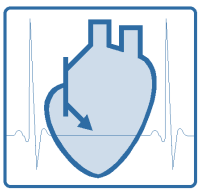


دانشگاه صنعتی امیرکبیر

دانشکده مهندسی پزشکی

گزارش تمرین پنجم پردازش تصویر

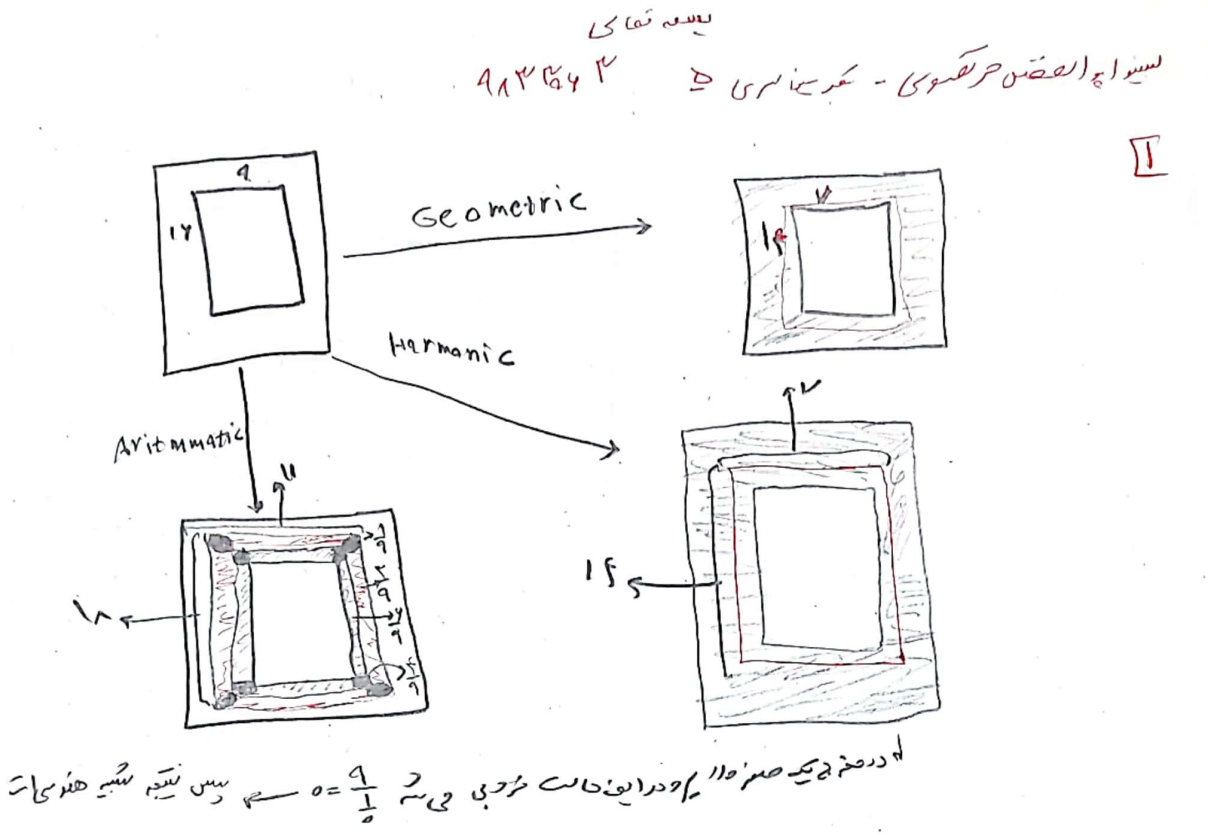
دانشجو

سیدابوالفضل مرتضوی ۹۸۳۳۰۶۳

دی ماه ۱۴۰۱

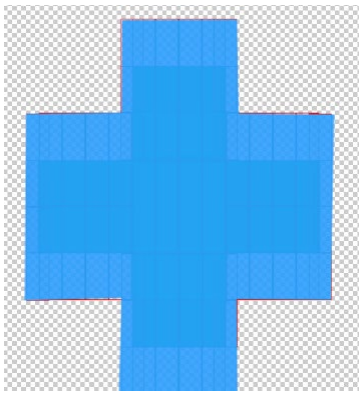
فهرست مطالب

