortaya koyuldu (Cheng vd. 2012). Bu metotta BIF 'i geliştirmek için Gabor fonksiyonu kullanıldı. Metodun hassasiyeti %86.30, doğruluğu %91,90 olarak kaydedilmiştir. Sarı nokta belirlemek için ters anomali segmentasyonu kullanıldı (Köse vd. 2008). Bu metot %90 doğruluk sağlamıştır. Case Based Reasoning(CBR) ve Dynamic Time Warping(DTW) metodu ise normal ve sarı nokta ayrımı için ortaya konulmuştur (Hijazi vd. 2012). Metodun duyarlılığı %86 olarak sınıflanmıştır. İstatistiksel veri çıkarımı için kullanılan ters segmentasyon, sarı nokta görüntülerinden sağlıklı ve sağlıksız alanların belirlenmesi için kullanılmaktadır (Köse vd. 2010). sağlıklı alanların belirlenmesinde %92.76, sağlıksız alanların belirlenmesinde ise $\%96-100$ aralığında doğruluk kaydedilmiştir. Wavelet, GLCM, color, histogram tabanlı özellikler ve Weighted Frequent Sub-Graph Mining(WFSM) normal ve sarı noktalı sınıfların belirlenmesi için kullanılmıştır (Hijazi vd. 2015). Bu metodun doğruluğu $\%99,9$ olarak kaydedilmiştir.

Bu yapılan çalışmalar görüntünün kontrastını iyileştirmek için Contrast Limited Adaptive Histogram Equalization (CLAHE) denilen görüntüyü kullanılacak kontrast seviyesine getiren hazırlık süreci ile başlar. Daha sonra LCP tabanlı özellikler normal ve sarı noktalı sınıfları birbirinden ayırır. Daha sonra çıkarılan özelliklerin istatistiksel önemi t-testi kullanılarak değerlendirilir. Son olarakta seçilen önemli özellikler, çeşitli çekirdek fonksiyonları ile DT, k-NN, NB, PNN ve SVM sınıfladırıcıları ile sıralanır. Sınıflamaların performansı ise 10 kat çapraz doğrulama stratejisi ile değerlendirilir.

Önemli sayılabilecek birkaç retina altı(fundus) görüntü veri topluluğu bulunmaktadır. Bunların bazıları;

$(i)-$ Private, Hindistandaki Kasturba Tıp okulundaki Opthalmology bölümünden alınan 270'er normal ve sarı noktalı fundus görüntüden oluşmaktadır.

$(ii)-$ ARIA (Automated Retinal Image Analysis), St Pauls Eye Unit ve Liverpool Üniversitesindeki kliniklerden toplanan 101 normal, 60 sarı noktalı görüntüden oluşmaktadır.

$(iii)-$ STARE (Structured Analysis of the Retina), Amerikada bulunan Kalifornia üniversitesi ve emekliler topluluğu tıp merkezi tarafından 36 normal, 47 sarı noktalı görüntüden oluşmaktadır.

4.2 Özellik Çıkarımı

Göz alanında gözün derecesel bozukluğu (miyop, hipermetrop, v.s.) dışındaki kusurların çoğu (cerrahi yaralanmalar hariç) kan, göz damarlarının bozulması sonucu oluşan rahatsızlıklardan ileri gelmektedir. Derecesel bozukluklar gözün uzunluğu ve kırma gücünün fazla veya az olması ile ilgilidir. Örneğin miyopta gözün ön-arka çapı gözün kırıcı bileşenlerine göre fazla uzundur. Bu sebeple oradaki açısal durumdan derece ölçümü yapılır. Özellik çıkarılması konusu görüntü üzerinde işlemler yapma gereksiniminden dolayı zorlayıcıdır. Tezin bu kısmında özellik çıkarımı konusunda göz alanına girmeden dermatoloji alanındaki kullanımına değinilecektir.

4.2.1 İyi-kötü huylu ben tespiti

Özellik çıkarımı konusunda uygulama alanı ararken dermatoloji alanında da birkaç konu detaylı incelendi. Bunlardan en uygun olanı bilimsel açıdan kullanılan iyi-kötü huylu ben tespitidir. Burada ben üzerinde malinite tanısı koymak için kaos oluşumları incelendi.

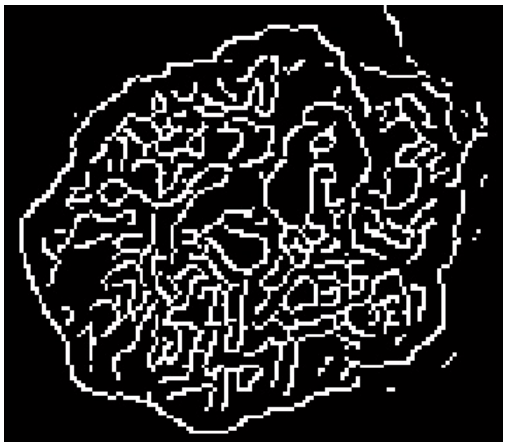
Kaos denilen kavram, bende yapı ve renk asimetrisi olarak tanımlanır. Malinitenin 8 ipucundan birini sağlıyorsa kaos vardır denir ve hastaya biyopsi yapılması önerilir. Malinitenin 8 ipucu (Rosendahl vd. 2016),

1– Eksantrik yapısız alan.
2– Kalın retiküler veya dallanan çizgiler.
3– Gri veya mavi yapılar.
4– Periferal siyah nokta veya klodlar.
5– Segmental radyal çizgiler.
6– Beyaz çizgiler.
7– Polimorf damarlar.
8– Sırtlarda paralel çizgiler.

şeklindedir.

Şekil 4.3: Vücutta ben görüntüsü

Görüntüde kaosun belirtileri görülür. Dallanan çizgiler, gri yapılar, segmental radyal çizgiler, beyaz çizgiler gibi kaos belirtileri bende mevcuttur. Görüntüdeki renk ayrımı kısmı gayet açık şekilde bir threshold değeri belirleyerek elde edilir. Çizgiler ile ilgili kısım için ise filtreden geçirerek ben, şekil 4.4'deki gibi yorumlamaya uygun hale getirilir.



Şekil 4.4: Canny filtresinden geçirilmiş ben görüntüsü

şeklindedir. Daha detaylı hale getirmek için ise filtreden geçirilen görüntünün son hali kenarlardaki eğrilik (tortuosity) değerlerine göre uygun bir programda yorumlanır.

4.2.2 Retinadan kan damar özelliklerinin çıkarılması

Bu kısımda retina altı görüntülerden kan damarlarının özellik çıkarımı anlatılacaktır. Bu bölüm aynı zamanda iris tanıma ve parmak izi tespiti ile de benzerlik göstermektedir.

