GÖRÜNTÜ İŞLEME

Süperpiksel Bölütleme Uygulaması

CEREN
YAŞAR
14011020

Yöntem

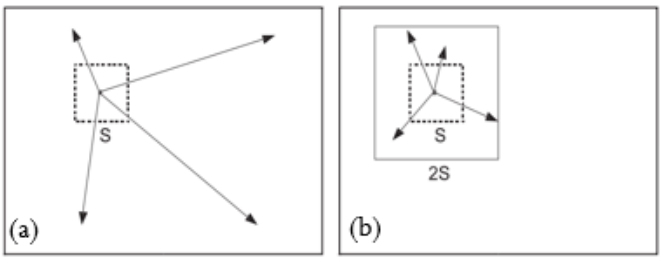
Süper piksel yönteminde her süperpixel grubu için belirlenen merkezlerin SxS alandaki pixellerinin merkezlerle olan mesafelerini aşağıdaki şekildeki gibi hesaplıyoruz.

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2}$$

$$S = \sqrt{N / K}$$



(a) standard k-means searches the entire image. (b) SLIC searches a limit region.

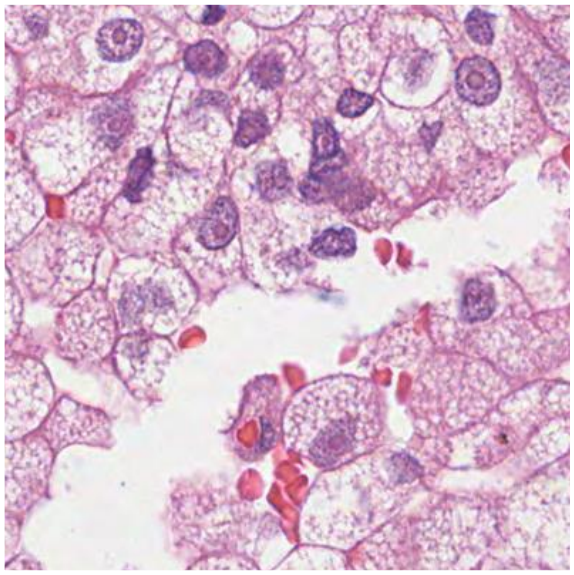
Bunu önce RGB değerler için gerçekleştirip yeni gruplamalar yapıyoruz. Sonra her grubun yeni merkezleri hesaplanıyor ve bu merkezlerin eski ve yeni değerleri arasındaki fark belirli bir değerin altındaysa La*b* değerleri için de aynı işlemler gerçekleştiriliyor.

En son her süper piksel grubunun parlaklık ortalaması o grubun tüm piksellerine yazılıyor.

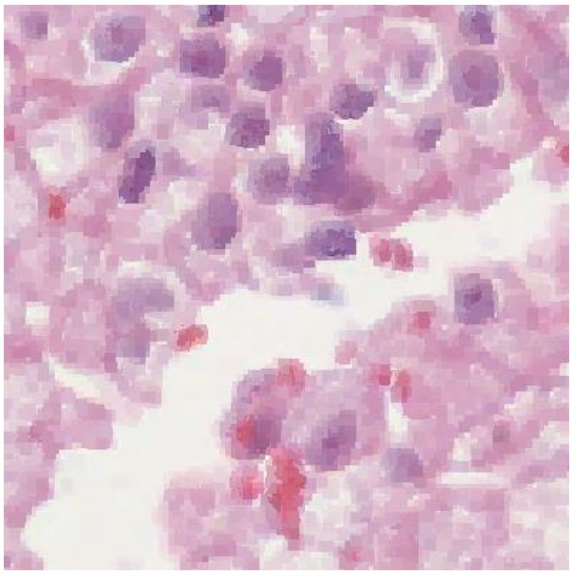
Uygulama

10100_11400

Orijinal

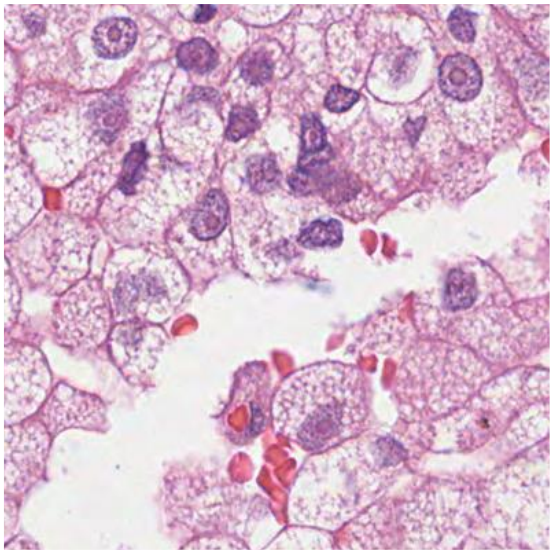


M=1 Süperpiksel=10000

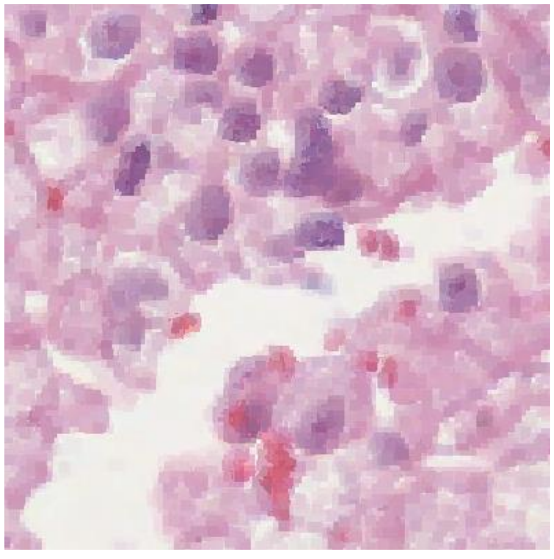


Süperpixel Bölütlemesi sonucunda resim segmentlere ayrıldı.

Orijinal

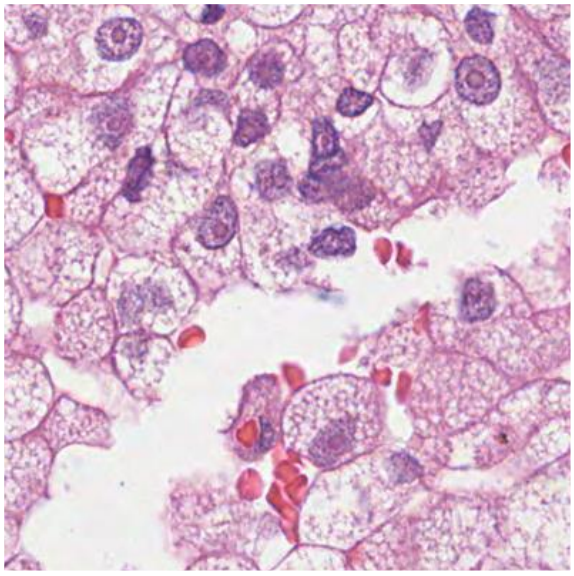


M=2 Süperpiksel=10000

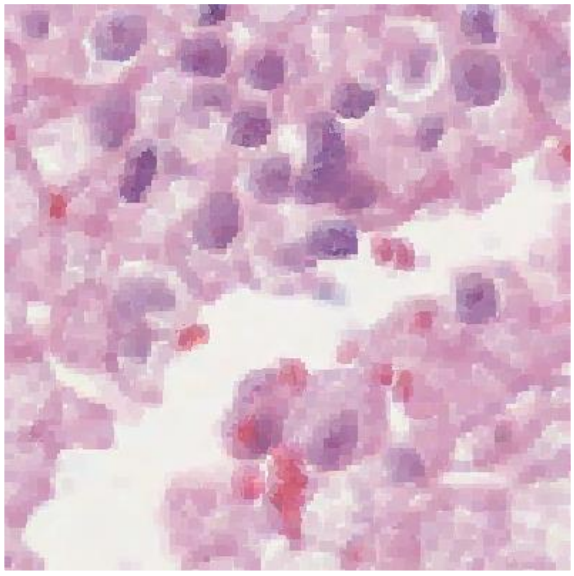


Önceki M değerine göre gözle görülür bir değişiklik yok.