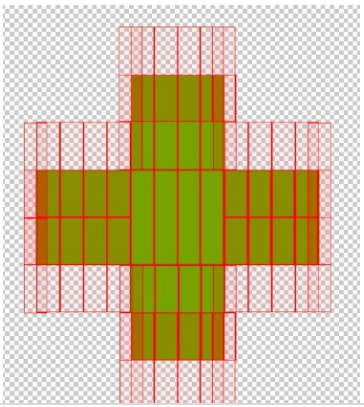
۱. سوال ۱)



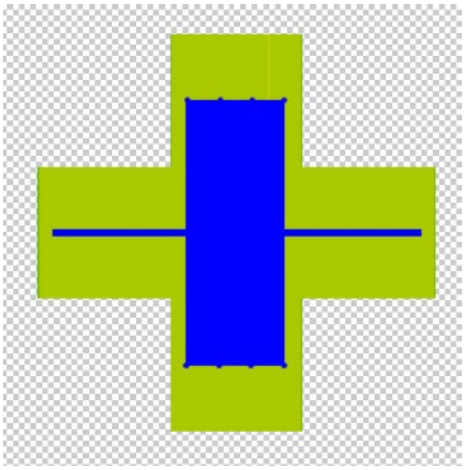
۲. سوال ۲)

به ترتیب مراحل انجام dilation(سمت چپ نتایج)

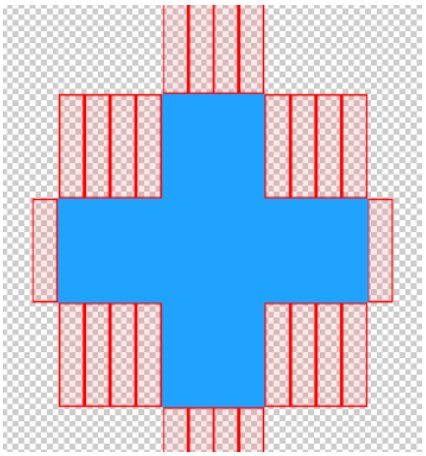


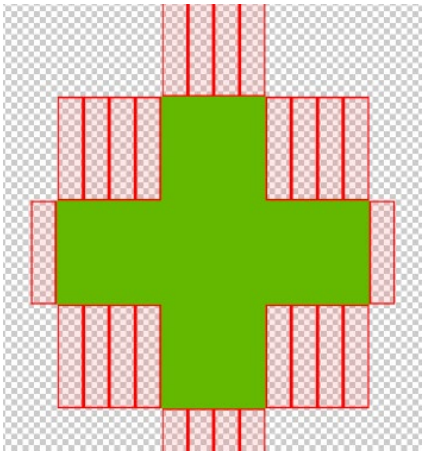


مراحل و نتیجهerosion (بخش آبی نتیجه)