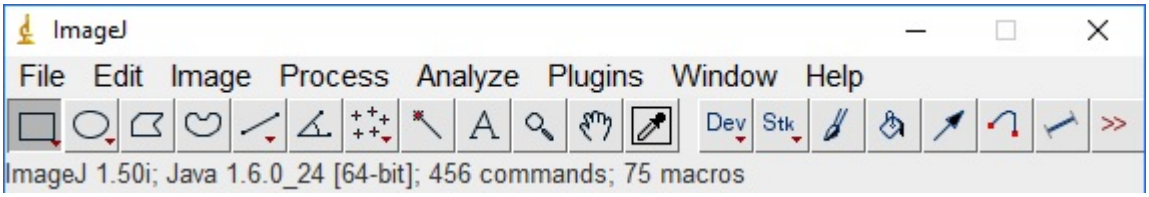
Göz alanında özellik çıkarımı konusunda çalışma yapmak istememizdeki sebep uygulama alanının geniş olması nedeniyledir. Örneğin; retina, kornea gibi konularda bu işlem

yapıldığında hem hasta için anlık durum bilgisi alınırken uygulanacak tedavi sonucunda gözün geldiği son durum da matematiksel değer ile ifade edilir. Bu durum hekimin gözün durumuna bakarak yorum yapmasındansa değerler üzerinden yorum yapmasını sağladığından dolayı hata payını minimize eder. Bu durum ayrıca yeni ilaçların testi içinde kullanılabilmektedir.

Bu aşamada Java kullananlar için **imagej** görüntü analiz programı kullanılabilir. Bu kısımda kısaca imagej programı anlatılacaktır.

4.2.2.1 Imagej

Java ile yazılmış bir arayüzdür. Program, çok sayıda plugin'den oluşmaktadır. Dışarıdan plugin'ler eklenebildiği için sürekli gelişim halindedir. Bu konuda çok az sayıda hekimin bildiği imagej'nin pluginlerinden **neuronj**'den yardım alınabilir.



Şekil 4.5: imagej programı kullanıcı arayüzü

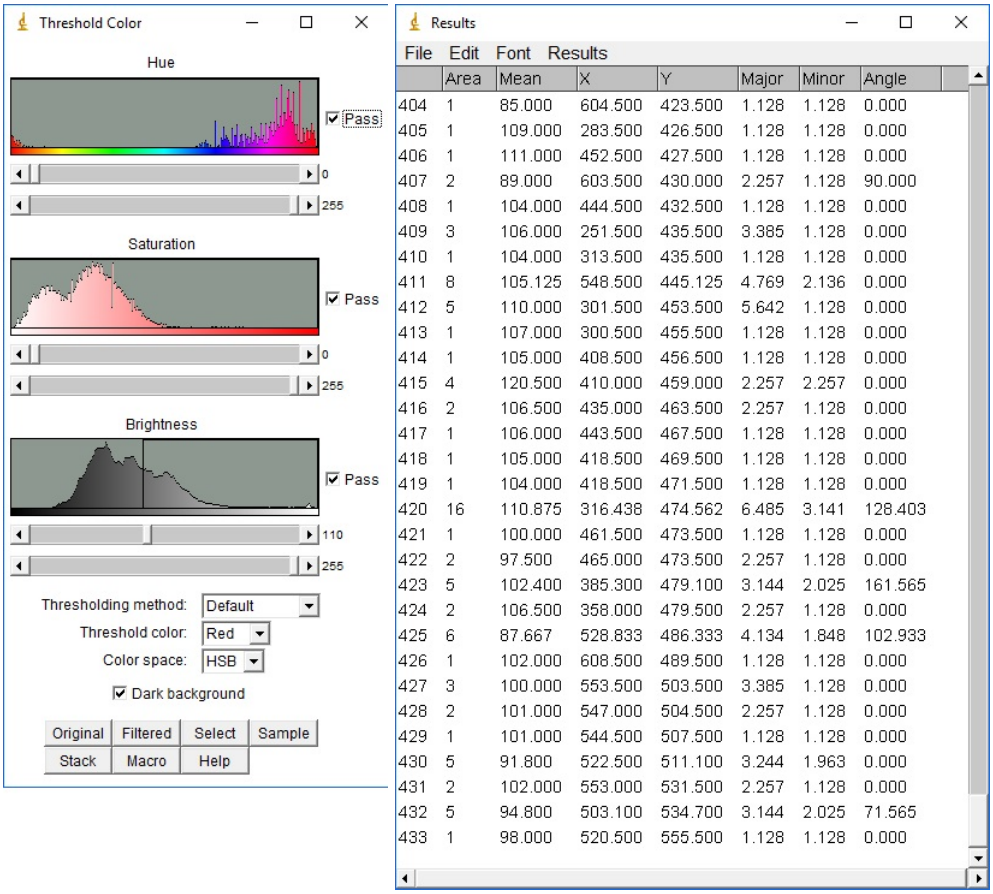
Programda Menu bar, Tool bar ve Status bar olmak üzere 3 bar bulunmaktadır. Tool bardaki ilk 4 tanesi görüntü üzerinde işlem yapılan alanın seçimiyle ilgilidir. 5., çizgi veya bölüm yapmak için, 6., iki çizgi arasında açıyı belirlemek için, 7., görüntü üzerinde kaç nokta seçileceğiyle ilgili olup seçilen noktanın özelliğini, yani parlaklığı v.s. gibi bilgileri kaydetmek için, 8., görüntünün sınırlarını belirlemek için, 9., görüntü üzerinde istenilen font ve yazı türü seçilerek yazı yazmak için, 10., görüntüyü büyültüp küçültmek için kullanılır. Diğerleri de görüntü üzerinde işlem yapmaya yardımcı araçlardır. Burada programın önemli kısımları menu bar'daki Process ve Analyze kısımları olup filtrelerle beraber görüntüler üzerinde çeşitli işlemler yapmaya yardımcı olan ana kısımdır. Process kısmında;

- Smooth, görüntü üzerindeki pürüzleri gidermeye yardımcı olur.
- Sharpen, görüntüyü keskinleştirerek alanların netleşmesini sağlar.
- Find edges, görüntünün köşelerinin belirlenmesini sağlar. Burada türev almaya dayalı operatörlerden birini program otomatik olarak kullanır.
- Find maxima, gürültü toleransını seçtikten sonra çıktı olarak o gürültünün üstünde kalan noktaları görüntüde belirtip list ile ilgili nokta değerlerini excel de listeleme işlemi yapmaktadır.
- Enhance contrast, görüntünün kontrastını arttırır.

- Noise, tuz ve biber gürültüsü gibi çeşitli gürültüler seçilerek görüntüye uygulanması sağlanır.
- Shadows, görüntü üzerinde seçilen yöne doğru gölgeleme işlemi yapar.
- Binary, görüntüye binary modda çeşitli işlemler yapılmasını sağlar.
- Math, görüntü üzerinde alanlar arası çarpma, bölme, and, or, xor, v.s. gibi operatörlerle işlem yapılmasını sağlar.
- FFT, bandpass filter, custom filter gibi filtrelerle işlem yapılmasını sağlar.
- Filters, gaussian blur, mean, median, unsharp mask v.s. gibi filtrelerle görüntü üzerinde işlem yapılmasını sağlar. Bunlarla beraber process kısmında batch, image calculator ve subtract background bulunmaktadır. Analyze kısmında ise ölçümler, kalibrasyon, histogram ve çizimle alakalı kısımlar bulunmaktadır.

imagej ile filtreleme haricinde yapılabilecek örnek bir uygulama,

image - adjust - color threshold'dan tonu, doygunluğu ve parlaklık ayarını belirleyip analyze - analyze particles'dan oluşan durum karşısında alanların ortalaması, aldığı büyük, küçük değerler ve aradaki açı sonucu elde edilir.



