YALOVA ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SEGMENTASYON YARDIMIYLA KENAR İYİLEŞTİRME YÖNTEMİ

YÜKSEK LİSANS TEZİ

Özlem MUTLU

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

EYLÜL 2017

YALOVA ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SEGMENTASYON YARDIMIYLA KENAR İYİLEŞTİRME YÖNTEMİ

YÜKSEK LİSANS TEZİ

Özlem MUTLU
(145105008)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Yrd. Doç. Dr. Osman Hilmi KOÇAL

EYLÜL 2017

YALOVA Üniversitesi Fen Bilimleri Enstitüsü'nün 145105008 numaralı Yüksek Lisans Öğrencisi **Özlem MUTLU**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"SEGMENTASYON YARDIMIYLA KENAR İYİLEŞTİRME YÖNTEMİ"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Yrd.Doç.Dr.Osman Hilmi KOÇAL**
Yalova Üniversitesi

Jüri Üyeleri : **Yrd. Doç. Dr.Osman Hilmi KOÇAL** .....
Yalova Üniversitesi

Doç. Dr. Müfit ÇETİN .....
Yalova Üniversitesi

Prof. Dr.Tuncay ERTAŞ .....
Uludağ Üniversitesi

Teslim Tarihi : **28 Temmuz 2017**
Savunma Tarihi : **12 Eylül 2017**

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışmasında, görüntü işleme alanında büyük önem teşkil eden nesnelerin kenarlarının bulunması ve iyileştirilmesi konusu araştırılmıştır. Günümüzde popüler olan kenar algoritmalarının eksiklikleri dikkate alınıp bu doğrultuda yeni bir algoritma sunulmuştur. Gradyan bilgisi kenar bulmanın temelini oluşturmasına rağmen algoritmamızda bu temel bilgiyi kullanmadan farklı parametreler kullanılarak benzer sonuçlar elde edilmiştir.

Tez çalışmamın ilerlemesinde bilgi birikimi ve tecrübelerini benden esirgemeyen sayın danışmanım Yrd. Doç. Dr. Osman Hilmi KOÇAL'a teşekkürlerimi sunarım.

Tez çalışmamın her aşamasında, sabırla yardımlarını esirgemeyen değerli arkadaşım Hüsniye Kevser BAYRAKTAR'a teşekkür ederim.

Eğitim hayatım boyunca her türlü maddi ve manevi desteklerini hiçbir zaman esirgemeyip bugünlere gelmemde emeği ve katkısı bulunan, dualarını hiç bir zaman eksik etmeyen kıymetli annem Kişmiş, babam Fahrettin, kardeşlerim ve HERKESE en içten teşekkürlerimi sunarım.

Eylül 2017

Özlem MUTLU
Bilgisayar Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

RGB	**:** Red Green Blue
JPG	**:** Joint Picture Experts Groups
LoG	**:** Laplacian of Gaussian
2B	**:** İki Boyutlu
MATLAB	**:** Matrix Labratory
D	**:** Doğu
GD	**:** Güney Doğu
GB	**:** Güney Batı
K	**:** Kuzey
KD	**:** Kuzey Doğu
KB	**:** Kuzey Batı
SKG	**:** Segmentasyonla Kenar Güçlendirme
NR	**:** Number of similar pixels on Region
SF	**:** Sol Fark
SAF	**:** Sağ Fark
BPS	**:** Benzer Piksel Sayısı
FPS	**:** Farklı Piksel Sayısı

ŞEKİL LİSTESİ

Sayfa

SEGMENTASYON YARDIMIYLA KENAR İYİLEŞTİRME YÖNTEMİ

ÖZET

Görüntü işleme, bilgisayarlı görme, örüntü tanıma uygulamalarının başarısı çoğunlukla nesne kenarlarının bulunmasına dayanır. Bu konu üzerinde çok fazla çalışılmasına rağmen, yansıma, gürültü, eşik değeri gibi etmenler doğruluğu etkilediği için günümüzde hala kenar bulmada güçlü bir algoritma bulunmamaktadır. Yapılan çalışmada, bu algoritmaların detaylı incelenmesi sonucu kenar izlerken başarısız oldukları yerler göz önünde bulundurularak kenarların bulunması ve iyileştirmesine yönelik yeni bir yöntem olan segmentasyonla kenar güçlendirme (SKG) önerilmiştir.

Bu yöntemle nesne kenarlarını bulma işlemi farklı bir açıdan ele alınmıştır. Yöntemde diğer kenar bulma algoritmalarının uyguladığı piksel yoğunluklarındaki ani değişimlere bakılarak bölgeleri ayıran kenarları bulma yerine, bölgelerden kenarların elde edilip edilmeyeceğine cevap aranmıştır. MATLAB yazılımı kullanılarak yapılan çalışmada öncelikle Sobel, Prewitt, Roberts, Canny gibi bilinen yöntemlerle elde edilen kenarların çift tarafları kullanılarak kenarların kayıp ipuçlarını keşfetmek için gerekli tüm bilgiye sahip olan iki referans bölge (RB) oluşturulur. Ardından referans kenarın yönü dikkate alınarak devam ettirilecek kenarın tahmini yönleri belirlenir ve bu yönlerin doğrultusunda yeni bölgeler oluşturulur. Sonraki adımda, istatistiksel yöntemler kullanılarak oluşturulan bölgeler ile RB kıyaslanıp kenar ihtimali en yüksek olan yön ve bölge belirlenir. En son adımda, elde edilen bölgenin RB'ye benzeyen pikselleri kullanılarak optimizasyon teknikleri yardımıyla bölgede eğimli kenar çizilmiştir.

Sunulan kenar iyileştirme algoritması, Berkeley Segmentation Dataset ve The Hypermedia Image Processing Reference (HIPR) veritabanından alınan farklı kontrast değerlerine sahip farklı görüntüler ve anjiyografi veri seti üzerinde uygulanmış ve popüler olan kenar bulma yöntemlerinin buldukları sonuçlar ile karşılaştırma yapılmıştır. Elde edilen sonuçlar geliştirilen algoritmanın, diğer yöntemlerin buldukları kenarları bulabildiğini ve eksik buldukları kenarları tamamlayabildiğini göstermiştir.

EDGE REINFORCEMENT METHOD BY USING SEGMENTATION

SUMMARY

The success of image processing, computer vision, pattern recognition applications mostly depends on the detection of object edges. Although there is a lot of research on this subject, there are still no strong algorithms for edge detection, because factors such as reflection, noise, threshold value affect the accuracy. In this study, as a result of examining these algorithms in detail where they have failed at the trailing edge, a new method, edge reinforcement by using edge segmentation (SKG), is proposed for detecting and enhancing the edges.

By this method, detecting object edges process discussed from a different viewpoint. In the method, instead of detecting the edges separating the regions by viewing sudden changes in the pixel densities, applied by the other edge detection algorithms, an answer was sought to determine whether the edges can obtained from the regions. In this study developed by using MATLAB software, two reference regions (RB) which have all the information necessary for exploring the missing tips of the edges are formed by using the sides of the edges that first obtained by known methods like Sobel, Prewitt, Roberts, Canny. Then, the estimated directions of the edge to be continued are determined by considering the direction of the reference edge and new regions are formed in the orientation of these directions. In the next step, with statistical methods, by comparing created regions and the RB, the direction and the region with the greatest edge possibility are determined. In the last step, by using similar to RB pixels of the obtained area, curved edge of the region were drawn.

The presented edge reinforcement algorithm was applied to different images with different contrast values and the angiography data set from the Berkeley Segmentation Dataset and the Hypermedia Image Processing Reference (HIPR) database and compared with the results of popular edge detection methods. The results show that the developed algorithm can detect the edges that other methods have found and complete the missing edges.

1. GİRİŞ

Günümüzde, teknolojinin gelişmesine bağlı olarak, tıp, askeriye, güvenlik, adli bilişim, uydu görüntüleri, cep telefonları, görüntü iyileştirme, biyometrik tanıma, robotik uygulamalar gibi birbirinden farklı alanlarda görüntü analizi yapılmaktadır.

Kenarlar nesnelerin dış görünüşü hakkında bilgi verir. Bu yapısal özelliklerinden dolayı birçok görüntü işleme algoritmalarının da temelini oluşturur ve büyük önem teşkil eder. Kenarlar görüntülerdeki birçok verinin içinde nesneyi en iyi tanımlayan özellik olduğundan dolayı sınıflandırma, örüntü tanıma, nesne tanıma ve sıkıştırma algoritmalarında öznitelik olarak kullanılır. Ani değişimin olduğu piksellerin konumlarının hesaplanması sonucu elde edilen kenarlar görüntüleri bölgelere ayırdığı için bölütleme algoritmalarının da temelini oluşturur.

Görüntü işleme uygulamalarının başarısı çoğunlukla kenarların bulunmasına dayandığı için bu konu üzerinde çok fazla çalışılmıştır. Buna rağmen, yansıma, gürültü, eşik değer gibi önsel bilgiler doğruluğu etkilediği için kenar bulma işlemi hala bir problem olarak karşımıza çıkmaktadır [1-4].

Görüntü işleme, gerçek görme sistemini elektronik ortama aktarılması ile ilgilenir. İnsanların görme yeteneği kenarların algılamasıyla orantılıdır. Genel olarak kenarın sayısal ortamdaki tanımı piksel yoğunluklarının değiştiği yeri ifade eder [5].

Kitchen ve Rosenfeld ise kenarı, komşusu ile farklılık gösteren iki homojen bölge arasında sınır olarak ifade eder [6]. Böylece kenar, görüntü içerisinde anlamlı bölgeler oluşturmuş olur. Bu tez çalışmasında kenar tanımına farklı bir açıdan bakılarak, anlamlı bölgelerdeki bilgiyi kullanıp insan görme sistemine benzer şekilde kenarları tespit eden ve iyileştiren bir yöntem sunulmuştur.

Geliştirilen, segmentasyon yapılarak kenar güçlendirme yöntemi, doğru kenarları takip etmek için yararlanılan yeni bir tekniktir. Ayrıca yöntemde, segmentasyon ve kenar tespiti, en zor problemlerden olan gürültünün üstesinden gelmek için birleştirilmiştir. Ünlü kenar bulma algoritmaları yoğunluk farkından faydalanarak buldukları eğimleri gürültüden dolayı bazı bölgelerde bulamadıkları için, kenar

takibinde başarısız sonuçlar ortaya çıkarmaktadırlar. Bu sorunun çözümü için yeni ve mantıksal bir metot geliştirilmiştir.

1.1 Kenar Bulma Algoritmalarının Karşılaştığı Problemler

Yoğunlukları değişen piksel konumlarının bulunması için, görüntülerin türevinin alınmasına yardımcı olan filtreler kullanılır [7]. Filtrelerin görüntülere uygulanmasının temel nedeni ise işlem kolaylığı sağlamaktır. Fakat filtrenin boyutu ve katsayıları çeşitli problemlere neden olmaktadır.

Filtreyi oluşturan parametreler, komşu pikseller arasındaki yoğunluk farklarını hesaplayarak kenarları elde ettiğinden dolayı kenar arama işleminin daha küçük bir alanda yapılmasına neden olur. Daha az pikselin hesaba katılması o konumda oluşan hatadan (gürültü, aydınlama vb.) dolayı gerçekte var olan kenarları algılamayıp kopukluklar oluşturacaktır.

Filtre boyutu büyük seçildiği zaman nesnelerin birbirine çok yakın olduğu görüntülerde, filtreye karşılık gelecek alana birden fazla kenar girmiş olur. Bu durumda kenarların konumunun elde edilmesinde başarısız sonuçlar elde edilir. Filtre boyutunun küçük alınması, dar bir alanda çalışılmasına neden olduğu için gürültülerde kenar olarak kabul edilir. Fakat ideal olan filtre her türlü ortamda kenarları tespit edebilmelidir.

Kenar bulmanın başarısını düşüren en önemli problemlerden biri gürültülerdir. Bunlar, aydınlanma, bulanıklaşma ve veri kaybına sebep olduğundan dolayı istenmeyen piksellerdir.

Gürültü yüksek frekans bileşenleri içerir. Türev tabanlı kenar bulma algoritmaları da bu bileşenleri daha baskın duruma getirir. Önerilen çözüm görüntülere gauss filtresi uygulanmasıdır. Fakat bu optimum bir çözüm değildir, yapılan işlem standart sapmaya bağlıdır. Standart sapmanın küçük olması, ince detaylara kadar, gürültü sayılacak pikselleri bulur. Büyük olması durumunda ise, gürültüyü yok etmesinin yanında gerekli ayrıntıları yok edip kenar adayı olacak piksellerin kaybolmasına neden olacaktır [7-10].

Kenar bulmada karşılaşılan bir diğer problem ise kullanıcıya bağlı olan eşik değeridir. Türev tabanlı algoritmalarda tüm piksellerin gradyan değerleri hesaplanır. Kenar adayını temsil eden en iyi gradyan değerini seçmek için eşik değeri kullanılır.

Bu değer kenarların bulunması ile doğru orantılıdır. Büyük alınırsa çok fazla kenar küçük alınırsa daha az kenar bulunmuş olur.

Görüntü üzerinde bozulmalar, veri kaybı, bulanıklaşma, ışıklanmanın etkisi, geometrik bozulmalar kenar bulmanın kalitesini düşürdüğü için tüm resimlere uygulanabilecek bir kenar bulma yöntemi yoktur. Aynı zamanda, mevcut kenar bulma teknikleri her resimde aynı başarıyı sağlayamamaktadır.

Bu problemler özetlenirse, farklı basamaklarda kullanılan parametrelerin standart olmaması ve görüntülerdeki yapısal farklılar görüntülerdeki bütün kenarların algılanmasını zorlaştıracak ve kenar bulmada kullanılan yöntemlerin başarısını düşürecektir [11].

1.2 Literatür Araştırması

Görüntü işleme ve bilgisayarlı görme alanındaki bir çok çalışmada ön işlem olarak görüntülerdeki kenarlar kullanılmıştır. Kenar tespitinin doğruluğu ise uygulamaların sonucunu direkt etkilemektedir. Bu sebeple görüntülerdeki kenarların doğru şekilde bulunması fazlasıyla önem arz etmektedir. Literatürde var olan kenar bulma algoritmalarının eksikliklerini gidermeye çalışan veya farklı teknikler kullanarak kenarların algılanması ile ilgili yapılmış birçok çalışma bulunmaktadır.

Geleneksel kenar bulma algoritmalarının en iyisi olan Canny algoritmasının iyileştirilmesine yönelik birçok çalışma yapılmıştır [12-15]. Canny algoritmasındaki Gauss filtresinin uyarlanabilir olmadığı için varyans değerini seçiminin kullanıcı tarafından yapılması gerekmektedir [11]. Deng ve Wang Canny algoritmasındaki bu eksikliği gidermek için Gauss filtresi yerine morfolojik filtreleme kullanımı önermişlerdir. Deneysel sonuçlar geliştirilen yöntemin tuz ve biber gürültüsünde başarılı olduğunu göstermiştir. Weibin Rong ve arkadaşları gürültüye karşı daha gürbüz ve Canny'nin gürültüden etkilendiği için tespit edemediği zayıf kenarları da bulmayı hedefleyen bir yöntem sunmuşlardır [16].

Mehena ve Adhikaary [1] yaptıkları çalışmada yeni bir kenar bulma yöntemi önermişlerdir. Kenarların doğası gereği bilinen kenar bulma yöntemlerinin eşik değeri veya sabit başka değerler ile işlem yapmasını dikkate alarak bulanık mantık ile esnek bir model geliştirmişlerdir. Elde edilen sonuçlar Sobel, Canny ve

morfolojik yaklaşımlarla kıyaslanmış ve gürültülere duyarlılığın az olduğu başarılı sonuçlar elde edilmiştir.

Utkarsha ve Deokar, kenarın elde edilmesine dair problemlere yapay sinir ağları kullanarak çözüm getirmiştir. Prewitt, Sobel ve Canny ile karşılaştığında tatmin edici sonuçlar elde etmiştir [17].

J. Vasavada and S. Tiwari ileri beslemeli yapay sinir ağı ile kenar bulmada yeni bir yöntem sunmuşlardır. İleri beslemeli öğrenme algoritması ile hatanın minimizasyonu sağlanmıştır. Sinir ağları ile yapılan çalışmalarda eğitim verisi için genellikle ikili resimler kullanılmaktadır. Sunulan bu yöntemde ise eğitim verisi için Sobel filtresi uygulanarak hesaplanan gradyan değerleri ve standart sapma kullanılmıştır. Sonuçlar birçok yöntemle karşılaştırılmış ve geliştirilen yöntemin daha başarılı olduğu gözlenmiştir [3].

Elif Deniz Yiğitbaşı yapay zeka tekniği olan yapay arı kolonisi temelli, kenar bulmada türev tabanlı algoritmalar kullanmadan optimizasyon temelli bir yöntem geliştirmiştir [18].

