Kenar Tabanlı Düzey Kümeleriyle Tıbbi İmge Bölütleme İçin Mobil Uygulama

Mobile Application for Medical Image Segmentation with Edge Based Level Sets

Bekir Dizdaroğlu, Asiye Kengil, Balkıza Topaloğlu

Bilgisayar Mühendisliği Bölümü
Karadeniz Teknik Üniversitesi
bekir@ktu.edu.tr, asiye_2929@hotmail.com, mcrsb@hotmail.com

Özetçe

Bu bildiride, basit yapılı tıbbi imgelerin bölütlenmesi için Android tabanlı mobil telefonlarda çalıştırılabilecek bir uygulama geliştirilmiştir. Bölütleme yöntemi olarak kenar tabanlı bir düzey küme yaklaşımı kullanılmıştır. Bu yöntem, diğer düzey küme yöntemleriyle karşılaştırma yapıldığında, merkezi sonlu farklar gibi basit türev yaklaşımları içerdiğinden ve sonuç imgesini doğruluklu bir şekilde ürettiğinden tercih edilmiştir. Android işletim sisteminin tercih edilmesinin sebebi ise, piyasadaki kullanıcıların çoğunun bu işletim sistemli mobil cihazları kullanmasından dolayıdır. Başka bir ifadeyle, Android mobil cihazlar daha ucuzdur ve yazılım geliştirmek için ücretsiz bir platform ve ayrıca açık kaynak kod olanağı sunulmaktadır. Bu çalışmada uygulamanın başarımı mobil cihazlarda test edilmiştir ve basit yapılı imgelerin bölütlendiği görsel olarak kanıtlanmıştır.

Abstract

In this paper, an application that can run on Android based mobile phones is developed for segmentation of simple structured medical images. An edge-based level set approach is used as a segmentation method. This method is preferred since it contains simple differentiation schemes such as central finite differences and it generates properly the resulting image when compared with other level set methods. The reason for preferring the Android operating system is that the majority of customers in the market use of mobile devices with this operating system. In other words, Android mobile devices are cheaper, and a free platform for developing software and also open source codes are on offer. In this study, the performance of application on mobile devices has been tested and segmentation of simple structured images has been proved visually.

1. Giriş

Tıbbi imge işleme alanında en önemli konulardan bir tanesi de imge bölütleme işlemidir. Literatürede imge bölütleme işlemi için birçok yöntem geliştirilmiştir. Ama çoğu yöntemler, imgenin yapı ve doku bilgisinin karmaşıklık içermesi ya da imgedeki yeğinlik dağılımının homojen olmaması durumunda başarılı sonuçlar üretememektedir. Bu yüzden tıbbi imge işleme alanında, patolojik bölgelerinin analiz edilmesi için ve başka bir ifadeyle hastalıkların teşhis edilmesi için, klasik yaklaşımlarla karşılaştırma yapıldığında, düzey küme yaklaşımları daha fazla matematiksel formüller içermesine ve genelde işlem süreleri biraz daha fazla olmasına rağmen doğruluklu sonuç üretmeleri açısından tercih edilmektedir. Ayrıca teknolojinin gelişimine paralel olarak mobil cihazlar da günlük hayatta oldukça fazla kullanım alanı bulmaktadır.

Bu bildiride de kenar tabanlı düzey küme yaklaşımıyla basit yapılı tıbbi imgelerin bölütlenmesi için mobil tabanlı bir uygulama geliştirilmiştir. Uygulamada, açık kaynak kodlu Android işletim sisteminin tercih edilmesinin sebebi piyasada kullanılan mobil cihazların daha çok bu işletim sistemiyle çalışmasından dolayıdır.