	Area	Mean	X	Y	Major	Minor	Angle
404	1	85.000	604.500	423.500	1.128	1.128	0.000
405	1	109.000	283.500	426.500	1.128	1.128	0.000
406	1	111.000	452.500	427.500	1.128	1.128	0.000
407	2	89.000	603.500	430.000	2.257	1.128	90.000
408	1	104.000	444.500	432.500	1.128	1.128	0.000
409	3	106.000	251.500	435.500	3.385	1.128	0.000
410	1	104.000	313.500	435.500	1.128	1.128	0.000
411	8	105.125	548.500	445.125	4.769	2.136	0.000
412	5	110.000	301.500	453.500	5.642	1.128	0.000
413	1	107.000	300.500	455.500	1.128	1.128	0.000
414	1	105.000	408.500	456.500	1.128	1.128	0.000
415	4	120.500	410.000	459.000	2.257	2.257	0.000
416	2	106.500	435.000	463.500	2.257	1.128	0.000
417	1	106.000	443.500	467.500	1.128	1.128	0.000
418	1	105.000	418.500	469.500	1.128	1.128	0.000
419	1	104.000	418.500	471.500	1.128	1.128	0.000
420	16	110.875	316.438	474.562	6.485	3.141	128.403
421	1	100.000	461.500	473.500	1.128	1.128	0.000
422	2	97.500	465.000	473.500	2.257	1.128	0.000
423	5	102.400	385.300	479.100	3.144	2.025	161.565
424	2	106.500	358.000	479.500	2.257	1.128	0.000
425	6	87.667	528.833	486.333	4.134	1.848	102.933
426	1	102.000	608.500	489.500	1.128	1.128	0.000
427	3	100.000	553.500	503.500	3.385	1.128	0.000
428	2	101.000	547.000	504.500	2.257	1.128	0.000
429	1	101.000	544.500	507.500	1.128	1.128	0.000
430	5	91.800	522.500	511.100	3.244	1.963	0.000
431	2	102.000	553.000	531.500	2.257	1.128	0.000
432	5	94.800	503.100	534.700	3.144	2.025	71.565
433	1	98.000	520.500	555.500	1.128	1.128	0.000

Şekil 4.6: imagej programı uygulama görüntüsü

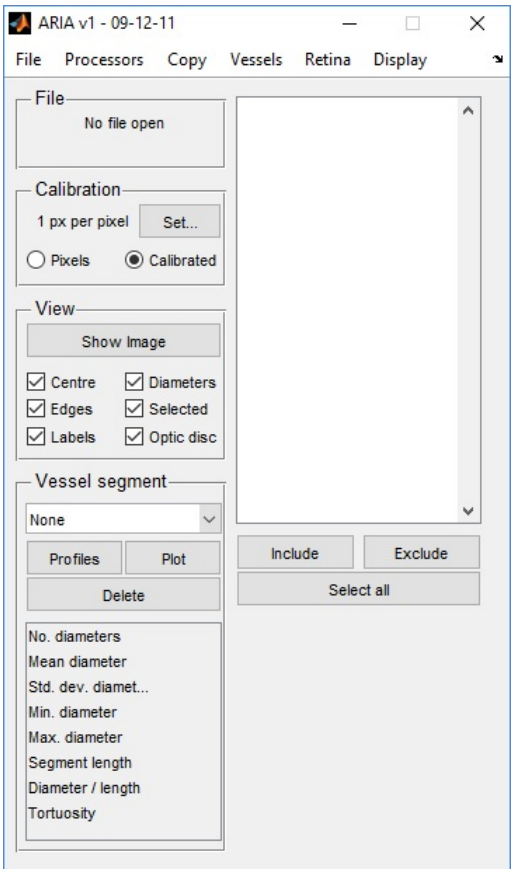
Java'da yazılan pluginlerden Alpha_Blending plugin'inin Java kodunun giriş kısmı (Burger ve Burge, 2007),

```
import ij.IJ;
import ij.ImagePlus;
import ij.WindowManager;
import ij.gui.GenericDialog;
import ij.plugin.filter.PlugInFilter;
import ij.process.*;
public class Alpha_Blending implements PlugInFilter {
static double alpha = 0.5; // Arka plan renginin golgelenme orani icin
ImagePlus fgIm = null; // Arka plan resmi icin
public int setup(String arg, ImagePlus imp) {
return DOES_8G;
}
public void run(ImageProcessor bgIp) {
if(runDialog()) {
ImageProcessor fgIp
fgIm.getProcessor().convertToByte(false);
fgIp = fgIp.duplicate();
fgIp.multiply(1-alpha);
bgIp.multiply(alpha);
bgIp.copyBits(fgIp, 0, 0, Blitter.ADD);
...
}
}
```

Tezin bu bölümünde retina görüntülerinden kan damarlarının çıkarılması konusunda Aria programı anlatılacaktır.

4.2.2.2 Aria

Matlab GUI ile yazılmış bir retina analiz programıdır. Arayüzü şekil 4.7'dedir.



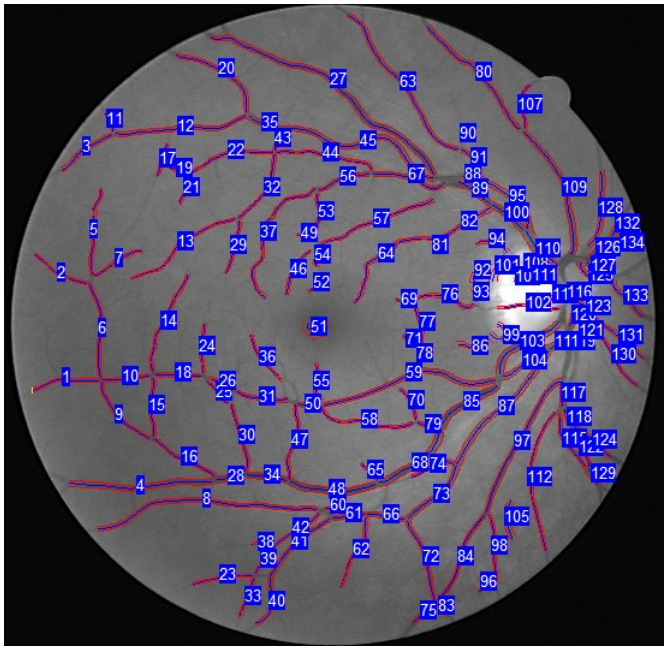
Şekil 4.7: Aria programı kullanıcı arayüzü

Program her bir damarın ortalama çapını, standart sapmasını, minimum, maksimum değerlerini, uzunluğunu ve eğriliğini piksel olarak hesaplar.

