بنام خدا

دانشگاه صنعتی اصفهان
دانشکده برق و کامپیوتر

تمرین سری ششم –پردازش تصاویر دیجیتال

استاد: دکتر سعید صدری

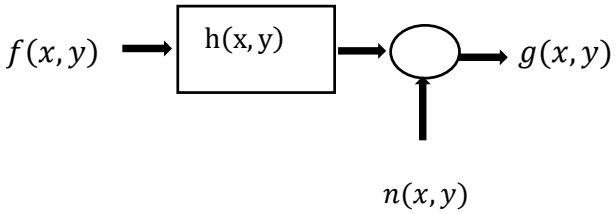
رضا سعادتی فرد ۹۴۱۱۳۹۴

پروانه رشوند ۹۴۱۰۱۲۴

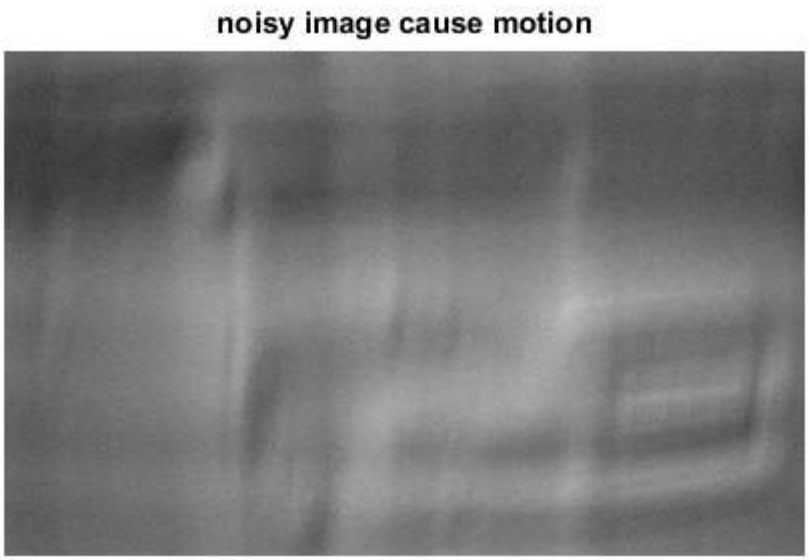
دی ماه ۹۵

سوال اول)(کد این قسمت در HW6_Q1 قرار دارد)

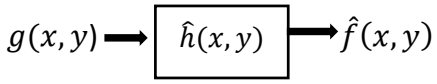
در این سوال قرار است که از روش‌های مربوط به image restoration تصویر یک اتومبیل را که در حال حرکت بوده و نویزی شده است را بهبود دهیم به نحوی که پلاک ماشین خوانده شود.



همانطور که مشاهده می‌نمایید ابتدا تصویر f از یک فیلتر خراب کننده عبور می‌کند که این فیلتر خراب کننده در اینجا فیلتر حرکت می‌باشد. سپس با یک نویز جمع شده و تصویرg را به ما می‌دهد. چیزی که هم اکنون ما در دست داریم همان تصویر g می‌باشد که به صورت زیر می‌باشد:



حال باید مطابق بلوک زیر یک $\hat{h}(x,y)$ طراحی نماییم که وقتی تصویر



نویزی شده‌ی $g(x,y)$ از آن عبور می‌کند، یک $\hat{f}(x,y)$ برای ما بدست آورد که حتی الامکان به $f(x,y)$ نزدیک باشد. بنابراین مقادیری که ما در دست داریم تنها $g(x,y)$ و $h(x,y)$ می‌باشند. در این سوال گفته شده که شیء گرفتار حرکت شده است لذا معادلات مربوط به فیلتر حرکت را می‌یابیم:

$$g(x,y) = \int_0^T f(x - x_0(t), y - y_0(t))dt \quad \rightarrow \quad G(U,V) = \iint g(x,y)e^{-iUx-iVy}\,dxdy$$

$$\rightarrow G(U,V) = \iint \int_0^T f(x - x_0(t), y - y_0(t))dt\; e^{-iUx-iVy}dxdy$$

$$\rightarrow G(U,V) = \int_0^T \iint f(x - x_0(t), y - y_0(t))\, e^{-iUx - iVy} dxdy\, dt = \int_0^T F(U,V) e^{-i(Ux_0(t) + Vy_0(t))}\, dt$$

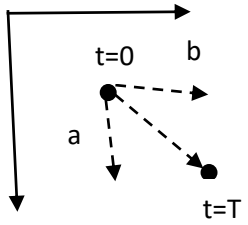
$$\rightarrow \mathrm{G(U,V)} = \mathrm{F(U,V)} \int_0^T e^{-i(Ux_0(t) + Vy_0(t))}\, dt$$

$$\rightarrow H(U,V) = \frac{G(U,V)}{F(U,V)} = \int_0^T e^{-i(Ux_0(t) + Vy_0(t))}\, dt$$

در حرکت روی خط مستقیم داریم:

$$x_0(t) = \frac{at}{T}$$

$$y_0(t) = \frac{bt}{T}$$



و لذا در این حالت داریم:

$$H(U,V) = \int_0^T e^{-\frac{i}{T}(aU+bV)t}\, dt = \frac{T}{-i(aU+bV)}\left(e^{-i(aU+bV)} - 1\right)$$

$$\rightarrow H(U,V) = \frac{2T}{aU+bV} e^{-i\frac{aU+bV}{2}} \sin\left(\frac{aU+bV}{2}\right) = Te^{-i\frac{aU+bV}{2}} \frac{\sin\left(\frac{aU+bV}{2}\right)}{\frac{aU+bV}{2}}$$

در نتیجه معادله مربوط به حرکت در حوزه فرکانس را یافتیم حال باید از دو روشCLS و وینر یک فیلتر بهبود دهنده طراحی نماییم.

روش اول) CLS

با توجه به بلوک دیاگرام های رسم شده در ابتدای مسیله در حوزه مکان داریم:

$$g(x,y) = f(x,y) ** h(x,y) + n(x,y)$$

هم نویز اعمالی دارای پهنای باند بزرگ است و هم اغییرات غیر طبیعی اعمال شده به f در اثر h. $\hat{h}(x,y)$ را طوری انتخاب می‌نماییم که حتی الامکان تغییرات خیلی سریع $f(x,y)$ تقلیل یابد.

اپراتور لاپلاسین را اگر بر $f(x,y)$ اعمال کرده و نرم آن را محاسبه نماییم ($\|Pf\|^2$) یک اندازه است از میزان تغییرات سریع در $f(x,y)$.

بنابراین $\hat{h}(x,y)$ ای مناسب است که $\|Pf\|^2$ حداقل شود.

از طرفی به صورت ماتریسی می‌توان نوشت : $n = g - fh \rightarrow \|n\|^2 = \|g - fh\|^2$

بنابراین مسیله‌ی تخمین $\hat{h}(x,y)$ در CLS به صورت مسیله بهینه سازی به صورت زیر در می‌آید:

یعنی مینیمایز شدن $\|Pf\|^2$ با شرط $\|g-fh\|^2-\|n\|^2=0$

از ضرایب لاگرانژ استفاده می‌نماییم $E=\|Pf\|^2+\lambda(\|g-fh\|^2-\|n\|^2)$

$$\rightarrow \mathrm{E}=f^TP^TPf+\lambda(g^Tg-g^Tfh-h^Tf^Tg+h^Tf^Tfh-n^Tn)$$

$$\frac{\partial E}{\partial f}=0 \quad \rightarrow \quad f=\frac{\lambda h^Tg}{\lambda h^Th+\|P\|^2} \qquad \rightarrow \qquad f=\frac{h^Tg}{h^Th+\gamma\|P\|^2}$$

که $\gamma=\frac{1}{\lambda}$

بنابراین اگر معادله‌ی به دست آمده را به حوزه فرکانس ببریم تخمین f به صورت زیر خواهد بود:

$$\hat{F}(U,V)=\frac{H^*(U,V)G(U,V)}{|H(U,V)|^2+\gamma\|P(U,V)\|^2}$$

بنابراین در الگوریتم CLS فیلتر بهبود دهنده به ص.رت زیر به دست می‌آید:

$$\widehat{H}(U,V)=\frac{\hat{F}(U,V)}{G(U,V)}=\frac{H^*(U,V)}{|H(U,V)|^2+\gamma\|P(U,V)\|^2}$$

که H(U,V) همان فیلتر خراب کننده مربوط به حرکت است که معادله آن را به دست آوردیم.