Orijinal

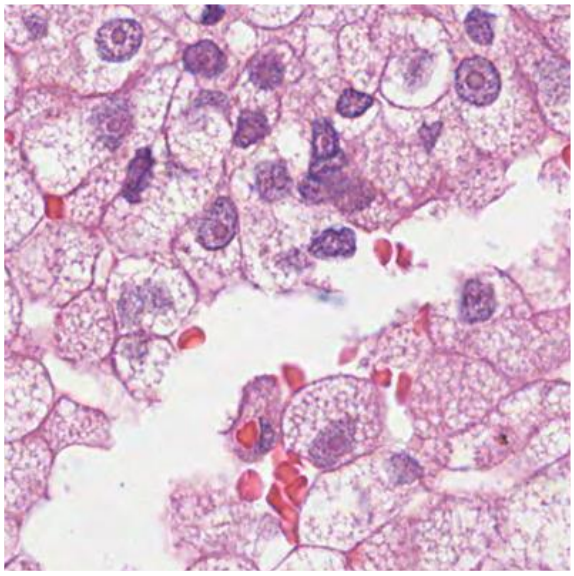


M=5 Süperpiksel=10000

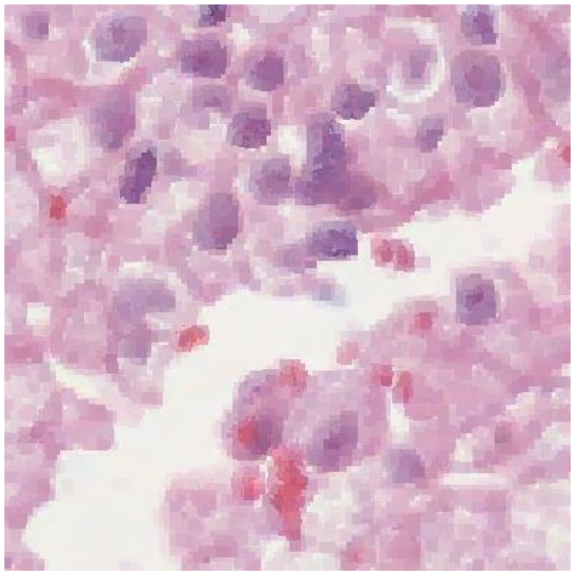


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

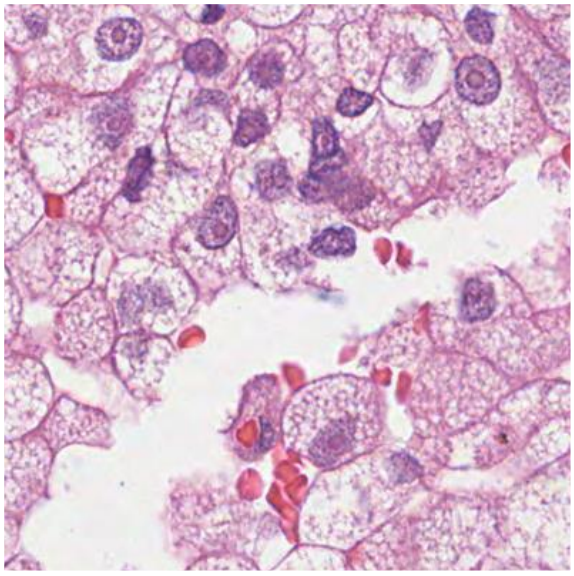


M=10 Süperpiksel=10000

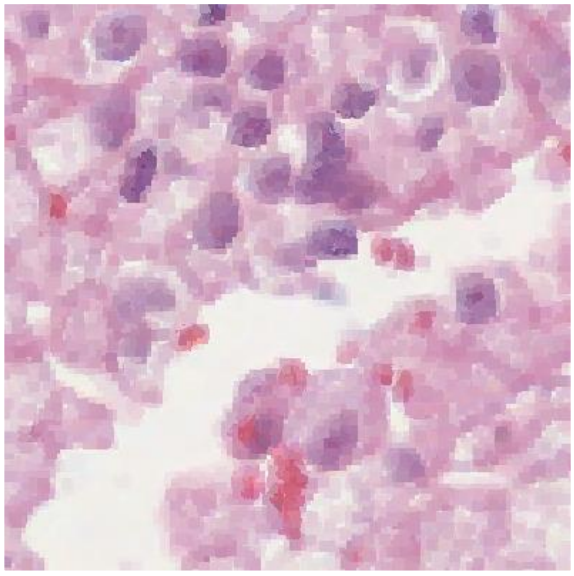


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

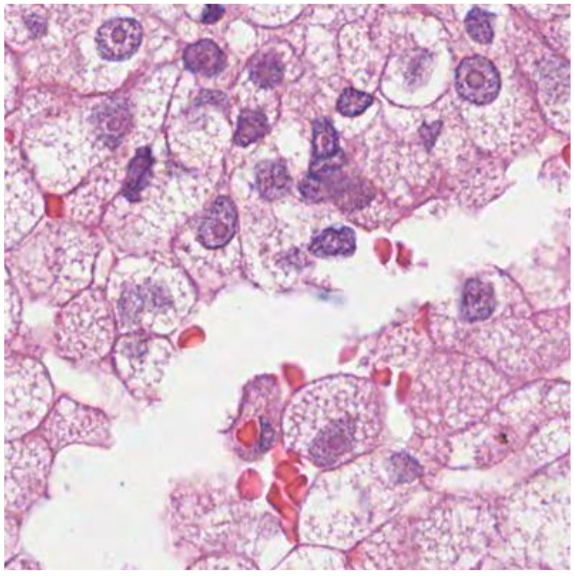


M=20 Süperpiksel=10000

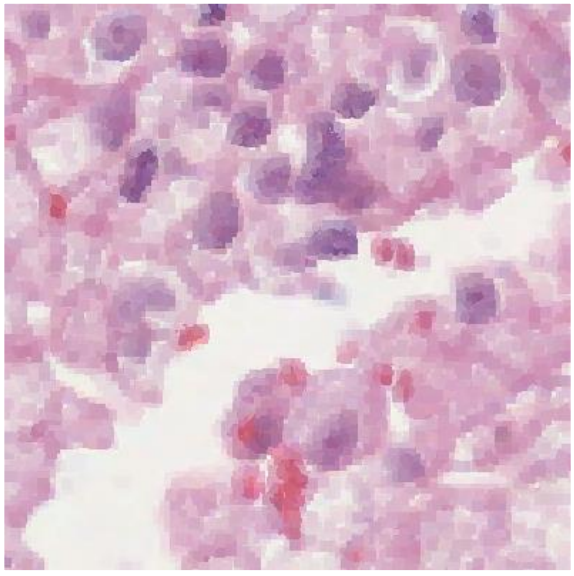


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

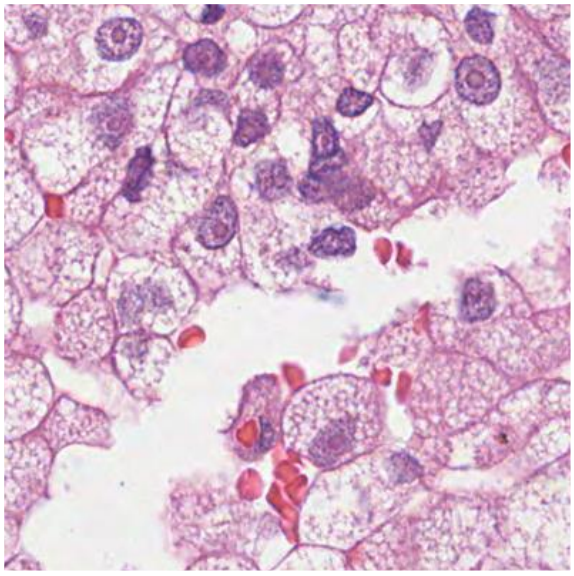


M=30 Süperpiksel=10000

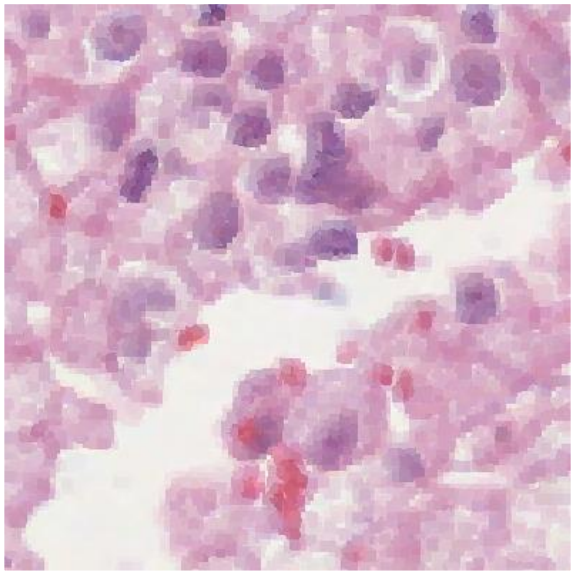


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal



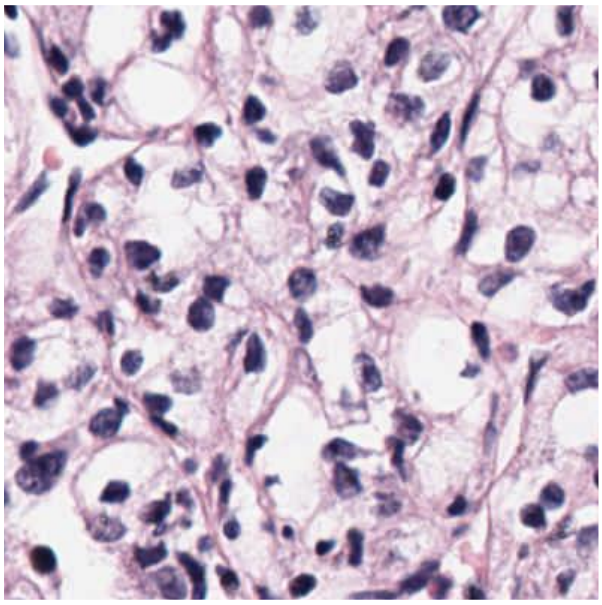
M=40 Süperpiksel=10000



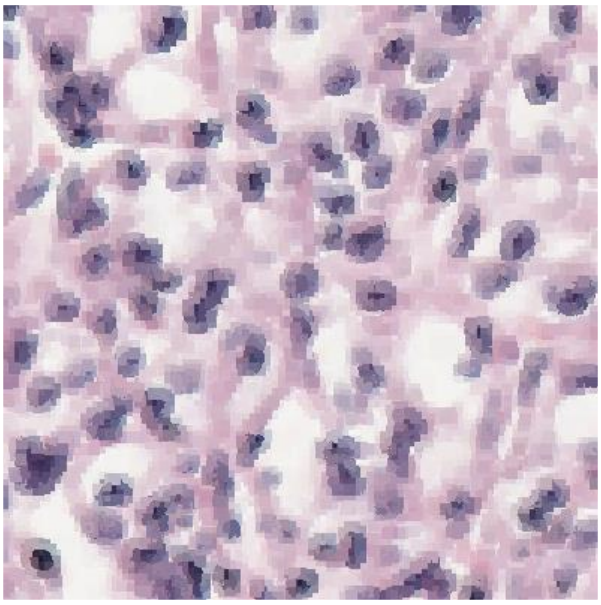
Önceki M değerine göre gözle görülür bir değişiklik yok.

14000_21400

Orijinal

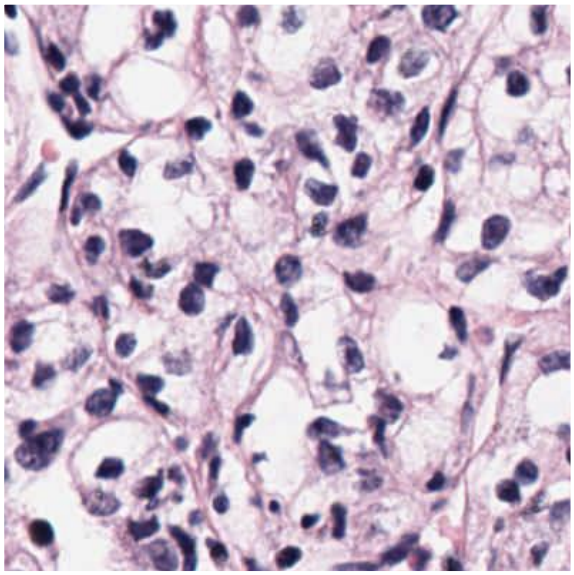


M=1 Süperpiksel=10000

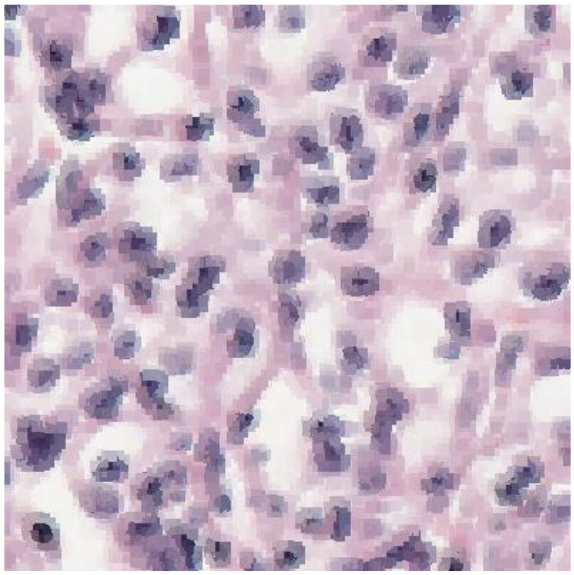


Süperpixel Bölütlemesi sonucunda resim segmentlere ayrıldı.