Aria, file, processors, copy, vessels, retina ve display kısımlarından oluşan bir menüye sahiptir. Menüye ek olarak arayüzde bulunan view kısmındaki checkbox'lar ile damarların çizimi için istenilenlerin seçilmesi sağlanır. Merkezler, kenarlar, damar numaraları, çap gibi damar özelliklerinden hangilerinin elde edilecek görüntüde görünmesi istendiği seçilir. Seçilen bir retina görüntüsüne karşılık damar görüntüsü şekil 4.8'dir.

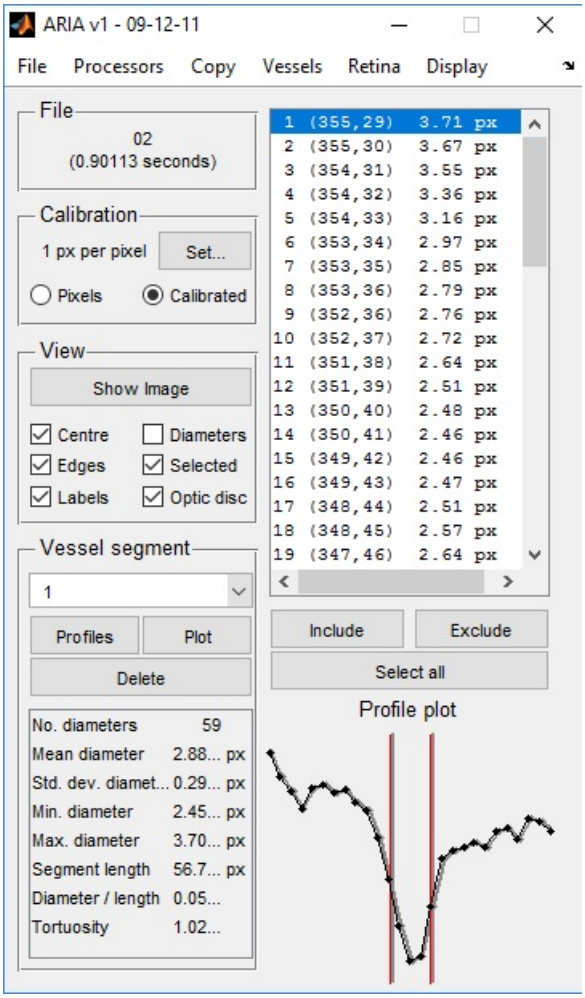
Şekil 4.8: Aria programı damar görüntüsü

Damarların detaylandırılmış olduğu görüntü ise şekil 4.9'dur.



Şekil 4.9: Aria programı detaylı damar görüntüsü

Arayüzden vessel segment kısmından numaralandırılmış damarlardan herhangi biri seçilerek özellikleri şekil 4.10'da görüldüğü gibi sol alttan incelenebilir.



Şekil 4.10: Aria programı damar özellikleri

5. LİTERATÜR ARAŞTIRMASI

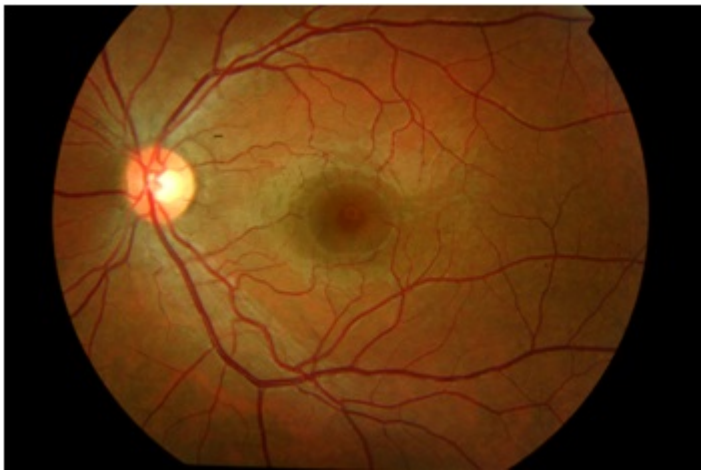
Yeni kenar belirleme filtresine geçmeden önce literatürde görüntünün özdeğer, özvektörleriyle yapılmış makalelerden ikisine değinelim. Bunlardan ilki Acevedo vd.(2014) hazırlamış oldukları kenar bulmada kullanılan assosyatif yaklaşımdır. Burada görüntü binary formata yani $0, 1$'li formata çevrildikten sonra Alpha-Beta Assosyatif hafıza ile işleme sokulur. Sonuç matrisi üzerinde 3×3'lük alanlar vektör olarak alınıp görüntü taranarak özdeğer, özvektörleri elde edilir. En iyi sonuç veren özdeğerine karşılık gelen özvektör ve bunun transpozu ile görüntü filtreden geçirilerek işlenmektedir. Burada yazarlar tek görüntü üzerinde çalıştıkları için filtreleme için görüntüden çıkan bütün özvektörleri inceleyip, uygun olanı seçtiler. Bu durumun retinaya uygulanamamasının sebebi retinal görüntülerde optik diskin bulunduğu alan çok parlak olduğu için binary görüntüye dönüldüğünde görüntünün sadece o ve ona yakın alanları kalmakta ve görüntü anlamsızlaşmaktadır. Uygun bir threshold değeri ile de şekil 5.1'deki gibi istenilen sonuç alınamamaktadır.

Şekil 5.1: Binary formatta retinal görüntü

Diğer bir çalışmada ise Sahbi vd.(2007) bozulmuş görüntüyü eski haline getirebilmek için graph laplacian ile çalışıp yine özvektörler aracılığıyla işlemler yaptılar. Bunlar gibi özdeğer ve özvektörler ile yapılmış literatürde çeşitli çalışmalar bulunmaktadır.

6. KENAR BELİRLEMEDE YENİ FİLTRELEME YÖNTEMİ

Tezin bu bölümünde filtreler kısmında detaylı olarak ele alınan filtrelerden farklı bir filtre elde edip diğer filtreler ile retinal görüntü üzerinde karşılaştırması yapılacaktır. Retinal görüntü olarak şekil 6.1'deki görüntü kullanılacaktır.