روش دوم: روش wiener

در این روش نیز به دنبال پیدا کردن یک فیلتر بهبود دهنده براساس فیلتر خراب کننده می‌باشیم در اینجا فیلتر حرکت بوده است. به علت طولانی بودن اثبات مربوط به این فیلتر از آوردن آن خودداری می‌نماییم و نتیجه نهایی را برای فیلتر وینر به دست می‌آوریم که عبارت است از:

$$\widehat{H}(U,V)=\frac{H^*(U,V)}{|H(U,V)|^2+\frac{|N(U,V)|^2}{|F(U,V)|^2}}$$

که در آن داریم:

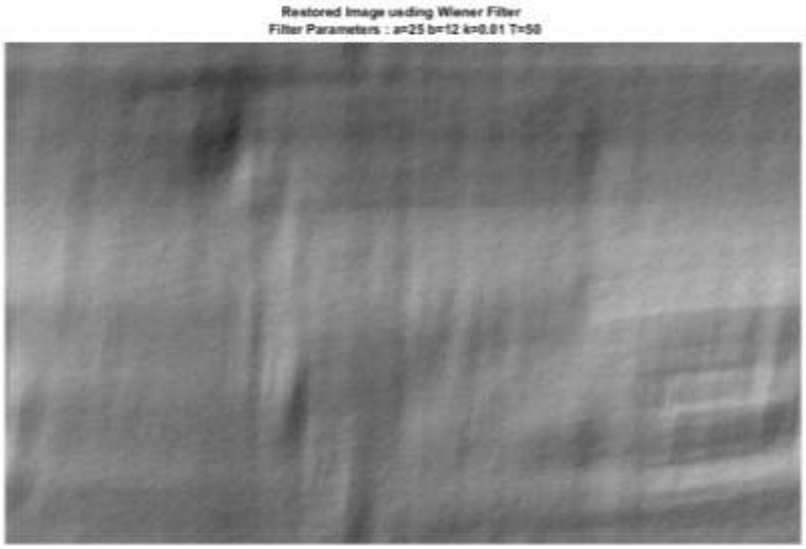
$\frac{|N(U,V)|^2}{|F(U,V)|^2}=\frac{{\sigma_n}^2}{\sigma_S^2}=\frac{1}{SNR}$=K

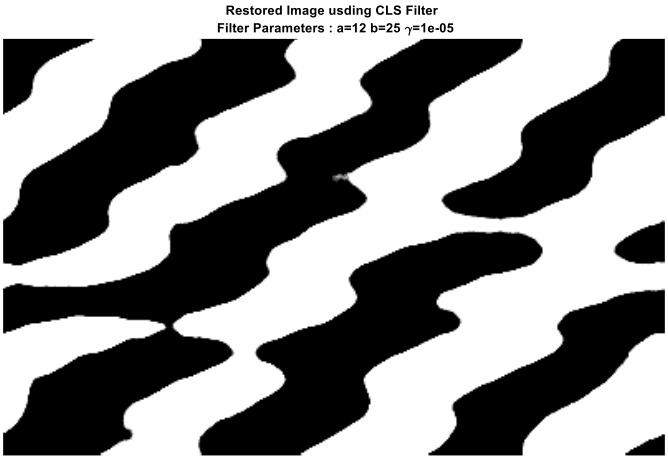
الگوریتم را بدین صورت اجرا می‌کنیم که ابتدا یک تصویر را توسط فیلتر خراب کننده مربوط به حرکت نویزی کرده و سپس آن را از دو روش فوق دی نویز می‌نماییم تا پلاک ماشین را تشخیص دهیم. برای این منظور از کد زیر استفاده می‌نماییم:

```
clc
clear
close all

%%%%%%% Create and Image in movement
r=imread('BMW.jpg');
r=rgb2gray(r);
figure,imshow(r),title('Original Image')
f=im2double(r);
[M,N] = size(f);

for a= 2:2
    for b=13:13
        T=40;
        F=fft2(f);

        for v=1:N
            for u=1:M
                o= (a*u + b*v)/2 ;
                H(u,v)= (T/o)    *  exp((-o) * 1i)   * sind(o);
                G1(u,v) = H(u,v) * F(u,v);
            end
        end

        %%%%% adding Noise
        n=0.003 * rand(M,N);
        Noise=fft2(n);

        G = G1 + Noise;
        figure,imshow(real(ifft2(G))),title(['Noisy Image a= ',num2str(a),'  b=
',num2str(b)])
    end
end

% %%%%%% Problem Image
% load('restoration.mat');
% figure, imshow(g),title('Original Image')
% G = fft2(g);
% [M,N] = size(g);



%%%% Generate Laplacian  for CLS Method
Lap = zeros(M,N);
temp = ones(3,3);
temp(2,2) = -8;
Lap(1:3,1:3) =temp;
LAP = fft2(Lap);
LAP2 = abs(LAP).^2;


for a=2:1:2
    for b=13:1:13
        for T=50:10:50
            H=[];
            for u=1:M
                for v=1:N
                    coef = (a*u + b*v)/2;
                    H(u,v) = (T/coef) * exp(-1i*coef) * sind(coef);
                end
            end
            H_conj = conj(H);
```

```
            H2 = H.*H_conj;

            %%%%%%%% Wener Method

            for k_Wiener=0.01:.01:0.01
                Wiener_Filt = H_conj./(H2 + k_Wiener);

                F_hat_Winer = G.* Wiener_Filt;
                f_hat_Winer = real(ifft2(F_hat_Winer));

                str_tit = sprintf('Restored Image usding Wiener Filter \n Filter
Parameters : a=%d b=%d k=%s T=%d',...
                    a,b,num2str(k_Wiener),T);
                figure,imshow(f_hat_Winer),title(str_tit)
            end

            %%%%%%% CLS Method
            for gamma=.001:.001:.001
                CLS_Filt = H_conj ./ (H2 + gamma.*(LAP2));
                F_hat_CLS = G.* CLS_Filt;
                f_hat_CLS = real(ifft2(F_hat_CLS));

                str_tit = sprintf('Restored Image usding CLS Filter \n Filter
Parameters : a=%d b=%d \\gamma=%s',a,b,num2str(gamma));
                figure,imshow(f_hat_CLS),title(str_tit)
            end

        end
    end
end
```

نتیجه برای روش وینر به صورت زیر خواهد بود:





بهترین نتیجه برای a=25 و b=12 به دست أمد ولی متاسفاه مناسب نبود و لذا ما خود یک تصویر را توسط فیلتر خواب کننده مربوط به حرکت نویزی نموده و سپس آن زا دی‌نویز نمودیم تا نتیجه حاصل از عملکرد درست الگوریتم مشخص شود.

تصویر اصلی که در واقع ما آن را نداریم به صورت زیر است:

تصور کنید در اثر حرکت نویزی شده و به صورت زیر درامده:

تصویر نویزی شده در واقع تصویری است که ما در دست داریم. حال با استفاده از کد نوشته شده و با توضیحاتی که داده شد تصویر را دی‌نویز می‌نماییم که به صورت زیر درخواهد آمد:

مشاهده می‌شود که با دی‌نویز نمودن تصویر با روش وینر و با پارامترهای مشخص شده در تصویر که به صورت سعی و خطا به دست آمده اند، می‌توان پلاک ماشین را خواند.

اما نتیجه حاصل از روشCLS به صورت زیر است:

نتیجه به صورتی است که پلاک ماشین کاملا قابل تشخیص است. با توجه به توضیحات گفته شده در ابتدای شوال پارامترهاا را به صورت سعی و خطا برابر با مقادیر مشخص شده در شکل در نظر گرفته ایم.

برای درست کردن اپراتور لاپلاسین مطابق مطالب عنوان شده در درس ابتدا اپراتور را به صورت زیر ایجاد میکنیم:

1	1	1
1	-8	1
1	1	1

سپس اندازه اپراتور را باید برابر با اندازه کل تصویر کنیم در نتیجه یک تصویر M * N ، که تمام درایه های آن ۰ است تولید میکنیم و این اپراتور فوق رو در مربع ۳*۳ ابتدایی آن قرار میدهیم . سپس با تبدیل fft گرفتن از آن اپراتور ما ایجاد می‌شود که کد آن نیز به صورت زیر است:

```
%%%% Generate Laplacian  for CLS Method
Lap = zeros(M,N);
temp = ones(3,3);
temp(2,2) = -8;
Lap(1:3,1:3) =temp;
LAP = fft2(Lap);
LAP2 = abs(LAP).^2;
```

در قسمت دیگر از سوال از ما خواسته شده است نحوه بدست آوردن γ بهینه در روش cls را بیان کنیم. در روش cls برای بدست اوردن گاما بهینه با با انتخاب یک گام افزیش b یک ترشولد a انتخاب میکنیم بگونه ای نرم

```
G - (H .* F_hat_1)
```

در فاصله $a_{+}^{-} \left\| \eta^{2} \right\|$ قرار بگیرد و اگر در این فاصله نبود گاما را به اندازه b کم و یا زیاد کنیم(اگر خارج از فاصله بود زیاد و اگر کمتر از فاصله بود کم میکنیم)

کد آن به صورت زیر است:

```
%%%%%%% optimzed gaamma

 delta = 1 ;                                    % tole tabe chegalie noise
 sigma = ((0.021)^2)* ((delta^2) / 12) ;   % varianse of noise
 mu = 0 ;                                       % mean of noise
 eta_2 = M*N*(sigma + (mu^2));

 a=.1;
 b=1e-2;
 gamma = 3e-6;

 exit =0;
 c=0;
 while exit==0
c=c+1
 CLS_Filt = H_conj ./ (H2 + gamma.*(LAP2));
 F_hat_1 = G.* CLS_Filt;
 nu = G - (H .* F_hat_1);
 rrr=norm(nu)^2;

 if rrr > (eta_2 + a)
 gamma = gamma - b ;
 elseif rrr< (eta_2 - a)
 gamma = gamma + b ;
 else
 exit=1;
 end

 end

CLS_Filt = H_conj ./ (H2 + gamma.*(LAP2));
F_hat_CLS_2 = G.* CLS_Filt;
f_hat_CLS_2 = real(ifft2(F_hat_CLS_2));
figure,imshow(f_hat_CLS_2)
```

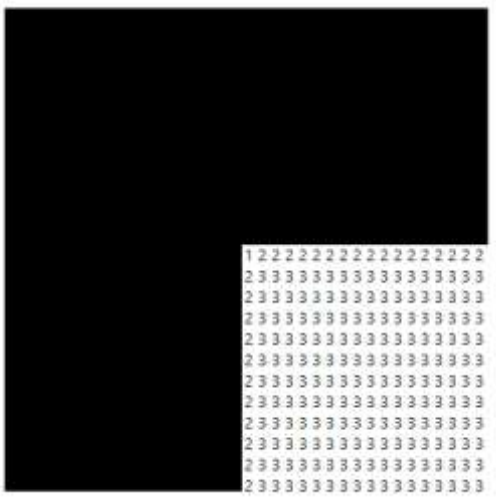
با توجه به اینکه به تصویر بهینه ای دست نیافیتم ، از محاسبه گاما به روش پیشنهادی نیز صرف نظر کردیم و تنها روش ارایه شد.