2. Android İşletim Sistemi

Android işletim sistemi, mobil cihazlar için geliştirilmiş ilk açık kaynak kodlu ve ücretsiz bir platformdur. Bu işletim sisteminin geliştirilmesi için *Google* başta olmak üzere birçok ticari şirket çalışmaktadır. Diğer mobil cihazlarla karşılaştırma yapıldığında, geliştirme ortamının açık platformlu ve ücretsiz oluşu müşterilere sunulan ürünlerin daha ucuz olmasını sağlamaktadır. Bu yüzden piyasadaki birçok mobil cihazlar Android işletim sistemli tabanlıdır. Ayrıca açık platformlu oluşundan dolayı yazılım geliştiriciler de yeni uygulamaları daha hızlı ve kolay bir şekilde yazabilmektedir.

2.1. Sistem Mimarisi

Android işletim sistemi, en üst katmanındaki Linux çekirdeği üzerine inşa edilmiştir. Bir sonraki katman C programlama diliyle yazılmış API (Application Programming Interface-Uygulama Programlama Arayüzü) destekli kütüphaneleri içermektedir. Aynı katmanda yürütüm yazılımı bulunmaktadır. Bu katman Java baytkodlarından dönüştürülen "*.dex*" uzantılı dosyaları kullanan Dalvik sanal makinesini içermektedir. Uygulama çatısında ise Java programlama diliyle yazılmış bir araç takımı kullanılmaktadır.

2.2. Uygulama Bileşenleri

Uygulamalar dört ana program modülünden oluşmaktadır [1]:

i. Etkinlik: Kullanıcı görsel arayüzüdür.
ii. Yayın alıcı: Akıllı telefonun kapatılması gibi dış bir olay tarafından tetiklendiğinde koşulan kayıtlı koddur.
iii. Servis: Arkaplanda çalışan programdır.

iv. İçerik sağlayıcı: Diğer uygulamalarla veri paylaşımını sağlar.

Ayrıca "*AndroidManifest.xml*" dosyası uygulama çalıştırıldığında hangi etkinliğin başlatılacağını tanımlar ve kamera erişimi gibi bazı izinlerin bildirimini yapar.

Herhangi bir Android yazılım demeti inşa edildiğinde "*.apk*" uzantılı bir dosya oluşturulur. Uygulamanın çalıştırması, bu dosyanın mobil cihaza yüklenmesiyle gerçekleştirilir.

3. Kenar Tabanlı Bölütleme Yaklaşımı

Kenar tabanlı düzey kümeleriyle imge bölütleme yaklaşımlarında genelde üç ana terimden oluşan enerji fonksiyoneli kullanılır:

$$E(\Phi) = \mu R(\Phi) + \lambda L(\Phi) + \alpha A(\Phi) \quad (1)$$

Denklemde, $R(.)$, $L(.)$ ve $A(.)$ sırasıyla düzenlileştirme terimini, uzunluk terimini ve alan terimini ifade etmektedir. μ, λ ve α ise ilgili terimler için ağırlık katsayılarıdır. Bu terimler için aşağıda verilen entegrallerin Euler-Lagrange yaklaşımına bağlı bayır inişi iteratif yöntemiyle düzey küme fonksiyonu dikkate alınarak minimize edilmesi gerekmektedir:

$$R(\Phi) = \int_\Omega \mathrm{p}(\Phi)\mathrm{d}\Omega$$
$$L(\Phi) = \int_\Omega \mathrm{g}\delta_\varepsilon(\Phi)|\nabla\Phi|\,\mathrm{d}\Omega \quad (2)$$
$$A(\Phi) = \int_\Omega \mathrm{g}H_\varepsilon(-\Phi)\mathrm{d}\Omega$$