Orijinal

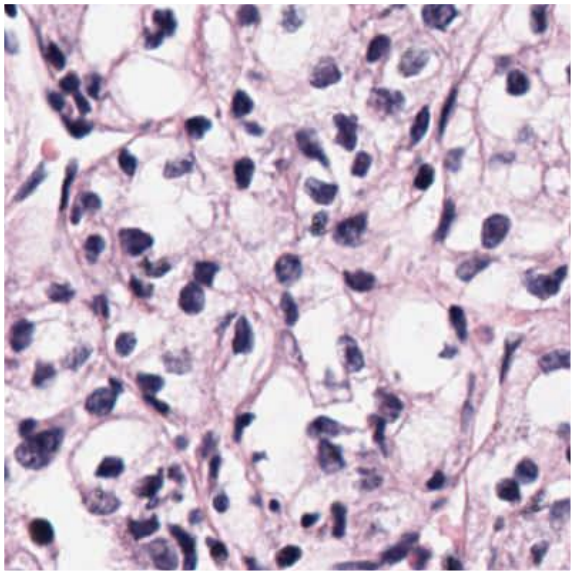


M=2 Süperpiksel=10000

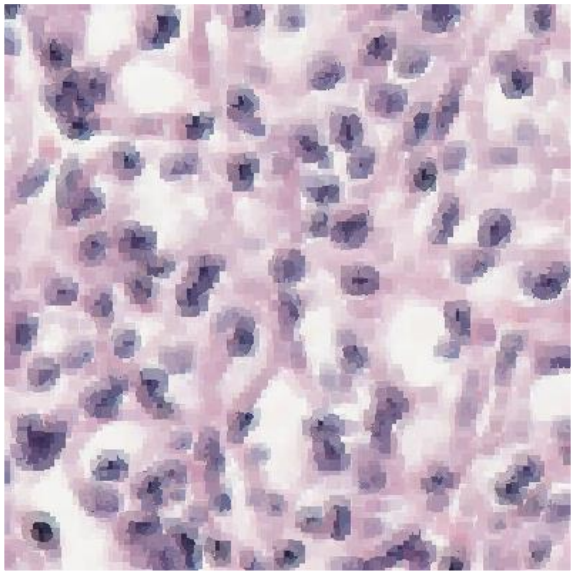


Önceki M değerine göre gözle görülür bir değişiklik yok.