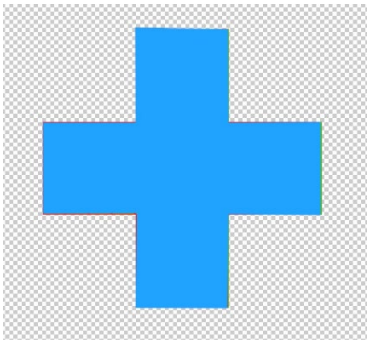


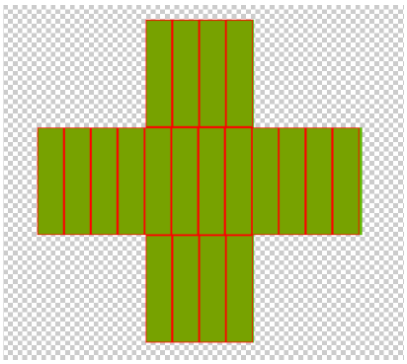
مراحل و نتیجه کلوزینگ



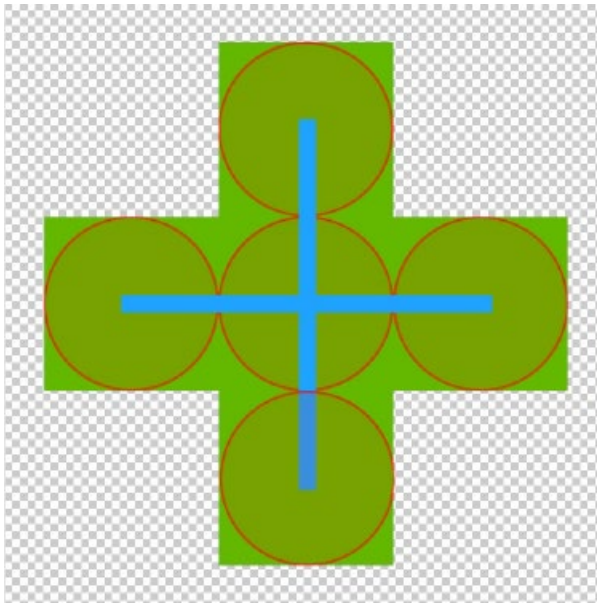


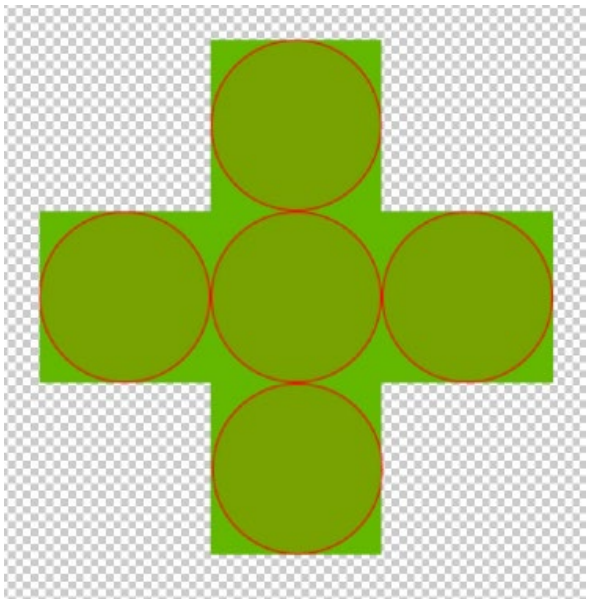
مراحل و نتیجه اوپنینگ