برای محاسبه k بهینه در روش وینر از سوپرویژن استفاده میکنیم. و البته گامای بهینه روش cls نیز آسان تر و بهتر است از سوپرویژن بدست آید.

سوال دوم)

الف) در این قسمت قرار است که مراحل الگوریتم تشخیص لبه‌ی canny را به صورت دستی برای یافتن لبه ها در تصویر زیر پیاده سازی نماییم:

تصویر مورد نظر یک تصویر ۲۰۰×۲۰۰ می‌باشد که پس از عبور از فیلتر گوسی به صورت زیر درامده است. یعنی سه چهارم آن صفر و بقیه نیز به صورت زیر می‌باشد:



Canny برای پیاده سازی الگوریتم خود گفت باید یک فیلتر h بهینه طراحی نمود که تصویر f از آن عبور نموده و تصویر g را نتیجه دهد و نصویر g لبه‌های f را آشکارسازد و در نهایت به این نتیجه رسید که h بهینه، مشتق اول تابع گوسی می‌باشد.

$$f(x,y) \rightarrow \boxed{h(x,y)} \rightarrow g(x,y)$$

در حالت یک بعدی:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}} \quad \rightarrow \quad h(x) = \frac{1}{\sqrt{2\pi}\sigma}\left(\frac{-x}{\sigma^2}\right) e^{\frac{-x^2}{2\sigma^2}}$$

$$\rightarrow \mathrm{g}(x) = f(x) * h(x) = f(x) * \frac{d}{dx}p(x) = \frac{d}{dx}(f(x) * p(x))$$

بنابراین لازم است در ابتدا تصویر را با یک فیلتر گوسی فیلتر نماییم و از آن‌جا که تصاویر دو بعدی هستند یک بار در جهت x و بار دیگر در جهت y از آن مشتق بگیریم. در صورت سوال، تصویر بالا را تصویر عبوری از فیلتر گوسی در نظر گرفته است لذا لازم است از همین تصویر مشتق در جهت x و y بگیریم. برای مشتق گیری از f در ۲ بعد باید از مشتق گیر سابل استفاده نماییم.

سابل در جهت x وy به صورت زیر می‌باشد:

سابل در جهت x

↓

	-1	0	1
-1	۰.۱	۰	۱
0	۰.۲	۰	۲
1	۰.۱	۰	۱

سابل در جهت y

↓

-۱	-۲	-۱
۰	۰	۰
۱	۲	۱

سابل در جهت x را h_1 و سابل در جهت y را h_2 می نامیم. برای به دست اوردن مشتق تصویر در این دو جهت، باید تصویر را توسط این دو ماتریس فیلتر نماییم یعنی ان را با این دو کانوالو نماییم. داریم:

$$\frac{df}{dx}(x,y) = f(x,y) ** h_1(x,y) = \sum_m \sum_n f(m,n) h_1(x-m, y-n)$$

$$\frac{df}{dx}(49,49) = \sum_m \sum_n f(m,n) h_1(49-m, 49-n)$$

ابتدا h_1 را معکوس می‌نماییم که به صورت روبرو در می‌آید:

	-1	0	1
-1		۰	-۱
0		۰	-۲
1		۰	-۱

همانطور که مشاهده می‌شود فیلتر h_1 تنها مولفه های 1- تا ۱ را شامل می‌شود و همچنین تصویر f نیز از پیکسل مرکزی یعنی(50,50) دارای مقدار است. لذا مقادیر را طوری جایگزین می‌نماییم که h_1 تنها شامل همین مقادیر شود و f نیز پیکسل های صفر را شامل نشود. لذا داریم:

$$\rightarrow \frac{df}{dx}(49,49) = f(50,50)h_1(-1,-1) = 1 \times 1 = 1$$

$$\frac{df}{dx}(49,50) = \sum_m \sum_n f(m,n) h_1(49-m, 50-n)$$

$$\rightarrow \frac{df}{dx}(49,50) = f(50,50)h_1(-1,0) + f(50,51)h_1(-1,-1) = 1 \times 0 + 2 \times 1 = 2$$

$$\frac{df}{dx}(50,49) = \sum_m \sum_n f(m,n) h_1(50-m, 49-n)$$

$$\rightarrow \frac{df}{dx}(50,49) = f(50,50)h_1(0,-1) + f(51,50)h_1(-1,-1) = 1 \times 2 + 2 \times 1 = 4$$

$$\frac{df}{dx}(50,50) = \sum_m \sum_n f(m,n) h_1(50-m, 50-n)$$

$$\rightarrow \frac{df}{dx}(50,50) = f(50,50)h_1(0,0) + f(50,51)h_1(0,-1) + f(51,50)h_1(-1,0) + f(51,51)h_1(-1,-1)$$

$$= 1 \times 0 + 2 \times 2 + 2 \times 0 + 3 \times 1 = 7$$

$$\frac{df}{dx}(50,51) = \sum_m \sum_n f(m,n)h_1(50-m, 51-n)$$

$$\rightarrow \frac{df}{dx}(50,51) = f(50,50)h_1(0,1) + f(50,51)h_1(0,0) + f(50,52)h_1(0,-1) + f(51,50)h_1(-1,1) +$$

$$f(51,51)h_1(-1,0) + f(51,52)h_1(-1,-1) = 1 \times -2 + 2 \times 0 + 2 \times 2 + 2 \times -1 + 3 \times 0 + 3 \times 1 = 3$$

$$\frac{df}{dx}(50,52) = \sum_m \sum_n f(m,n)h_1(50-m, 52-n)$$

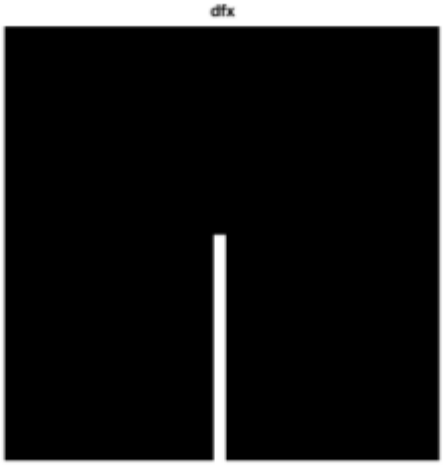
$$\rightarrow \frac{df}{dx}(50,52) = f(50,51)h_1(0,1) + f(50,52)h_1(0,0) + f(50,53)h_1(0,-1) + f(51,51)h_1(-1,1) +$$

$$+f(51,52)h_1(-1,0) + f(51,53)h_1(-1,-1) = 2 \times -2 + 2 \times 0 + 2 \times 2 + 3 \times -1 + 3 \times 0 + 3 \times 1 = 0$$

به همین ترتیب مراحل را ادامه می‌دهیم تا تمامی مولفه های $\frac{df}{dx}$ به دست آید. نتایج به صورت زیر خواهد بود و مشتق در جهت x تنها در پیکسل‌های (49:100,49) و (49:100 ,50) و(49:100 ,51) (49:100 ,100) مقدار دارند. این مقادیر عبارتند از:

$\frac{df}{dx}(49,49) = 1$	$\frac{df}{dx}(50,49) = 4$	$\frac{df}{dx}(51,49) = 7$	$\frac{df}{dx}(52{:}99,49) = 8$	$\frac{df}{dx}(100,49) = 6$
$\frac{df}{dx}(49,50) = 2$	$\frac{df}{dx}(50,50) = 7$	$\frac{df}{dx}(51,50) = 11$	$\frac{df}{dx}(52{:}99,50) = 12$	$\frac{df}{dx}(100,50) = 9$
$\frac{df}{dx}(49,51) = 1$	$\frac{df}{dx}(50,51) = 3$	$\frac{df}{dx}(51,51) = 4$	$\frac{df}{dx}(52{:}99,51) = 4$	$\frac{df}{dx}(100,51) = 3$
$\frac{df}{dx}(49,100) = -2$	$\frac{df}{dx}(50,100) = -7$	$\frac{df}{dx}(51,100) = -11$	$\frac{df}{dx}(52{:}99,100) = -12$	$\frac{df}{dx}(100,100) = -9$

و بقیه‌ی پیکسل‌ها صفر می‌باشند که در نتیجه‌ی آن مشتق تصویر در جهت x به صورت زیر در خواهد آمد که مقادیر روشنایی پیکسل ها را نیز در بالا محاسبه نمودیم:



در ادامه به محاسبه‌ی مشتق در جهت y می‌پردازیم. برای این منظور باید از h_2 استفاده کنیم. همانند قبل ابتدا h_2 را معکوس می نماییم که به صورت روبرو خواهد شد:

	-1	0	1
-1		۲	۱
0		۰	۰
1	۱	-۲	-۱

حال با توجه به شرایط ذکر شده برای مرحله قبل همین کار را برای این جهت تکرار می‌نماییم:

$$\frac{df}{dy}(x,y) = f(x,y) ** h_2(x,y) = \sum_m \sum_n f(m,n) h_2(x-m, y-n)$$

$$\frac{df}{dy}(50{,}50) = \sum_m \sum_n f(m,n) h_2(50-m, 50-n)$$

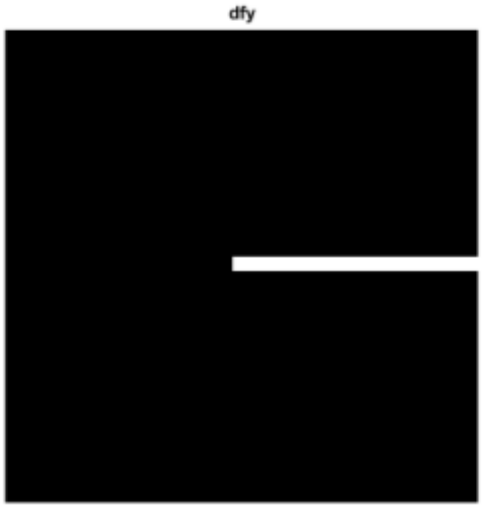
$$\rightarrow \frac{df}{dy}(50{,}50) = f(50{,}50)h_2(0{,}0) + f(50{,}51)h_2(0,-1) + f(51{,}50)h_2(-1{,}0) + f(51{,}51)h_2(-1,-1) =$$

$$= 1 \times 0 + 2 \times 0 + 2 \times 2 + 3 \times 1 = 7$$

پس از محاسبه تمامی مقادیر در تمامی پیکسلها مشاهده می‌نماییم که مشتق در جهت y تنها در پیکسل‌های(49,49:100) و (50,49:100) و (51,49:100) و (100,49:100) مقدار دارد که عبارتند از:

$\frac{df}{dy}(49,49) = 1$	$\frac{df}{dy}(50,49) = 2$	$\frac{df}{dy}(51,49) = 1$	$\frac{df}{dy}(49,52:99) = 8$	$\frac{df}{dy}(100,49) = -2$
$\frac{df}{dy}(49,50) = 4$	$\frac{df}{dy}(50,50) = 7$	$\frac{df}{dy}(51,50) = 3$	$\frac{df}{dy}(50,52:99) = 12$	$\frac{df}{dy}(100,50) = -7$
$\frac{df}{dy}(49,51) = 7$	$\frac{df}{dy}(50,51) = 11$	$\frac{df}{dy}(51,51) = 4$	$\frac{df}{dy}(51,52:99) = 4$	$\frac{df}{dy}(100,51) = -11$
$\frac{df}{dy}(49,100) = 6$	$\frac{df}{dy}(50,100) = 9$	$\frac{df}{dy}(51,100) = 3$	$\frac{df}{dy}(100,52:99) = -12$	$\frac{df}{dy}(100,100) = -9$

و بقیه‌ی پیکسل‌ها نیز صفر می‌باشند. لذا مشتق در جهت y به صورت زیر خواهد بود:



در ادامه باید اندازه گرادیان و زاویه‌ی θ را محاسبه نماییم:

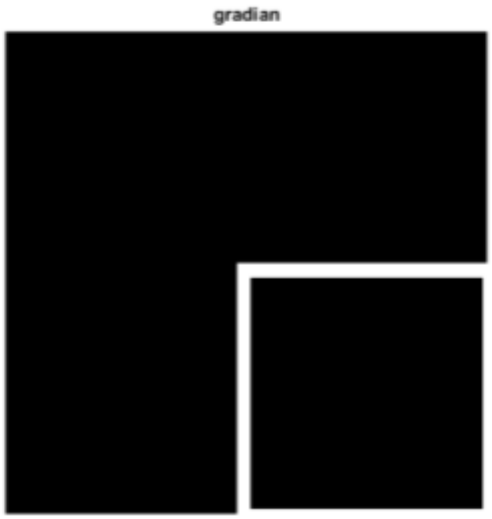
$$|\Delta| = \sqrt{\left(\frac{df}{dx}\right)^2 + \left(\frac{df}{dy}\right)^2}$$

$$\theta = arctan\frac{\frac{df}{dy}}{\frac{df}{dx}}$$

با توجه به مقادیر پیکسل ها در $\frac{df}{dy}$ و $\frac{df}{dx}$ بنابراین اندازه گرادیان در پیکسل‌های (49,49:100) و (50,49:100) و (51,49:100) و (100,49:100) و (52:100,49) و (50, 52:100) و(51, 52:100) (100, 52:100)مقدار دارند که برابرند با:

$\|\Delta(49,49)\| = 1.4142$	$\|\Delta(50,49)\| = 4.4721$	$\|\Delta(51,49)\| = 7.0711$	$\|\Delta(52:99,49)\| = 8$
$\|\Delta(49,50)\| = 4.4721$	$\|\Delta(50,50)\| = 9.8995$	$\|\Delta(51,50)\| = 1.4018$	$\|\Delta(52:99,50)\| = 12$
$\|\Delta(49,51)\| = 7.0711$	$\|\Delta(50,51)\| = 11.4018$	$\|\Delta(51,51)\| = 5.6569$	$\|\Delta(52:99,51)\| = 4$
$\|\Delta(49,52:99)\| = 8$	$\|\Delta(50,52:99)\| = 12$	$\|\Delta(51,52:99)\| = 4$	$\|\Delta(100,49)\| = 6.3246$
$\|\Delta(49,100)\| = 6.3246$	$\|\Delta(50,100)\| = 11.4018$	$\|\Delta(51,100)\| = 11.4018$	$\|\Delta(100,50)\| = 11.4018$
$\|\Delta(100,51)\| = 11.4018$	$\|\Delta(100,52:99)\| = 12$	$\|\Delta(52:99,100)\| = 12$	$\|\Delta(100,100)\| = 12.7279$

بنابراین گرادیان به صورت شکل زیر خواهد بود:

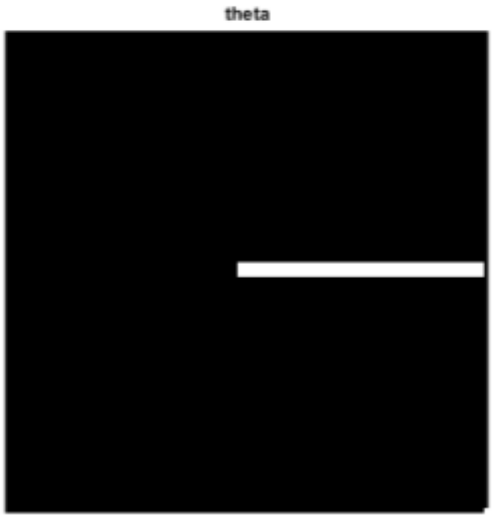


حال به محاسبه‌ی θ می‌پردازیم. در پیکسل‌هایی که $\frac{df}{dy}$ برابر با صفر می‌باشد، θ نیز صفر می‌باشد و در جاهایی که $\frac{df}{dx}$ صفر می‌باشد، θ برابر با ۹۰ درجه خواهد بود. بنابراین θ در پیکسل‌های (49,49:100) و (50,49:100) و (51,49:100) و (100,49:100) مقدار دارد که این مقادیر عبارتند از:

$\theta(49,49) = 45$	$\theta(50,49) = 26.5651$	$\theta(51,49) = 8.1301$	$\theta(100,49) = -18.4349$
$\theta(49,50) = 63.4349$	$\theta(50,50) = 45$	$\theta(51,50) = 15.2551$	$\theta(100,50) = -37.8750$
$\theta(49,51) = 81.8699$	$\theta(50,51) = 74.7449$	$\theta(51,51) = 45$	$\theta(100,51) = -74.7449$
$\theta(49,52:99) = 90$	$\theta(50,52:99) = 90$	$\theta(51,52:99) = 90$	$\theta(100,52:99) = -90$
$\theta(49,100) = -71.5651$	$\theta(50,100) = -52.1250$	$\theta(51,100) = -15.2551$	$\theta(100,100) = 45$

توجه شود که زوایای منفی را از ۱۸۰ کم کرده تا مثبت شوند.