Orijinal

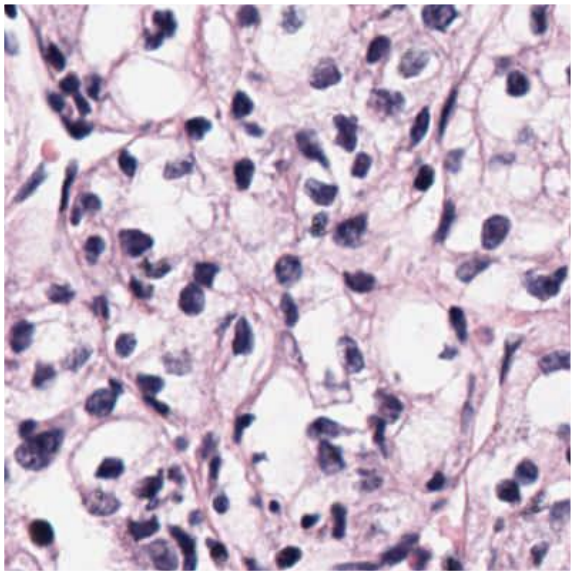


M=5 Süperpiksel=10000

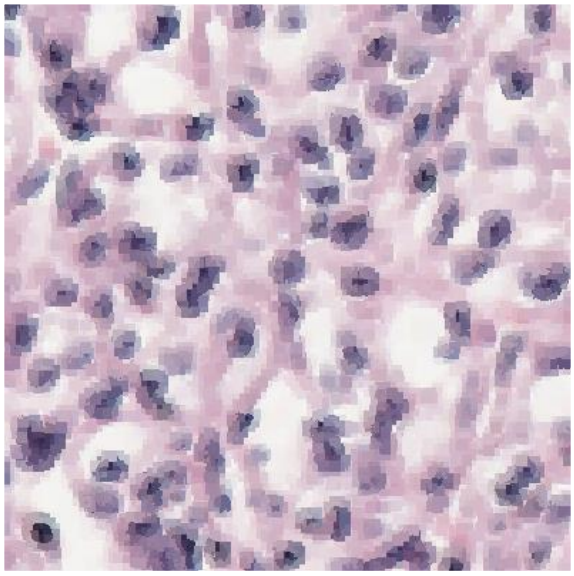


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

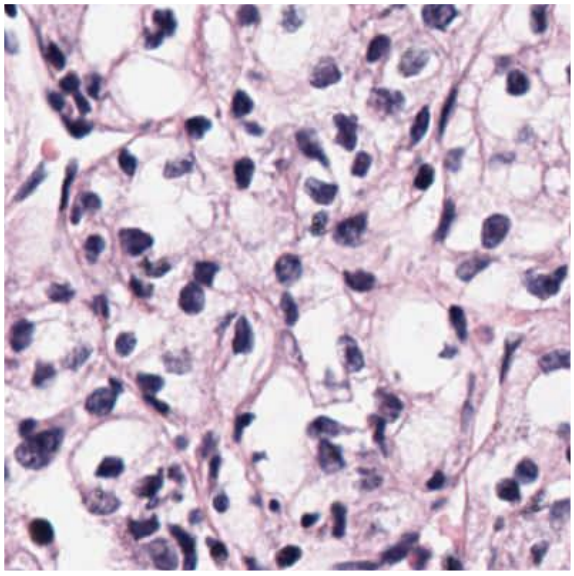


M=10 Süperpiksel=10000

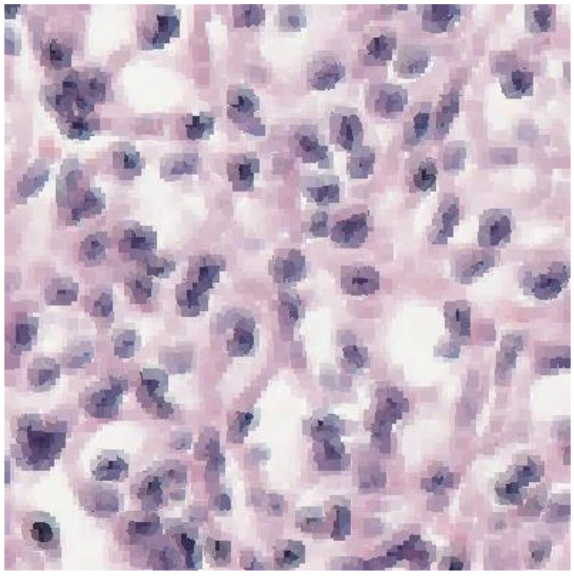


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

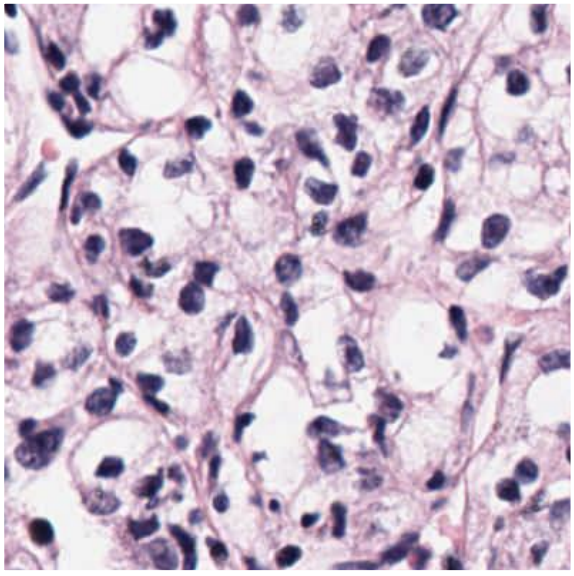


M=20 Süperpiksel=10000

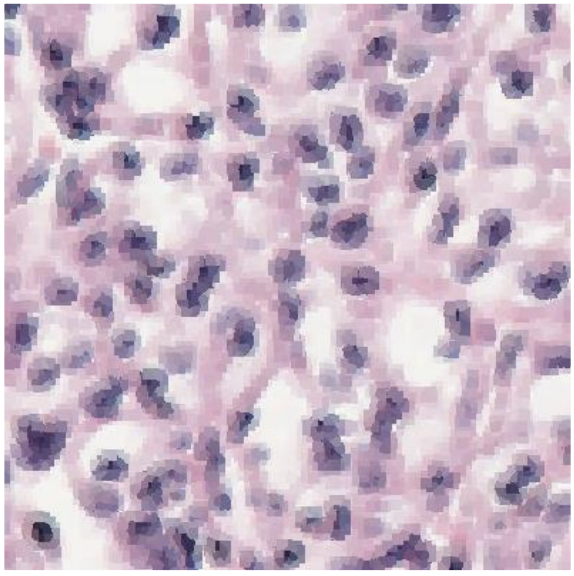


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

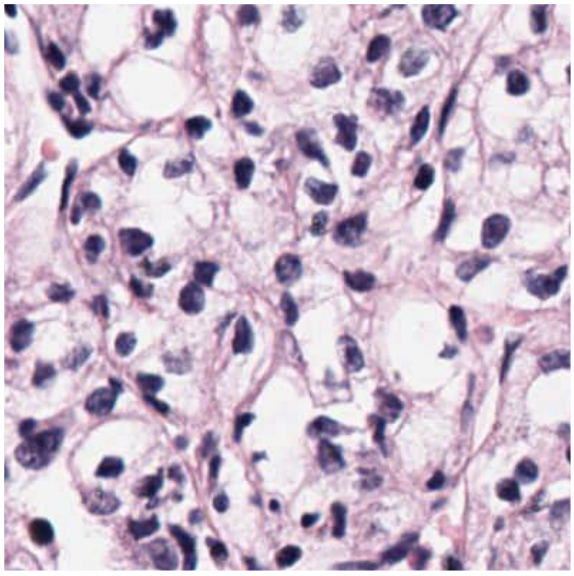


M=30 Süperpiksel=10000

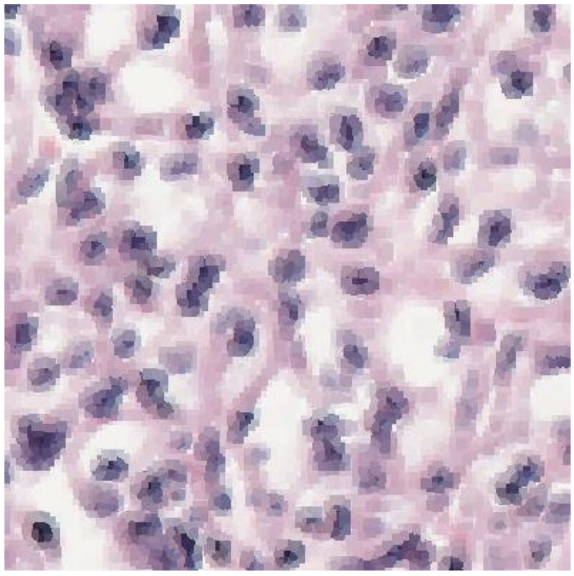


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal



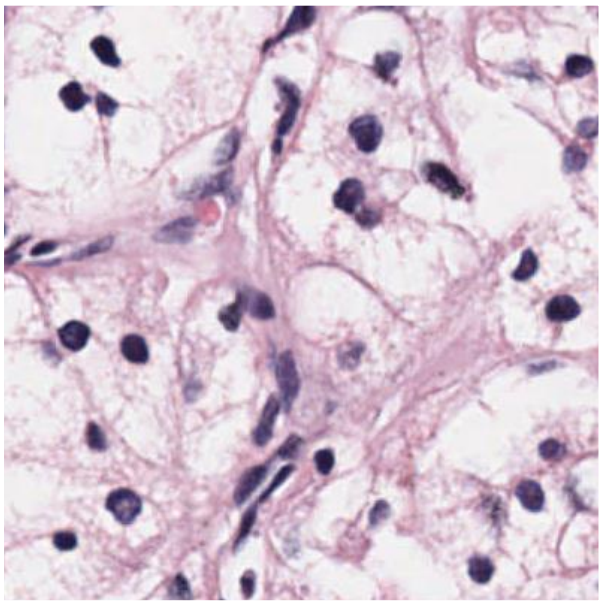
M=40 Süperpiksel=10000



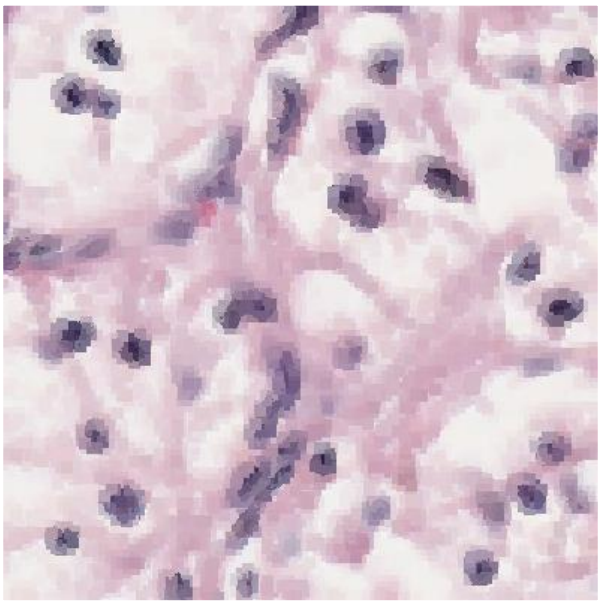
Önceki M değerine göre gözle görülür bir değişiklik yok.

20500_10700

Orijinal

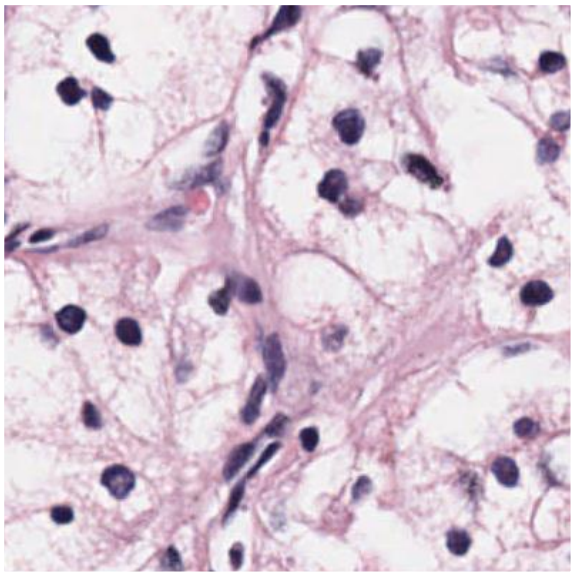


M=1 Süperpiksel=10000

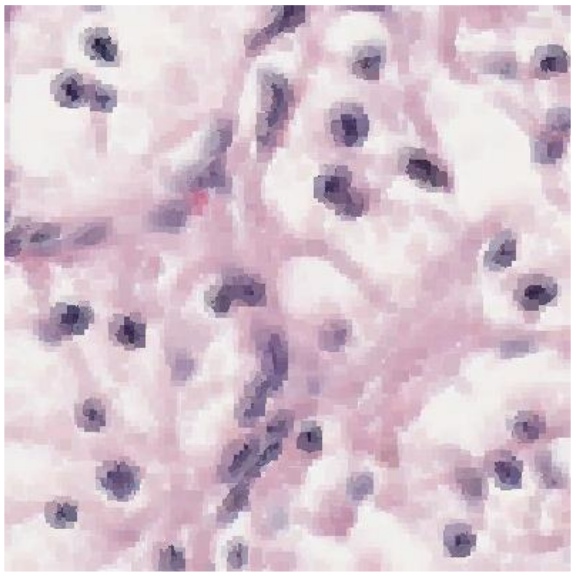


Süperpixel Bölütlemesi sonucunda resim segmentlere ayrıldı.

Orijinal

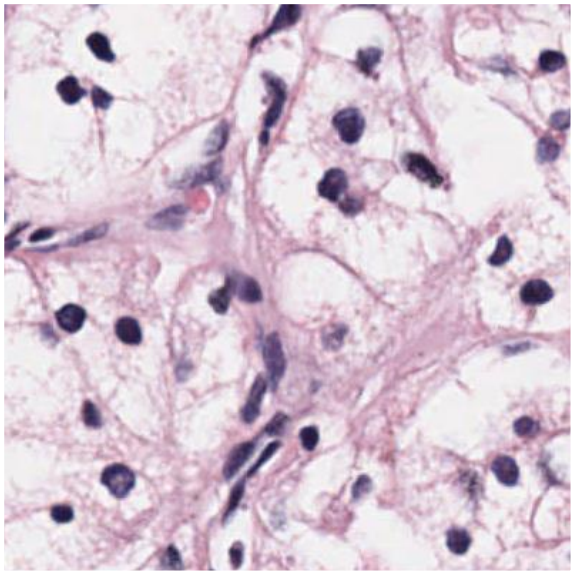


M=2 Süperpiksel=10000

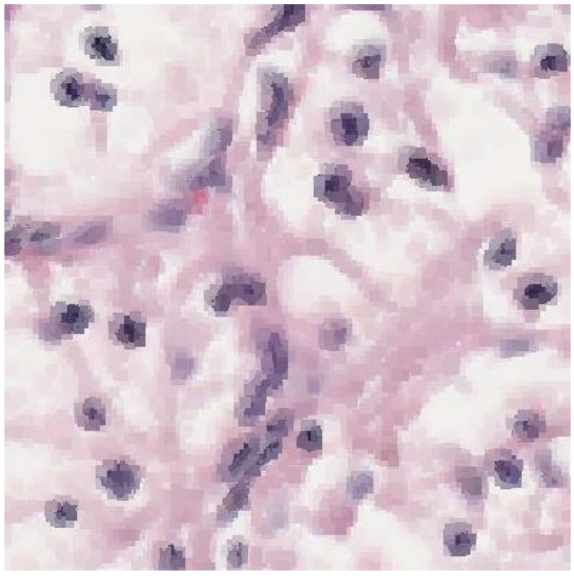


Önceki M değerine göre gözle görülür bir değişiklik yok.