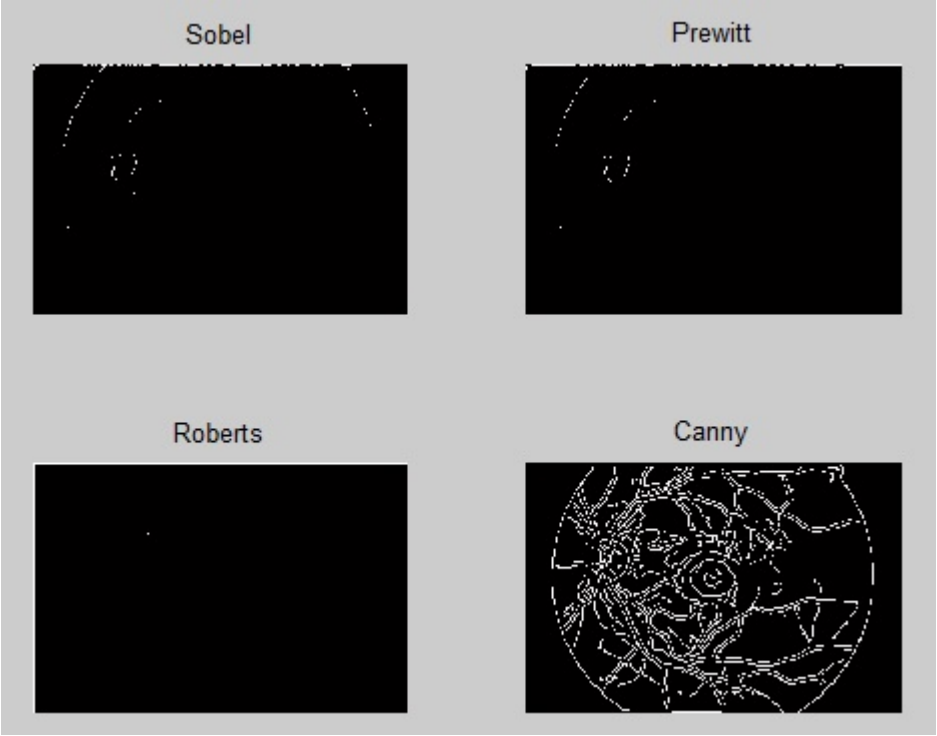


Şekil 6.1: Retinal görüntü

Görüntüyü bilinen kenar bulma filtrelerinden Sobel, Prewitt, Roberts ve Canny filtreleri ile filtreleyelim.

```
>> x=imread('D:\Matlab\iris3.jpg');
>> y=rgb2gray(x);
>> a=edge(y,'sobel');
>> b=edge(y,'prewitt');
>> c=edge(y,'roberts');
>> d=edge(y,'canny');
>> subplot(2,2,1),imshow(a),title('Sobel');
>> subplot(2,2,2),imshow(b),title('Prewitt');
>> subplot(2,2,3),imshow(c),title('Roberts');
>> subplot(2,2,4),imshow(d),title('Canny');
```

Kod bloğu ile oluşan görüntüler şekil 6.2'dedir.

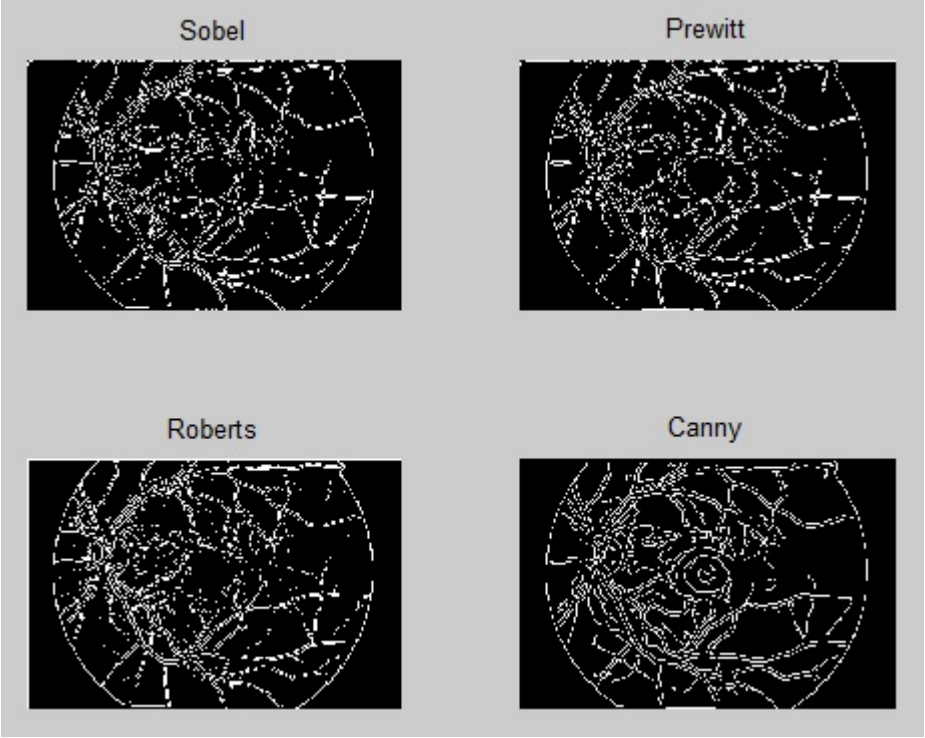


Şekil 6.2: Çeşitli filtrelerle filtrelenmiş retinal görüntü

elde edilen görüntülerden en iyisi Canny kenar bulma filtresi olup diğerleri son derece zayıf kalmıştır. Diğer filtreler için bu görüntüde en uygun görünen threshold değeri ile tekrar kod düzenlendiğinde,

```
>> x=imread('D:\Matlab\iris3.jpg');
>> y=rgb2gray(x);
>> a=edge(y,'sobel',0.02);
>> b=edge(y,'prewitt',0.02);
>> c=edge(y,'roberts',0.02);
>> d=edge(y,'canny');
>> subplot(2,2,1),imshow(a),title('Sobel');
>> subplot(2,2,2),imshow(b),title('Prewitt');
>> subplot(2,2,3),imshow(c),title('Roberts');
>> subplot(2,2,4),imshow(d),title('Canny');
```

oluşan görüntü şekil 6.3'tedir.



Şekil 6.3: Uygun threshold değerleriyle filtrelenmiş retinal görüntü

Çeşitli kenar bulma filtrelerinde kullanılan filtreler incelendiğinde işleme sokulan filtre matrislerin değer kaybı olmaması için elemanlarının toplamları sıfır alınmaktadır. Örneğin Sobel için

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

matrisleri, Prewitt için

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} ve \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

matrisleri, Roberts için

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} ve \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