Türev tabanlı kenar bulma algoritmaları genelde dar bir alandan gradyan hesabı yapmaktadır. Hazer geliştirdiği yöntemde ise bulanık topoloji ve bağlantı haritalarını kullanarak daha geniş bir pencere içerisinde hesaplama yapmıştır [19].

Topoloji kullanarak kenar algılayan diğer bir çalışma ise Ayber'in çalışmasıdır. Bu çalışmasında topolojik özellikleri kullanan iki yöntem geliştirmiştir. İlk adım olarak orta piksellerin diğer piksellere nasıl bağlandığını gösteren bağlantı haritasını hesaplar. İlk yöntemde öz uyarlamalı yöntemle kenar bulmuş, ikinci yöntemde ise elde ettiği bağlantı haritasına Sobel kenar bulma algoritmasını uygulamıştır. Geliştirilen yöntemle daha ince kenarlar elde etmiştir [20].

Roushdy çalışmasında kenar bulma yöntemlerinin performanslarını karşılaştırarak elde ettiği sonuçlara göre, gürültülü resimlerde Canny yöntemi Prewitt ve Sobel'den daha başarılı sonuçlar elde etmiştir. Ayrıca, ön işlem olarak morfolojik filtrelerin kullanılmasıyla görüntülerde istenmeyen gürültüleri gidererek daha iyi sonuçlar elde edilmiştir [13].

Medikal görüntülerin gürültü oranının yoğun olduğu göz önüne alındığında bahsedilen kenar bulma yöntemlerinin başarı oranı düşmektedir. Bu sebeple tıbbi görüntüler için yeni kenar bulma yöntemi arayışları devam etmektedir [6].

1.3 Tezin Amacı ve Organizasyonu

Bu çalışmada türev tabanlı kenar bulma algoritmaları gibi filtreye gerek duymayan segmentasyon tabanlı kenar bulma yöntemi geliştirildi. Algoritma, farklı özelliklere sahip resimlerin kenarlarında iyileştirme, gerekli kenarlarda ekleme, silme ve kopuk parçaları tamamlamayı hedeflemektedir.

Kenarın gürültülü yerlerde gerçek konumundan sapmasından dolayı kenar sürekli bir yönde takip edilememektedir. Görüntülerde hangi adımda hangi tip gürültü olduğunun tespiti zor bir problemdir ve görüntülerdeki tüm kenarların bulunmasını olumsuz etkilemektedir. Bu yüzden kenarlar üzerinde bölgesel aramalar yapılarak bu probleme çözüm aranmıştır.

Daha geniş bir alan referans alınarak kenarların incelenmesi için segment edilmiş bölgelerde çalışılmıştır. Sonucunda gradyanların azaldığı veya ufak gürültülerden dolayı bulunamayan kenarlar tespit edilmeye çalışılmıştır.

Geliştirilen algoritmanın işlem yaptığı durumlar;

- Çeşitli problemlerden dolayı kenar noktalarında oluşan kopuklukları tamamlayarak gerçek kenar noktalarında süreklilik sağlar.
- Kenar bulma algoritmaları etrafta daha iyi bir kenar bulunurken kenar ihtimali düşük olan tarafa doğru sapılarak yanlış kenar bulunduğu durumlarda doğru tarafta kenar bulunmasını sağlar.
- Diğer yöntemlerin kenarı sonlandırdığı yerlerde devam etmesi gereken kenarları ekler.

Ufak gürültülerden etkilenip kenar olarak davranıldığı noktalarda bölgesel inceleme yapıldığı için, daha fazla bilgi kullanarak gürültü pikseller hesaba katılmadan daha doğru sonuçlar elde edilir.

Algoritmanın uygulamasında herhangi bir gürültü giderici algoritma kullanılmamıştır. Böylece, Gauss ve benzeri filtreler gibi kenarlar üzerinde bilgi kaybına neden olmadığı için daha doğru kenarlar üretilir. Ayrıca kullanıcıya bağlı parametrelere ihtiyacı en aza indirilmiştir.

Kenar bulma algoritmalarının başarı oranı artırılmış, aynı zamanda Sobel, Prewitt, Canny yöntemlerinin bulduğu ve bulamadığı kenarlar elde edilmiştir.

Makale organizasyonu şu şekilde düzenlenmiştir. Tezde anlatılan konu kapsamında temel kavramları anlatmak için, popüler olan kenar bulma teknikleri ayrıntıları ile ikinci bölümde anlatılmıştır. Üçüncü bölümde, geliştirilen algoritmanın detaylı anlatımı yapılmış, ön işlemler ve temel işlemler olarak iki temel alt başlık şeklinde anlatılmıştır. Deneysel çalışmalar bölümünde birçok farklı resimde yeni teknikle üretilen kenarların sonuçları gösterilmiştir. Sonuç kısmında genel olarak yöntem değerlendirilmiştir.

2. KENAR BULMA TEKNİKLERİ

Resimlerde nesnelerin sınırlarını algılayabilmek için renk geçişlerini bulmak kenar bulma algoritmaların temel amacıdır. Temelde iki tip kenar belirleme yöntemi vardır. İlki, birinci dereceden türev tabanlı olan Sobel, Roberts, Prewitt vb. yöntemler iken diğeri ikinci dereceden türev tabanlı olan LOG ve Canny gibi yöntemlerdir.

Görüntülerdeki kenarları algılamak için yaygın olarak kullanılan kenar bulma operatörleri ise Prewitt [21], Sobel [22], Canny [23], Marr Hilldreth [5] kenar bulmak için geliştirilmiş bilinen en iyi operatörlerdir. Bunlar birçok çalışmada kullanılmış ve iyileştirmeler yapılmıştır.

2.1 Birinci Türeve Dayalı Teknikler

Türev değişim ile alakalı bir durumdur. Fonksiyonun herhangi bir parametresi değişirken fonksiyonun nasıl değiştiğini ifade eder. Türevin görüntü işlemedeki tanımı ise piksellerin değişim oranıdır. Görüntülerde renk yoğunluk farkından faydalanılarak kenarların tespiti sağlanır.

2.1.1 Yönlere bağlı kısmi türev

f(x,y), x ve y koordinatına denk gelen bir pikselin sayısal değerini ifade eder.

x ve y değişkenlerine sahip sürekli bir ***f***(x,y) fonksiyonu olursa;

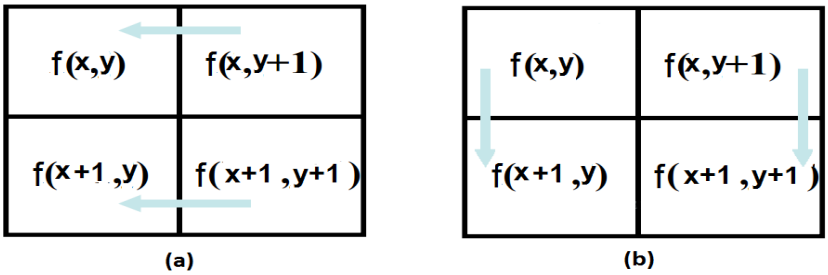
$$\frac{\partial f}{\partial y} = \lim_{h \to 0} \frac{f(x+h) - f(x,y)}{h} \qquad \textbf{(2.1)}$$

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(y+h) - f(x,y)}{h} \qquad \textbf{(2.2)}$$

Yukarıdaki denklem 2.1 x'in sabit tutulup fonksiyonun y'ye göre değişimini, denklem 2.2 y'nin sabit tutulup fonksiyonun x'e göre türevini ifade eder. Resimler sürekli değildir, ayrık fonksiyonla ifade edilir. Bundan dolayı yukarıda kullanılan formül, h değeri 1 alınarak denklem 2.3 gibi olur.

$$\frac{\partial f}{\partial x} = f(x) - f(x-1) \qquad \frac{\partial f}{\partial y} = f(y) - f(y-1) \tag{2.3}$$

Yukarıda yazılan eşitlikler iki boyutlu görüntüler üzerinde uygulanırsa görüntünün x yönünde birinci türevi (Şekil 2.1);



Şekil 2.1 : (a)Yatayda türev, (b)Dikeyde türev.

$$\frac{\partial f}{\partial x}(x,y) = (f(x,y+1) - f(x,y)) + (f(x+1,y+1) - f(x+1,y)) \tag{2.4}$$

$$\frac{\partial f}{\partial y}(x,y) = (f(x+1,y) - f(x,y)) + (f(x+1,y+1) - f(x,y+1)) \tag{2.5}$$

Denklem 2.4 ve 2.5, Şekil 2.1'in matematiksel olarak modellenmesidir. Bu çıkarma işleminin, görüntülerin her pikselinde bu şekilde yapılması maliyetli bir iştir. Bu nedenle yukarıdaki sonucun aynısını verecek 2.6 ve 2.7'deki denklemler ile bir matris tanımlanıp sinyal işlemede çok kullanılan konvolüsyon işleminin yapılması yeterli olacaktır.

$$\frac{\partial}{\partial x} = g = \begin{bmatrix} 1 & -1 \end{bmatrix} \tag{2.6}$$

$$\frac{\partial}{\partial y} = g = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{2.7}$$

Konvolüsyon, girdi görüntüsü üzerinde filtre gezdirdikten sonra oluşan yeni piksel yoğunluk değerinin amaca uygun hesaplanıp resme yerleştirilme işlemidir. Uygulamalarda farklı amaçlar için kullanılan çeşitli filtreler bulunmaktadır. Bunlar; Medyan, Gauss, Mean, Sobel ve Prewitt'tir. Kenar bulma yöntemlerinde öncelikle gürültü giderici filtreler kullanılır.

$$f \otimes g = \sum_{x}^{M} \sum_{y}^{M} f_{x,y} g_{x,y} \tag{2.8}$$

f : giriş fonksiyonu

g : filtre

$g_{x,y}$: x. satır, y. sütundaki filtrenin değeri

$f_{x,y}$: orjinal görüntünün, filtrenin x. satır, y. sütununa karşılık gelen değeri

M : g filtresinin boyutu

2.1.2 Gradyan

Herhangi bir noktanın gradyanı, fonksiyonun o noktadaki en hızlı artışı için hareket edilmesi gereken yönü gösterir. Gradyan vektörünün uzunluğu artışın miktarını ifade eder. Görüntülerde gradyan ise renk yani yoğunluk değişiminin hangi yöne doğru ne kadar olduğunu belirtir. İki değişkenli bir fonksiyonun türevi, kısmı türev ile alınır. Aşağıdaki denklem 2.9 ile iki boyutlu bir görüntü fonksiyonun gradyanı hesaplanır.

f(x,y) fonksiyonun gradyanı;

$$\nabla f = \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} j \tag{2.9}$$

i: x yönüne doğru birim vektördür

j: y yönüne doğru birim vektördür

$$\nabla f \equiv grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.10}$$

$$mag(\nabla f) = \sqrt{g_x^2 + g_y^2} \tag{2.11}$$

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_x}{g_y}\right] \tag{2.12}$$

f(x,y) türevi, kenarların yönü ve gradyanın büyüklüğü hakkında bilgi verir. Görüntülerin yatay ve dikey yöndeki kısmi türevlerinin toplanması ile gradyan genlik değerleri bulunur. Türev değerinin büyüklüğü bölgeler arasında büyük bir farklılığın olduğunu bildirir [24].

2.1.3 Roberts algoritması

Roberts filtresi, görüntüler üzerinde basit ve hızlı bir şekilde gradyan değerlerini hesaplaması sonucu, kenar bilgisini ifade eden yüksek frekans bölgelerini belirtir. Roberts operatörleri aşağıdaki şekillerde yatay (GX) ve dikey (GY) verilmiştir.

Şekil 2.2 : Roberts operatörleri.

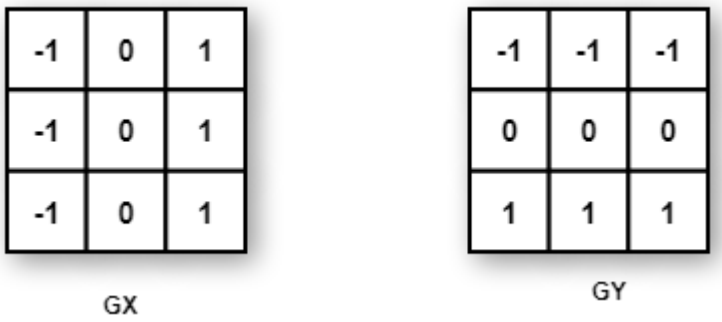
Oluşturulan iki Roberts operatörü eğimi 45 derece olan kenarlara daha fazla ağırlık verir. Bu filtreler görüntüye ayrı ayrı uygulanır. Daha sonra her bir pikselin toplam gradyanın büyüklüğünü bulmak için denklem 2.11'deki gibi hesaplanır.

GX filtresinin görüntüye uygulanması, piksellerin siyahtan beyaza doğru yani kontrastın büyük olduğu yöne ilerlendiğini ifade eder. GY operatörü ise 90 derece yatay yönde aynı işlemi yapar. Bu algoritmanın kullanılma nedeni, hesaplamada sadece toplama ve çıkarma işlemi yaptığı için hesaplamayı çok hızlı yapmasıdır. Filtrenin küçük olması, gürültüye karşı aşırı hassas davranıp gürültülü noktaları algılamasına neden olacaktır.

Görüntülerdeki kenar noktalarının yönü ve gradyan büyüklüğü çok farklılık gösterdiğinden dolayı Roberts operatorü farklı frekans ve yönleri bulunan görüntülerde farklı sonuçlar üretir [25].

2.1.4 Prewitt algoritması

Prewitt kenar bulma algoritması diğer Sobel, Roberts gibi gradyan yardımıyla kenarın yönünü ve büyüklüğünü bulmaya çalışan algoritmalara alternatif bir yöntemdir. Bu operatör Sobel ile aynı denklemleri kullanır. Fakat filtrenin merkezine yakın komşulara farklı ağırlık vermez.



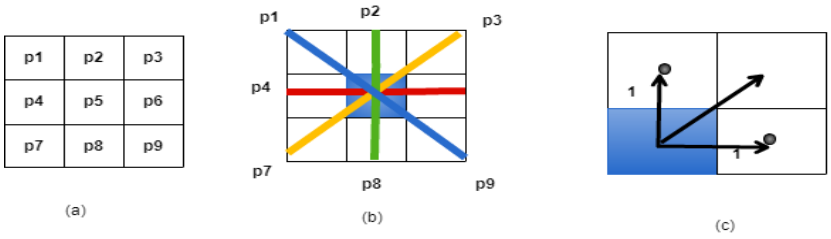
Şekil 2.3 : Prewitt operatörleri.

Şekil 2.3'te görünen GX filtresi sıfırın bulunduğu satır dikey olduğu için görüntü üzerinde dikey kenar bulur. Aynı şekilde yatayda bulunan sıfır satırı yatay olarak kenarları algılar [20].

2.1.5 Sobel algoritması

Yaygın olarak kullanılan Sobel kenar bulma algoritması ilk basamakta gürültüleri gidermek için yumuşatma filtresi kullanır. Başka bir şekilde ifade edilirse resme alçak geçiren filtre uygular. Bu işlemin yapılması Sobel algoritmasının bir sonraki basamaklarda daha iyi kenarlar elde etmesini sağlamıştır. İkinci adımda görüntülere maskenin uygulanması ile belirgin hale getirilen kenarlar üçüncü adımda ise daha önce hesaplanan bir eşik değeri ile kıyaslanarak eşik değerinden büyük olanlar objenin kenarı olarak seçilir. Eğer büyük değilse seçilme işlemi yapılmaz. En son basamaktaki temel problem, eşik değeri belirleme işlemi tüm görüntü üzerinden yapıldığı için resmin her bölgesinde objelerin kenarlarını belirlemede doğru sonuçlar verememesidir.

Şekil 2.4.a'da 3x3 komşulukta 9 pikselin yoğunluk değerleri gösterilmiştir. Şekil 2.5.b'de merkez noktasından komşularına olan türev yönünü belirleyen türev yönleri çizilmiştir.



Şekil 2.4 : (a) Pikselin yoğunluk değeri, (b) Piksellerin türev yönleri, (c) Birim vektör.

Birim vektör: $\frac{\vec{A}}{|\vec{A}|}$

$$\vec{A} = \sqrt{A^2{}_x + A^2{}_y} \quad \textbf{(2.13)}$$

Şekil 2.4.c vektörün, türev yönü [1,1] olan birim vektördür, bu birim vektörün çarpılması ile vektör elde edilir.