Orijinal

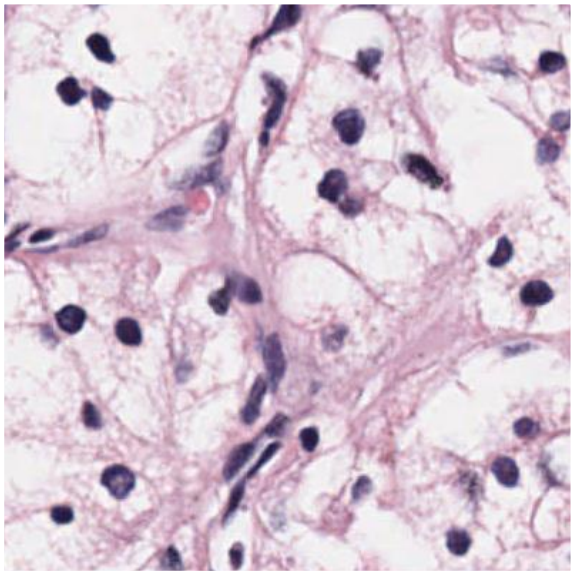


M=5 Süperpiksel=10000

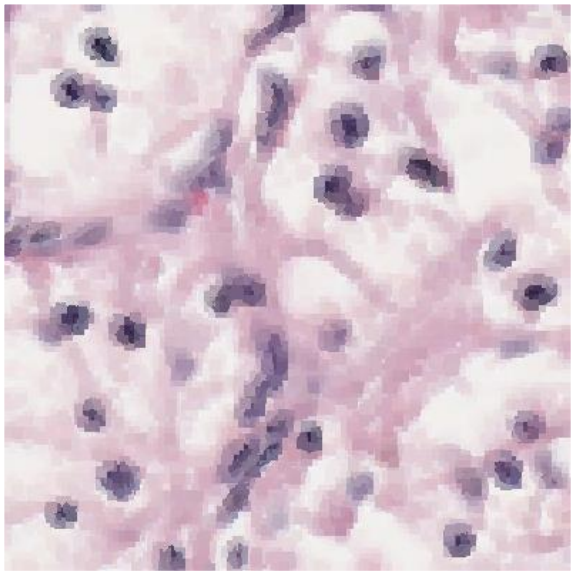


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

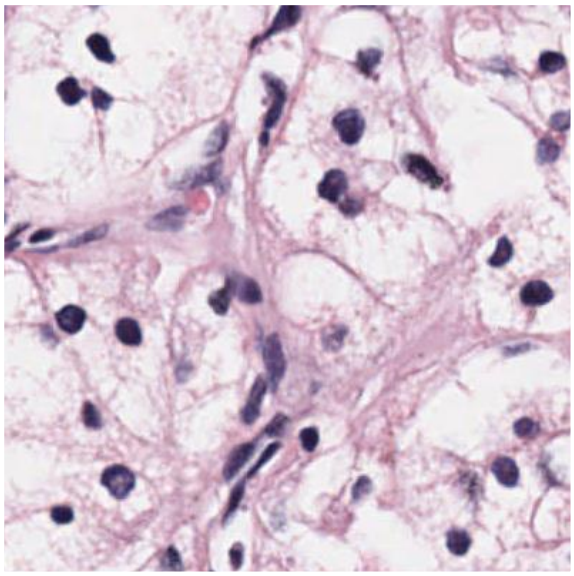


M=10 Süperpiksel=10000

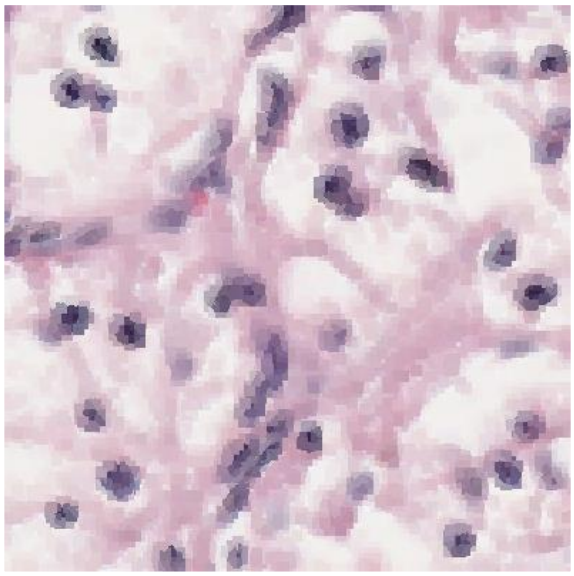


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

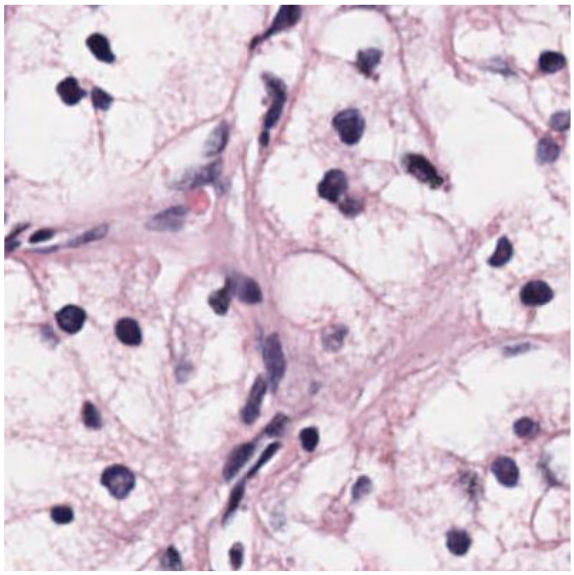


M=20 Süperpiksel=10000

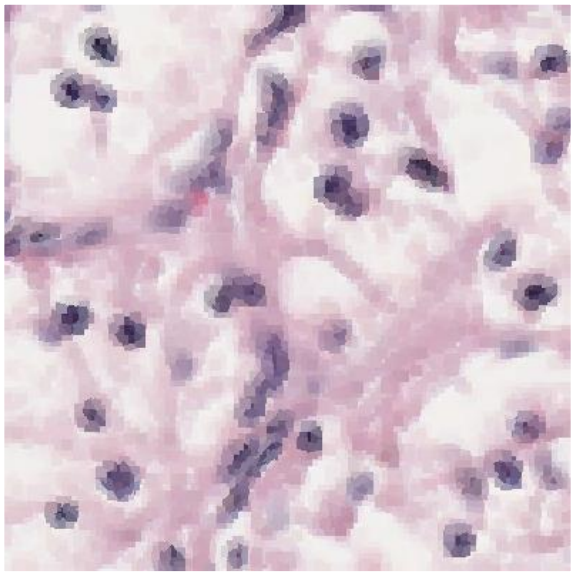


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

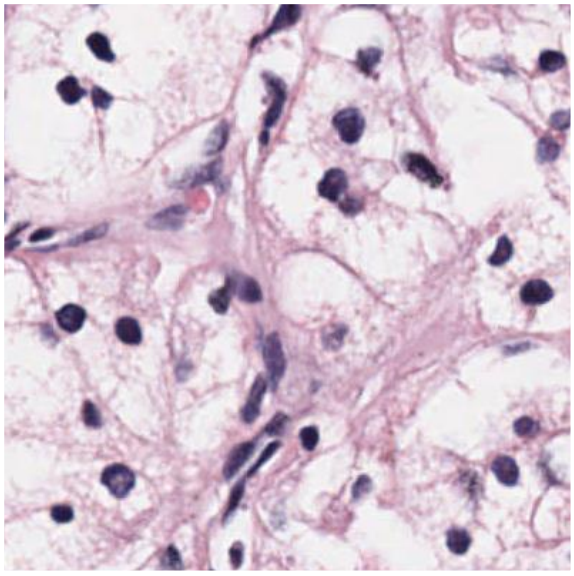


M=30 Süperpiksel=10000

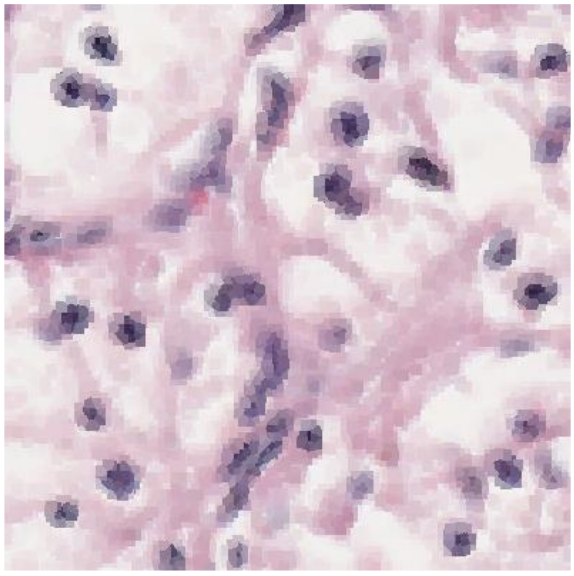


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal



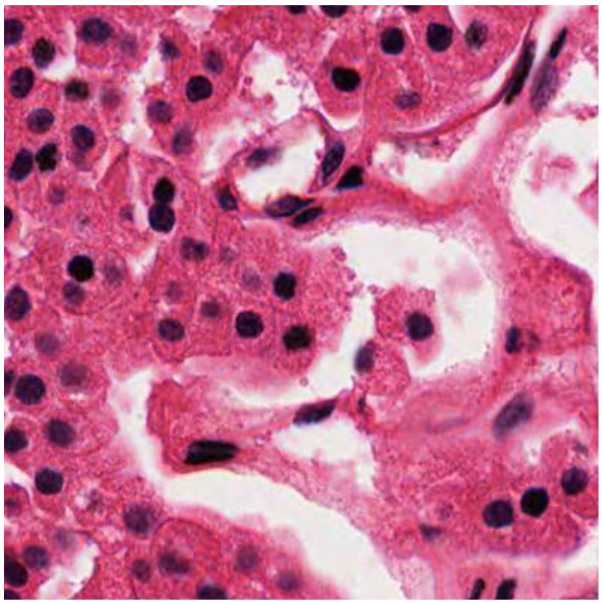
M=40 Süperpiksel=10000



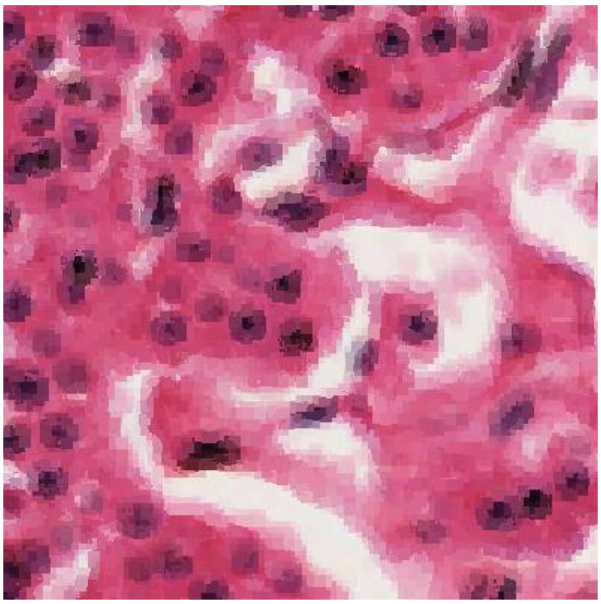
Önceki M değerine göre gözle görülür bir değişiklik yok.

29200_21400

Orijinal

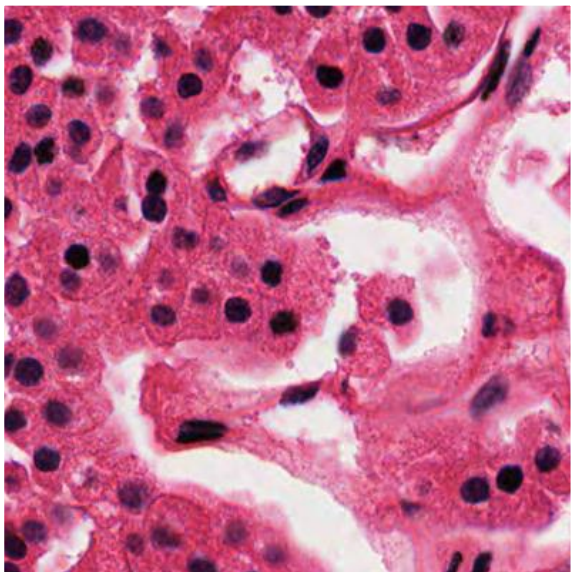


M=1 Süperpiksel=10000

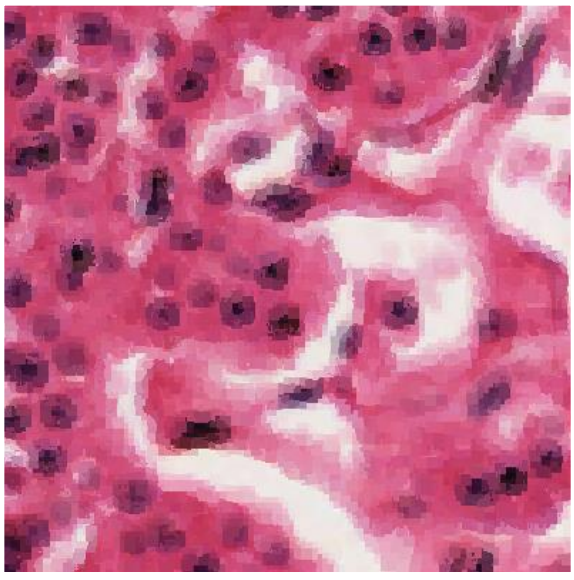


Süperpixel Bölütlemesi sonucunda resim segmentlere ayrıldı.

Orijinal

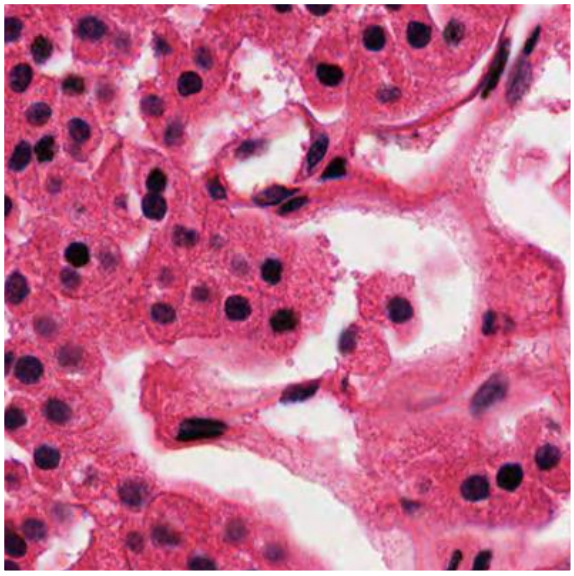


M=2 Süperpiksel=10000

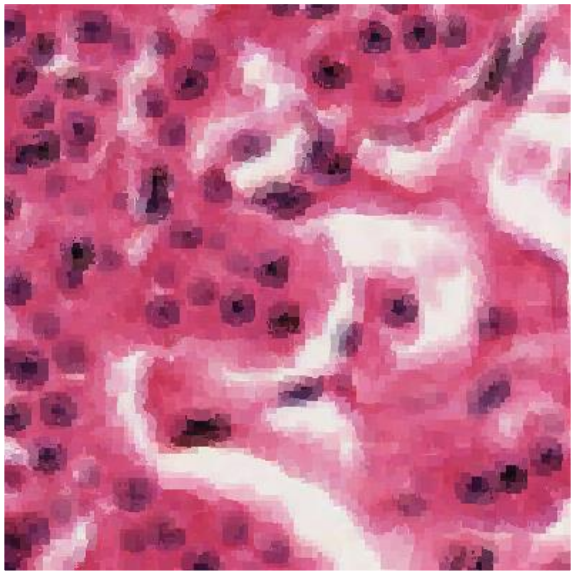


Önceki M değerine göre gözle görülür bir değişiklik yok.