$R(.)$ düzenlileştirme terimi bağlı minimizasyon işlemi

$$\frac{\partial\Phi_\mathrm{R}}{\partial t} = \mu\mathrm{div}(\mathrm{d}(|\nabla\Phi|)\nabla\Phi) \quad (3)$$

olarak elde edilir [2]. Denklemdeki, $\mathrm{d}(|\nabla\Phi|) = 1 - |\nabla\Phi|^{-1}$ fonksiyonuyla Φ düzey küme fonksiyonu yönbağımlı bir şekilde yumuşatılmaktadır. $\mathrm{d}(\cdot)$ fonksiyonu, $\mathrm{p}(\cdot)$ potansiyel fonksiyonuna bağlı olarak aşağıdaki gibi alınmaktadır [2]:

$$\mathrm{d}(x) = x^{-1} \times (\partial\mathrm{p}(x)/\partial x)$$

Burada, potansiyel fonksiyonu olarak aşağıdaki eşitlik kullanılabilir [2]:

$$\mathrm{p}(x) = \frac{1}{2}(x-1)^2 \quad (4)$$

İmgedeki kenar bilgisinin elde edilmesi için gradyan vektör alanından yararlanılmaktadır:

$$\mathrm{g} = (1 + |\nabla(\mathrm{G}_\sigma * \mathrm{f})|^2)^{-1} \quad (5)$$

Uzunluk teriminin minimizasyon işlemi

$$\frac{\partial\Phi_\mathrm{L}}{\partial t} = \lambda\delta_\varepsilon(\Phi)\mathrm{div}\left(\mathrm{g}\frac{\nabla\Phi}{|\nabla\Phi|}\right) \quad (6)$$

olarak bulunur [2].

Alan teriminin minimizasyon işlemi ise,

$$\frac{\partial\Phi_\mathrm{A}}{\partial t} = \alpha\delta_\varepsilon(\Phi)\mathrm{g} \quad (7)$$

olarak elde edilir.

Φ düzey küme fonksiyonununa ilk değer ataması yapmak için iki düzeyli bir fonksiyon kullanılmaktadır [2]:

$$\Phi_{ilklendirme} = \begin{cases} -c_0 & \Omega_0 \text{ bölgesinde} \\ c_0 & \Omega\backslash\Omega_0 \text{ bölgesinde} \end{cases} \quad (8)$$

Burada, Ω_0 kullanıcı tarafından başlangıçta işaretlenen bir bölge ve c_0 sabit bir değerdir.

Alan terimindeki α katsayısının işaretine göre, Φ düzey küme fonksiyonunun sıfır düzey çevriti ya genişletilir ya da daraltılır.

H_ε yumuşatılmış birim basamak fonksiyonu ve $\delta_\varepsilon = H'_\varepsilon$ yumuşatılmış birim dürtü fonksiyonu için aşağıdaki eşitlikler alınmıştır [2]:

$$H_\varepsilon(x) = \begin{cases} \frac{1}{2}\left[1 + \frac{\pi}{\varepsilon} + \frac{1}{\pi}\sin\left(\frac{\pi x}{\varepsilon}\right)\right], & |x| \le \varepsilon \\ 1, & x > \varepsilon \\ 0, & x < -\varepsilon \end{cases} \quad \text{ve}$$
$$\delta_\varepsilon(x) = \begin{cases} \frac{1}{2\varepsilon}\left[1 + \cos\left(\frac{\pi x}{\varepsilon}\right)\right], & |x| \le \varepsilon \\ 0, & |x| > \varepsilon \end{cases} \quad (9)$$

Burada ε yumuşatma sabitidir.

Bütün yukarıda verilen ifadeler dikkate alındığında Φ küme fonksiyonunun minimizasyonu için aşağıdaki ifade verilebilir:

$$\frac{\partial\Phi}{\partial t} = \frac{\partial(\Phi_\mathrm{R} + \Phi_\mathrm{L} + \Phi_\mathrm{A})}{\partial t} \quad (10)$$

Bölütleme işleminde düzey küme fonksiyonunu en uygun bir şekle sokma işlemi içinse aşağıdaki ifade kullanılır:

$\Phi_{(t+1)} = \Phi_{(t)} + \tau \times \partial\Phi_{(t)}/\partial t$, burada τ zaman adımı sabitidir. $\mathbf{n} = (u, v)^T$, imge sınırına dik bir birim vektör olmak üzere, Denklem (10), $\frac{\partial\Phi}{\partial\mathbf{n}} = \nabla\Phi \cdot \mathbf{n} = u\frac{\partial\Phi}{\partial x} + v\frac{\partial\Phi}{\partial y} = 0$ eşitliğine bağlı Neumann sınır koşullarına göre çözülmektedir. Bu durumda, imge sınırında $\frac{\partial\Phi}{\partial x} = 0$ ve $\frac{\partial\Phi}{\partial y} = 0$ eşitliklerinin sağlanması yeterli olmaktadır. Kısmi türev alma işleminde ise Taylor serisi açılımından yararlanılmaktadır.