Yukarıda bölüm 2.1.2'de gradyan tanımında kullanılan denklem 2.10 ve 2.11 kullanılarak merkez pikselin komşularına olan vektörlerin türevleri toplamı ile aşağıdaki gradyan denklemi elde edilir [28]. Toplam gradyanın büyüklüğü şu şekilde hesaplanır;

$$G = \frac{(P_3 - P_7)}{R}\frac{[1,1]}{R} + \frac{(P_1 - P_9)}{R}\frac{[-1,1]}{R} + (P_2 - P_8).[0,1] + (P_6 - P_1).[1,0] \quad \textbf{(2.14)}$$

R=$\sqrt{2}$ alınır çünkü p3 ile p5 arasındaki Öklid uzaklığı $\sqrt{2}$ 'dir.

$$G = \frac{1}{2}((p_3 - p_7, p_3 - p_7) + (-p_1 + p_9, p_1 - p_9)) + ((0, p_2 - p_8) + (p_6 - p_1, 0)) \quad \textbf{(2.15)}$$

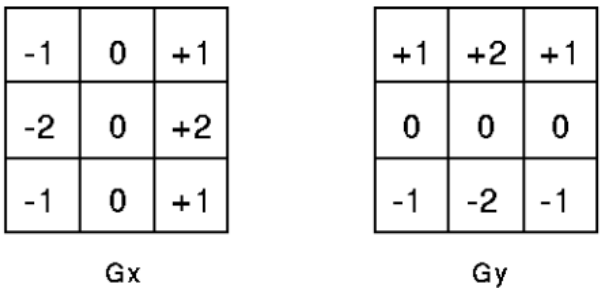
$$2G = (p_3 - p_7 - p_1 + p_9, p_3 - p_7 + p_1 - p_9) + 2(p_6 - p_1, p_2 - p_8) \quad \textbf{(2.16)}$$

$$2G = (p_3 - p_7 - p_1 + p_9 + 2(p_6 - p_1), p_3 - p_7 + p_1 - p_9 + 2(p_2 - p_8)) \quad \textbf{(2.17)}$$

Bu denklemlerin sonucunda aşağıdaki yatay ve dikey Sobel filtresi elde edilmiştir. Böylece tekrarlı bir şekilde bu uzun işlemler yapılmadan denklem yardımıyla çıkarılan yatay ve dikey Sobel filtreleri ile konvolüsyon işlemi sonucunda tüm görüntüdeki pikseller işleme alınıp elde edilen bütün gradyan değerleri kenar bilgisini tutan matrise yazılır.

$$G_x = p_3 - p_7 - p_1 + p_9 + 2(p_6 - p_1) \quad \textbf{(2.18)}$$

$$G_y = p_3 - p_7 + p_1 - p_9 + 2(p_2 - p_8) \quad \textbf{(2.19)}$$



Şekil 2.5 : Sobel operatörleri

$G = \sqrt{G_x^2 + G_y^2}$ işlemi uzun sürdüğü için genelde $|G| = |G_x| + |G_y|$ denklemi kullanılır.

Eğer G belirlenen T gibi bir eşik değerinden büyükse f(x,y) pikseli kenardır. Nesnedeki bölge aynı piksel yoğunluğunda ise hesaplanan bütün gradyanlar aynı değer sahip olur [26].

2.2 İkinci Türeve Dayalı Teknikler

Görüntüler üzerinde ikinci derece türev, ince kenarlar ve farklılık oluşturan noktaları tespit etmede, nesneleri netleştirme işlemlerinde birinci türeve göre daha etkili sonuç üretmektedir. Görüntüleri x ve y gibi iki farklı yönde ikinci türevinin alınması Laplacian olarak ifade edilir.

Birinci türevin en büyük olduğu yerde ikinci türev sıfır olur. Bu bilgi kullanılarak sıfıra geçiş noktaları kenarları temsil eder.

Bölüm 2.1.1 kısımında fonksiyonu birinci türevinin görüntüler üzerinde uygulama notasyonu verilmiştir. Bu bilgiler ışığında ikinci türev ise;

$$\frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x) \tag{2.20}$$

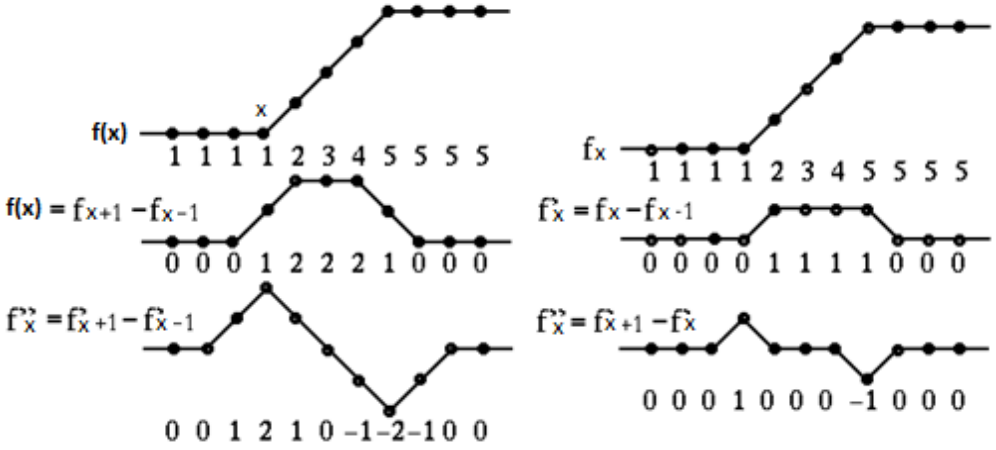
$$\frac{\partial f'(x)}{\partial x} = \frac{\partial^2 f}{\partial x^2} = f'(x+1) - f'(x) \tag{2.21}$$

$$(f(x+1)')' = f(x+1) - f(x) \tag{2.22}$$

$$(f(x)')' = f(x) - f(x-1) \tag{2.23}$$

$$\frac{\partial f'(x)}{\partial x} = \frac{\partial^2 f}{\partial x^2} = f(x+1) - f(x) - f(x) + f(x-1) \tag{2.24}$$

Birinci türev ile ikinci türev arasında nasıl bir ilişki olduğunu anlamak için Şekil 2.6'yı incelediğimizde birinci türev yokuş boyunca sıfır değildir. Fakat ikinci türev yokuşun başında ve sonunda sıfırdır. Bu durum birinci türevin daha kalın, ikinci türevin de daha ince kenarlar çıkardığını gösterir. Aynı gürültü noktasında, ikinci türevin elde ettiği genlik değeri birinci türeve göre daha büyük olması ani değişimlerin bulunmasında ikinci türevin daha hassas olduğunu ifade eder. İkinci türev gürültülü resimlerde birinci türeve göre daha keskin kenarlar bulur.



Şekil 2.6 : Birinci ve ikinci türevin incelenmesi.

İkinci türevin işareti, kenarın içine doğru ilerlediğinden dolayı işaretin değiştiği nokta, kenarın parlak bölgeden koyu(negatif) bölgeye veya koyu bölgeden parlak(pozitif) bölge tarafından hangisine geçtiğini belirtir.

Görüntülerde her pikselin türevinin alınması için aşağıdaki denklem 2.25 ve 2.26 elde edilen Şekil 2.7.a'daki maske ile konvolüsyon işlemi uygulanır.

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad \textbf{(2.25)}$$

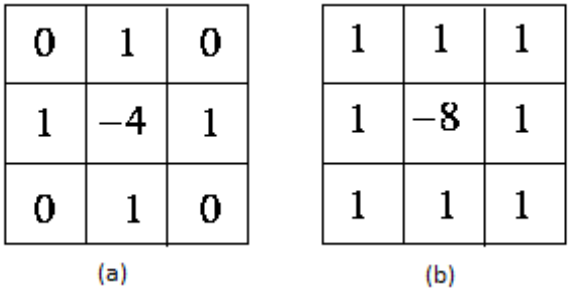
$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \qquad \textbf{(2.26)}$$

Ve,

$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \qquad \textbf{(2.27)}$$

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \qquad \textbf{(2.28)}$$

Bu eşitlikteki notasyona bağlı, herhangi bir yöne bağlı olmayan, resmin her tarafından keskinleştirme sağlayan aşağıdaki filtre oluşturulur.



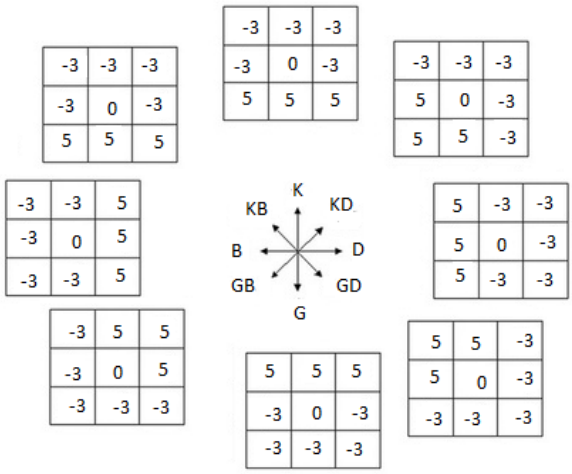
Şekil 2.7 : İkinci türev operatörleri.

Şekil 2.7.a'dan köşegendeki değerleri kapsayacak şekilde yeni bir filtre elde edilirse köşegendeki koordinatlara eklediğimiz terimler denklem 2.26 ve 2.27 kullanılarak elde edildiği için her köşegende -2f(x,y) vardır. 4 köşegen eklendiği için filtredeki orta piksel -8 olacaktır.

Laplace maskesi ile filtrelenmiş olan görüntülerde sadece sıfırın olduğu piksellerin işaretini değil aynı zamanda sıfır olmayıp işaret değişen yerlerde orijinal görüntünün ayrıt noktalarını belirler. Filtrelenmiş görüntüde işaretin değiştiği noktaları belirlemek için komşuların da işaret değişimine bakılır. Her hangi bir işaret değişimi varsa ve pikseldeki yoğunluk farkları belirlenen eşik değerinden büyükse o piksel kenar listesine eklenir.

2.3 Kirsch Algoritması

Kirsch algoritması bilgisayar bilimcisi olan Russell A. Kirsch tarafından bulunmuştur. Tek bir tane çekirdek filtre vardır. Bu filtre sekiz yönde 45 derece açıyla döndürülerek 8 farklı filtre şekildeki gibi oluşturulur [27,28] (Şekil 2.8).



Şekil 2.8 : Kirsch operatörü.

2.4 Canny Algoritması

Kullanılan kenar bulma teknikleri arasında en popüler olandır. Nesneler arasındaki farklılığı mümkün olduğu kadar gerçeğe en yakın bulur. Canny algoritması gürültüleri gidermek için görüntülere gauss filtresi uygular, devamında diğer kenar bulma operatörlerinin kullandığı herhangi bir filtre yardımıyla birinci türev işlemleri sonucunda elde edilen türev genliklerinden kenara dik yönünde komşu piksellerden en büyük gradyan değerini alan piksel muhtemel kenar noktası seçilir [9, 24, 29].

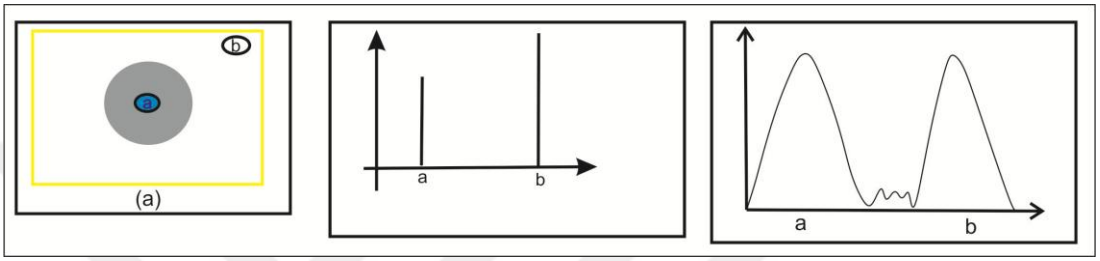
Canny kenarları tespit ederken, bütün gerçek kenarları algılayıp yanlış kenar olmaması, muhtemel kenarın bölgesel yerinin gerçek kenar ile arasındaki farkın en aza indirilmesi son olarak olası kenar noktalar arasında sadece bir tanesinin doğru kenar noktası seçilmesi gibi üç temel amacı hedefler. Bu amaçlar doğrultusunda Canny algoritmasının adımları aşağıda listelenmiştir.

2.4.1 Görüntünün Gauss ile yumuşatılması

Görüntülerde oluşan gürültü, kamera hareketinden dolayı meydana gelen bulanıklaşma veya ışık kaynağı, sensör, optik fokus, fotoğraf çekimi sırasında nesne ve kameranın göreceli konumu gibi unsurlardan kaynaklanan hatalardan dolayı oluşan gerçek görüntüler dışında istenmeyen piksellerin tamamına gürültü denir. Birçok gürültü giderme tekniği vardır. Canny bu işlem için Gauss gürültü giderme tekniğini kullanmıştır.

F(x,y) giriş görüntüsündeki (Şekil 2.10) her (x,y) farklı bir gri değer ile temsil edilmesinden dolayı rastgele bir süreç olarak ele alınır. Rastgele ifade edilen bu görüntü fonksiyonunu modellemek için olasılık kuramından yararlanılır.

Elektrik sinyalinden dijital görüntü elde edilmesinin her aşamasında piksellerde rastgele gürültüler oluşur. Belli frekans boyunca sürekli değişmeyen düzgün bir frekans aralığına sahip gürültüye beyaz gürültü denir. Beyaz gürültünün genlik değerleri histogram şekli gibi bir gauss eğrisidir. Bu matematiksel olarak modellenebilir [24].



Şekil 2.9 : Görüntülerde gauss gürültüsünün gösterimi.

$$f(x) = e^{-\frac{e^2}{2\sigma^2}} \tag{2.29}$$

Normal yoğunluk fonksiyonunda $\sigma = 1$ alınırsa a ve b arasındaki olasılık Şekil 2.9'da gösterilen sınırlar arasında kalan alana eşittir. Bu alan f(x) fonksiyonun integrali ile denklem 2.30 ve 2.31 ile hesaplanır.

$$\int f(x)dx = \int_a^b e^{-\frac{e^2}{2\sigma^2}} dx \tag{2.30}$$

$$P(x) = e^{\frac{(b-a)^2}{2\sigma^2}} \tag{2.31}$$

P(x) olasılık yoğunluk fonksiyonu ile oluşturulan bir Gauss gürültüsüdür.

Gauss fonksiyonu iki boyutlu görüntüler için yazılırsa:

$$G(x,y) = e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{2.32}$$

Gauss ile görüntünün konvolüsyonu sonucu daha yumuşak bir görüntü elde edilir:

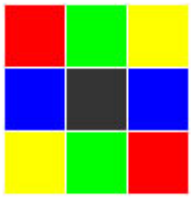
$$f_y = G(x,y) * f(x,y) \tag{2.33}$$

Normal dağılımda ortalamadan uzaklaştıkça değer küçülür, görüntüler üzerinde ise piksellerin komşuları arasındaki mesafe arttıkça değer azalır.

Elde edilen yumuşatılmış görüntüde her bir pikselin açısını ve yönünü bulmak için eşitlik 2.10 ve 2.11 ile açı ve gradyan büyüklüğü hesaplanır.

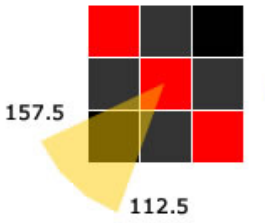
2.4.2 Maksimum olmayanların bastırılması

Bir önceki basamakta görüntünün gradyanın alınması ile bir pikselde daha büyük, kalın kenarlar elde edilir. Bu aşamada elde edilen kenar piksellerin inceltilmesi yapılacaktır.



Şekil 2.10 : F(x,y)

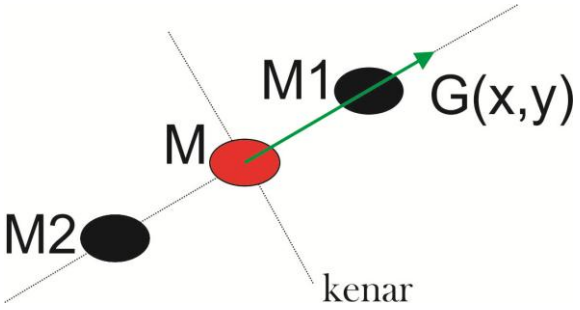
F(x,y) görüntüsünde, gri pikselin etrafında olabilecek kenar sayısı dörttür. Bunlar kuzeyden güneye olan yeşil, doğudan batıya mavi, çapraz olan kırmızı ve sarıdır. Siyah pikselden gradyan yönü kullanılarak kenarın nereye gittiği tahmin edilir.



Şekil 2.11 : Dikey kenar.

Kırmızı piksellerin kenar olup olmadığını belirlemek için gradyan değerlerinin bu siyah noktalarda maksimum olup olmadığının kontrol edilmesi gerekir. Bunun için sol üst piksel ve sağ alt piksel karşılaştırılır.