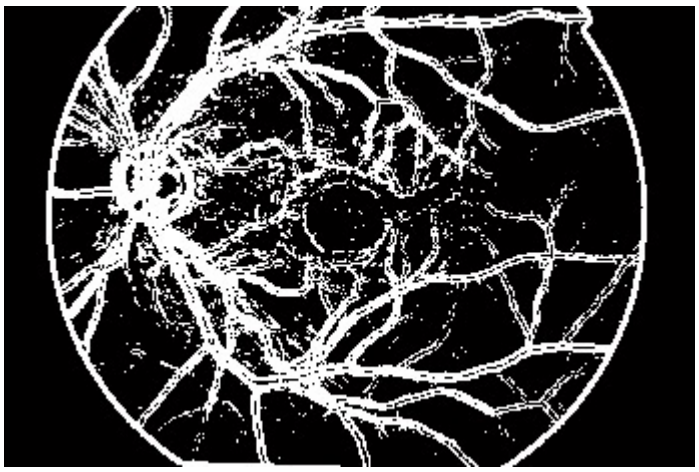
matrisleri kullanılmaktadır. Bu kullanılan matrisler görüntünün aldığı ışığa ve renk yoğunluğuna son derece duyarlıdırlar. Bu sebeple hassasiyeti azaltmak için her görüntüye kendine özgü bir filtre oluşturduk. Oluşturduğumuz filtre için görüntüyü 3×3'lük alanlara ayırarak görüntü üzerindeki bu alanları vektörler kabul edip her 3×3'lük alan için özdeğer ve özvektörlerini bulduk. Toplamsal olarak, bütün görüntü içinde elemanları toplamı sıfıra en yakın özvektör matrisini alıp bu matrisi, elemanlar toplamı sıfır olan matrisine yaklaştırarak filtre matris olarak kabul ettik. Oluşan filtre matrisi de işleme sokarken yine görüntü kayıplarını minimuma indirmek için kendi etrafında 90° döndürmeler ile 4 farklı filtre matrisi oluşturduk. Bu 4 matris ile görüntü matrisini işleme sokup belirlenen threshold değerinin üzerinde kalan alanları belirledik. Bu anlatılan kısım için yazılan fonksiyon ve alt fonksiyon,

```
function [  ] = Tez( adres )
boyut=ndims(adres);
if boyut==3
    adres=rgb2gray(adres);
end
t=ozvektor(adres);
toplam=sum(sum(t));
sayac1=0;
sayac2=0;
for i=1:3
    for j=1:3
    if t(i,j)<=0
        sayac1=sayac1+1;
    else
        sayac2=sayac2+1;
    end
    end
end
if toplam<=0
    sayi=abs(toplam/sayac2);
    for i=1:3
        for j=1:3
            if t(i,j)>0
                t(i,j)=t(i,j)+sayi;
```

```
            end
        end
    end
else
    sayi=abs(toplam/sayac1);
    for i=1:3
        for j=1:3
            if t(i,j)<=0
                t(i,j)=t(i,j)-sayi;
            end
        end
    end
end
y1=t;
y2=[t(3,1),t(2,1),t(1,1);t(3,2),t(2,2),t(1,2);t(3,3),t(2,3),t(1,3)];
y3=[t(3,3),t(3,2),t(3,1);t(2,3),t(2,2),t(2,1);t(1,3),t(1,2),t(1,1)];
y4=[t(1,3),t(2,3),t(3,3);t(1,2),t(2,2),t(3,2);t(1,1),t(2,1),t(3,1)];
filtre1=imfilter(adres,y1);
filtre2=imfilter(adres,y2);
filtre3=imfilter(adres,y3);
filtre4=imfilter(adres,y4);
[x,y]=size(adres);
damarlar=zeros(x,y);
filtreler=zeros(1,4);
threshold=input('Esikleme icin bir threshold degeri giriniz:');
for i=1:x
    for j=1:y
        filtreler(1)=filtre1(i,j);filtreler(2)=filtre2(i,j);
        filtreler(3)=filtre3(i,j);filtreler(4)=filtre4(i,j);
        eb=filtreler(1);
        for k=1:4
            if filtreler(k) > eb
                eb=filtreler(k);
            end
        end
```

```
        if(eb>threshold)
            damarlar(i,j)=eb;
        end
    end
end
imshow(damarlar);
end
function [f]=ozvektor(a)
[x,y]=size(a);
ek=100;
for i=1:x-2
    for j=1:y-2
    k=[a(i,j),a(i,j+1),a(i,j+2);...
        a(i+1,j),a(i+1,j+1),a(i+1,j+2);...
        a(i+2,j),a(i+2,j+1),a(i+2,j+2)];
    [w,z]=eig(double(k));
    toplam=sum(sum(w));
    if abs(toplam) < ek
        ek=toplam;
        f=w;
    end
    end
end
end
```

şeklindedir. Fonksiyon retinal görüntüye uygulandığında elde edilen filtreli görüntü şekil 6.4'tedir.



Şekil 6.4: Filtremiz ile filtrelenen retinal görüntü

görüntüde threshold değeri olarak 30 seçilmiştir. Görüntüdeki gürültüyü minimuma indirmek için çıktı resminde 3×3'lük alanların ortancası alındı. Kod bloğu,

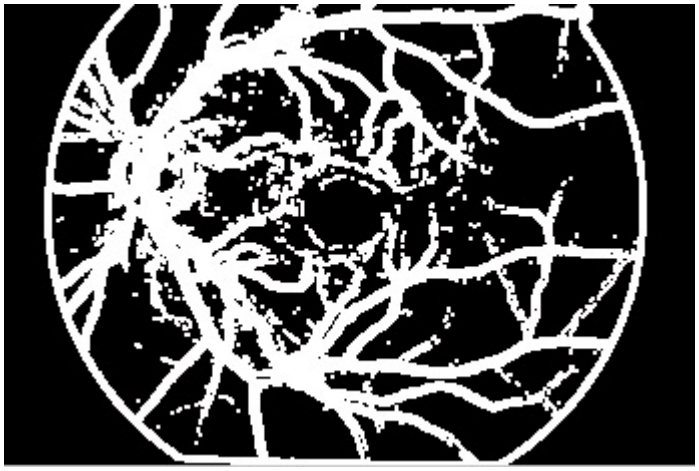
```
function [ ] = Tez2( adres )
boyut=ndims(adres);
if boyut==3
    adres=rgb2gray(adres);
end
t=ozvektor(adres);
toplam=sum(sum(t));
sayac1=0;
sayac2=0;
for i=1:3
    for j=1:3
    if t(i,j)<=0
        sayac1=sayac1+1;
    else
        sayac2=sayac2+1;
    end
    end
end
if toplam<=0
    sayi=abs(toplam/sayac2);
    for i=1:3
        for j=1:3
```

```
            if t(i,j)>0
                t(i,j)=t(i,j)+sayi;
            end
        end
    end
else
    sayi=abs(toplam/sayac1);
    for i=1:3
        for j=1:3
            if t(i,j)<=0
                t(i,j)=t(i,j)-sayi;
            end
        end
    end
end
y1=t;
y2=[t(3,1),t(2,1),t(1,1);t(3,2),t(2,2),t(1,2);t(3,3),t(2,3),t(1,3)];
y3=[t(3,3),t(3,2),t(3,1);t(2,3),t(2,2),t(2,1);t(1,3),t(1,2),t(1,1)];
y4=[t(1,3),t(2,3),t(3,3);t(1,2),t(2,2),t(3,2);t(1,1),t(2,1),t(3,1)];
filtre1=imfilter(adres,y1);
filtre2=imfilter(adres,y2);
filtre3=imfilter(adres,y3);
filtre4=imfilter(adres,y4);
[x,y]=size(adres);
g=zeros(x,y);
filtreler=zeros(1,4);
threshold=input('Esikleme icin bir threshold degeri giriniz:');
for i=1:x
    for j=1:y
        filtreler(1)=filtre1(i,j);filtreler(2)=filtre2(i,j);
        filtreler(3)=filtre3(i,j);filtreler(4)=filtre4(i,j);
        eb=filtreler(1);
        for k=1:4
            if filtreler(k) > eb
                eb=filtreler(k);
```

```
            end
        end
        if(eb>threshold)
            g(i,j)=eb;
        end
    end
end
damarlar=g;
for i=2:x-1
    for j=2:y-1
    b=[];z=0;
for k=i-1:i+1
    for l=j-1:j+1
        z=z+1;
        b(z)=g(k,l);
    end
end
for m1=1:length(b)-1
    byk=m1;
    for m2=m1+1:length(b);
        if b(m2)>b(m1);
            byk=m2;
        end
    end
    if m1~=byk
        gecici=b(m1);
        b(m1)=b(byk);
        b(byk)=gecici;
    end
end
damarlar(i,j)=b(5);
    end
end
imshow(damarlar);
end
```