4. Deneysel Çalışmalar

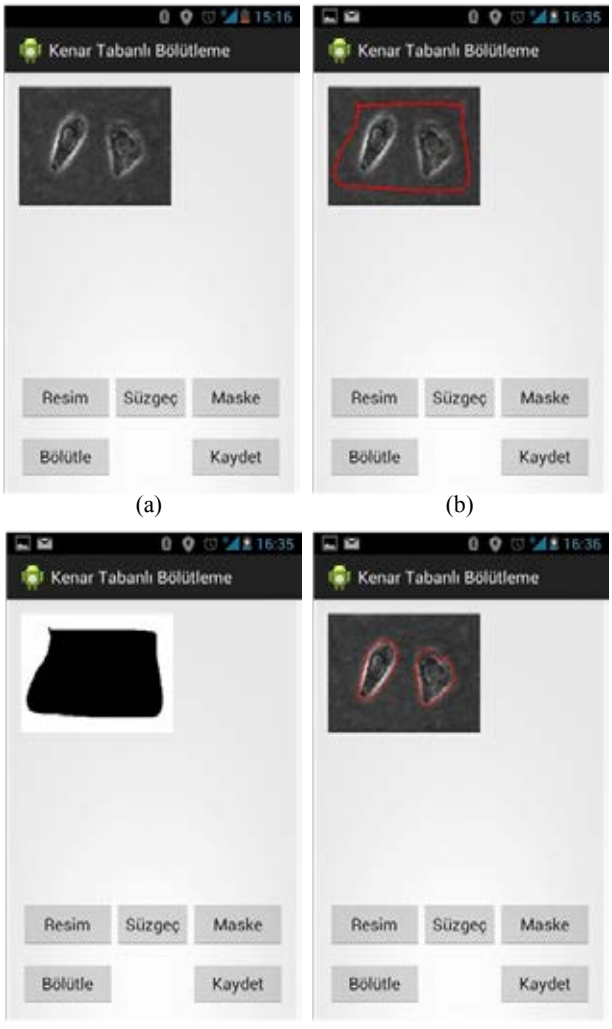
Android işletim sistemine bağlı bir uygulama, Eclipse ve Android SDK (Software Development Kit-Yazılım Geliştirme Aracı) kullanılarak yazılmıştır. Uygulamanın Android öykünücüdeki çalışması denendikten sonra, uygulamaya ait "*.apk*" uzantılı dosya mobil cihaza aktarılmıştır. Uygulamadaki işlem adımları aşağıda verilmiştir:

i. Girdi imgesini al.
ii. Girdi imgesini Gauss süzgecinden geçir.
iii. Bölütlenecek bölgeyi, nesnenin dış veya iç kısmından kabaca seç/işaretle.
iv. Seçilen bölgeye bir maske imgesi uydur ve maske imgesine göre düzey küme fonksiyonuna ilk değer ataması yap.
v. Bölütleme işlemini gerçekleştir.