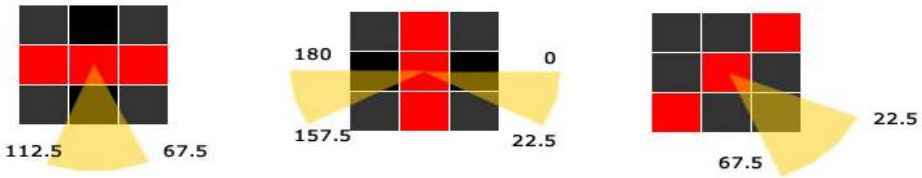
Her bir piksel için yukarıda hesaplanan gradyan yönüne göre Şekil 2.12'de olduğu gibi M2 ve M1 gibi iki komşu seçilir.



Şekil 2.12 : Komşu pikseler.

$$|G(M_2)| < |G(M)| \geq |G(M_1)| \tag{2.34}$$

M'deki pikselin gradyan değeri her iki komşusundan küçükse değeri sıfır yapılarak aday kenar noktasından silinir. Eğer büyükse kenar olarak kayıt edilir.



Şekil 2.13 : Muhtemel kenar yönleri.

Şekil 2.11'deki kırmızı piksel kenarı ifade eder. Merkez piksel siyah piksellerle karşılaştırılır.

67.5 - 112.5: Gradyan yönü yukarıdan aşağı olur. Kenar ise soldan sağa doğrudur çünkü bir kenar her zaman gradyan yönüne diktir.

0-22.5 veya 157.5-180: Gradyan yatay olunca kenarda dikey olur, sol ve sağ pikseller alınarak kontrol edilir.

Böylece her piksel 90, 0, 45, 135 değerleri arasında karşılık gelen bir açı değeri yönünde iki piksel seçilir. Eğer ortadaki piksel seçilen bu iki pikselden büyük değilse aday kenar noktasından silinir. Bu işlemler sonucunda kenara dik olan en büyük gradyan değerini alan kenar olarak belirlenir [15].

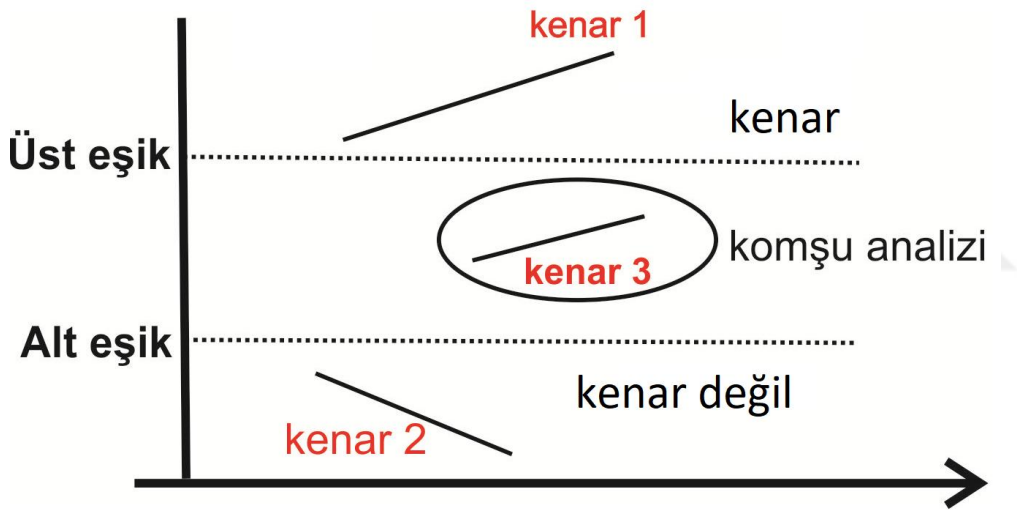
2.4.3 Eşikleme

Gradyan büyüklükleri hesaplanan piksellerin kenara dik yönünden maksimum olmayanların aday kenar noktasından silinmesi sonucunda kenarlar inceltilmiş olur. Bu aşamadan sonra kenarların sürekliliği bozulmuş olursa kenarlar arasında boşluklar oluşabilir. Bu aşamada iki farklı eşik değeri kullanılarak kenar adayı olan

piksellerin sürekliliği sağlamak için boşlukların doldurulması veya hatalı kenar piksellerinin silinmesi gerekir.

Diğer kenar algılama algoritmaları tek eşik kullanarak çözüm geliştirmişlerdir. Fakat tek eşik kullanılmasında eğer eşik değeri küçük seçilirse yanlış pozitif olarak adlandırılan gerçekte kenar olmayan hatalı kenar pikselleri artar. Eğer eşik değeri artırılırsa gerçek kenar pikselleri başka bir ifade ile yanlış negatifler silinir. Canny bu sorunları çözmek E_A alt eşik ve $E_Ü$ üst eşik gibi iki farklı değer kullanılır.

Elde edilen genlik değerleri, üst eşikten büyükse kenar olarak alınır. Eğer alt eşikten küçükse aday piksellerden silinir. Genlik değeri alt ve üst eşik değerleri arasında ise kenar adayının komşuları arasında üst eşiği aşan bir piksel var ise kenar adayı kenar olarak seçilir. Eğer komşusu yoksa aday listesinden silinir.



Şekil 2.14 : Canny eşikleme yöntemi.

2.5 Marr-Hildreth Algoritması

Marr ve Hildreth türev tabanlı kenar algılama algoritmaları gürültülü resimlerde çok hassas çalıştıkları için ikinci türev uygulamadan önce gürültüleri azaltmak için gauss filtresi uygulanması gerektiğini çalışmalarında belirtirler. Kenar algılama operatörlerinde olması gereken iki özellik üzerinde durmuşlardır. Birincisi, piksel yoğunluğundaki ani geçişlerin algılanmasıdır. Belirtilen geçişleri yakalamak için yönlü operatör kullanan birinci türev yerine ikinci türev kullanır çünkü görüntünün herhangi bir noktasındaki pikselin yönden bağımsız olarak fark değerini ölçer.

Böylece farklı yönlerdeki yoğunluk değişimini ölçeklemek için çok sayıda filtre kullanmasına gerek kalmaz. İkincisi, gürültünün çeşidine göre ayarlanabilen bir filtrenin görüntüler üzerinde uygulanması farklı ölçekte görüntüler üretmesine neden olacağı için farklı kenar görüntüleri elde edilir. Görüntüyü farklı ölçeklerde ele almak için alçak geçiren filtre çeşidi olan Şekil 2.10'daki Canny'deki gauss fonksiyonunun yönlü türevi alınarak LoG filtresi elde edilmiştir (Şekil 2.16).

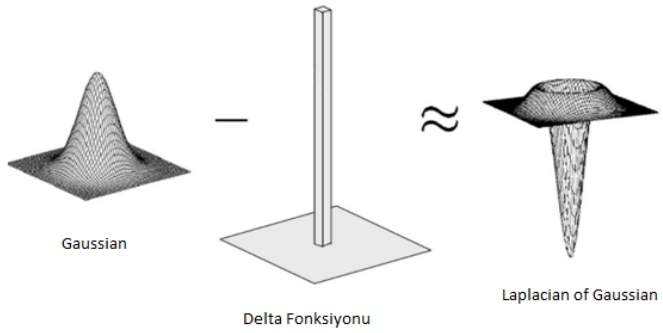
$$\nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{G^2(x,y)}{\partial y^2} \tag{2.35}$$

$$= \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] \tag{2.36}$$

$$= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.37}$$

$$= \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.38}$$

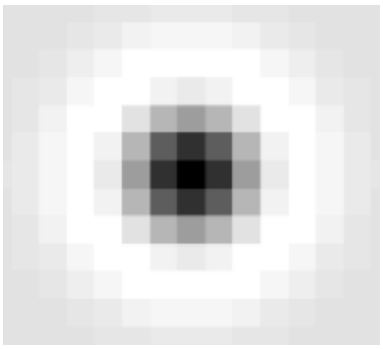
Sonunda LoG notasyonu elde edilir. Bu fonskiyonun 3B çizimi aşağıdaki şekilde gösterilmiştir.



Şekil 2.15 : Laplacian of Gauss gösterimi.

Meksika şapkası olarak bu görüntünün tersi alınarak aşağıdaki filtre elde edilmiştir.

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0



Şekil 2.16 : Gauss filtresi.

Bu filtrelerin görüntülerle konvolüsyonuyla, kenarlar arasında keskin geçişleri azaltarak daha bulanık görüntüler elde edilir. Marr ve Hildreth'in temel amacı görüntüleri farklı ölçeklerde ele almaktı. Bundan dolayı gauss fonksiyonu kullanılmıştır. Notasyondaki σ standart sapmayı ifade eder. Bu değerin değiştirilmesi ile Gauss filtresinin boyutu değişerek farklı ölçeklerde görüntüler elde edilir. Standart sapmanın artırılması sebebiyle görüntülerde bulanıklaşma artışı sıfır geçişlerin sayısını da artırır. Bu görüntülerde sıfır geçiş noktalarını tespit etmek için ikinci türev uygulanırsa gürültülerden arındırılmış daha belirgin kenarlar elde edilir. Fakat gerçek kenarların koordinatlarını doğru bulamayabilir. Eğer standart sapmayı azaltılırsa gürültülere karşı daha hassas davranır [3, 5].

2.6 Segmentasyon

Görüntü işlemenin temel adımlarından biridir. Genellikle nesneleri baz alarak görüntü analiz yapan çalışmalarda örneğin biyomedikal görüntülerde kullanılır. İki temel faydası bulunmaktadır. Birincisi nesnelerin temel karakteristiğini çıkarır ve kullanıcının sade ve öz veriler üzerinde analiz yapmasını sağlar. Böylece gereksiz piksellerle uğraşmak yerine segmentasyon sonucu elde edilen nesnelerin pikselleri kullanılır. Bir diğeri, nesnelerin dokusu, morfolojisi, fiziksel özelliği hakkında bilgileri elde ederek tanımlama ve sınıflandırmada kullanılır. Bu nedenle segmentasyon işleminin en az hata ile bulunması yapılan işlemin kalitesini artıracaktır.

Segmentasyon, bazı parametreler kullanarak görüntüyü homojen piksel gruplarına ayırma işlemidir. Bu parametreler parlaklık, renk, yoğunluk gibi her bir nesnenin özelliklerini belirler.

Haralick ve Saphiro'ya [30] göre segmentasyon yönteminin temel özellikleri;

- Segment edilmiş her bir bölge, bazı parametreler (piksel yoğunluk değeri, doku) açısından benzer özelliklere sahip olmalıdır.
- Her bölge tek tiptir.
- Bölgeler çakışmamalıdır. Komşu bölgeler farklı değerlere sahip olmalıdır.

Görüntülerin bölütlenmesi için yüzlerce yöntem bulunmaktadır. Bunlar genelde ani değişimleri baz alarak görüntüyü parçalara ayırırsa kenar tabanlı, eğer benzerlikleri hesaba katarak işlem yaparsa alan tabanlı yöntemler olarak ayrılır. Segmentasyon için oluşturulan bu yöntemlerin başarılı olması, görüntüye ve yapılan çalışmaya bağlı olarak değişiklik gösterir. Bundan dolayı tüm görüntülerde çalışan otomatik bir segmentasyon bulunmamaktadır.

2.6.1 Kenar tabanlı segmentasyon

Nesnelerin fiziksel özellikleri ile kenarları doğrudan ilişkilidir. Dolayısıyla görüntünün pek çok fiziksel özelliği kenar bilgisinden ortaya çıkarılabilir. Kenar tabanlı segmentasyonda ilk olarak eğime göre görüntü satır satır taranarak maksimum nokta saptanmaktadır. Başlangıç noktasına ulaşana kadar bu işlem devam eder. Başlangıç noktasına ulaşıldığında bir sonraki maksimum nokta saptanır ve tarama devam eder.

2.6.2 Bölge tabanlı segmentasyon

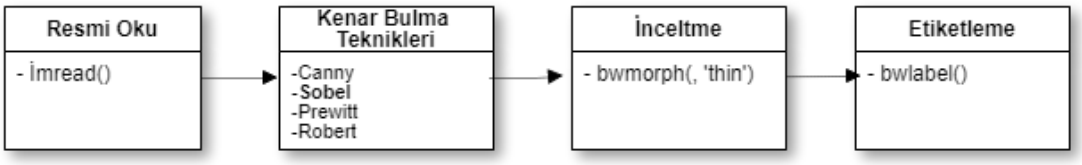
Bu yöntemde görüntüdeki gri seviye piksel değerlerinin benzerliğini kullanır. Benzerliklerine göre görüntü farklı bölgelere ayrılır. Eşikleme (thresholding), büyütme (growing), ve yarma - kaynaştırma (split-and-merge) işlemleri uygulanır.

3. SEGMENTASYONLA KENAR GÜÇLENDİRME ALGORİTMASI

Bu tez çalışmasında geliştirilen yöntem, segment edilmiş bölgelerin kenar çizelecek kadar benzer olup olmadığına karar vererek kopuk pikselleri tamamlar ve yeni kenarlar oluşturur.

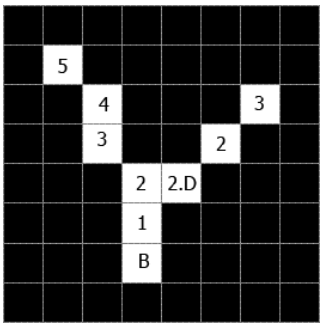
Bilinen kenar bulma algortimaları ile elde edilen kenarlar, başlangıç aşamasında referans bölge oluşturmak için kullanılmıştır. Referans kenarların sağ ve solunda referans bölgeler oluşturulmuştur. Sonraki adımda kenarın devam ettirilmesi gerekliliğine karar verebilmek için kenarın ilerleme ihtimali olan yönlerde bölgeler oluşturulmuştur. Bu bölgelerden referans bölgeye belirli bir kriteri sağlayarak en çok benzeyen bölgenin yönü, kenarın devam ettirilmesi gereken yön olarak belirlenmiştir. Geliştirilen yöntem basamakları bu bölümde ayrıntılı olarak anlatılmıştır.

3.1 Ön İşlemler



Şekil 3.1 : Ön işlem.

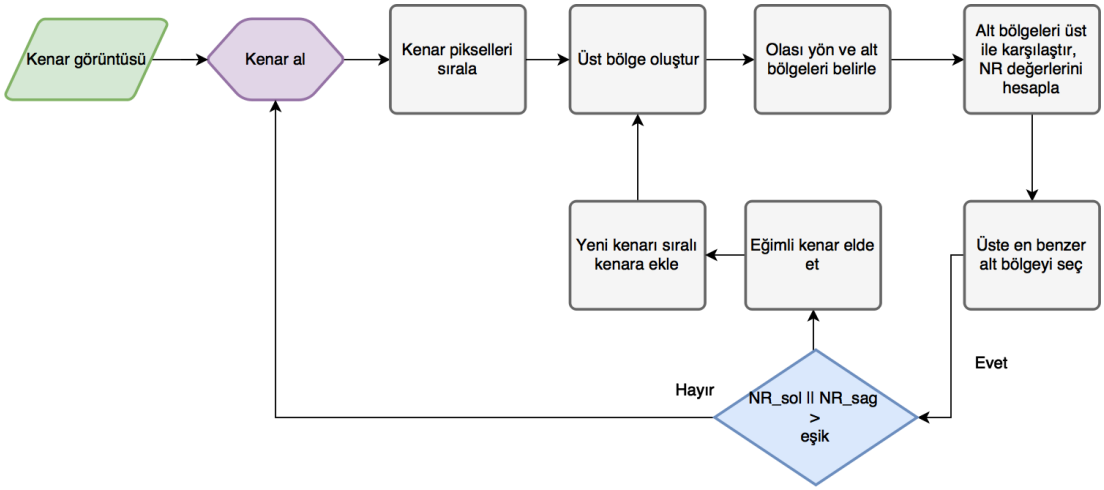
Kenar belirleme yöntemleri ile elde edilen kenarların üzerindeki pikseller sırayla alınarak inceleme işlemi yapılacağı için analiz yapılacak kenarların düzenli ve sıralı olması gerekir. Bu işlem için kenarların öncelikle inceltilip tek piksel kalınlığına indirgenmesi gerekir. Bunun için MATLAB bwmorph komutu 'thin' morfolojik işlem parametresi ile kullanılmıştır.



Şekil 3.2 : Bağlı pikseller.