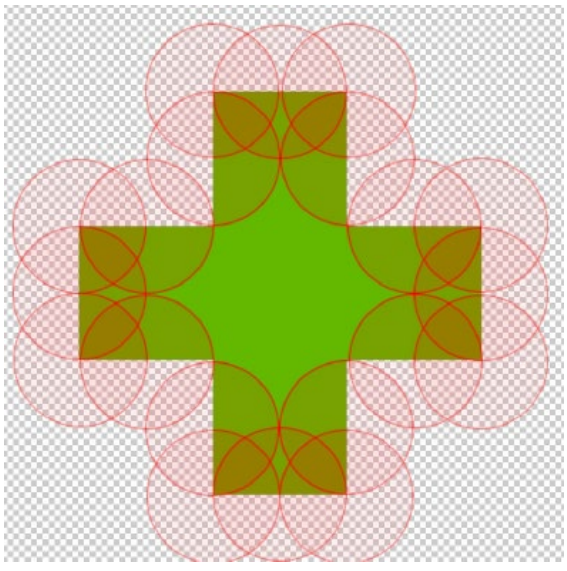


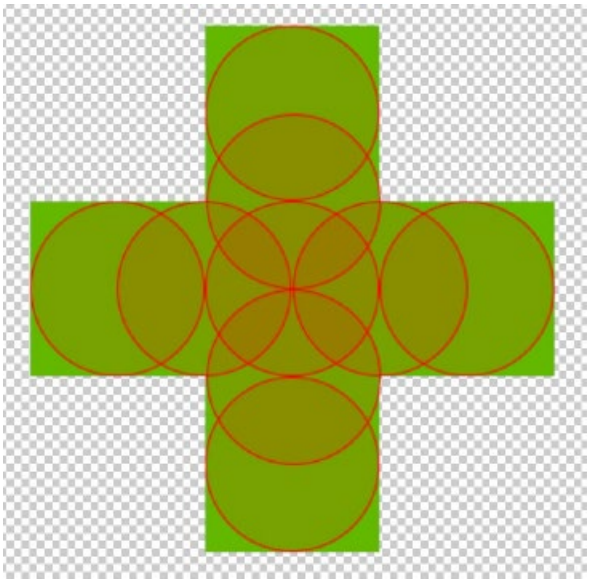
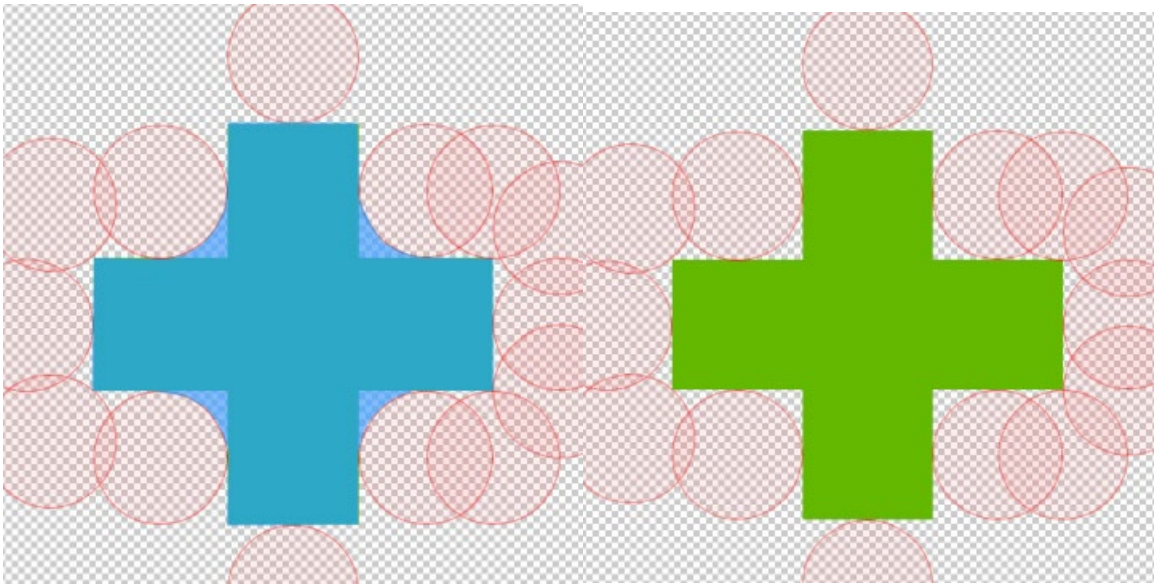


مراحل قبل برای دیسک: بخش آبی نتیجه است.







۳. سوال۳)

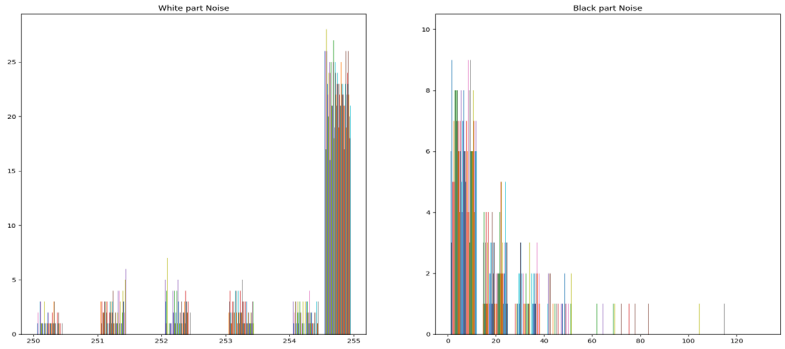
۳-۱ بخش ۱و۲و۳:

ابتدا تصویر به صورت خاکستری خوانده شدو سپس با کد زیر هیستوگرام بخش‌های سفید و سیاه رسم شد.

```
MRI_Noisy=cv.imread('MRI_Noisy.png',0)
Sample_MRI=np.zeros([10,260])
Sample_MRI=MRI_Noisy[40:50,40:300]
```

```
Sample_MRI2=MRI_Noisy[0:30,0:300]
plt.figure()
plt.subplot(1,2,1)
plt.title('White part Noise')
MRI_hist=plt.hist(Sample_MRI2)
plt.subplot(1,2,2)
plt.title('Black part Noise')
MRI_hist2=plt.hist(Sample_MRI)
```

خروجی کد بالا به صورت زیر است.



همانطور که ملاحظه می‌شود نویز بخش سفید توزیع یکنواخت داشته و نویز بخش سیاه توزیع نمایی دارد. با توجه به سکل توزیع یکنواخت و وجود نقاط تاریک در بخش سفید می‌توان نتیجه گرفت که نویز دارای علامت منفی است. در نویز نمایی با توجه به نزولی بودن نتیجه می‌شود که این نویز دارای مقادیر منفی است و این نمودار به همین خاطر نزولی شده است.

۳-۲ بخش ۴و۵:

در این بخش تابع زیر برای اعمال فیلتر تطبیقی برروی تصویر پیاده‌سازی شد.

```
def adaptive_filter(img2,ker):

    img=img2.copy()
    var_noise_l=[]
    for i in range(img.shape[0]-ker):
        for j in range(img.shape[1]-ker):
            window=img[i:i+ker,j:j+ker]
            var_noise_l.append(np.var(window))
    var_noise=np.mean(np.array(var_noise_l))

    img_paded=np.pad(img,int(ker/2))

    for i in range(img.shape[0]):
```

```
        for j in range(img.shape[1]):
            window_2=img_paded[i:i+ker,j:j+ker]
            var_window=np.var(window_2)
            mean_window=np.mean(window_2)
            img[i,j]=img[i,j]-var_noise/var_window*(img[i,j]-mean_window)
    return img.astype('uint8')
```

در اینجا ابتدا واریانس‌نویز را یافته سپس در هر پنجره واریانس و میانگین را حساب کرده و در فرمول مربوطه جایگذاری می‌کنیم.

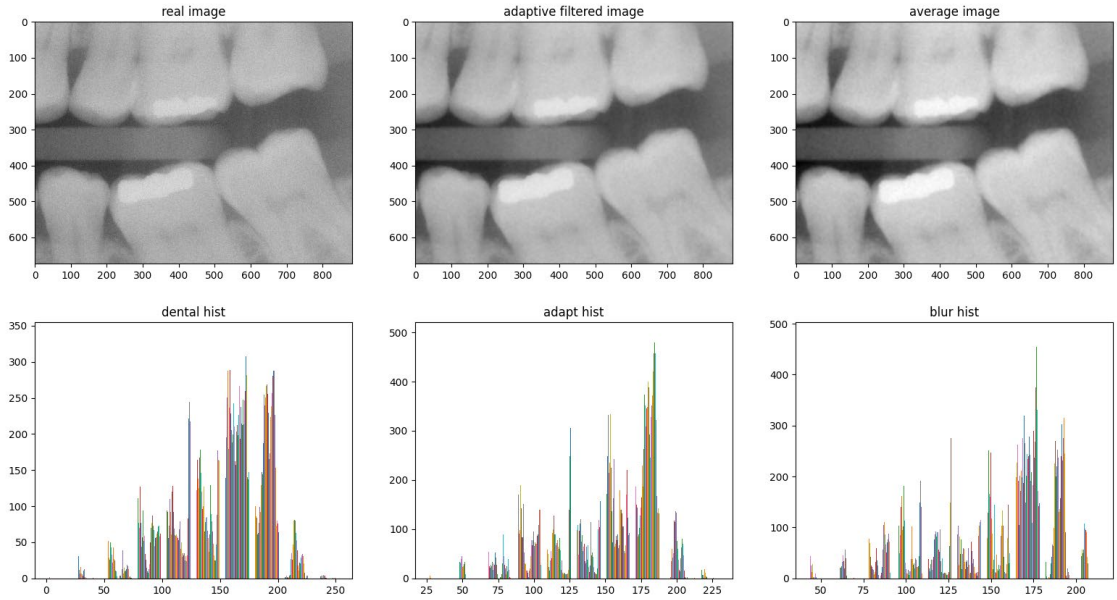
```
Dental_Noisy=cv.imread('DentalXray_Noisy.png',0)
adaptive_filtered_img=adaptive_filter(Dental_Noisy,7)
```