لذا زوایای به دست آمده برای پیکسل‌ها به صورت زیر خواهند شد:



حال باید زوایای به دست آمده را به صورت زیر کوانتیزه نماییم:

$$if \quad \theta < 22.5^{\circ} \quad \rightarrow \quad \hat{\theta} = 0$$

$$if \quad 22.5 \ll \theta < 67.5 \quad \rightarrow \quad \hat{\theta} = 45$$

$$if \quad 67.5 \ll \theta < 112.5 \quad \rightarrow \quad \hat{\theta} = 90$$

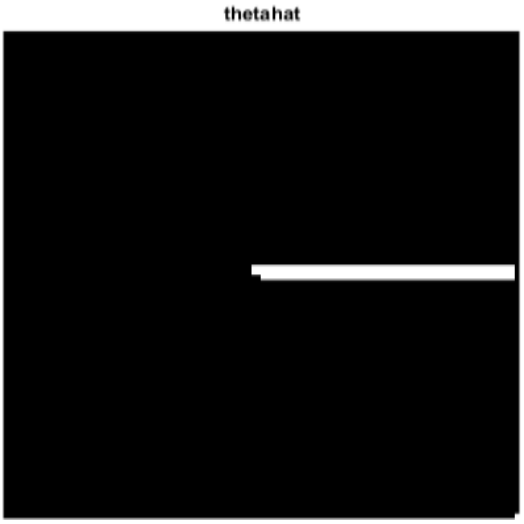
$$if \quad 112.5 \ll \theta < 157.5 \quad \rightarrow \quad \hat{\theta} = 135$$

$$if \quad 157.5 \ll \theta < 180 \quad \rightarrow \quad \hat{\theta} = 0$$

لذا زوایا به صورت زیر خواهند شد:

$\hat{\theta}(49,49) = 45$	$\hat{\theta}(50,49) = 45$	$\hat{\theta}(51,49) = 0$	$\hat{\theta}(100,49) = 0$
$\hat{\theta}(49,50) = 45$	$\hat{\theta}(50,50) = 45$	$\hat{\theta}(51,50) = 0$	$\hat{\theta}(100,50) = 135$
$\hat{\theta}(49,51) = 90$	$\hat{\theta}(50,51) = 90$	$\hat{\theta}(51,51) = 45$	$\hat{\theta}(100,51) = 90$
$\hat{\theta}(49,52:99) = 90$	$\hat{\theta}(50,52:99) = 90$	$\hat{\theta}(51,52:99) = 90$	$\hat{\theta}(100,52:99) = 90$
$\hat{\theta}(49,100) = 90$	$\hat{\theta}(50,100) = 135$	$\hat{\theta}(51,100) = 0$	$\hat{\theta}(100,100) = 45$

که در نتیجه آن $\hat{\theta}$ به صورت زیر خواهد شد:



در مرحله‌ی بعد باید عملیات NON-Maximum Suppression را به صورت زیر انجام دهیم:

$$if \;\; \hat{\theta}(x,y) = 0 \;\; \rightarrow \;\; if\; \Delta(x,y) > \Delta(x-1,y) \;\; \& \;\; \Delta(x,y) > \Delta(x+1,y) \;\; \rightarrow \;\; M_f(x,y) = \Delta(x,y)$$

$$if \;\; \hat{\theta}(x,y) = 45 \;\; \rightarrow \;\; if\; \Delta(x,y) > \Delta(x-1,y-1) \;\; \& \;\; \Delta(x,y) > \Delta(x+1,y+1) \;\; \rightarrow \;\; M_f(x,y) = \Delta(x,y)$$

$$if \;\; \hat{\theta}(x,y) = 90 \;\; \rightarrow \;\; if\; \Delta(x,y) > \Delta(x,y-1) \;\; \& \;\; \Delta(x,y) > \Delta(x,y+1) \;\; \rightarrow \;\; M_f(x,y) = \Delta(x,y)$$

$$if \;\; \hat{\theta}(x,y) = 135 \;\; \rightarrow \;\; if\; \Delta(x,y) > \Delta(x+1,y-1) \;\; \& \;\; \Delta(x,y) > \Delta(x-1,y+1) \;\; \rightarrow \;\; M_f(x,y) = \Delta(x,y)$$

لذا داریم:

$$\hat{\theta}(51{,}49) = 0 \;\; \& \; \Delta(51{,}49) = 7.0711 \;\; \& \;\; \Delta(50{,}48) = 0 \;\; \& \Delta(52{,}50) = 12$$

$$\rightarrow \;\; \Delta(51{,}49) < \Delta(52{,}50) \;\; \rightarrow \;\; Mf(51{,}49) = 0$$

$$\hat{\theta}(51{,}50) = 0 \;\; \& \; \Delta(51{,}50) = 1.4018 \;\; \& \;\; \Delta(50{,}49) = 4.4721 \;\; \& \Delta(52{,}51) = 4$$

$$\rightarrow \;\; \Delta(51{,}50) < \Delta(50{,}549) \;\; \& \;\; \Delta(51{,}50) < \Delta(52{,}51) \;\; \rightarrow \;\; Mf(51{,}50) = 0$$

$$\hat{\theta}(50{,}50) = 45 \;\; \& \; \Delta(50{,}50) = 9.8995 \;\; \& \;\; \Delta(51{,}51) = 5.6569 \;\; \& \Delta(49{,}49) = 1.4142$$

$$\rightarrow \;\; \Delta(50{,}50) > \Delta(50{,}549) \;\; \& \;\; \Delta(50{,}50) < \Delta(49{,}49) \;\; \rightarrow \;\; Mf(50{,}50) = \Delta(50{,}50) = 9.8995$$

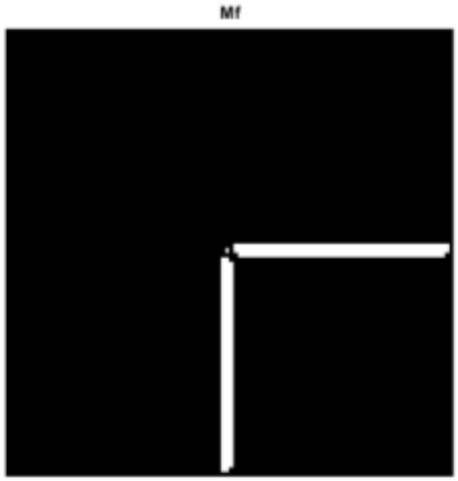
$$\hat{\theta}(100{,}51) = 90 \;\; \& \; \Delta(100{,}51) = 11.4018 \;\; \& \;\; \Delta(100{,}50) = 11.4018 \;\; \& \Delta(100{,}52) = 12$$

$$\rightarrow \;\; \Delta(100{,}51) < \Delta(100{,}52) \;\; \rightarrow \;\; Mf(100{,}51) = 0$$

به همین ترتیب برای تمامی زوایا مقدار Mf را حساب می‌نماییم که برابر است با:

$Mf(49{,}49{:}\,51) = 0$	$Mf(51{,}49{:}\,52) = 0$
$Mf(49{,}52{:}\,99) = 8$	$Mf(51{,}53{:}\,98) = 4$
$Mf(49{,}100) = 0$	$Mf(51{,}99{:}\,100) = 0$
$Mf(50{,}49) = 0$	$Mf(52{:}\,99{,}49) = 8$
$Mf(50{,}50) = 9.8995$	$Mf(52{:}\,99{,}50) = 12$
$Mf(50{,}51) = 0$	$Mf(52{:}\,98{,}51) = 4$
$Mf(50{,}52{:}\,99) = 12$	$Mf(99{,}51) = 0$
$Mf(50{,}100) = 0$	$Mf(100{,}49{:}\,51) = 0$

و نتیجه به صورت شکل زیر خواهد بود:



تصویر Mf شامل تمام پیکسل‌هایی است که احتمال لبه بودن را دارند. حال یک حد بالا و یک حد پایین در نظر می‌گیریم. حد بالا را ۱,۹ و حد پایین را نصف ان یعنی ۰,۹۵ در نظر می‌گیریم و یک تصویر R می‌سازیم که به صورت زیر ساخته شده است:

$$if\ Mf(x,y) \gg HT\ \rightarrow R(x,y) = 1$$

بنابراین داریم:

$$Mf(49{,}52{:}99) = 8 \quad \rightarrow \quad R(49{,}52{:}99) = 1$$
$$Mf(50{,}50) = 9.8995 \quad \rightarrow \quad \mathrm{R}(50{,}50) = 1$$
$$Mf(50{,}52{:}99) = 12 \quad \rightarrow \quad R(50{,}52{:}99) = 1$$
$$Mf(51{,}53{:}98) = 4 \quad \rightarrow \quad R(51{,}53{:}98) = 1$$
$$Mf(52{:}99{,}49) = 8 \quad \rightarrow \quad R(52{:}99{,}49) = 1$$
$$Mf(52{:}99{,}50) = 12 \quad \rightarrow \quad R(52{:}99{,}50) = 1$$
$$Mf(52{:}98{,}51) = 4 \quad \rightarrow \quad R(52{:}98{,}51) = 1$$

حال در اطراف $Mf(x,y)$ می‌گردیم و می‌بینیم آیا $Mf(x,y)$ ای وجود دارد که از LT و یا همان حد پایین بزرگ تر باشد. آن را هم در R برابر با یک می‌گیریم به صورت زیر:

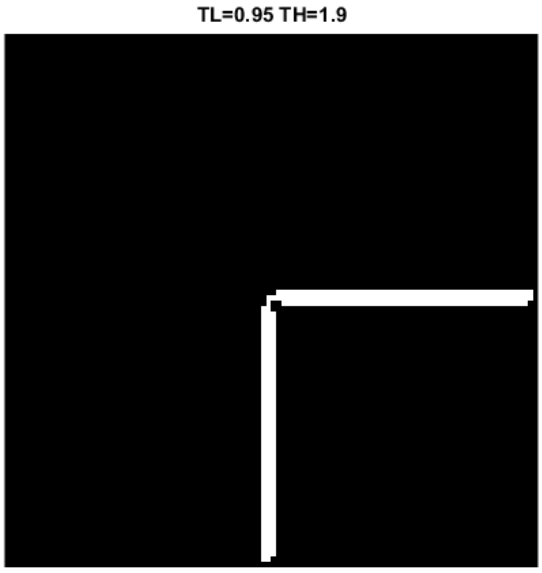
$$if\ \ \hat{\theta}(x,y) = 0 \quad \rightarrow \quad if\ Mf(x-1,y) > TL\ \ \&\ \ Mf(x+1,y) > TL\ \rightarrow\ R(x,y) = 1$$
$$if\ \ \hat{\theta}(x,y) = 45\ \rightarrow if\ Mf(x-1,y-1) > TL\ \ \&\ \ Mf(x+1,y+1) > TL\ \rightarrow\ R(x,y) = 1$$
$$if\ \ \hat{\theta}(x,y) = 90 \quad \rightarrow \quad if\ Mf(x,y-1) > TL\ \ \&\ \ Mf(x,y+1) > TL \quad \rightarrow\ R(x,y) = 1$$
$$if\ \ \hat{\theta}(x,y) = 135\ \rightarrow \quad if\ Mf(x-1,y+1) > TL\ \ \&\ \ Mf(x+1,y-1) > TL\ \rightarrow\ R(x,y) = 1$$