Orijinal

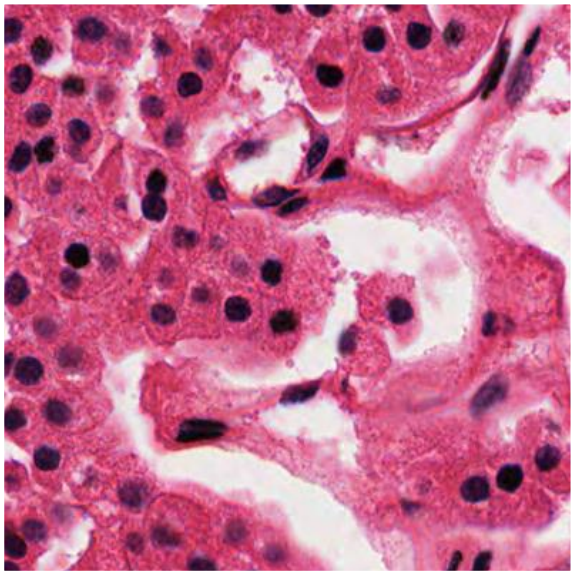


M=5 Süperpiksel=10000

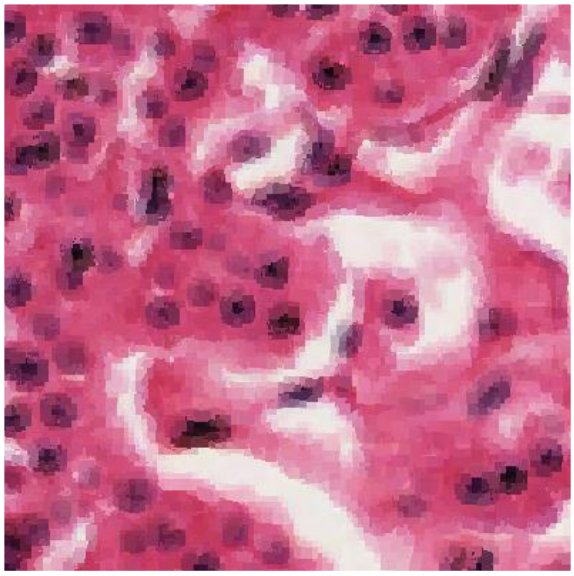


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

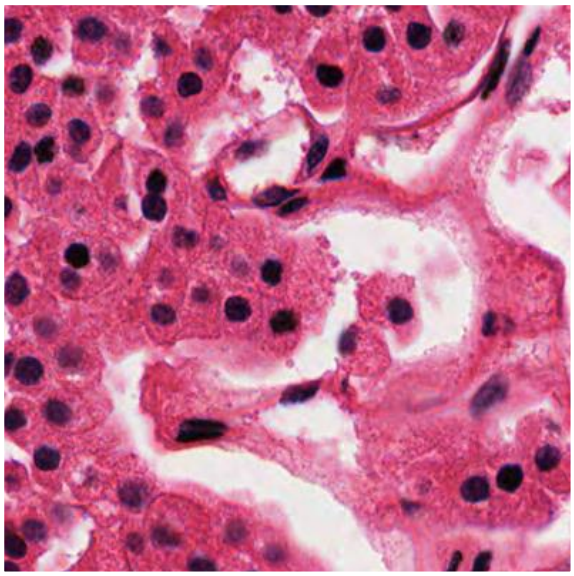


M=10 Süperpiksel=10000

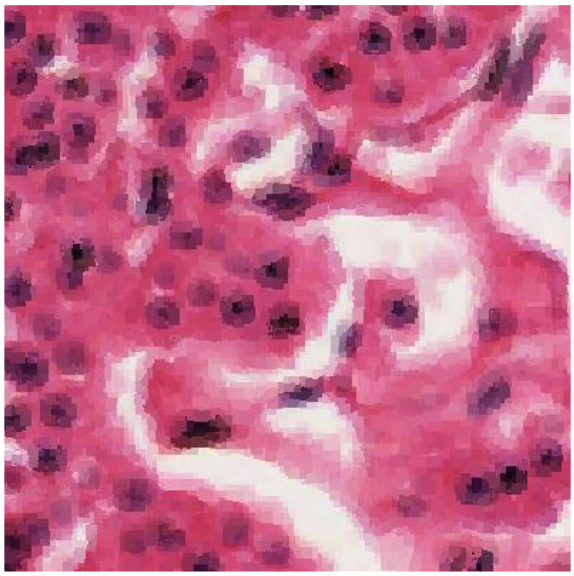


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

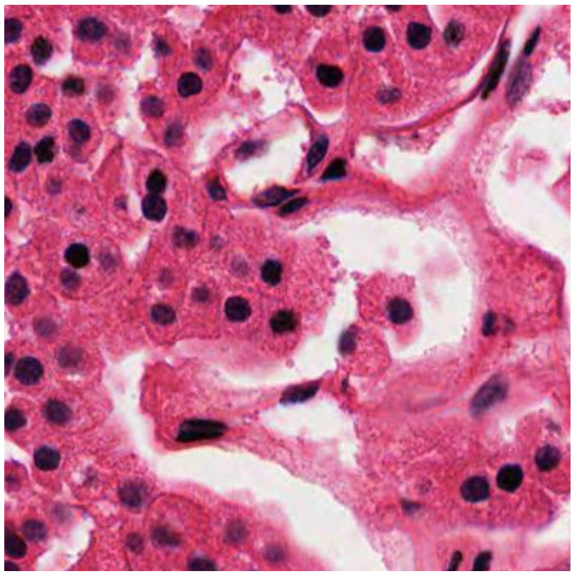


M=20 Süperpiksel=10000

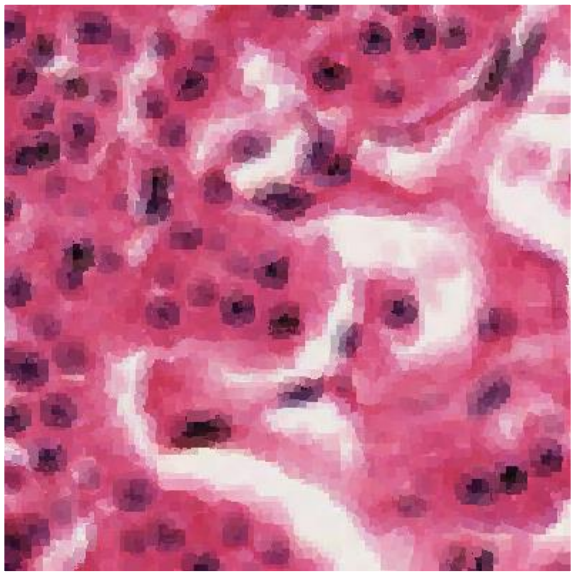


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

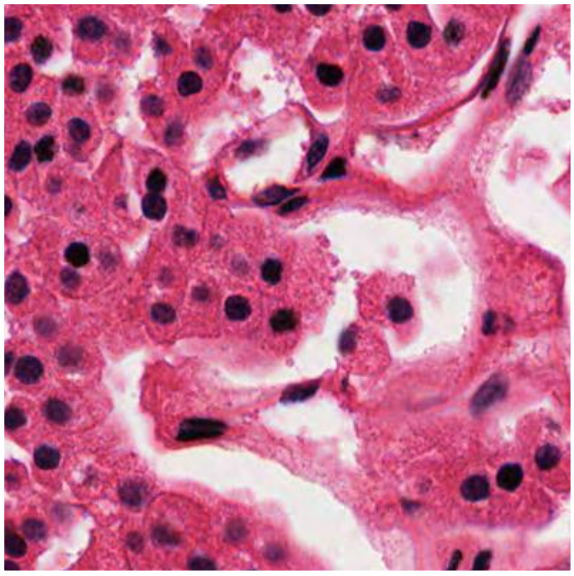


M=30 Süperpiksel=10000

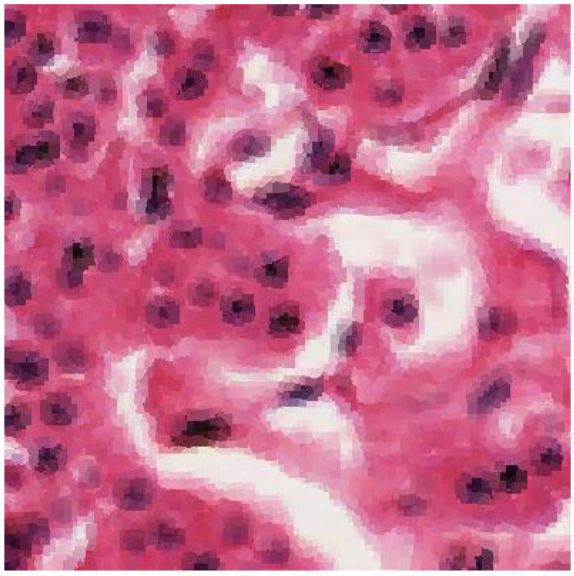


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal



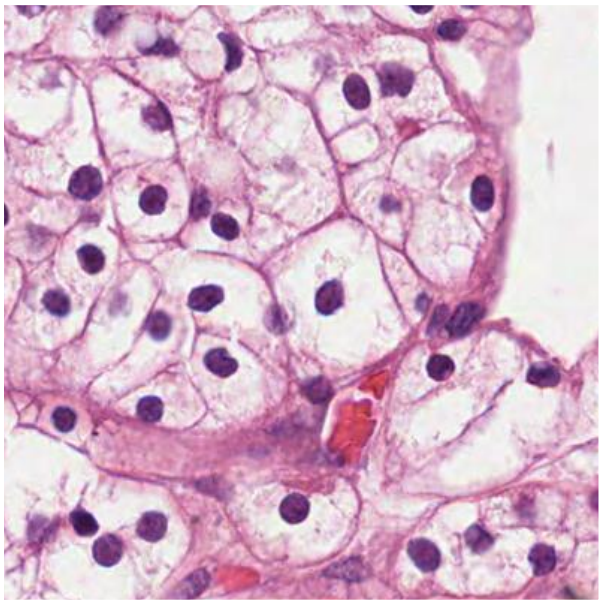
M=40 Süperpiksel=10000



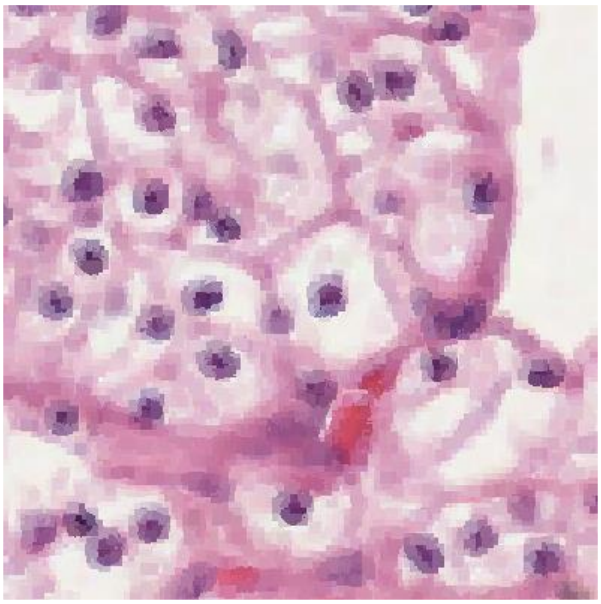
Önceki M değerine göre gözle görülür bir değişiklik yok.