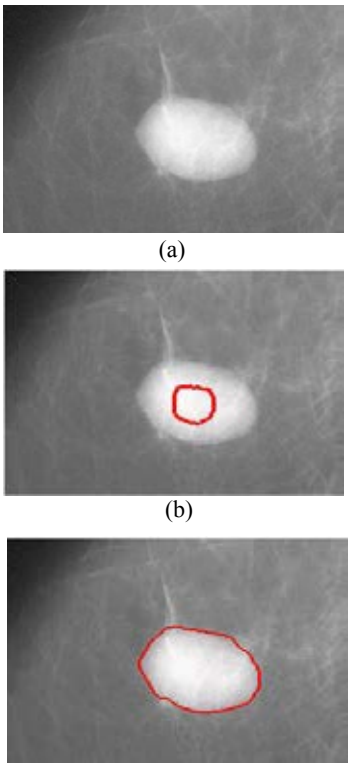
Test işlemleri için 249×195 boyutlarında bir mikroskop görüntüsü ve 241×202 boyutlarında bir ultrason görüntüsü kullanılmıştır.

Yöntemde, [2]'deki çalışmada önerilen parametre değerlerine bağlı olarak $\tau = 5$, $\mu = 0.2/\tau$, $\lambda = 5$, $\alpha = \pm 3$, $\varepsilon = 1.5$, $c_0 = 2$ ve gürültü azaltma işleminde kullanılan Gauss süzgecinin standart sapması ise $\sigma = 1.5$ alınmıştır. İterasyon sayısı ise 100 değerine setlenmiştir.

Şekil 1.'de uygulamaya ait ekran görünümleri verilmiştir. Şekil 1.d'den görülebileceği gibi basit yapılı test görüntüsünün bölütlenmesi başarılı bir şekilde gerçekleştirilmiştir. Şekil 1.d'deki görüntü bölütlemesi için mobil tabanlı bir cihazdaki işlem süresi yaklaşık 30 saniyedir. Şekil 2.'de ise bir ultrason görüntüsüne ait bölütleme sonucu verilmiştir.



Şekil 1: Mobil uygulamanın ekran görünümleri: (a) İki hücreye ait bir mikroskobik test imgesi, (b) kullanıcı tarafından nesnenin kabaca seçilmesi, (c) seçime uydurulan maske imgesi ve (d) bölütlenmiş sonuç imgesi.



Şekil 2: Göğüs bölgesinden tıbbi görüntüleme tekniğiyle üretilmiş bir ultrason görüntüsünün bölütlenmesi: (a) test imgesi, (b) kullanıcı tarafından patolojik bölgenin kabaca işaretlenmesi ve (b) bölütlenmiş sonuç imgesi.

Uygulamaya ait bazı kod parçaları Ek 1.'de verilmiştir.

5. Sonuçlar

Bu çalışmada basit yapılı tıbbi imgelerin bölütlenesi için mobil tabanlı bir yazılım geliştirilmiştir. İleriki çalışmalarda, sunucu bilgisayara bağlanarak veritabanından seçilecek bir tıbbi imgedeki herhangi bir patolojik bölgenin mobil bir cihazda kullanıcı etkileşimiyle kabaca işaretlenmesi ve bölütleme işleminin sunucu bilgisayarda gerçekleştirilerek akıllı telefona aktarılması için bir uygulama geliştirilecektir.

6. Kaynakça

[1] Billiauwas, I., and Borjean, K., "Image recognition on an Android mobile phone", *Master Thesis*, De Nayer Institute, Netherlands, 2009.

[2] Li, C., Xu, C., Gui, C., and Fox, M. D., "Distance regularized level set evolution and its application to image segmentation", *IEEE Trans. Image Process.*, vol. 19, no. 12, pp. 3243–3254, 2010.