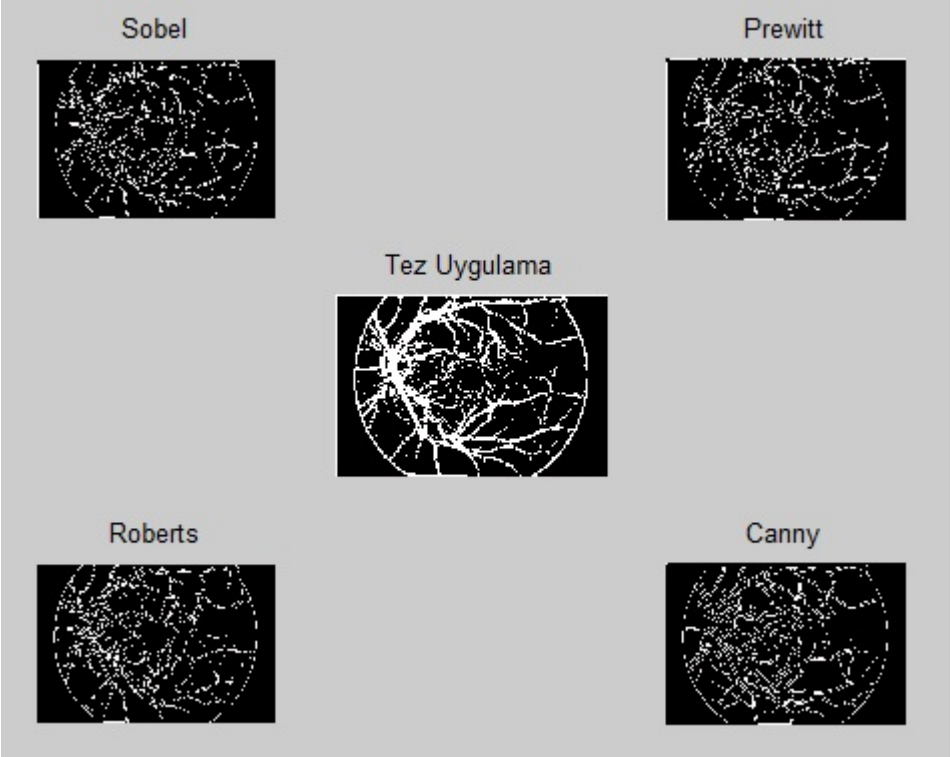
```
function [f]=ozvektor(a)
[x,y]=size(a);
ek=100;
for i=1:x-2
    for j=1:y-2
    k=[a(i,j),a(i,j+1),a(i,j+2);...
        a(i+1,j),a(i+1,j+1),a(i+1,j+2);...
        a(i+2,j),a(i+2,j+1),a(i+2,j+2)];
    [w,z]=eig(double(k));
    toplam=sum(sum(w));
    if abs(toplam) < ek
        ek=toplam;
        f=w;
    end
    end
end
end
```

şeklindedir. Fonksiyonun retinal görüntüye uygulanmış hali şekil 6.5'tedir.



Şekil 6.5: 3×3'lük ortancası alınmış filtremiz ile filtrelenen retinal görüntü

Diğer filtreler ve oluşturduğumuz filtreden çıkan sonuçlar şekil 6.6'dadır.



Şekil 6.6: Filtrenin diğer filtrelerle görsel olarak karşılaştırılması

Görüntüde görüldüğü gibi oluşturulan filtre ile diğer filtreler karşılaştırıldığında Canny filtrenin biraz daha iyi sonuç verdiği, yeni oluşturduğumuz filtrenin ise diğer filtrelerle aynı hatta çoğu bakımdan daha iyi sonuç verdiği gözlenmiştir.

7. SONUÇ VE ÖNERİLER

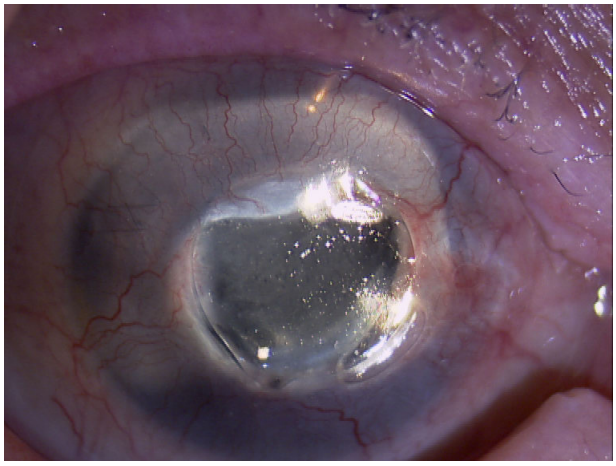
Tez çalışması kapsamında farklı bir yöntemle yeni filtre elde edildi ve retinal görüntüde diğer filtreler ile görsel karşılaştırması yapıldı. Uygulama sonucunda filtrelerin birbirine yakın sonuçlar verdiği, hatta oluşturulan filtrenin bazı filtrelerden iyi sonuç verdiği görüldü.

Görüntünün özvektörleri aracılığıyla oluşturulan filtreyi diğer filtrelerden ayıran özellik ise diğer filtreler, filtreleme işlemi için sabit bir filtre matrisi kullanırken, yöntemimiz her görüntü için görüntüye özel filtre matrisi oluşturmaktadır. Bu durum görüntünün yorumlanması anlamında kolaylık sağlamaktadır.

Dinamik aralığı, yani görüntünün renk aralığı dar olan görüntülerde, oluşturduğumuz filtre, görüntünün içindeki özvektör matrisini kullandığı için diğer filtrelere göre daha iyi sonuç verir.

Filtreleme işleminde gürültüyü minimum seviyeye getirmek için görüntünün 3×3'lük alanlarının ortancası alındı. Bu durum görüntüdeki gerçek damar genişliğini bir kademe arttırdı. Bu alanda yapılacak bundan sonraki çalışmaların konusu, gerçek damar genişliğine, yine gürültü seviyesini minimumda tutarak nasıl ulaşılabileceği olabilir.

Şekil 7.1'deki korneal görüntü ele alınırsa,



Şekil 7.1: Korneal görüntü

Bundan sonraki çalışmalarda bu yapılan yaklaşım veya diğer yaklaşımlarla korneadan damar özelliklerinin çıkarılmasını da çalışma yapmak için uygun bir alan olarak görüyoruz.

KAYNAKLAR DİZİNİ

Acevedo, E., Acevedo, A., Martinez, F., Chavez, A., Velasco, P.,K., 2014, Associative approach for edge detection, International Conference on Systems, Man, and Cybernetics, San Diego, USA, p. 152-157.

Anonim, 2000, United Nations the World Population Prospects, http://www.un.org/esa/population/publications/wpp2000/highlights.pdf, erişim tarihi: 26.05.2017

Anonim, 2006, Image Processing, http://cpsc.ualr.edu/milanova/image_processing/week1/introduction_07_06.pdf, erişim tarihi: 26.05.2017.