۳-۳ بخش ۶و۷:

بااستفاده از کد زیر فیلترمیانگین‌گیری برروی تصویر اعمال شد.

```
ker=np.ones((7,7),np.uint8)/49
blur_img=cv.filter2D(Dental_Noisy,-1,ker)
```

خروجی بخش‌های قبلی به صورت زیر است.



همانطور که قابل ملاحظه است تصویر مربوط به فیلتر میانگین گیری چون هر پیکسل در سرنوشت خود به اندازه بقیه پیکسل‌ها دخیل است و به طور کلی تصویر دارای نقاط سفید زیادی است روشن‌تر شده است اما در فیلتر تطبیقی چنین نیست.

۴. سوال۴)

۴-۱ بخش ۱و۲:

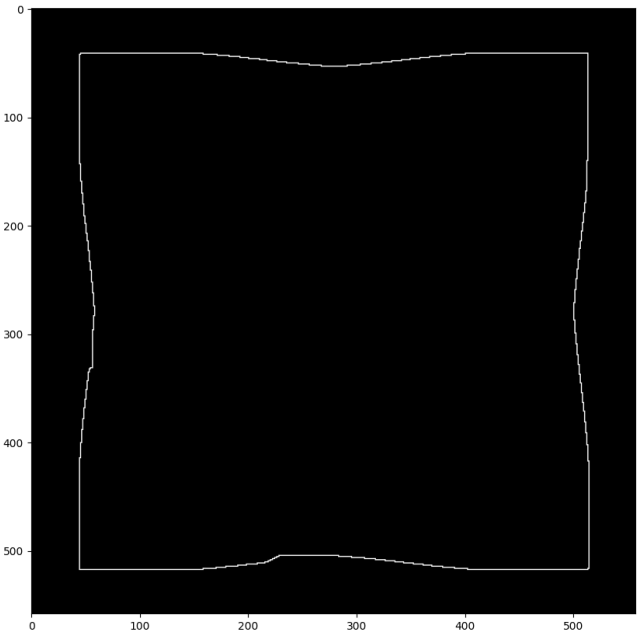
در این بخش از سوال از ما خواسته شده تا مرز های بیرونی تصویر را بدست بیاوریم برای این کار ابتدا نیاز است با استفاده از عملیات های ریخته گری عکس را دایلیت کنیم اما با کرنلی بزرگ تر از حد معمول چون میخواهیم قسمت های درونی عکس جزو مرز ها حساب نشود

سپس با استفاده از کرنل ۳*۳ ایرود میکنیم تا عکس به اندازه ی بسیار کمی تراشیده شود.

سپس عکس اول را از عکس بدست امده کم میکنیم تا لبه های عکس بدست بیاید.

```
kernel = np.ones((3,3), np.uint8)
kernel2 = np.ones((100,100), np.uint8)
kernel3 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(50,50))
kernel4 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(95,95))
img_dilation = cv2.dilate(img4_1, kernel2, iterations=1)
img_erosion = cv2.erode(img_dilation, kernel, iterations=1)
final_image  = img_dilation - img_erosion
plt.figure()
plt.imshow(final_image,'gray')
```

خروجی این بخش در تصویر زیر قابل مشاهده است.