İnceltmenin ardından piksellerin sıralanması işlemine geçilir. Başlangıçta kenarın sıralı bir şekilde dizilmesi için kenarın uç noktasının bulunması gerekir. Uç nokta, çevresinde sadece bir piksel bulunan noktadır. Uç piksel belirlendikten sonra komşu piksellere bakılarak sıralama yapılır. Pikseller sıralanırken bulunulan pikselin çevresinde birden fazla piksel varsa dallanma olduğu tespit edilir. Bu durumda, dallanan kenarlardan biri, dallanma noktasına ulaşana kadar elde edilmiş sıralı kenarın devamına eklenir. Diğer dallar ise ayrı birer kenar olarak alınıp ayrıca sıralanır. Bu şekilde kenar bulma teknikleri tarafından üretilen kenarların piksel koordinatları takip sıralarına göre kaydedilir. Şekil 3.1'deki örnekte B noktasından sıralama işlemi başlatılmış, 1 nolu pikselin komşusu birden fazla piksel olduğu için dallanma olduğu tespit edilip soldaki dal başlangıç dalı üzerinden devam ettirilip 2.D ile belirtilen pikselden yeni bir dal başlatılmıştır.

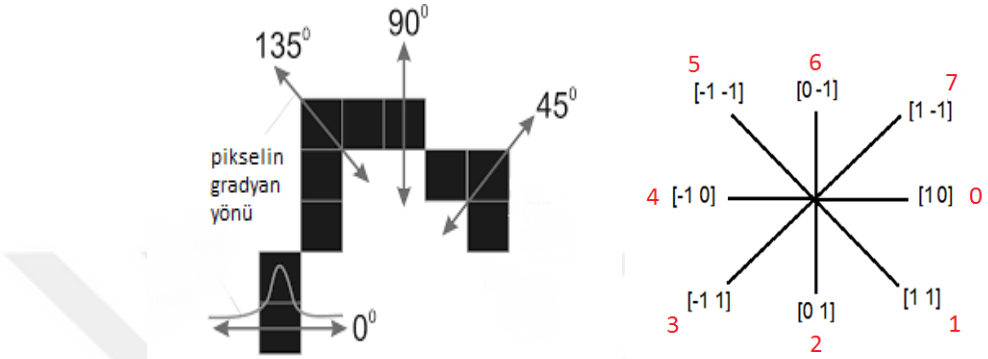
3.2 Temel İşlemler



Şekil 3.3 : Temel işlemler akış diyagramı.

3.2.1 Bölge oluşturulması

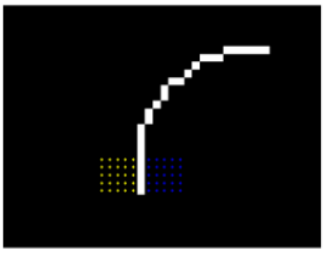
Bölgelerin oluşturulması için daha önce kenar bulma tekniklerinin elde ettiği kenarların hangi yöne doğru ilerlediği ve ilerleme ihitimali olan yönler tek tek hesaplanır. Bir önceki bölümde tek pikselle indirgenen kenarların üzerinde 5 piksel alınır. Her bir pikselin koordinatına göre yönü hesaplanarak yön histogramı oluşturulur (Şekil 3.4).



Şekil 3.4 : Yönler.

Kenar üzerinde alınan 5 pikselin yönü bir sonraki komşu pikselin ilişkisine bakılarak oluşturulur. Bunun için her bir pikselden ötelenmiş piksel çıkarılarak (Şekil 3.5) yönler hesaplanır.

X	Öteleme	Fark	Y	Öteleme	Fark
201	201	0	294	299	-5
201	201	0	295	294	1
201	201	0	296	295	1
201	201	0	297	296	1
201	201	0	298	297	1
201	201	0	299	298	1



Şekil 3.5 : Bölgeler.

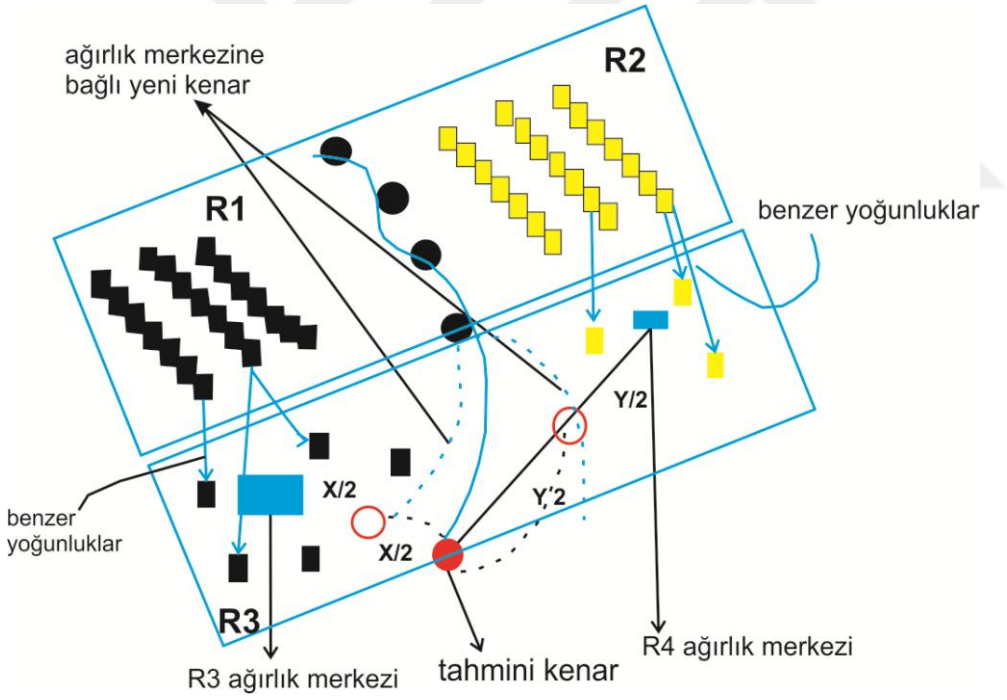
Hesaba katılan 5 pikselden elde edilen 5 yönden kenar ihitimali olan yönlerin seçilmesi işlemi yapılır . Bu işlem için son pikselin yönü, sayıca en fazla olan yön ve bunun sağ ve sol komşusu seçilerek bu yönlerin her birinin iki yanında 5x5 boyutunda bölgeler oluşturulur. Bu bölgeler bundan sonra üst bölge olarak isimlendirilecektir.

Gradyan tabanlı kenar bulma yöntemleri gürültülü bölgeler ve kontrastın düşük olduğu alanlarda gradyan zayıfladığı için ideal kenarı bulamazlar. Geliştirilen yöntemle kenarlar üzerinde ilerlerken segmentasyon ile elde edilen bölge bilgisi kullanıldığı için gradyanın zayıfladığı yerlerde hatalı kenar oluşumu engellenmiş olur.

3.2.2 Benzer bölge tespiti

Kenar bulma algoritmalarından elde edilen kenarlarda piksel ilerlerken algoritmanın doğruluğunu kontrol etmek için elde edilen bölgelerin birbiri ile benzerliğinden faydalanılır. Hatalı ya da eksik olan kenarlar geliştirilen algoritma ile düzeltilemeye çalışılır.

Bölgelerin birbirleri ile benzerliğini kıyaslarken üst bölgelerin ortalama ve standart sapma bilgileri kullanılmıştır. Alt bölgelerin kendi üst bölgelerine benzeyen piksel sayıları, NR, denklem 3.3'te belirtilen kıyaslama ile hesaplanmıştır.



Şekil 3.6 : SKG yöntemi.

$$\mu_B = \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} f(x,y)}{NM} \qquad \textbf{(3.1)}$$

$$\sigma_B = \sqrt{\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(f(x,y)-\mu_B)^2}{NM}} \quad \textbf{(3.2)}$$

$$|\mu_1 - \sigma_1| < f(x,y)_{R3} < |\mu_1 + \sigma_1| \quad \textbf{(3.3)}$$

$$NR3 < T \quad NR4 < T \quad \textbf{(3.4)}$$

Burada, μ_1 üst bölge ortalaması, σ_1 üst bölge standart sapması, $f(x,y)_{R3}$ R3 bölgesinin görüntü yoğunluk değerlerini gösterir. Belirtilen işlemler R4 bölgesi için de tekrarlanır. Elde edilen NR değerleri belirlenen eşik değerinin (T) üzerinde ise bölgelerin uyumlu olduğuna karar verilir. Eğer NR belirlenen eşik değerinden daha düşükse bu, kenar bulma tekniğinin hatalı veya eksik bir kenar bulduğunu ve iyileştirme yapmamız gerektiğini gösterir.

3.3 Yeni Kenarların Oluşturulması

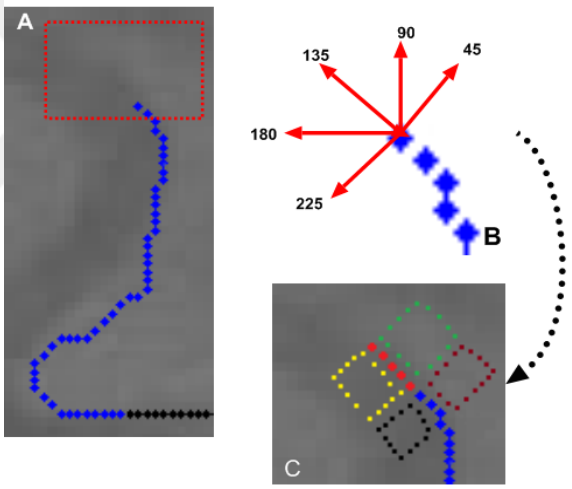
Bu adıma Şekil 3.6'da görüldüğü gibi, farklı iki koşulda ulaşılır. Birincisi, kenar belirleme operatörlerinin belirlediği kenarlar üzerinde ilerlerken elimizde daha önceden belirlenen kenarlar olduğu için alt bölge ve üst bölgeleri kolaylıkla oluşturup ikinci adımdaki kıyaslama teknikleri kullanılarak doğru kenarın çizilip çizilmediği kontrol edilir. Eğer doğru kenar çizilmemişse muhtemel kenarlar üretilene kadar tekrarlı olarak yeni kenarın çizilmesi gerekir. İkincisi, kenar operatörlerinin bulduğu kenarlar devam etmesi gerekirken sonlanmış olabilir veya kenarlar arasında kopukluk gerçekleşebilir. Bu sorunların çözümü için yukarıda belirtilen iki adımda hatanın belirlendiği pikseller veya kenarın sonladığı pikseller etrafında geniş bir alan seçilerek doğru kenar için bölgeler aranır.

İyi kenar bulma işleminde üzerinde durulması, gereken üç temel problem bulunmaktadır. Birincisi, istenilen bölgeyi hangi yönlerde oluşturarak arama yapılması gerekir? İkincisi, aranılan bölgenin özellikleri neler olmalıdır? Üçüncüsü, seçilecek bölgeye karar verildikten sonra bu bölgenin hangi pikselleri muhtemel kenarı temsil eder? Bu soruların cevabı aşağıda adım adım açıklanmıştır.

Adım 1: Doğruluğunu denklem 3.3'teki kıyaslama mantığını kullanarak kesinleştirilen kenar üzerinde referans alınan 5 pikselin her birini kendinden bir

önceki pikselle kıyaslayıp yönleri hesaplanır. Yeni belirlenecek kenar en son pikselin doğrultusunda devam edebilir veya alınan 5 piksel yönlerinde en fazla hangi yön var ise o yön ve o yönün bir öncesi ve sonrasındaki yönler alınarak yön listesi oluşturulur.

Adım 2: Bir önceki adımda oluşturulan yönlerin doğrultusunda bütün alt bölgeler oluşturulur. Şekil 3.7'deki A'da görülen siyah renkli kenar gradyan tabanlı kenar bulma tekniklerinin bulduğu kenardır. Görüntü üzerinde dikdörtgen bölgede yeni kenar araması yapılır. B'de arama yapılan yön listesi ve buna bağlı yön listesinin alt bölgeleri oluşturulur. C'de görüldüğü gibi kenar 135 yönündedir. Bu yönde alt bölge ve üst bölge oluşturulur. Bunlar sarı renkli kısım sol tarafın alt bölgesi, yeşil kısım sol tarafın üst bölgesi, kırmızı sağ tarafın alt bölgesi, mavi sağ tarafın üst bölgesidir. Ayrıca 135 derecelik yönün etrafındaki yönler olan 90 ve 180 yönleri içinde bölge oluşturma işlemi yapılır.



Şekil 3.7 : Yeni kenar oluşturma.

Adım 3: Önceki adımda üç farklı yönde oluşturulan alt bölgelerden referans olarak kullanılan üst bölgeye en çok benzeyeni kıyaslama teknikleri kullanılarak kenar olabilecek en iyi bölge seçilir.

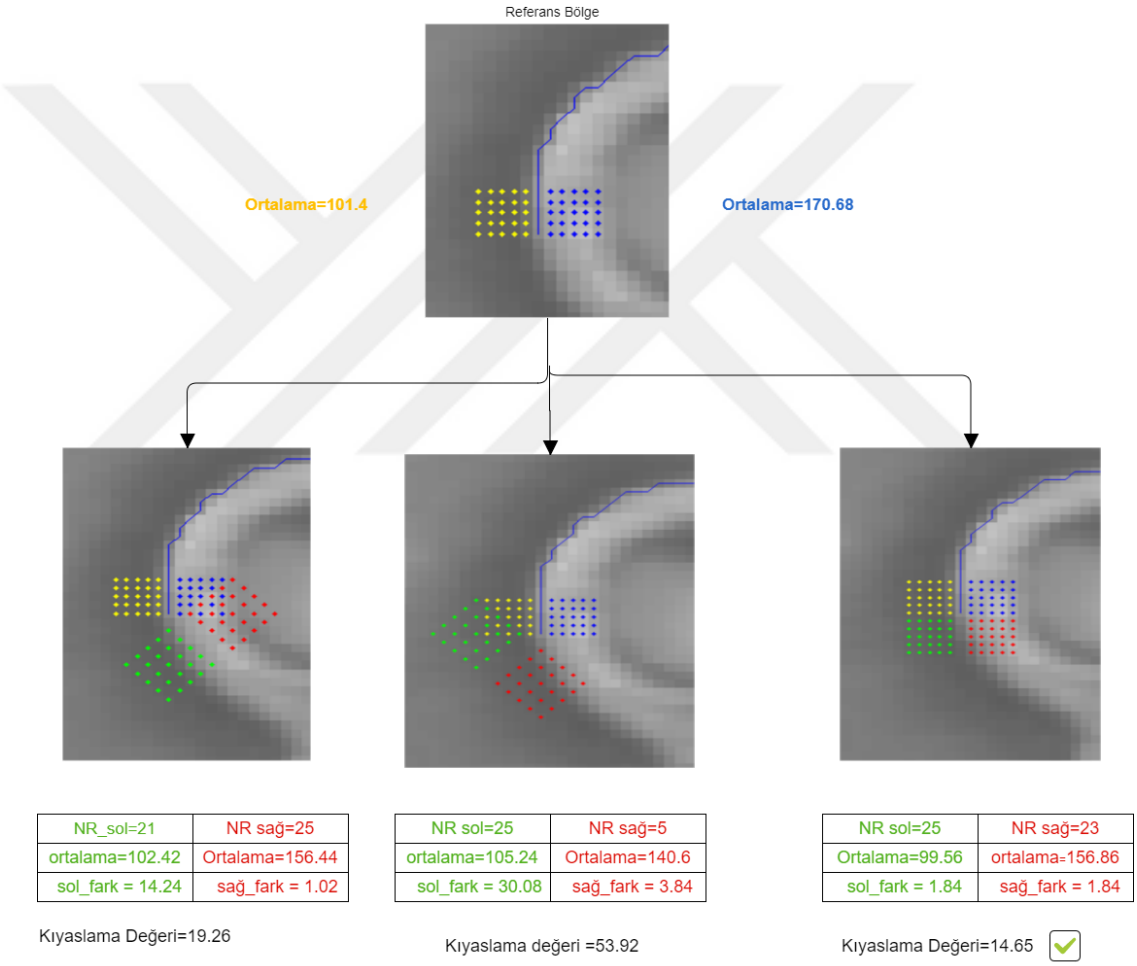
$$\mu_{SF} = |\mu_1 - \mu_3| \tag{3.5}$$

$$\mu_{SAF} = |\mu_2 - \mu_4| \tag{3.6}$$

$$FPS = BPS - (NR_3 + NR_4) \tag{3.7}$$

$$KD = FPS + \mu_{SF} + \mu_{SAF} \quad \textbf{(3.8)}$$

Üst bölgenin sapma değerinin büyük olması alt bölgenin daha çok pikselin üst bölgeye benzer olarak belirlenmesine sebep olmaktadır. Bu durumlarda kıyaslanacak üç yön için de benzer piksellerin sayısı (NR) aynı çıkabilmektedir. Denklem 3.5 ve 3.6'da hesaplanan üst ve alt bölgelerinin sol ve sağının ortalama farkları(μ_{SF} , μ_{SAF}) bu sebeple kıyaslamada ikinci bir kriter olarak kulanılmıştır. Üst bölgelere benzemeyen piksel sayıları(FPS) ve ortalama farkları toplanarak kıyaslama kriteri olan KD değeri hesaplanmıştır. Örnek bir anlatım Şekil 3.8'de gösterilmiştir.