Ek 1: Uygulamaya ait Java programlama dilinde yazılmış bazı kod parçaları.

```
//width = inputImage.getWidth(); height = inputImage.getHeight();
//Gradyan vektörünü merkezi farklar yaklaşımına göre hesaplama
public void getGradient(double[] input, double[] inputx, double[] inputy) {
 for(int y=1; y<height-1; y++) for(int x=1; x<width-1; x++) {
       inputx[x+width*y]=(input[x+1+width*y]  - input[x-1+width*y])/2;
       inputy[x+width*y]=(input[x+width*(y+1)]- input[x+width*(y-1)])/2;
 } }
```

```
//Laplas işleci
public void getLaplace(double[] input, double[] output){
 for(int y=1; y<height-1; y++) for(int x=1; x<width-1; x++)
  output[x+width*y]= input[x+1+width*y]+input[x-1+width*y]+
                        input[x+width*(y+1)]+input[x+width*(y-1)]-
                           4*input[x+width*y];
}
//Birim vuruş fonksiyonunu hesaplama
public void getDiracDelta(double[] fi, double[] diracDelta, double epsilon) {
 double temp = 1.0/(2*epsilon);
 for(int y=0; y<height; y++) for(int x=0; x<width; x++)
  if( fi[x+width*y] <= epsilon &&  fi[x+width*y] >= -epsilon)
   diracDelta[x+width*y]=temp*(1+Math.cos(Math.PI*fi[x+width*y]/ epsilon));
  else diracDelta[x+width*y]=0;
}
//Neuman sınır koşullarını hesaplama
public void NeumannBoundaryCondition(double[] fi) {
 fi[0+width*0] = fi[2+width*2];
 fi[width-1+width*0] = fi[width-3+width*2];
 fi[0+width*(height-1)] = fi[2+width*(height-3)];
 fi[width-1+width*(height-1)] = fi[width-3+width*(height-3)];
 for(int x=1; x<width-1; x++){
  fi[x+width*0] = fi[x+width*2];
  fi[x+width*(height-1)] = fi[x+width*(height-3)];}
 for(int y=1; y<height-1; y++) {
  fi[0+width*y] = fi[2+width*y];
  fi[width-1+width*y]= fi[width-3+width*y]; }
}
//Kenar tabanlı düzey küme bölütleme işlemi
public void EdgeEvolution(double[] fi, double[] g, double lambda, double mu, double alfa,
                          double epsilon, double timestep, int numIter) {
 double[] K,laplace,diracDelta,gx,gy,fix,fiy,tempx,tempy,terms = 0,denominator = 0;
 //Dinamik bellekte yerel değişkenlere yer ayırma ve içerik sıfırlama
 //Burada bazı kodlar çıkartıldı…
 getGradient(g,gx,gy); //Gradyan vektörü hesapla
 for(int k=0; k<numIter; k++){
  NeumannBoundaryCondition(fi); //Düzey küme fonk.nu için Neumann sınır koşulları
  getGradient(fi,fix, fiy);
  for(int y=0; y<height; y++) for(int x=0; x<width; x++){
    denominator = Math.sqrt(fix[x+width*y] * fix[x+width*y] +
                  fiy[x+width*y] * fiy[x+width*y]+1e-10);
    fix[x+width*y] /= denominator; //Düzey küme fonk.nunun Gradyan  vektörünü
    fiy[x+width*y] /= denominator; //normalize etme
  }
 getGradient(fix,tempx, tempy);
 for(int y=0; y<height; y++) for(int x=0; x<width; x++)
   K[x+width*y]=tempx[x+width*y];
 for(int y=0; y<height; y++) for(int x=0; x<width; x++)
   tempx[x+width*y] = tempy[x+width*y] = 0;
 getGradient(fiy,tempx,tempy);
 for(int y=0; y<height; y++) for(int x=0; x<width; x++)
   K[x+width*y] += tempy[x+width*y];
 getDiracDelta(fi,diracDelta,epsilon); //Birim vuruş fonksiyonu
 getLaplace(fi,laplace); //Laplas hesaplama
 for(int y=0; y<height; y++) for(int x=0; x<width; x++){
     terms = mu * (laplace[x+width*y]-K[x+width*y])+         //Düzenlileştirme terimi
             lambda * diracDelta[x+width*y] *
                    (gx[x+width*y] * fix[x+width*y]+y[x+width*y]*fiy[x+width*y]+
                     g[x+width*y]*K[x+width*y])+             //Uzunluk terimi
             alfa * g[x+width*y]*diracDelta[x+width*y];      //Alan terimi
     fi[x+width*y]+=timestep*terms;//Düzey küme fonksiyonunu optimize etme
  }
 }
}
```