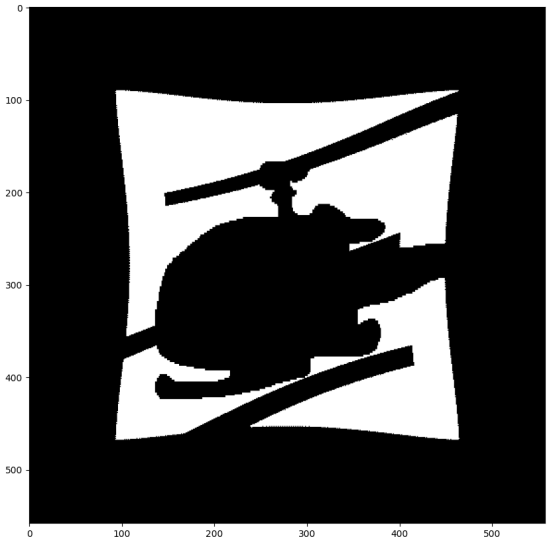
۴-۲ بخش ۳:

تفاوتی ندارند اما برای اطمینان ازاین امر تابع زیر نوشته شد

```
def morpho_diff(img,img2):
    A=255*np.ones([img2.shape[0],img2.shape[1]])
    mor_diff_img=np.zeros([img.shape[0],img.shape[1]]).astype('uint8')
    img2c=A-img2
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i,j] ==255 and img2c[i,j]==255:
                mor_diff_img[i,j]=255


    return mor_diff_img
```

خروجی کد بالا به صورت تصویرزیر است.



۴-۳ بخش ۴و۵:

برای این قسمت طبق کتاب عمل میکنیم در ابتدا عملیات کلوزینگ را انجام میدهیم سپس با تصویر خروجی عملیات اپنینگ را انجام میدهیم.

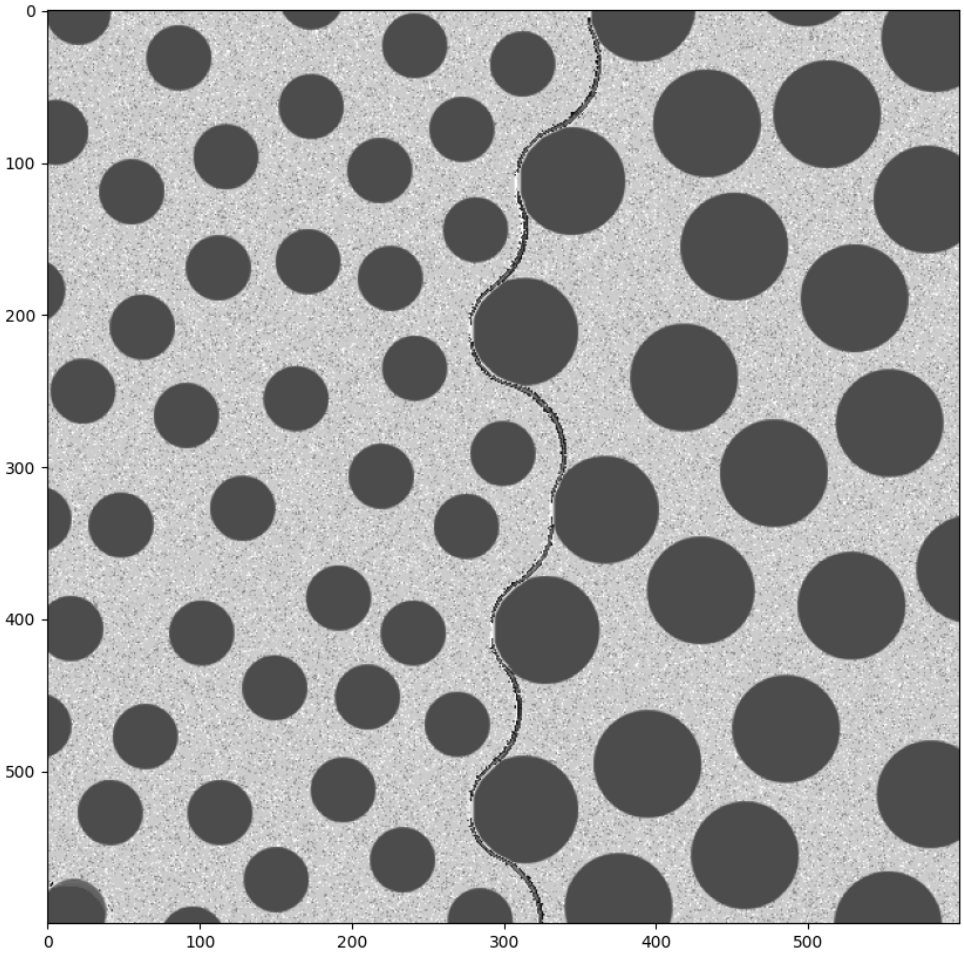
تصویر بدست امده را یک بار دایلیت و یک بار ایرود میکنیم سپس این دو را از یکدیگر کم میکنیم تا لبه ی تغییر بافت ها بدست بیاید.