Şekil 3.8 : Yön seçimi.

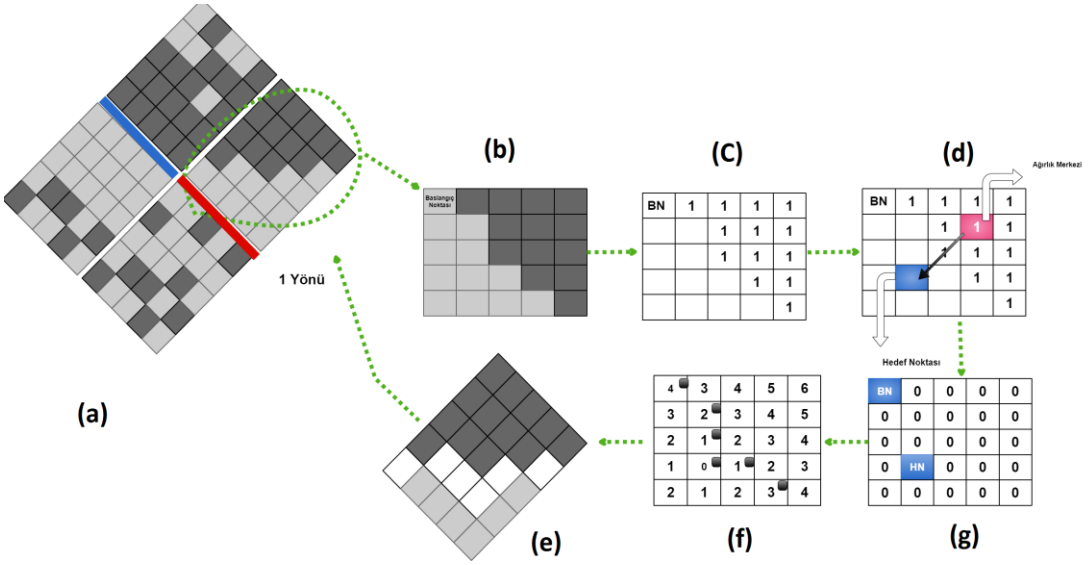
Yukarıdaki 3.8 denklemi yardımıyla elde edilen kıyaslama değeri en küçük olan alt bölge yeni kenarın çizilmesi için seçilmiştir.

Adım 4: Kenarın çizileceği bölgede kenarın nesne çevresine yakın ve daha eğimli geçmesi yeni oluşturulacak kenarın gerçeğe daha yakın olmasını sağlayacaktır. Şekil

3.9'da seçilen bölge koordinat düzleminde düzeltilerek, yukarıda anlatılan kıyaslama teknikleri kullanılarak üst ve alt bölge arasındaki benzer pikseller 1 (Şekil 3.9.c), diğer pikseller 0 olacak şekilde bölge ikili olarak yeniden oluşturulur. Oluşturulan ikili bölgenin ağırlık merkezi tespiti için değeri 1 olan piksellerin yatay ve dikeydeki konumlarının ortalamaları aşağıdaki denklem kullanılarak Şekil 3.9.d'de görüldüğü gibi mavi piksel olarak hesaplanır.

$$x_A = \frac{1}{N}\sum_{i=1}^{N} x_i \quad y_A = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{3.9}$$

Bölgenin kenara yakın olan en üst pikseli başlangıç noktası, ağırlık merkezinin dışında bir nokta hedef nokta alınır (Şekil 3.9.d).



Şekil 3.9 : Bölgede yeni kenarın çizilmesi.

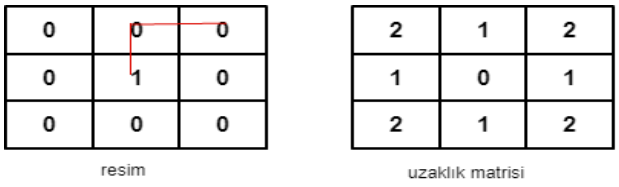
Sol üst köşesinden başlanarak Şekil 3.9.g'de gösterilen hedef noktasına doğru en kısa yolun çizilmesi ile yeni kenarlar elde edilir.

Başlangıç noktasında hedefe olan en yakın mesafenin bulunması için, matristeki her bir pikselin hedef noktasına olan uzaklığı bulunur. Uzaklık hesaplamasında pikseller birbiri arasındaki mesafelerin mutlak değerlerinin toplanmasıyla elde edilir.

(x1,y1) ile (x2,y2) arasındaki uzaklık;

$$|x1 - x2| + |y1 - y2| \tag{3.10}$$

MATLAB'ta bu uzaklık matrisi bwdist komutu sonucunda Şekil 3.9.f elde edilir. bwdist komutunun çalışma mantığı ise,



Şekil 3.10 : MATLAB bwdist çalışma mantığı.

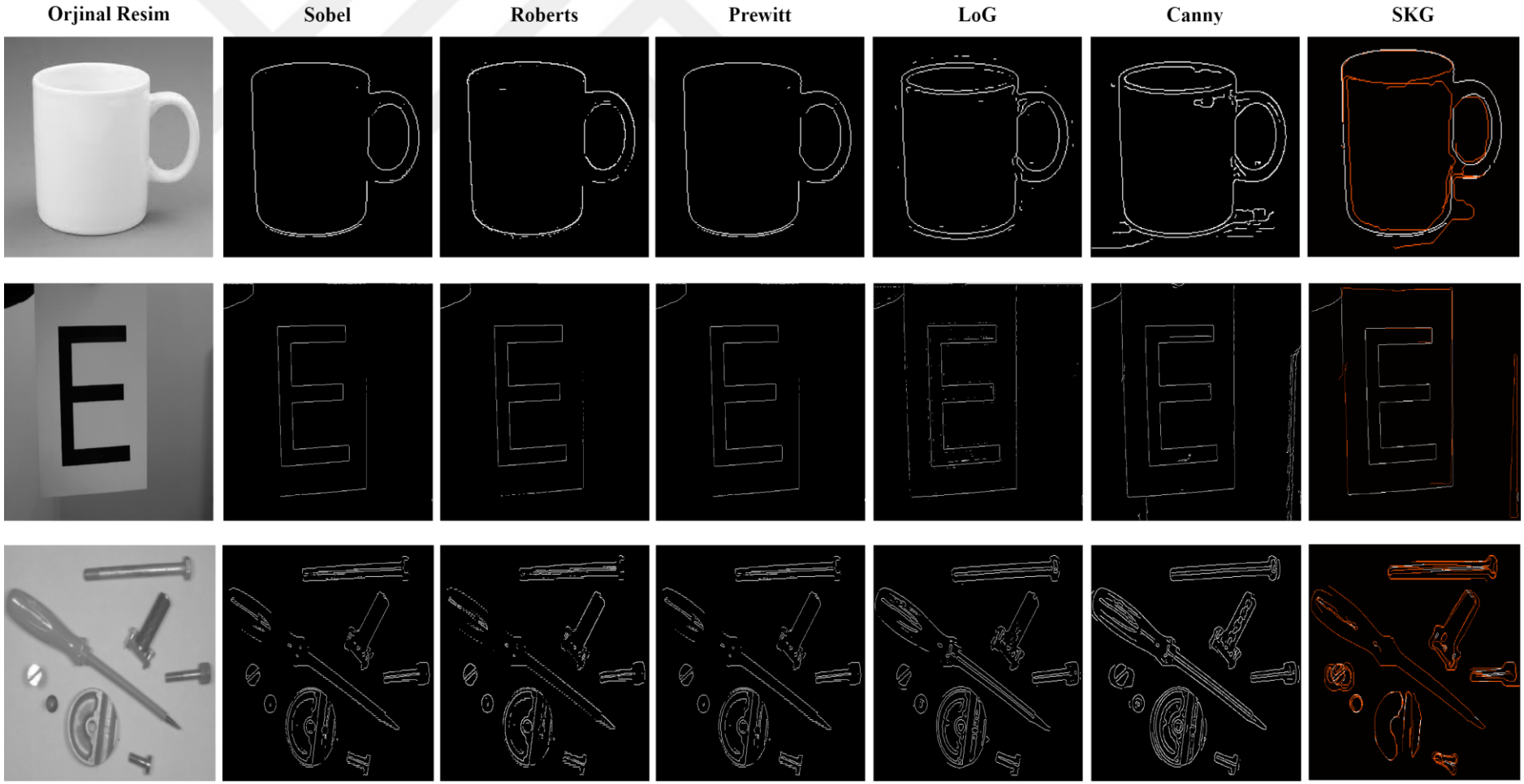
İkili resimde hedef noktası dışındaki pikselleri sıfırlayarak 8 komşu pikseller arasındaki uzaklık mesafesini ölçer. Yatay ve dikey arasındaki uzaklık 1 birim köşegen pikseller arasında mesafe ise iki birimdir. Bu yöntemle Şekil 3.9.f elde edilir. Başlangıç noktasından hedefe en kısa yoldan ilerlemek için, komşu piksellerden en küçük olan daha önce var olan benzer piksellerden değilse, o nokta yeni kenar pikseli olarak kayıt edilir. Hedefe ulaştıktan sonra komşu piksellerden en büyük olan fakat var olan benzer piksel değil ise kenar noktası seçilir.

4. DENEYSEL ÇALIŞMALAR

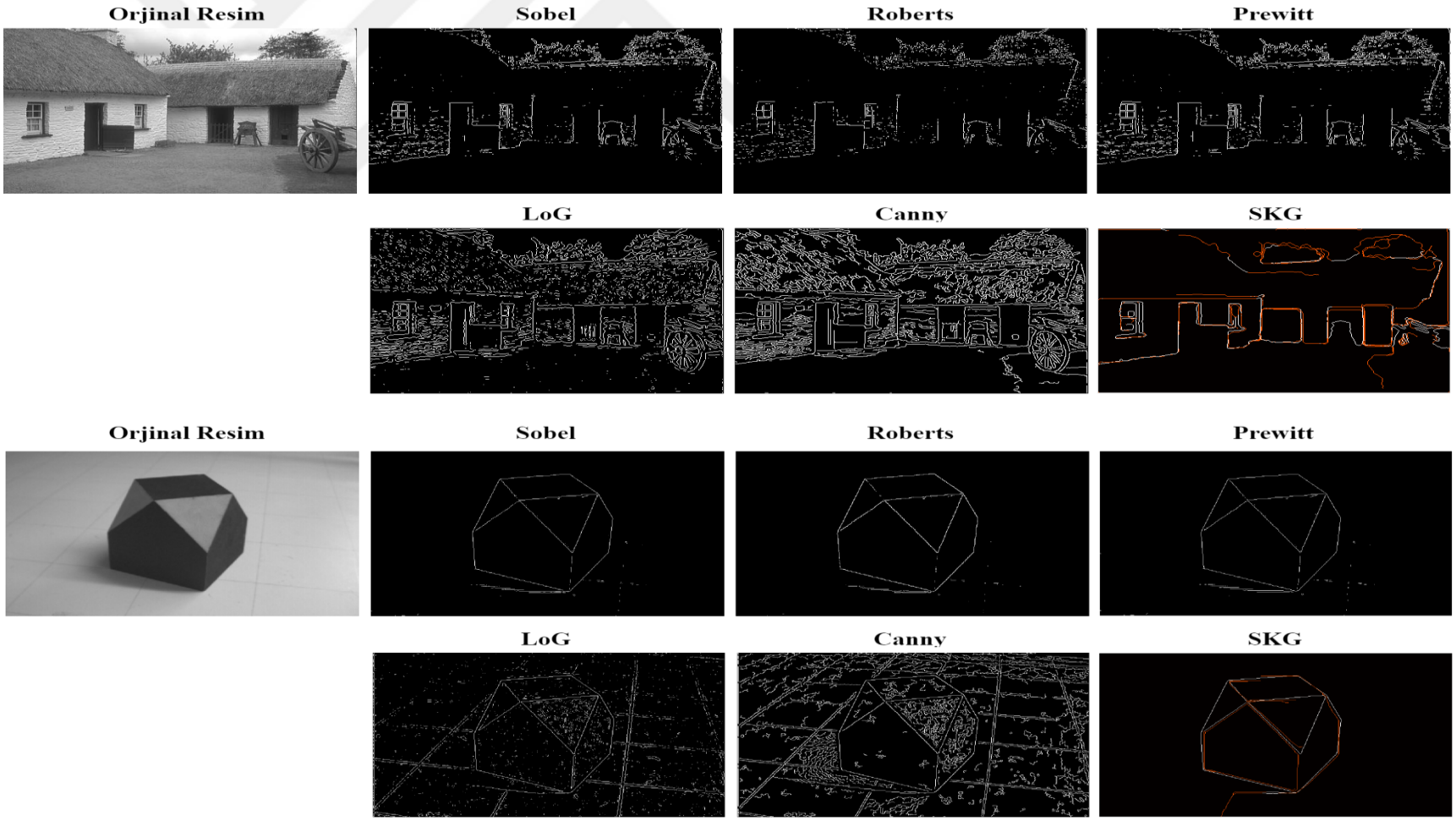
Bu bölümde, geliştirilen yöntem ve diğer kenar bulma yöntemleri ile elde edilen sonuçlar karşılaştırılmıştır. Yöntem MATLAB yazılımı ile geliştirilmiştir. Uygulamada kullanılan görüntüler Berkeley Segmentation Dataset ve The Hypermedia Image Processing Reference(HIPR) kaynaklarından alınmıştır. Anjiyo veri seti ise hastaneden etik izinle alınmıştır. İmgeler 512x512 çözünürlükte ve 8 bit DICOM formatındadır.

4.1 Sonuçlar ve Karşılaştırma

Geliştirilen yeni yöntemin güvenilirliğini artırmak için kenar bulma tekniklerinin başarısız olduğu kontrastı düşük, gürültülü ve gürültüsüz resimler seçilmiştir. Yazılım, gürültülü olan anjiyo görüntüleri ve doğa resmi, 2-B nesne görüntüleri gibi farklı yapıdaki resimler üzerinde çalıştırılmıştır. Kenar bulma yöntemleri uygulanırken eşik, sigma değeri gibi parametreler MATLAB'ın otomatik olarak belirlediği şekilde kullanılmıştır. Geliştirilen algoritmanın uygulanmasında gereken referans kenarlar ise Sobel ve Canny kenar bulma yöntemlerine farklı sigma ve eşik değerleri verilerek elde edilmiştir. Kenarları bağlı pikseller şeklinde bulmada daha başarılı oldukları için bu yöntemler tercih edilmiştir. Anjiyografi görüntülerinde ise kenar bulma tekniği olarak Canny uygulanmıştır. Sobel, anjiyografi resimlerinde başarı oranı düşük olduğu için kullanılmamıştır.

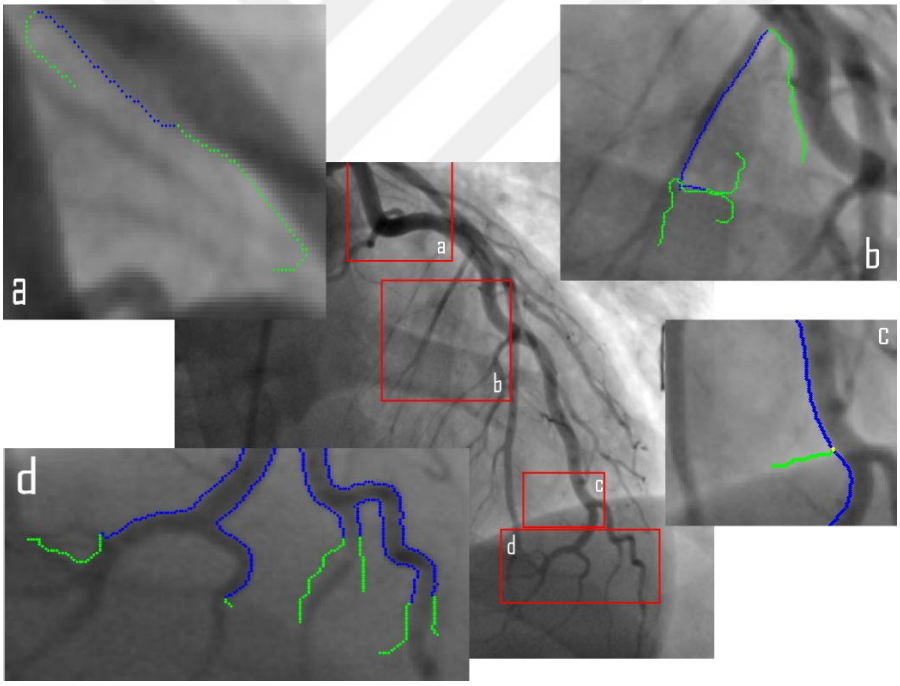


Şekil 4.1 : Kenar bulma algoritmalarının karşılaştırılması.