داریم:

$\hat{\theta}(50,51) = 90 \quad \& \ Mf(50,50) = 9.8995 > TL \ \& Mf(50,52) = 12 > TL \quad \rightarrow \quad R(50,51) = 1$

$\hat{\theta}(51,50) = 0 \quad \& \ Mf(50,50) = 9.8995 > TL \ \& Mf(52,50) = 12 > TL \quad \rightarrow \quad R(51,50) = 1$

و در نهایت R به صورت زیر خواهد شد:



ب) (کد قسمت ب و ج در HW6_Q2_b قرار دارد)

در این قسمت قرار است الگوریتم کنی را در متلب پیاده سازی نماییم. با توجه به توضیحات مربوط به محاسبات دستی که در بالا به آن اشاره شد، کد مربوطه مثلا برای تصویر بالا به صورت زیر خواهد شد:

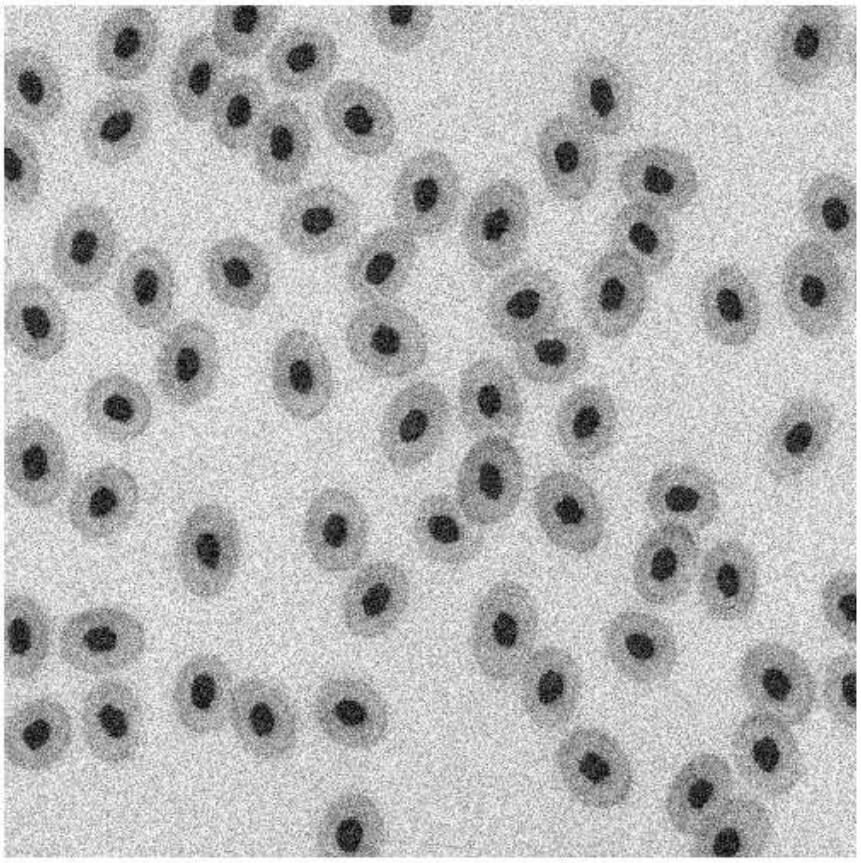
```
f=zeros(100,100);
f(1:100,1:49)=0;
f(1:49,49:100)=0;
f(50,50)=1;
f(50,51:100)=2;
f(51:100,50)=2;
f(51:100,51:100)=3;
figure
imshow(f);
[M,N]=size(f);
dx = [-1 0 1; -2 0 2; -1 0 1];
dy =dx';

dfx=imfilter(f,dx);
dfy=imfilter(f,dy);

%%%%%%%%%%%%%%%% finding theta and gradian %%%%%%%%%%%%%
for m=1:M
    for n=1:N
        theta(m,n)=atan(dfy(m,n)/dfx(m,n));
        T(m,n)=(180/pi)*theta(m,n);
```

```
        gradian(m,n)=sqrt((dfx(m,n))^2+(dfy(m,n))^2);
    end
end

%%%%%%%%%%%%%%%%%%%%%% theta quantization and finding thetahat
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for m=1:M
    for n=1:N
         if     T(m,n)<22.5,                    T1(m,n)=0;
         elseif T(m,n)>=22.5  & T(m,n)<67.5,   T1(m,n)=45;
         elseif T(m,n)>=67.5  & T(m,n)<112.5,  T1(m,n)=90;
         elseif T(m,n)>=112.5 & T(m,n)<157.5,  T1(m,n)=135;
         elseif T(m,n)>=157.5 & T(m,n)<180,    T1(m,n)=0;
         end
    end
 end

 %%%%%%%%%%%%%%%%%%%%%%% non-maximum supression %%%%%%%%%%%%%%%%%%%%%%%%%%%%
 Mf=zeros(M,N);
 for m=2:M-1
    for n=2:N-1
            if T1(m,n)==0    & gradian(m,n)>=gradian(m-1,n)    &
gradian(m,n)>=gradian(m+1,n),  Mf(m,n)=gradian(m,n);end
            if T1(m,n)==45   & gradian(m,n)>=gradian(m-1,n-1) &
gradian(m,n)>=gradian(m+1,n+1),Mf(m,n)=gradian(m,n);end
            if T1(m,n)==90   & gradian(m,n)>=gradian(m,n-1)    &
gradian(m,n)>=gradian(m,n+1),  Mf(m,n)=gradian(m,n);end
            if T1(m,n)==135  & gradian(m,n)>=gradian(m-1,n+1) &
gradian(m,n)>=gradian(m+1,n-1),Mf(m,n)=gradian(m,n);end
    end
 end

 %%%%%%%%%%%%%%%%%%%%%%%%% edge detection %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 R=zeros(M,N);
TH=1.9;
TL=TH/2;
 for m=1:M
    for n=1:N
            if Mf(m,n)>=TH
                R(m,n)=1;
            end
    end
 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for m=2:M-1
    for n=2:N-1
            if T1(m,n)==0    & Mf(m-1,n)>TL    & Mf(m+1,n)>TL   ,  R(m,n)=1;end
            if T1(m,n)==45   & Mf(m-1,n-1)>TL  & Mf(m+1,n+1)>TL , R(m,n)=1;end
            if T1(m,n)==90   & Mf(m,n-1)>TL    & Mf(m,n+1)>TL   ,  R(m,n)=1;end
            if T1(m,n)==135  & Mf(m-1,n+1)>TL  & Mf(m+1,n-1)>TL , R(m,n)=1;end
    end
 end
 figure,imshow(R),title(['TL=',num2str(TL),' TH=',num2str(TH)]);
```

ج)

در این قسمت تصویر تعدادی از سلول‌های خونی داده شده و قرار است این تصویر با یک نویز گوسی جمع شده و از طریق الگوریتم کنی که در بالا پیاده سازی نمودیم، سلول‌ها را با محیط بیرونیشان از یکدیگر جدا نماییم:

پس از خاکستری نمودن تصویر و اضافه کردن نویز، تصویر زیر را خواهیم داشت:



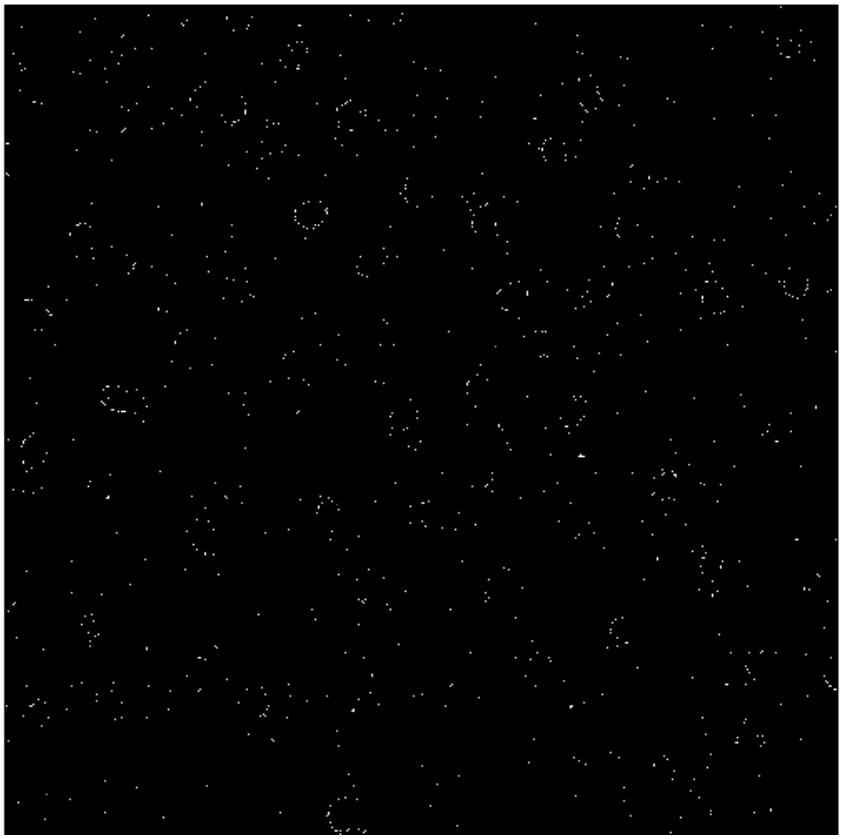
ابتدا تصویر را از یک فیلتر گوسی عبور داده و سپس الگوریتم کنی را برای آن پیاده سازی نموده و پارامترها را طوری تنظیم کرده که بهترین نتیجه حاصل شود.

```
ff=imread('bloodcel_95.jpg');
ff=rgb2gray(ff);
ff=im2double(ff);
[M,N]=size(ff);
g=ff+0.1*randn(M,N);
figure
imshow(g);
w=4;
Gause=fspecial('gaussian',w,w/6);
f=imfilter(g,Gause);
figure,imshow(f,[]),title('filtered image');

[M,N]=size(f);
dx = [-1 0 1; -2 0 2; -1 0 1];
dy =dx';

dfx=imfilter(f,dx);
dfy=imfilter(f,dy);

%%%%%%%%%%%%%%%%% finding theta and gradian %%%%%%%%%%%%%
for m=1:M
    for n=1:N
        theta(m,n)=atan(dfy(m,n)/dfx(m,n));
        T(m,n)=(180/pi)*theta(m,n);
        gradian(m,n)=sqrt((dfx(m,n))^2+(dfy(m,n))^2);
    end
```

```
end

%%%%%%%%%%%%%%%%%%%%%% theta quantization and finding thetahat
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for m=1:M
    for n=1:N
         if     T(m,n)<22.5,                    T1(m,n)=0;
         elseif T(m,n)>=22.5  & T(m,n)<67.5,   T1(m,n)=45;
         elseif T(m,n)>=67.5  & T(m,n)<112.5,  T1(m,n)=90;
         elseif T(m,n)>=112.5 & T(m,n)<157.5,  T1(m,n)=135;
         elseif T(m,n)>=157.5 & T(m,n)<180,    T1(m,n)=0;
         end
    end
 end

 %%%%%%%%%%%%%%%%%%%%%%% non-maximum supression %%%%%%%%%%%%%%%%%%%%%%%%%%
 Mf=zeros(M,N);
 for m=2:M-1
    for n=2:N-1
            if T1(m,n)==0    & gradian(m,n)>=gradian(m-1,n)    &
gradian(m,n)>=gradian(m+1,n),  Mf(m,n)=gradian(m,n);end
            if T1(m,n)==45   & gradian(m,n)>=gradian(m-1,n-1) &
gradian(m,n)>=gradian(m+1,n+1),Mf(m,n)=gradian(m,n);end
            if T1(m,n)==90   & gradian(m,n)>=gradian(m,n-1)    &
gradian(m,n)>=gradian(m,n+1),  Mf(m,n)=gradian(m,n);end
            if T1(m,n)==135  & gradian(m,n)>=gradian(m-1,n+1) &
gradian(m,n)>=gradian(m+1,n-1),Mf(m,n)=gradian(m,n);end
    end
 end

 %%%%%%%%%%%%%%%%%%%%%%%% edge detection %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 R=zeros(M,N);
TH=0.8;
TL=0.7;
 for m=1:M
    for n=1:N
            if Mf(m,n)>=TH
                R(m,n)=1;
            end
    end
 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for m=2:M-1
    for n=2:N-1
            if T1(m,n)==0    & Mf(m-1,n)>TL    & Mf(m+1,n)>TL   ,  R(m,n)=1;end
            if T1(m,n)==45   & Mf(m-1,n-1)>TL  & Mf(m+1,n+1)>TL , R(m,n)=1;end
            if T1(m,n)==90   & Mf(m,n-1)>TL    & Mf(m,n+1)>TL   ,  R(m,n)=1;end
            if T1(m,n)==135  & Mf(m-1,n+1)>TL  & Mf(m+1,n-1)>TL , R(m,n)=1;end
    end
 end
 figure,imshow(R),title(['canny ede detection with''TL=',num2str(TL),'
TH=',num2str(TH)]);
```

حال نتایج را برای آستانه‌های مختلف مشاهده می‌نماییم. در ابتدا TH را یک در نظر گرفته و TL=TH/2 (بهتر است که TLنصفTH باشد) نتیجه به صورت زیر است:

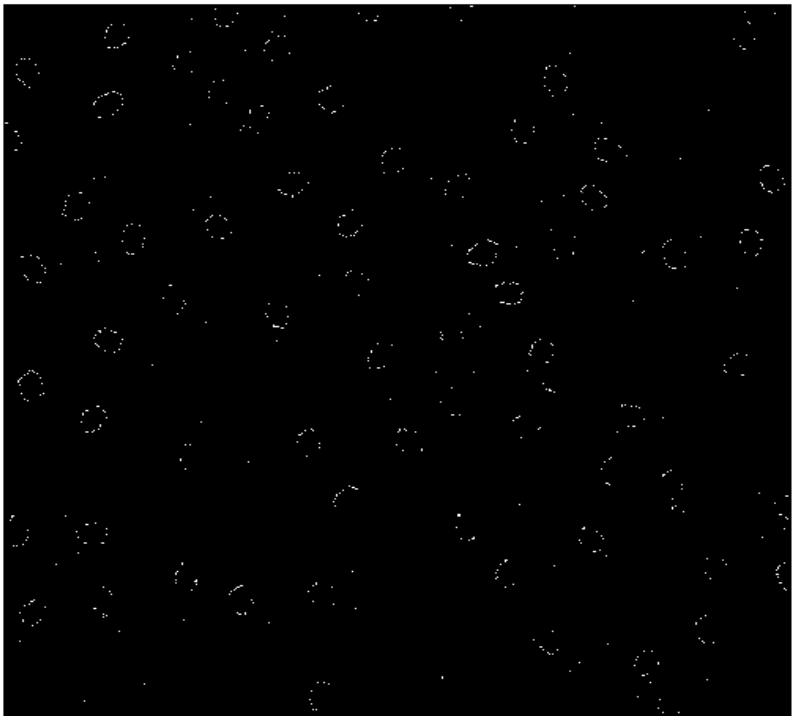
TL=0.5 TH=1



مشاهده می‌شود که بسیاری از لبه‌های ضعیف حذف شده‌اند که این نشان می‌دهد مقدارTH را زیاد در نظر گرفته ایم.
همچنین نویزها هم به عنوان لبه آشکار شده اند این موضوع نیز نشان می‌دهد که TL را مقدار پایینی گرفته ایم که نویزها را هم به عنوان لبه در نظر گرفته است.

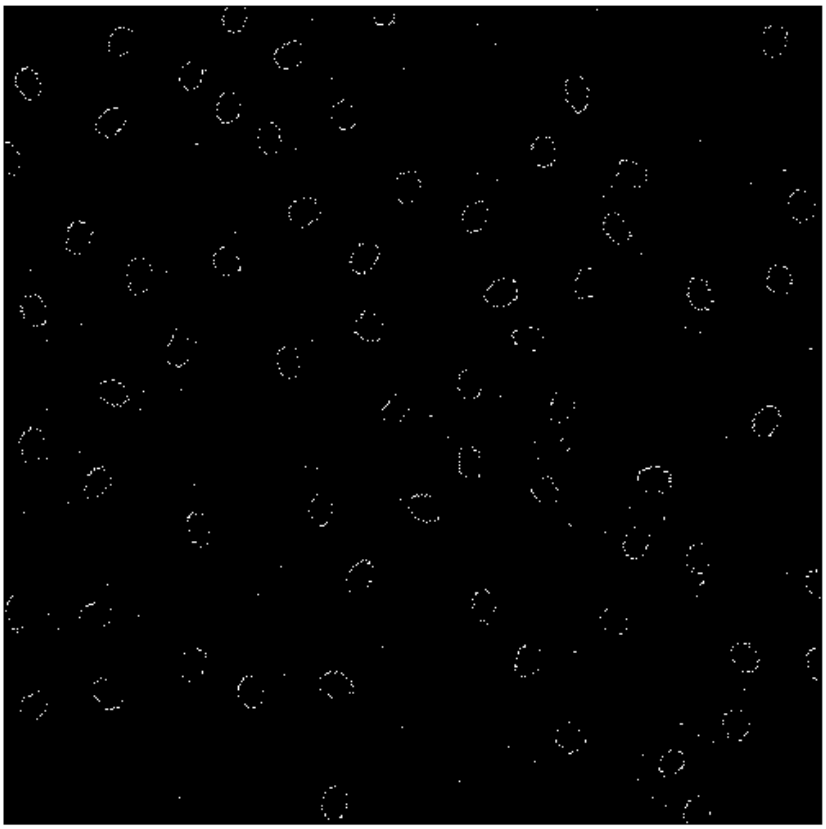
لذا TH را اندکی کاهش و TL را قدری افزایش می‌دهیم نتیجه به صورت زیر خواهد بود:

TL=0.6 TH=0.9



محیط سلول‌ها که در انتخاب قبلی لبه‌ی ضعیف تلقی شده و حذف شده بود بیشتر آشکار گردید ولی همچنان مقداری نویط داریم و محیط سلول‌ها هم هنوز مقداری مبهم است لذل مجددا TH را کاهش و TL را افزایش می‌دهیم و نتیجه زیر حاصل می‌شود:

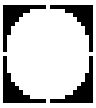
TL=0.7 TH=0.8



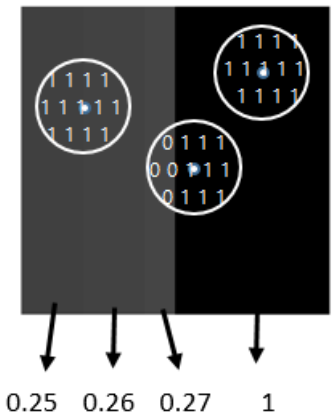
مشاهده می‌شود که تقریبا نتیجه بهتر شد به ازای TH=0.8 و TL=0.7. توجه شود که اگر TH را خیلی پایین نیز بگیریم مجددا نویزها را هم به عنوان لبه در نظر گرفته و نتیجه خراب می‌شود. لذا بهترین نتیجه به ازای TH=0.8 و TL=0.7 به دست آمد که به صورت supervision با TH=1 و TL=TH/2 شروع کردیم و براساس مشاهدات آنها را تغییر داده تا به بهترین جواب برسیم.