31300_16200

Orijinal

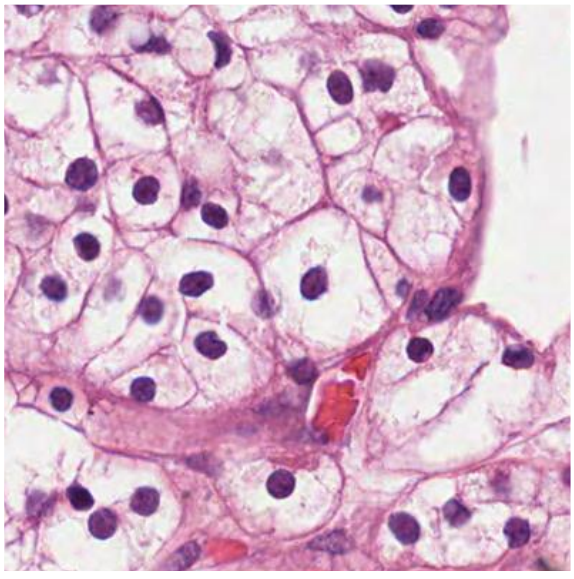


M=1 Süperpiksel=10000

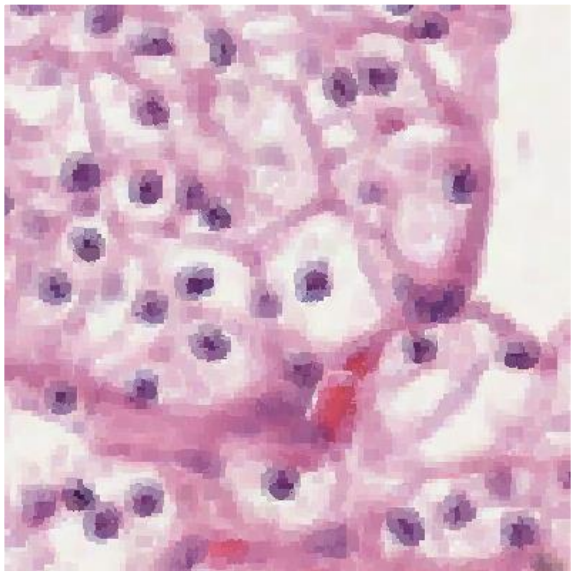


Süperpixel Bölütlemesi sonucunda resim segmentlere ayrıldı.

Orijinal

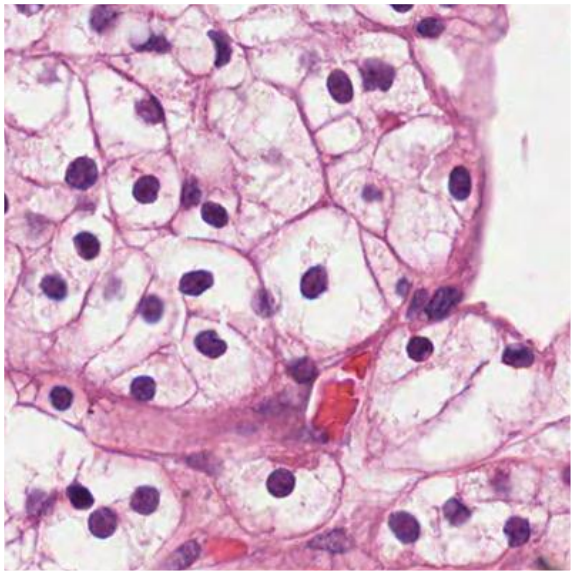


M=2 Süperpiksel=10000

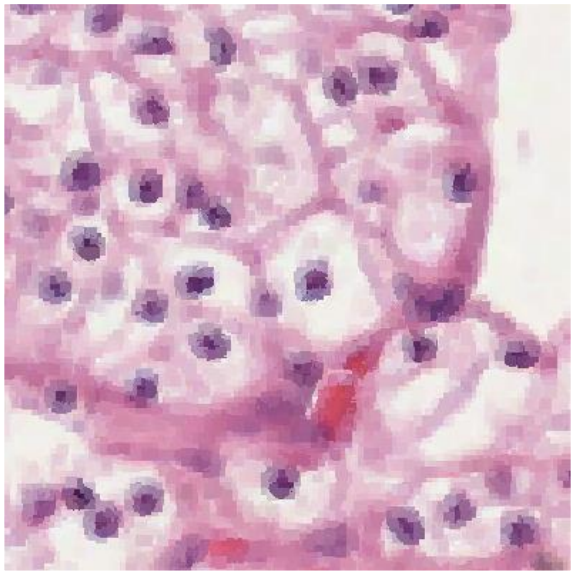


Önceki M değerine göre gözle görülür bir değişiklik yok.