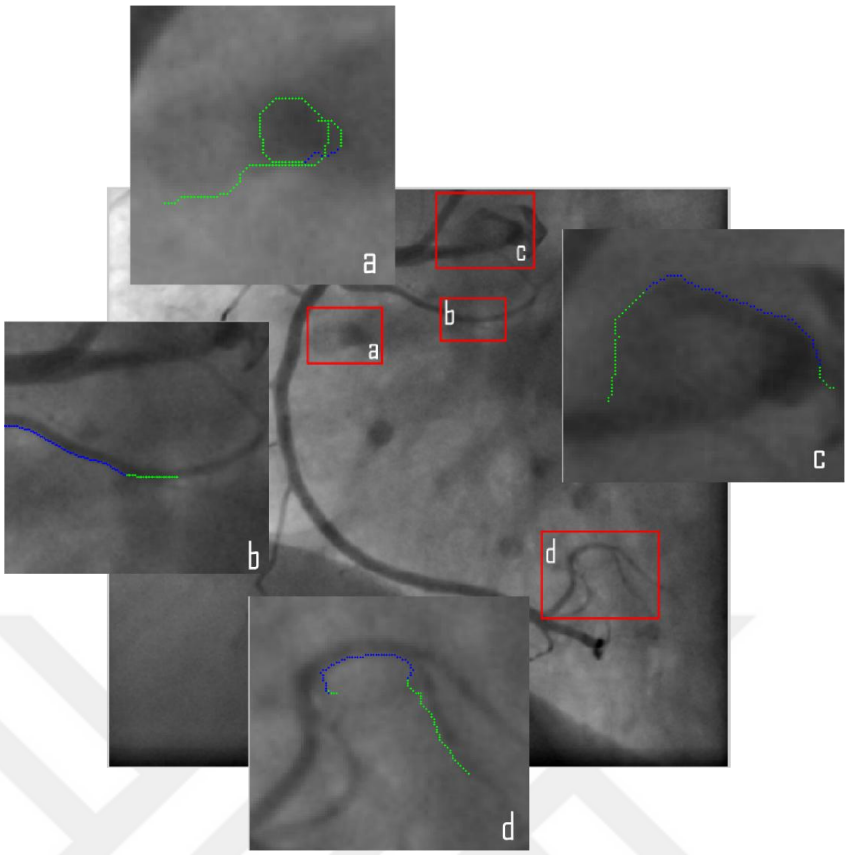


Şekil 4.2 : Kenar bulma algoritmalarının karşılaştırılması.

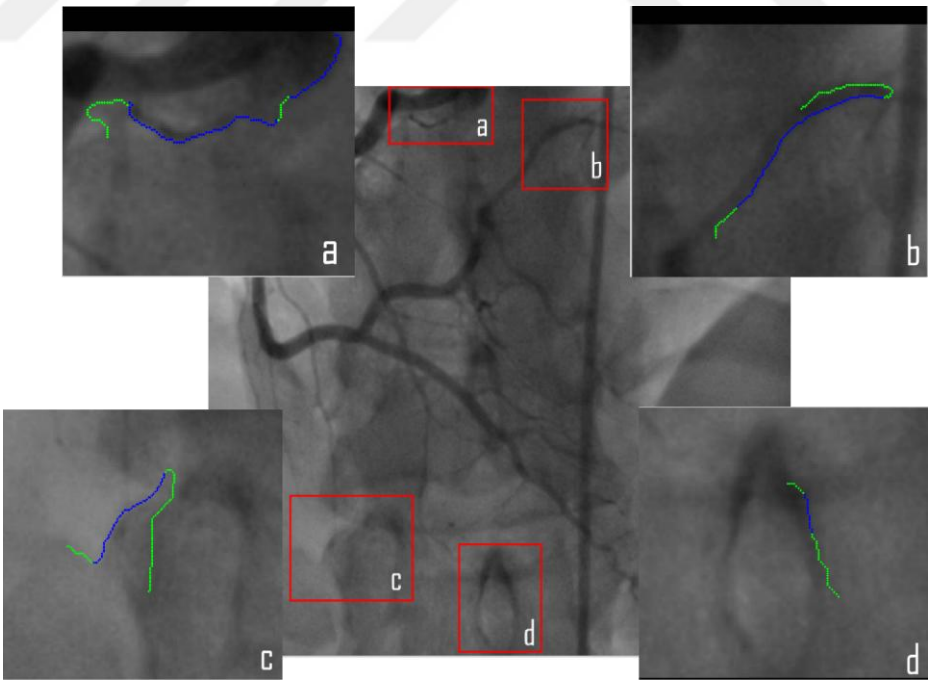
Şekil 4.2'de farklı kenar bulma yöntemlerinden ve geliştirilen yöntemden elde edilen sonuçlar gösterilmiştir. Görüntülerdeki beyaz renkli bölgeler bilinen kenar bulma yöntemleri ile elde edilen kenarları, turuncu renkli kısımlar ise sunulan yöntemin oluşturduğu kenarları göstermektedir. Önerilen yöntemde kontrastı düşük olan görüntülerde daha iyi bir kenar çıkarımı için sağ ve sol bölgelerin ortalama değerlerinin birbirine yakın olduğu durumlarda da kenar bulma işlemi devam ettirilmiştir. Bu yöntem sayesinde görüntülerde sadece ön planda bulunan ana nesnelerin kenarlarının çıkarımı sağlanmaktadır. Yöntem, Canny ve LoG gibi gelişmiş kenar bulma yöntemlerinin ana nesneyi bulmanın yanı sıra arka planda olan ve istenmeyen kenarları bulması problemine de bir çözüm getirmektedir. Ayrıca, gürültü sebebiyle diğer kenar bulma yöntemleri nesnelerin kenarlarını kopuk bulmaktadır. Bu yöntem, segmentasyon temelli çalıştığı için kenarları kopukluk olmadan, birbirini takip eden pikseller şeklinde bulmaktadır.



Şekil 4.3 : Anjiyo görüntüsünde SKG uygulaması.



Şekil 4.4 : Anjiyo görüntüsünde SKG uygulaması.



Şekil 4.5 : Anjiyo görüntüsünde SKG uygulaması.

Anjiyografi görüntülerine algoritmanın uygulanması ile elde edilen sonuçlar Şekil 4.3, 4.4 ve 4.5'te görülmektedir. Görüntülerde mavi renkli kenarlar Canny, yeşil

renkli olanlar ise algoritmanın ürettiği ve Canny'nin bulamadığı yeni kenarlardır. Şekil 4.5.a görüntüsünde kopuk kenar birleştirilmiştir. Şekil 4.3.c görüntüsünde sarı renkli piksel Canny'nin ürettiği kenarın yanında başka bir kenar olduğunu göstermektedir. O noktadan başlanarak yeni bir kenar çizilmiştir. Diğer şekillerde de Canny tarafından gürültü veya düşük kontrasttan dolayı bulunamayan kenarlar bulunmuştur.

5. SONUÇ VE ÖNERİLER

İnsan çocukluğundan beri zekâsını kullanarak nesneleri tanımak için çevresinde gördüğü görüntüler hakkında bilgi edinir. İyi bir kenar bulma algoritmasından beklenen şey bölgesel gradyan değişikliğine dayanmak yerine bazı genel özellikler de ekleyerek insanın görme yeteneğine yaklaşmasıdır. Bu çalışma belli bir yerde Canny, Sobel gibi sıradan kenar operatörleri ile mükemmel insan gözünün arasındaki boşluğu doldurmaya çalışıyor. Çalışma insanın görmesine benzer bir teknikle, segment edilmiş bölge bilgisini kullanarak kenar bulma algoritmalarına destek sağlar. Algoritma kontrastın ve türevin çok fazla azaldığı kenarların izini devam etmek için yok olmuş kenarların yerine segment edilmiş bölge bilgisi kullanır. Geliştirilen teknikle Berkeley Segmentation Dataset ve The Hypermedia Image Processing Reference (HIPR) veritabanından alınan görüntüler ve medikal anjiyo görüntüleri üzerinde çalışılmış ve çıkan sonuçlar incelenmiştir. Algoritma kontrastı çok düşük anjiyo görüntülerinde kalp damarlarının sınırlarını iyileştirmiştir.

Segmentasyon yöntemi ile kenar bulma algoritması, kenar bulma algoritmalarının herhangi bir nedenle bulamadığı bölgelerde gerçekte var olan kenarı bulur. Deneysel çalışmaların sonucunda, diğer algoritmalara kıyasla arka plan gürültüsünden arındırılmış görüntülerdeki nesnelerin kenarlarında iyileştirmeler görülmüştür.

KAYNAKLAR

[1] **Khaire, P. A., Thakur, N. V.,** 2012, Image edge detection based on soft computing approach, *International Journal of Computer Applications*, 51, 12-14.

[2] **Suzuki, C. T. N., Gomes, J.F., Falcao, A. X., Papa, J. P., Hoshino-Shimizu S.,** 2013, Automatic segmentation and classification of human intestinal parasites from microscopy images, *IEEE Transactions on Biomedical Engineering*, 60(3), 803–812.

[3] **Vasavada, J., Tiwari, S.,** 2013, An Edge Detection Method for Grayscale Images based on BP Feedforward Neural Network, *International Journal of Computer Applications*, 67(2), 22-28.

[4] **McGaffin, M. G., Fessler, J. A.,** 2015, Edge-Preserving Image Denoising via Group Coordinate Descent on the GPU, *IEEE Transactions on Image Processing*, 24(4), 1273-1281.

[5] **Hildreth, E. C.,** 1985, Edge Detection, *Artificial Intelligence Laboratory, Massachusetts Institute of Tecnology*, A. I. Memo No. 858, September.

[6] **Kumar, M., Singh, S.,** 2012, Edge Detectıon And Denoısıng Medical Image Using Morphology, International Journal of Engineering Sciences & Emerging Technologies, 66-72

[7] **Shah, M.,** 1997, Fundamentals of Computer Vision (PDF), University of Central Florida, Orlondo.

[8] **Heath, M.D., Sarkar,S., Sanocki, T., Bowyer, K.W.,** 1997, A robust visual method for assessing the relative performance of edge-detection algrithms, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19, 1338-1359.

[9] **Shin, M.C., Goldgof, D., Bowyer, K.W.,** 2001, Comparison of edge detector performance through use in an object recognition task, *Computer Vision and Image Understanding*, 84(1), 160-178.

[10] **Peli, T., Malah, D.,** 1982, A Study of Edge Detection Algorithms, Computer Graphics and Image Processing, 20, 1-21.

[11] **Deng, J., Wang, G., Yang, X.,** 2013, Image edge detection algorithm based on improved canny operator, *International Conference on Wavelet Analysis and Pattern Recognition*, Tianjin, 168-172.

[12] **Mainin, R., Aggarwal, H.,** 2009, Study and comparison of various image edge detection techniques, International Journal of Image Processing, 3(1), 1-11.

[13] **Roushdy, M.,** 2006, Comparative study of edge detection algorithms applying on the grayscale noisy image using morphological filter, ICGST, International Journal of Graphics, Vision, and Image Processing, 6, 17–23.

[14] **Juneja, M., Sandhu, P. S.,** 2009, Performance evaluation of edge detection techniques for images inspatial domain, *International Journal of Computer Theory and Engineering*, 1(5), 614-621

[15] **Kaur, R.,** 2016, Edge Detection of an Image Using an Improved Canny Algorithm: A Review, International Journal of Advanced Research in Computer and Communication Engineering, 5(8), 587-590.

[16] **Deng, C. Wang, G., Yang, X.,** 2013, Image Edge Detection Algorithm Based on Improved Canny, Operator Proceedings of the 2013 International Conference on Wavelet Analysis and Pattern Recognition, Tianjin, 14-17 July

[17] **Kale, U., Deokar, S. M.,** 2014, An Edge Detection Method Using Back Propagation Neural Network, International Conference On Industrial Automation And Computing, 5, 7-11

[18] **Yiğitbaşı, E.,** 2014, Yapay Arı Kolonisi Optimizasyonu İle Kenar Bulma, Yüksek Lisans Tezi, Selçuk Üniversitesi, Fen Bilimleri Enstitüsü.

[19] **Hazer, M.,** 2007, Bulanik Topolojiye Dayali Kenar Bulma Algoritmasi, Yüksek Lisans Tezi, Anadolu Üniversitesi, Fen Bilimleri Enstitüsü.

[20] **Aybar, E.,** 2003, Topolojik Kenar İşleçleri, Doktara Tezi, Anadolu Üniversitesi, Fen Bilimleri Enstitüsü.

[21] **Prewitt, J. M. S.,** 1970, Object Enhancement and Extraction, Picture Processing and Psychopictorics, 75-149, Eds. B. Lipkin ve A. Rosenfeld, New York.

[22] **Sobel, I.,** 1970, Camera Models and Perception, Doktora Tezi, Stanford University CA.

[23] **Canny, J.,** 1986, A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8, 679-700.

[24] **Gonzalez, R., Wintz, P.,** 1987, Digital Image Processing(3th ed.), Publisher: Addison–Wesley.

[25] **Roberts, L. G.,** 1965, Machine Perception of Three-Dimensional Solids, *Optical and Electro-Optical Information Processing*, 159-197, Eds. J. Tippett, et al., MIT Press

[26] **Sobel, I.,** 1990, An Isotropic 3×3 Gradient Operator, Machine Vision for Three–Dimensional Scenes, 376-379, Eds. Freeman, H., Academic Press, New York.

[27] **Xiao, W., Hui, X.,** 2010, An improved canny edge detection algorithm based on predisposal method for image corrupted by gaussian noise, *World Automation Congress*, 113-116.

[28] **Kirsch, R.,** 1971, Computer determination of the constituent structure of biological images, Computers and Biomedical Research, 4, 315–328.

[29] **Ramamurthy, B., Chandran, K. R.,** 2011, Content based image retrieval for medical images using Canny edge detection algorithm,*International Journal of Computer Applications*, 17, 32-37

[30] **Haralick, R. M., Shapiro, L. G.,** 1985, Image Segmentation Techniques, Computer Vision, Graphics and Image Processing, 29, 100-132

EKLER

EK A : Kaynak Kodları

EK A

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Referans kenarı dikkate alarak doğru yönde yeni kenarların
% oluşturulmasını sağlar
% Parametreler:
% @ust_bolge: orjinal resim boyutunda, tek kenar içeren BW matris
% @im: incelenen görüntü
% @sonuc_ed: elde edilen yeni kenarları barındıran matris
% Sonuç:
% @sonuc_ed: algoritma ile yeni bulunan kenaraların eski kenarlarla
% birleştirildiği matris
function sonuc_ed = yeniKenarOlustur(ust_bolge,im,sonuc_ed)
flag=true;
oran=0.3;

%referans bölgeye göre ilerlenebilecek yönlerden optimum olanı seçer
[bolge_yon, alt_bolge]=bolgeArama(ust_bolge,im);
if(isempty(alt_bolge.x))
   return;
end

%kenarları bul
while(flag)
   if (~isempty(alt_bolge.x))
      [ alt_bolge,NR_sol,NR_sag ] = kiyasla(ust_bolge,alt_bolge);

      %sağ ve sol bölgelerde birbirine benzerlik fazla ise hesaplamayı sonlandır
      if((abs( round(alt_bolge.m_sol)-round(alt_bolge.m_sag))<3) )
         break;
      end
      %alt bölgenin benzerlik oranı hangi bölgede fazla ise o yönde ilerle
      if(NR_sag>(numel(alt_bolge.sag)*oran) || NR_sol>(numel(alt_bolge.sol)*oran))
         if(NR_sol==NR_sag)
            if(alt_bolge.m_sag >alt_bolge.m_sol)
               [flag,ust_bolge,alt_bolge,bolge_yon,sonuc_ed] =
sagBolgedeKenarOlustur(im,ust_bolge,alt_bolge,bolge_yon,sonuc_ed);
            else
               [flag,ust_bolge,alt_bolge,bolge_yon,sonuc_ed] =
solBolgedeKenarOlustur(im,ust_bolge,alt_bolge,bolge_yon,sonuc_ed);
            end
         elseif(NR_sag>(numel(alt_bolge.sag)*oran))
            [flag,ust_bolge,alt_bolge,bolge_yon,sonuc_ed] =
sagBolgedeKenarOlustur(im,ust_bolge,alt_bolge,bolge_yon,sonuc_ed);
         elseif(NR_sol>(numel(alt_bolge.sol)*oran))
            [flag,ust_bolge,alt_bolge,bolge_yon,sonuc_ed] =
solBolgedeKenarOlustur(im,ust_bolge,alt_bolge,bolge_yon,sonuc_ed);
         end
      else
         sonuc_ed(ust_bolge.y(1),ust_bolge.x(1))=4;
         flag=false;
```