Anonim, 2010, Computers and Informatics, http://bu.edu.eg/portal/uploads/computers%20and%20informatics/computer%20science/838/crs_6520/Files/chapter%201_introduction.pdf, erişim tarihi: 26.05.2017.

Barraga, E.S., Murray, V., Agurto, C., Pattichis, M.S., Russell, S., vd., 2009, Multi-scale AM.Fm for lesion phenotyping on age-related macular degeneration, 22nd IEEE International Symposium on Computer-Based Medical Systems, CBMS, Albuquerque, New Mexico, USA, p. 1-5.

Bensbeh, Z., Cohen, L.D., Mimoun, G., Coscas, G., 2001, A new approach of geodesic reconstruction for drusen segmentetion in eye fundus images, IEEE Trans. Med. Imaging, p. 1321-1333.

Bhuiyan, A., Xiao, D., Yogesan, K., 2014, A review of disease grading and remote diagnosis for sight threatening eye condition: age related macular degeneration, J. Compute. Sci. Sys. Bid, p. 62-71.

Blackledge, J.M., 2005, Digital Image Processing Mathematical and Computational Methods, Horwood Publishing, p. 473-479.

Brandon, L., Hoover, A., 2003, Drusen detection in a retinal image using multi-level analysis, Medical Image Computing and Computer-Assisted Intervention MICCAI 2003, Montreal, Canada, p. 618-625.

KAYNAKLAR DİZİNİ (devam)

Burger, W., Burge, M.J., 2007, Digital Image Processing, Springer, p. 507

Burlina, P., Freund, D., Dupos, B., Bressler, N., 2011, Automatic screening of age-related macular degeneration and retinal abnormalities, Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Boston, Massachussetts, USA, p. 3962-3966.

Chapdar, A., Chakravarthy, U., Verma, D., 2003, Age related macular degeneration, Br. Med, p. 485

Qidwai, U., Chen, C.H., 2009, Digital Image Processing, CRC, p. 2-6.

Cheng, J., Wong, D.W.K., Cheng, X., Liu, J., Tan, N.M., vd., 2012, Early age-related macular degeneration detection by focal biologically inspired feature, 19th IEEE International Conference on Image Processing (ICIP), Lake Bueno Vista, Florida, USA, p. 2805-2808.

De Jong, P.T., Age-related macular degeneration, 2006, N. Engl J. Med, p. 1474-1485.

Evans, J.R., 2001, Risk factors for age-related macular degeneration, Prog. Retn. EyeRes. p. 227-253

Freund, D.E., Bressler, N., Burlina, P., 2009, Automated detection of drusen in the macula, IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Boston Massachusetts, USA, p.61-64.

Gonzalez, R.C., Woods, R.E., Eddins, S.L., 2009, Digital Image Processing, Using Matlab, Prentice Hall, p. 328-349.

Hijazi, M.H.A., Coenen, F., Zheng, Y., 2012, Data mining techniques for the screening of age-related macular degeneration, Knowl. Based Syst., p. 83-92.

Hijazi, M.H.A., Coenen, F., Zheng, Y., 2015, Data mining for AMD screening: a classification based approach, Int. J. Simul. Syst. Sci. Technol., p. 57-68.

Köse, C., Şevik, U., Gençalioğlu, O., 2008, Automatic segmentation of age-related macular degeneration in retinal fundus images, Comput. Biol. Med., p. 611-619.

KAYNAKLAR DİZİNİ (devam)

Köse, C., Şevik, U., Gençalioğlu, O., İkibaş, C., Kayikiçioğlu, T., 2010, A statistical segmentation method for measuring age-related macular degeneration in retinal fundus images, J. Med. Syst., p. 1-13.

Liang, Z., Wong, D.W., Liu, J., Chan, K.L., Wong, T.Y., 2010, Towards automatic detection of age-related macular degeneration in retinal fundus images, Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Buenos Aires, Argentina, p. 4100-4103.

McAndrew, A., 2004, An Introduction to Digital Image Processing with Matlab, Notes for SCM2511 Image Processing, p. 11-24.

Mora, A.D., Vieira, P.M., Manivannon, A., Fonseca, J.M., 2011, Automated drusen detection in retinal images using analytical modelling algorithms, Biomed. Eng. Online, p. 59.

Namee, B.M., 2012, Utah, http://www.utah.edu/~gerig/cs6640_F2012/Materials/ImageProcessing1_Introduction_Bryan_Mac_Namee.pdf, erişim tarihi: 26.05.2017.

Niemeijer, M., Van Ginneken, B., Russell, S.R., Suttorp-Schulten, M.S., Abramoff, M.D., 2007, Automated detection and diffrentiation of drusen, exudates and cotton-wool spots in digital color fundus photographs for diabetic retinopathy diagnosis, Investig. Opthalmol. Vis. Sci., p. 2260-2267.

Rapantzikar, K., Zervakis, M., Balas, K., 2007, Detection and segmentation of drusen, exudates and cotton-wool spots in digital color fundus photographs for diabetic retinopathy diagnosis, Investig. Opthalmol. Vis. Sci., p. 2260-2267.

Rosendahl, C., Cameron, A., Tschandi, P., Bulinska, A., Gourhant, J-Y, vd., 2016, A Dermatoscopic Decision Algorithm for Pigmented Skin Malignancy, Gpcme, p. 3-7.

Sahbi, H., Etyngier, P., Audibert, J.Y., Kerivan, R., 2007, Graph Laplacian For Interactive Image Retrieval, Research Report, p. 7-32.

KAYNAKLAR DİZİNİ (devam)

Soliz, P., Russell, S., Abramoff, M., Murillo, S., Pattichis, M., vd., 2009, Independent component analysis for vision-inspired classification of retinal images with age-related macular degeneration, IEEE Southwest Symposium on Image Analysis and Interpretation, SSIAI 2008, Santa Fe, New Mexico, USA, p. 1-5.

Soloman, C., Breckon, T., 2008, Fundamentals of Digital Image Processing, Wiley-Blackwell, p. 49-62.

Quellec, G., Russell, S.R., Abramoff, M.D., 2011, Optimal filter framework for automated, instantaneous detection of lesions in retinal images, IEEE Trans. Med. Imaging, p. 523-533.

Qureshi, S., 2005, Embedded Image Processing, Springer, p. 211-219.

Wong, W.L., Su, X., Li, X., Cheung, C.M.G., Klein, R., vd., 2014, Global prevalence of age-related macular degeneration and disease burden projection for 2020 and 2040: a systematic review and meta-analysis, Lancet Glob. Health2, p. 106-116.

Yavuz, H.S., Çevikalp, H., Edizkan, R., 2013, A comprehensive comparison of features and embedding methods for face recognition, Turkish Journal of Electrical Engineering & Computer Sciences, p. 5.

Yılmaz, İ., Güllü, M., Baybura, T., Erdoğan, A.O., 2010, Renk Uzayları ve Renk Dönüşüm Programı (RDP), Afyon Kocatepe Üniversitesi Fen Bilimleri Dergisi, 2, 2, 19-35

Zhou, H., Wu, J., Zhang, J., 2010, Digital Image Processing, Ventus Publishing ApS, p. 53-58.