Orijinal

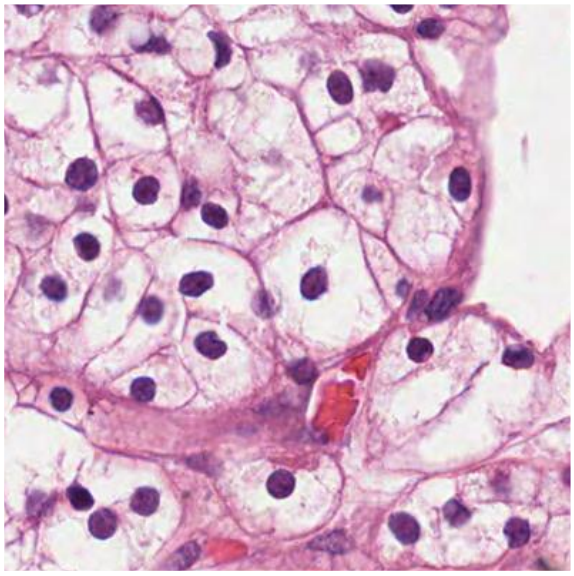


M=5 Süperpiksel=10000

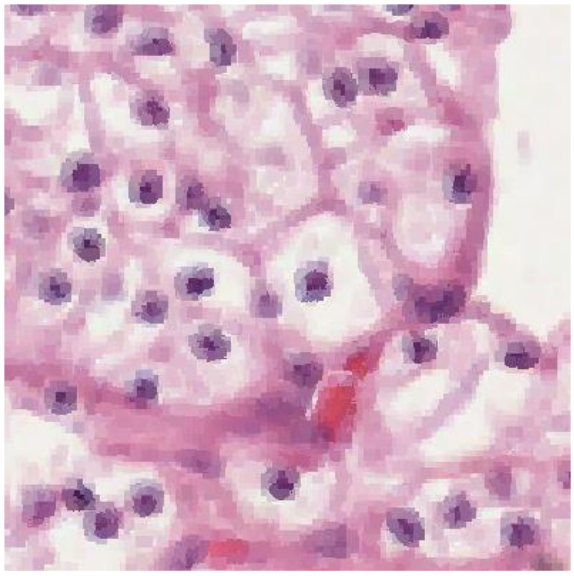


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

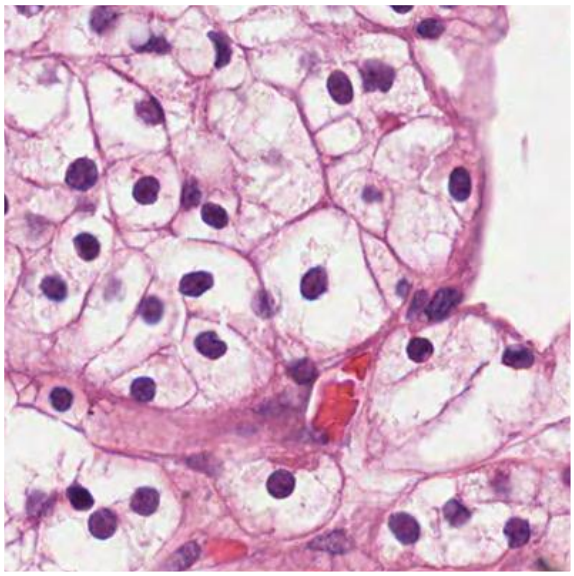


M=10 Süperpiksel=10000

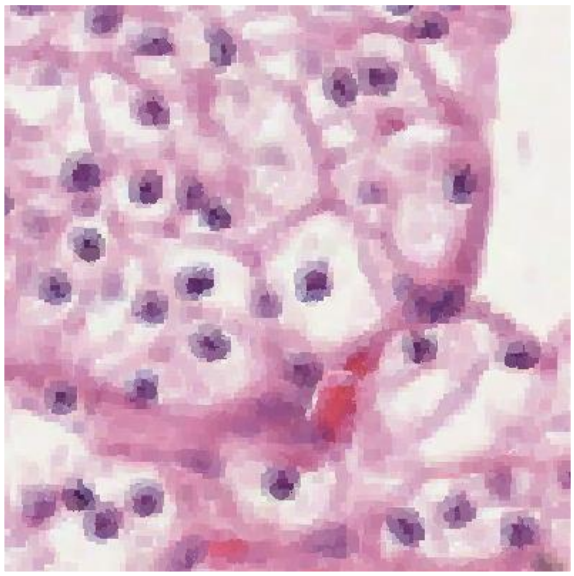


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

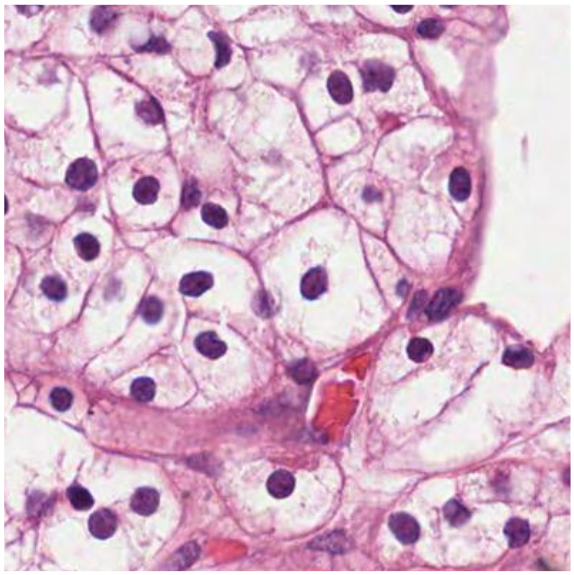


M=20 Süperpiksel=10000

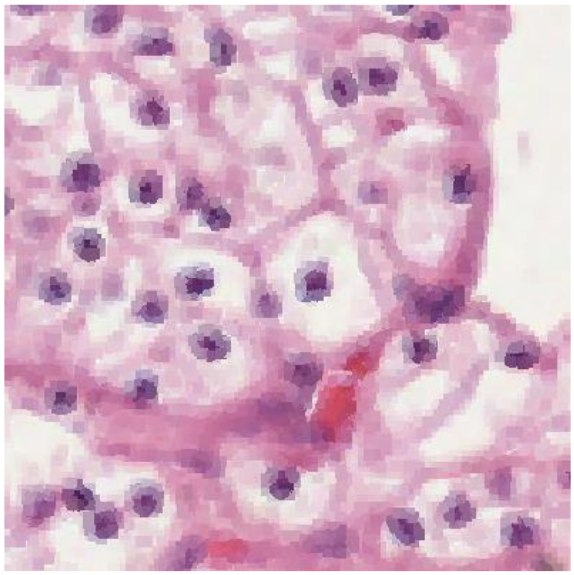


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal

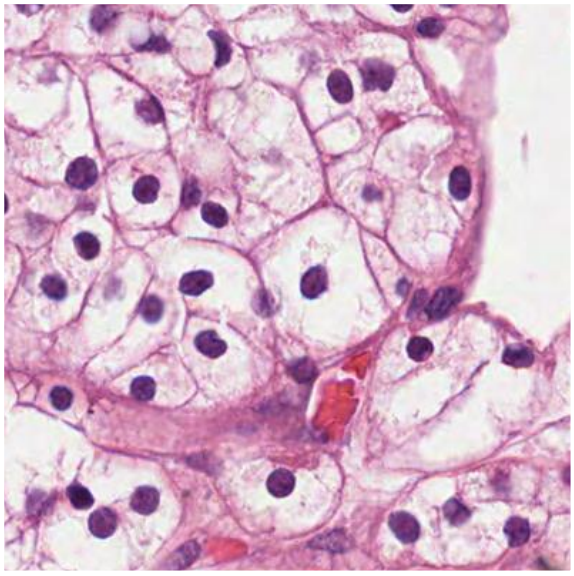


M=30 Süperpiksel=10000

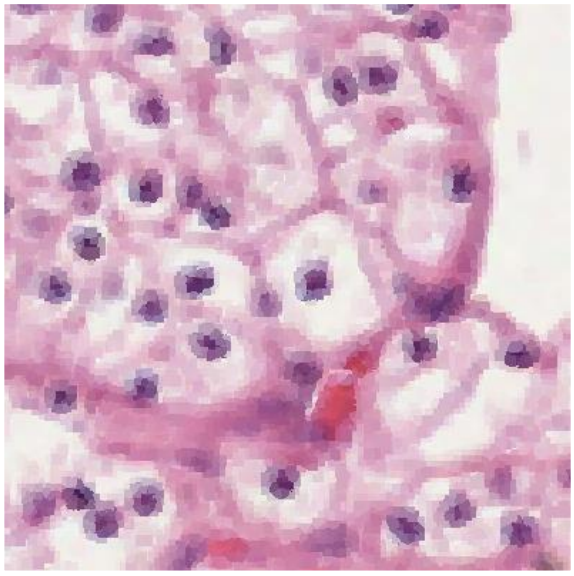


Önceki M değerine göre gözle görülür bir değişiklik yok

Orijinal



M=40 Süperpiksel=10000



Önceki M değerine göre gözle görülür bir değişiklik yok.

Sonuç

Resimlere dikkatli baktığımda M değeri büyüdükçe süper piksel bölütlemesinde grupların dışında kalan daha küçük grupların ortadan kalktığını görüyorum. Segmentasyon için daha yararsız olan küçük gruplar gidiyor ve bu süper pixellerin daha bağlı gruplar halini almasını sağlıyor. Sanırsam bize ayırt etme olayında fayda sağlayan bir eyleme dönüşüyor.

İkinci olarak üstteki olaya bağlı olarak kompaktlığın arttığını düşünüyorum. Şöyle ki M değerindeki artışa göre gruplar daha birleşik bir hal alıyor ki bu da benzer parçalar arasında daha az aralık olduğu anlamına geliyor.

Kaynak Kod

```
import sys
import cv2
from math import sqrt
import numpy as np
from operator import itemgetter

def initClusCenter(Clus,img,k):
    row=img.shape[0]
    col=img.shape[1]
    s=int(sqrt(row*col/k))
    print(s)
    c=[]
    t=int(s/2-1)
    c.append(t)
    c.append(t)
    c.append(img[t][t][0])
    c.append(img[t][t][1])
    c.append(img[t][t][2])
    Clus.append(c)
    k=0
    i=t
    j=i+s
    while i < row:
        while j < col:
            #print ("value: "+str(i)+" " +str(j))
            c=[]
            c.append(i)
            c.append(j)
            c.append(img[i][j][0])
            c.append(img[i][j][1])
            c.append(img[i][j][2])
            Clus.append(c)
            j+=s
        i+=s
        j=t
def bright_ave(New,Clus,img,l):
    row=img.shape[0]
    col=img.shape[1]
    for i in range(0,len(Clus)):
        sum_l=0
        sum_a=0
        sum_b=0
        sum=0
        ave_l=0
        ave_a=0
        ave_b=0
        j=0
        #print("len["+str(i)+"] :"+ str(len(l[i])))
        while j < len(l[i]):
            #print("j : "+str(j))
            x=l[i][j]
            y=l[i][j+1]
            sum_l+=img[x][y][0]
            sum_a+=img[x][y][1]
            sum_b+=img[x][y][2]
            sum+=1
            j+=2
        ave_l=sum_l // sum
        ave_a=sum_a //sum
```

```
        ave_b=sum_b //sum
        j=0
        while j < len(l[i]):

            x=l[i][j]
            y=l[i][j+1]
            #print("ilk L a b : " +str(img[x][y][0])+" " +str(img[x][y][1])+"
"+str(img[x][y][2]))
            img[x][y][0]=ave_l
            img[x][y][1]=ave_a
            img[x][y][2]=ave_b
            j+=2


def newClusCen(New,Clus,img,l):
    row=img.shape[0]
    col=img.shape[1]
    for i in range(0,len(Clus)):
        sum_row=0
        sum_col=0
        sum_r=0
        sum_g=0
        sum_b=0
        sum=0
        newCenter=[]
        j=0
        while j < len(l[i]):
            x=l[i][j]
            y=l[i][j+1]
            sum_row+=x
            sum_col+=y
            sum_r+=img[x][y][0]
            sum_g+=img[x][y][1]
            sum_b+=img[x][y][2]
            sum+=1
            j+=2
        newCenter.append(sum_row//sum)
        newCenter.append(sum_col//sum)
        newCenter.append(sum_r//sum)
        newCenter.append(sum_g//sum)
        newCenter.append(sum_b//sum)
        New.append(newCenter)




set=["10100_11400","14000_21400","20500_10700","29200_21400","31300_16200"]

def superPix(string,k_pix):
    for element in set:
        fileName= element + string
        try:
            fout=open(fileName,"rb")
        except:
            print ("Cannot open file ", filename, "Exiting … \n")
            sys.exit()
        img = cv2.imread(fileName)
        Clus=[]
        initClusCenter(Clus,img,k_pix)
```

```
        row=img.shape[0]
        col=img.shape[1]
        s=int(sqrt(row*col/k_pix))
        #initialize distance and label for every pixel
        l=[]
        d=[]

        for i in range(0,row):

            dtuple=[]
            for j in range(0,col):
                #ltuple.append(-1)
                dtuple.append(sys.maxsize)
            #l.append(ltuple)
            d.append(dtuple)
        m=40
        control=True
        while control==True:
            for k in range(0,len(Clus)):
                #print("Clus["+ str(k)+"][0] " + str(Clus[k][0]))
                #print("Clus["+ str(k)+"][1] " + str(Clus[k][1]))
                if Clus[k][0]-s < 0 :
                    stRow=0
                else:
                    stRow=Clus[k][0]-s
                if Clus[k][0]+s > row:
                    endRow=row
                else:
                    endRow=Clus[k][0]+s
                if Clus[k][1]-s < 0 :
                    stCol=0
                else:
                    stCol=Clus[k][1]-s
                if Clus[k][1]+s > col:
                    endCol=col
                else:
                    endCol=Clus[k][1]+s
                ls=[]
                for i in range(stRow,endRow):
                    for j in range(stCol,endCol):
                        #print("k "+ str(k) + " i "+ str(i)+ " j " +str(j))
                        dc=sqrt((Clus[k][2]-img[i][j][0])**2 + (Clus[k][3]-
img[i][j][1])**2 + (Clus[k][4]-img[i][j][2])**2)
                        dxy=((Clus[k][0]-i)**2 + (Clus[k][1]-j)**2)**0.5
                        ds=((dc)**2 + (((dxy/s)**2)*(m)**2))**0.5
                        if( ds < d[i][j]):
                            d[i][j]=ds
                            ls.append(i)
                            ls.append(j)
                l.append(ls)
            New=[]
            newClusCen(New,Clus,img,l)
            result=0
            for i in range(0,len(Clus)):
                result+= abs((Clus[i][0]-New[i][0])+(Clus[i][1]-New[i][1]))
                #print("result "+ str(result))
            print("result "+ str(result))
            if result < 10:
                control= False
            else:
                for i in range(0,len(Clus)):
                    Clus[i]=New[i]
```

```
        #convert image to Lab
        Lab_img= cv2.cvtColor(img, cv2.COLOR_RGB2LAB)

        Lab_l=[]
        for i in range(0,row):
            for j in range(0,col):
                d[i][j]=sys.maxsize

        #find Cluster Centers
        control=True
        while control==True:
            print(len(Clus))

            for k in range(0,len(Clus)):
                #print("Clus["+ str(k)+"][0] " + str(Clus[k][0]))
                #print("Clus["+ str(k)+"][1] " + str(Clus[k][1]))
                if Clus[k][0]-s < 0 :
                    stRow=0
                else:
                    stRow=Clus[k][0]-s
                if Clus[k][0]+s > row:
                    endRow=row
                else:
                    endRow=Clus[k][0]+s
                if Clus[k][1]-s < 0 :
                    stCol=0
                else:
                    stCol=Clus[k][1]-s
                if Clus[k][1]+s > col:
                    endCol=col
                else:
                    endCol=Clus[k][1]+s
                ls=[]
                for i in range(stRow,endRow):
                    for j in range(stCol,endCol):
                        #print("k "+ str(k) + " i "+ str(i)+ " j " +str(j))
                        dc=sqrt((Clus[k][2]-Lab_img[i][j][0])**2 + (Clus[k][3]-
Lab_img[i][j][1])**2 + (Clus[k][4]-Lab_img[i][j][2])**2)
                        dxy=((Clus[k][0]-i)**2 + (Clus[k][1]-j)**2)**0.5
                        ds=((dc)**2 + (((dxy/s)**2)*(m)**2))**0.5
                        if( ds < d[i][j]):
                            d[i][j]=ds
                            ls.append(i)
                            ls.append(j)
                Lab_l.append(ls)
            New=[]
            newClusCen(New,Clus,Lab_img,Lab_l)
            result=0
            for i in range(0,len(Clus)):
                result+= abs((Clus[i][0]-New[i][0])+(Clus[i][1]-New[i][1]))
                #print("result "+ str(result))
            print("result "+ str(result))
            if result < 5:
                control= False
            else:
                for i in range(0,len(Clus)):
                    Clus[i]=New[i]
        #write image
        bright_ave(New,Clus,Lab_img,Lab_l)
        RGB_img= cv2.cvtColor(Lab_img, cv2.COLOR_LAB2RGB)
        FileNameNew= element +"_" +str(m)+ string
        cv2.imwrite(FileNameNew,RGB_img)
```

```
print("İşlemi gerçekleştiriyor...")
superPix(".tiff",10000)
print("Bitti.")
```