```
    end
  end
end
end

function [flag,ust_bolge,alt_bolge,bolge_yon,sonuc_ed] =
solBolgedeKenarOlustur(im,ust_bolge,alt_bolge,bolge_yon,sonuc_ed)
baslangic=[ust_bolge.x(end);ust_bolge.y(end)];
ro_sol=ust_bolge.ro_sol;
alt_bolge_sol=alt_bolge.sol;
bolge=BWYap(ust_bolge.m_sol,ro_sol,alt_bolge_sol);
[y,x]=egimliKenarCiz(bolge);
[x,y]=YXSifirla(bolge_yon,x,y);
[yon,x,y]=yonSol(bolge_yon,x,y);
[x,y]=caprazYonXY(yon,x,y,baslangic);
[flag,ust_bolge,alt_bolge,bolge_yon,sonuc_ed]=siradakiAdimiBelirle(x,y,im,ust_bol
ge,sonuc_ed,alt_bolge);
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
% Verilen bölgede kenarın eğimli çizilebilmesini sağlar
% Parametreler:
% @ilk_bolge: eğimli kenarın oluşturulacağı bölge
% Sonuç:
% @x: eğimli olarak elde edilen kenarın x koordinatları
% @y: eğimli olarak elde edilen kenarın y koordinatları
function [x, y] = egimliKenarCiz(inceleneck_bolge)
%örnek bölge=[0 1 1 1 1
%             0 0 1 1 1
%             0 0 1 1 1
%             0 0 0 1 1
%             0 0 0 0 1];

[alanx,alany]=find(inceleneck_bolge==1);
alanxmean=round(mean(alanx));
alanymean=round(mean(alany));
bolge=zeros(5,5);

%sol=(x,y+2)
if((alanxmean==3 && alanymean==1) || (alanxmean==3 &&
alanymean==2))
    bolge((alanxmean),round(alanymean+2))=1;
%sol ve yukari=(x+2,y+2)
elseif((alanxmean==2 && alanymean==2))
    bolge((alanxmean+2),round(alanymean+2))=1;
%sol ve asagi=(x-2,y+2)
elseif((alanxmean==4 && alanymean==2))
    bolge((alanxmean-2),round(alanymean+2))=1;
%sag=(x,y-2)
```

```
elseif((alanxmean==3 && alanymean==4)|| (alanxmean==3 &&
alanymean==5))
    bolge((alanxmean),round(alanymean-2))=1;
%sag ve yukarı=(x+2,y-2)
elseif((alanxmean==2 && alanymean==4))
    bolge((alanxmean+2),round(alanymean-2))=1;
%sag ve aşağı=(x-2,y-2)
elseif((alanxmean==4 && alanymean==4))
    bolge((alanxmean-2),round(alanymean-2))=1;
%yukarı=(x+2,y)
elseif((alanxmean==1 && alanymean==3)||(alanxmean==2 &&
alanymean==3))
    bolge((alanxmean+2),round(alanymean))=1;
%asagı=(x-2,y)
elseif((alanxmean==5 && alanymean==3)||(alanxmean==4 &&
alanymean==3))
    bolge((alanxmean-2),round(alanymean))=1;
%orta=(x-2,y)
elseif((alanxmean==3 && alanymean==3))
    bolge((alanxmean+2),round(alanymean-2))=1;
else
end;

D2=bwdist(bolge,'cityblock');
ekle=max(max(D2))+1;
im_boyut=size(D2);
resim=zeros(im_boyut+2);
resim(:,:)=ekle;
resim(2:end-1,2:end-1)=D2;

%5x5 lik bitwise bölge genişletilir
genis_incelenecek_bolge=zeros(im_boyut+2);
genis_incelenecek_bolge(:,:)=max(max(inceleneck_bolge))+1;
genis_incelenecek_bolge(2:end-1,2:end-1)=inceleneck_bolge;


baslangic=[2;2];
adim=[];
adim=[adim baslangic];
b_x=baslangic(1,1);
b_y=baslangic(2,1);
%{
bölge_yapısı=struct('sol_ust',[],'ust',[],'sag_ust',[],
                    'sol',[],'merkez',[],'sag',[],
                    'sol_alt',[],'alt',[],'sag_alt',[]);
%}
while(~(resim(b_x,b_y)==0))
    % b_x,b_y merkezli 3*3 lük bölge al
    [bul,bolge]=komsu(resim,b_x,b_y);
    % bulunan 3*3 bölgenin en küçük elemanını bulup o yönde
ilerle
    min_deger=min(min(bul));

[adim,baslangic]=sonraki_adim(resim,genis_incelenecek_bolge,mi
n_deger,baslangic,bolge,adim,b_x,b_y);
```

```
    b_x=baslangic(1,1);
    b_y=baslangic(2,1);
end
[xresim ,yresim]=ind2sub(size(resim),find(resim==0));
baslangic=[xresim; yresim];
b_x=baslangic(1,1);
b_y=baslangic(2,1);
adim1=[];

while(~(resim(b_x,b_y)==ekle))
   [bul,bolge]=komsu(resim,b_x,b_y);
    max_deger=max(max(bul));

[adim1,baslangic]=sonraki_adim(resim,genis_incelenecek_bolge,m
ax_deger,baslangic,bolge,adim1,b_x,b_y);
    b_x=baslangic(1,1);
    b_y=baslangic(2,1);
end

sonuc=[adim adim1];
sonuc(:,end)=[];
y=sonuc(1,1:end)-ones(1,size(sonuc,2));
x=sonuc(2,1:end)-ones(1,size(sonuc,2));
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
% Kenarları dallara ayırarak birbirini takip eden pikselleri
sıralar
% Parametreler:
% @kenar_matrisi: orjinal resim boyutunda, tek kenar içeren BW
matris
% Sonuç:
% @sirali_kenar_tum_dallar: bir labelden üretilen tüm dalların
contur yapısı(x,y)
function [ sirali_kenar_tum_dallar] = pikselSirala(
kenar_matrisi )
sirali_kenar_tum_dallar={};
sirali_kenar=[]; %yapısı: [x; y], size(2,n)
m=1;

%matristeki kenarin koordinatlarını al
%ind2sub: lineer indexi 2B koordinata çevirir
[xresim,yresim]=ind2sub(size(kenar_matrisi),find(kenar_matrisi
==1));
baslangic_noktasi=[];

%labelin uç noktasını belirler. Uç nokta çevresinde sadece 1
pixel olan
%noktadir.
for i=1:length(xresim)
    cevredeki_noktalar =
sekizKomsuyaBak([xresim(i);yresim(i)],kenar_matrisi);
```

```
    if(size(cevredeki_noktalar,2) == 1)
        baslangic_noktasi=[xresim(i);yresim(i)];
        break;
    end
end
dal_baslangiclari=baslangic_noktasi;

%dinamik olarak olusan tum kenar dallarının piksellerini
sırala
while( m <= size(dal_baslangiclari,2))
    bulunulan_nokta=dal_baslangiclari(:,m);
    flag=true;
    while(flag)
        cevredeki_noktalar  =
sekizKomsuyaBak(bulunulan_nokta,kenar_matrisi);
        %dallanma yada kenarın son noktasına gelinme durumu
değerlendir
        if(size(cevredeki_noktalar,2) > 1)
            sirali_kenar=[sirali_kenar bulunulan_nokta];
            dal_baslangiclari=[dal_baslangiclari
cevredeki_noktalar(:,2:end)];
            %kulanılan pixelleri tekrar ulaşılamaması için
temizle

kenar_matrisi(bulunulan_nokta(1),bulunulan_nokta(2))=0;

kenar_matrisi(cevredeki_noktalar(1,:),cevredeki_noktalar(2,:))
=0;
            bulunulan_nokta=cevredeki_noktalar(:,1);
        elseif(size(cevredeki_noktalar,2) == 1)
            sirali_kenar=[sirali_kenar bulunulan_nokta];

kenar_matrisi(bulunulan_nokta(1),bulunulan_nokta(2))=0;
            bulunulan_nokta=cevredeki_noktalar;%dizideki tek
elemanı al
        else
            sirali_kenar=[sirali_kenar bulunulan_nokta];
            sirali_kenar_tum_dallar{m}=swap(sirali_kenar);
            %kulanılan pixelleri tekrar ulaşılamaması için
temizle

kenar_matrisi(bulunulan_nokta(1),bulunulan_nokta(2))=0;
            m=m+1;
            sirali_kenar=[];
            flag=false;
        end
    end
end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Oluşturulacak yeni kenarın hangi yönde olacağını belirler
% Parametreler:
% @ust_bolge: üst bölge bilgisi
% @im: incelenen görüntü
% Sonuç:
% @secilen_alt_bolge: belirlenen alt bölge bilgisi
% @secilen_yon: belirlenen alt bölge kenarının yönü
function [secilen_yon,secilen_alt_bolge] =
bolgeArama(ust_bolge,im)
deltax=ust_bolge.deltax;
deltay=ust_bolge.deltay;
%delta pointlerinin eğimlerini bulup yönleri döndürür
yonler=cevir(deltax,deltay);
%son pikselin yönü alınır
yon1=yonler(end);
%yonler dizisinden olası sekiz yon icin her yonden kac tane
var oldugu
%bilgisi
liste=[];
for n=0:7
    liste=[liste length(find(yonler(:)==n))];
end
deger=find(liste==max(liste));
deger=deger-ones(1,length(deger));
farklar=abs((deger-yon1));
yon2=deger(find(farklar==min(farklar)));
yon2=[yon2-1 yon2 yon2+1];
if(~isempty(find(yon1==yon2)))
    yon_listesi=yon2;
else
    yon_listesi=[yon2 yon1];
end
yon_listesi=yonTasmaKontrol(yon_listesi);

for j=1:length(yon_listesi)
    x=ust_bolge.x;
    y=ust_bolge.y;
    m_yonler=yon_listesi(j)*ones(1,length(yonler)+1);
    [ko_x,ko_y]=cevir2koordinat(m_yonler);
    x=x(end)*ones(length(m_yonler),1);
    y=y(end)*ones(length(m_yonler),1);
    x=x+ko_x';
    y=y+ko_y';
    alt_bolge = bolgeOlustur(x,y,im);
    alt_bolge_listesi(j)=alt_bolge;
end
tahmini_alt_bolge=[];
min_fark=256;
genel_toplam=1000;
for s=1:length(alt_bolge_listesi)
    alt_bolge=alt_bolge_listesi(s);
    if(isempty(alt_bolge.x))
        continue;
    end
    [alt_bolge, NR_sol,NR_sag]=kiyasla(ust_bolge,alt_bolge);
```

```
    sol_fark=(abs(ust_bolge.m_sol-alt_bolge.m_sol));
    sag_fark=(abs(ust_bolge.m_sag-alt_bolge.m_sag));
    min_deger=min(sol_fark,sag_fark);
    NR_farki=50-abs(NR_sol+NR_sag);
    toplam=sag_fark+sol_fark+NR_farki ;

    if(genel_toplam>toplam)
        min_fark=min_deger;
        genel_toplam= toplam;

tahmini_alt_bolge={yon_listesi(s),alt_bolge,NR_sol+NR_sag};
    elseif (genel_toplam==toplam)
        if(min_fark>min_deger)
            min_fark=min_deger;
            genel_toplam= toplam;

tahmini_alt_bolge={yon_listesi(s),alt_bolge,NR_sol+NR_sag};
        end
    end
end

secilen_alt_bolge=tahmini_alt_bolge{2};
secilen_yon=tahmini_alt_bolge{1};

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
% Alt bölgenin üst bölgeye benzerliği hesaplar ve alt bölge
bilgilerini
% günceller
% Parametreler:
% @ust_bolge: referans alınacak üst bölge
% @alt_bolge: referans bölge ile kıyaslanacak alt bölge
% Sonuç:
% @alt_bolge: üst bölgeye benzerliğine göre ortalama ve sapma
değeri güncellenen alt bölge
% @NR_sol: sol alt bölgenin sol üst bölgeye benzeyen pixel
sayısı
% @NR_sag: sağ alt bölgenin sağ üst bölgeye benzeyen pixel
sayısı
function [ alt_bolge,NR_sol,NR_sag ] = kiyasla(
ust_bolge,alt_bolge)
m_sag=ust_bolge.m_sag;
m_sol=ust_bolge.m_sol;
ro_sag=ust_bolge.ro_sag;
ro_sol=ust_bolge.ro_sol;
alt_sag=alt_bolge.sag;
alt_sol=alt_bolge.sol;

liste_sag=bolgeBul(alt_sag, m_sag, ro_sag);
liste_sol=bolgeBul(alt_sol, m_sol, ro_sol);
```

```
%bölgelerdeki sapma değerleri çok büyükse bölgede gürültü veya
birden fazla nesne var demektir
[sag_length,sag_ort,sag_ro]=lengthOrtRo(liste_sag);
[sol_length,sol_ort,sol_ro]=lengthOrtRo(liste_sol);

NR_sag =sag_length;
alt_bolge.m_sag = sag_ort;
alt_bolge.ro_sag = sag_ro;

NR_sol = sol_length;
alt_bolge.m_sol = sol_ort;
alt_bolge.ro_sol = sol_ro;
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
% Verilen kenarın çift tarafında bölgeler oluşturur
% Parametreler:
% @x: kenar x koordinatları
% @y: kenar y koordinatları
% @im: incelenen görüntü
% Sonuç:
% @bolge: sağ-sol bölgeler ve kenar bilgisi
function bolge = bolgeOlustur(x,y,im)
bolge=struct('m_sol',[],'m_sag',[],...
    'ro_sol',[],'ro_sag',[],...
    'sol',[],'sag',[],...
    'deltax',[],'deltay',[],...
    'x',[],'y',[]);
gercek_x=x; gercek_y=y;
deltay=y-circshift(y,1);
deltax=x-circshift(x,1);
deltax(1)=[]; deltay(1)=[];
x(end)=[]; y(end)=[];
sagy=[];sagx=[];soly=[];solx=[];

%kenara dik olan bölgeleri oluştur
for i=1:5
    sagy=[sagy,round(-i*deltax)];
    sagx=[sagx,round(i*deltay)];
    soly=[soly,round(i*deltax)];
    solx=[solx,round(-i*deltay)];
end

sagx_kontrol=[]; sagy_kontrol=[];
solx_kontrol=[]; soly_kontrol=[];

for i=1:5
    sagx_kontrol=[sagx_kontrol x+sagx(:,i)];
    sagy_kontrol=[sagy_kontrol y+sagy(:,i)];
    solx_kontrol=[solx_kontrol x+solx(:,i)];
    soly_kontrol=[soly_kontrol y+soly(:,i)];
end
```

```
% alınan bölgerin resim sınırlarında taşma olup olmadığı
kontrolü
tasma_kontrol=length(find(sagx_kontrol > size(im,2) |
sagy_kontrol > size(im,1)...
    | solx_kontrol > size(im,2)|soly_kontrol > size(im,1)...
    | sagx_kontrol <= 0 | sagy_kontrol <= 0| solx_kontrol <=
0|soly_kontrol <= 0));
if(~tasma_kontrol)
    %     Debug: Seçilen bölgeleri resim üzerine işaretler
    %     for i=1:5
    %         hold on; plot(x+sagx(:,i),y+sagy(:,i),'.b');
    %         hold on; plot(x+solx(:,i),y+soly(:,i),'.y');
    %     end
    en=size(im,1);
    sol=[];sag=[];
    %bolgelerdeki pixel yoğunluklarını al
    for  i=1:5
        sol=[sol,im((x+solx(:,i)-1)*en+y+soly(:,i))];%lineer
koordinata göre işlem yapar
        sag=[sag,im((x+sagx(:,i)-1)*en+y+sagy(:,i))];
    end

    bolge.sol=double(sol);
    bolge.sag=double(sag);
    bolge.m_sol=mean(sol(:));
    bolge.ro_sol=std(sol(:));
    bolge.m_sag=mean(sag(:));
    bolge.ro_sag=std(sag(:));
    bolge.deltay=deltay;
    bolge.deltax=deltax;
    bolge.x=gercek_x;
    bolge.y=gercek_y;
end
end
```

ÖZGEÇMİŞ

Ad Soyad: Özlem MUTLU

Doğum Yeri ve Tarihi: Van - 1988

E-Posta: ozlemmutlu2011@gmail.com

Lisans: Yalova Üniversitesi – Bilgisayar Mühendisliği

Yüksek Lisans : Yalova Üniversitesi – Bilgisayar Mühendisliği

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- **Mutlu, Ö., İskurt, A., Bayraktar, H. K.,** 2016, Reinforcement of Edge Detection by Concurrent Segmentation on Images with Elongated Structures, 2nd International Congress of Technology, Management and Social Sciences-16, 19, 1-5.