سپس تصویر خروجی را باتصویر اولیه جمع می‌کنیم تا لبه‌ها مشخص شود.

کد زیر این مراحل را انجام می‌دهد.

```
closing = cv2.morphologyEx(Blobs_img, cv2.MORPH_CLOSE, kernel3)
opening = cv2.morphologyEx(closing, cv2.MORPH_OPEN,kernel4)
img_dilation = cv2.dilate(opening, kernel, iterations=1)
img_erosion = cv2.erode(opening, kernel, iterations=1)
finalimg = img_dilation - img_erosion
Blobs_det =Blobs_img +finalimg
plt.figure()
plt.imshow(Blobs_det,'gray')
```

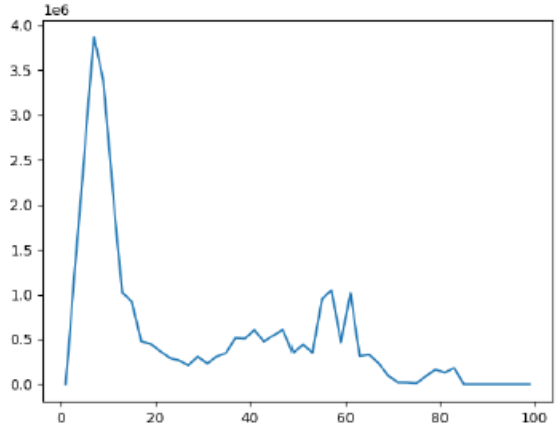
خروجی این بخش به صورت تصویر زیر است.



۴-۴ بخش۶:

در این مرحله باید عملیات اپنینگ را با کرنل‌هایی با سایز های مختلف انجام دهیم. و در هرمرحله تفاوت جمع شدت پیکسل ها را بررسی کنیم در نقاطی از نمودار که تفاوت شدت به یکباره زیاد میشود میتوان متوجه شد که کرنل با سایزمناسبی اعمال شده و در آن نقطه مقادیر زیادی ازدانه ها حذف شده اند. کد این بخش در زیر آمده است.

```
list = []
y1 = range(1,5)
i=1
plt.figure()
for k in [10,40,50,80]:
    circlekernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(k,k))
    opening = cv2.morphologyEx(Blobs_img, cv2.MORPH_OPEN,circlekernel)
    plt.subplot(1,4,i)
    plt.imshow(opening,'gray')
    plt.title(f'{i}th stage')
    current = np.float64(opening.sum())
    list.append(current)
    i+=1
difflist = [0]
k= 0
for i in list:
    if(k!=0):
        diffrence =  np.abs( i - list[k-1])
        difflist.append(diffrence)
    k = k+1
plt.figure()
plt.plot(y1,difflist, label = "line 1")
```



با توجه به نمودار فوق خروجی در کرنل‌سایزهای ۱۰و۴۰و۵۰و۸۰ تغییرات زیادی داشته و تصاویر مربوط به این کرنل‌سایز ها در تصویر زیر آمده است.