ج) (کد این قسمت در HW6_Q2_c قرار دارد)

در این قسمت قرار است که از الگوریتم susan به منظور شناسایی لبه‌های تصویر قبل استفاده نماییم. در روش susan از یک nucleus و یا هسته به صورت زیر استفاده می‌نماییم:



و سطح روشنایی پیکسل مرکزی هسته را $I(x_0, y_0)$ می‌نامیم. سپس هسته سطح تصویری که می‌خواهیم لبه‌های آن را آشکارسازی نماییم را جاروب می‌کند و پیکسل مرکزی هسته با پیکسل مورد نظرمان که روشنایی آن را با $I(x, y)$ نمایش می‌دهیم مقایسه می‌شود. اگر اختلاف سطح روشنایی این دو پیکسل از یک سطح آستانه‌ای کوچکتر باشد، مقدار آن را در مکان همان پیکسل و در یک هسته به نام C، ۱ می‌گذاریم در غیر این صورت صفر. مثلا تصویر زیر را در نظر بگیرید که میزان روشنایی پیکسل‌ها روی ان مشخص شده است:



اختلاف روشنایی های نیمه چپ تصویر در حد ۰,۰۱ می‌باشد.و با سمت راست که یک لبه قوی است اختلاف بیشتری دارند. همچنین پیکسل‌های سمت راست تصویر نیط املا یکنواختند. لذا اگر در این تصویر آستانه‌ی مورد نظر را برای نمونه ۰,۱ فرض

نماییم، مادامی که هسته موردنظر به طور کامل در سمت چپ تصویر و یا سمت راست تصویر می‌باشد، اختلاف سطح روشنایی پیکسل مرکزی هسته و پیکسل‌های دیگر در هیچ حالتی از ۰,۱ بیشتر نخواهد بود و لذا مقادیر همه پیکسل ها در C برابر با یک خواهد بود. اما به محض اینکه هسته وارد نیمه راست تصویر شود که یک لبه‌ی قوی است، از آنجا که اختلاف روشنایی پیکسل مرکزی و تعدادی از زپیکسل‌ها که در طرف دیگر قرار گرفته اند زیاد می‌شود یعنی از آستانه ما بیشتر می‌شود، لذا مقدار معادل صفر برای آن‌ها در نظر گرفته می‌شود.

سپس در ادامه تعداد کل یک های C را شمرده و آن را n می‌نامیم. این n به صورت زیر با n_max مقایسه می شود و R را تشکیل داده و به این صورت لبه های تصویر براساس حد استانه ما مشخص می شوند.

$$R(x_0,y_0)=\begin{cases}0.75n_{max}-n & if\ n<0.75n_{max}\\ 0 & else\end{cases}$$

بنابراین در الگوریتم susan ابعاد هسته و همچنین سطح آستانه می‌توانند نتایج متفاوتی را برای ما بدست دهند.

حال مجددا تصویر سلول‌های خونی را در نظر گرفته و الگوریتم susan را برای لبه‌یابی آن امتحان می‌نماییم.

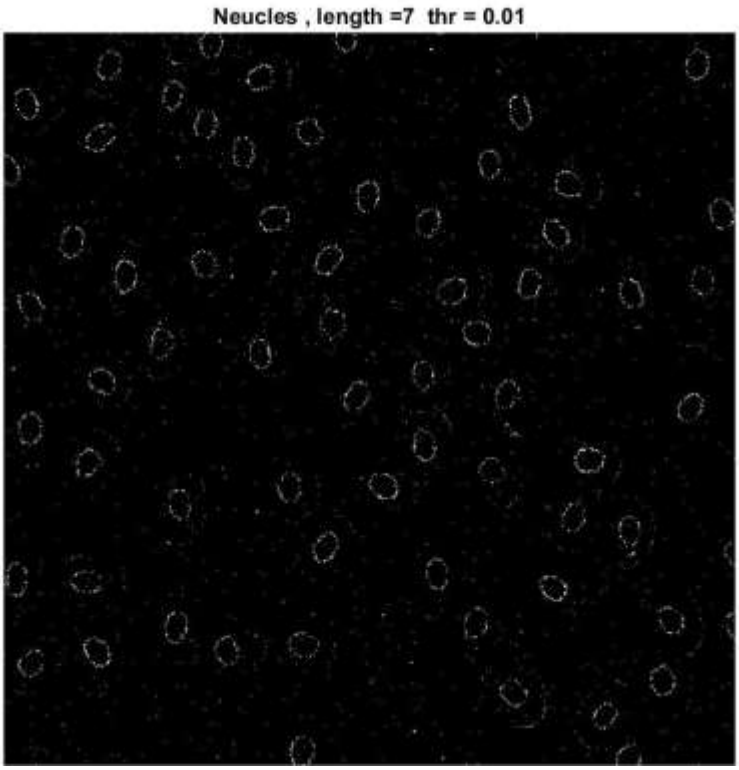
کد برنامه به صورت زیر می‌باشد:

```
f = imread('bloodcel_95.jpg');
f = rgb2gray(f);
f = im2double(f);
[M,N] = size(f);
f = f + .01.*randn(M,N);

figure,imshow(f), title('Original Image')

%%%%%% SUSSAN
for w = 11:1:11
    for thr_sussan =.105:.005:.105

        N_Length = 2*w + 1;

        %%%% zero-pad f
        ff = zeros(M+2*w , N+2*w);
        ff(w+1:end-w , w+1:end-w) = f;


        Nucleus = zeros(N_Length , N_Length);

        for m=-w:w
            for n=-w:w
                r = sqrt(m.^2 + n.^2);
                if ceil® <= w
                    Nucleus(m+w+1,n+w+1) = 1;
                end
            end
        end
         figure, imshow(Nucleus);

        n_max = length(find(Nucleus));
        p =0;
        for m=w+1:M-(w+1)
            p=p+1; q=0;
            for n=w+1:N-(w+1)
                q=q+1;
                center_pixel = ff(m,n);

                temp = zeros(N_Length,N_Length);
```

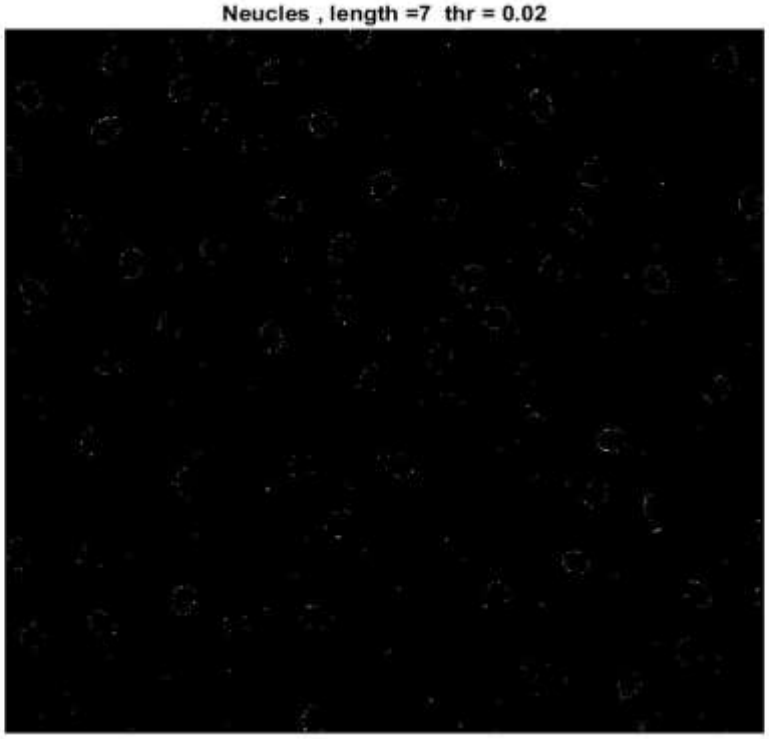
```
                for mm = -w:w
                    for nn = -w:w
                        temp(mm+w+1,nn+w+1) = Nucleus(mm+w+1,nn+w+1) .*
abs(center_pixel - ff(m+mm,n+nn));
                    end
                end
                temp_2(p,q) = length(find(temp < thr_sussan));


            end
        end
        temp_3 = .75*n_max - temp_2;
        ind = find(temp_2 < .75*n_max);
        [MM,NN] = size(temp_2);
        temp_4 = zeros(MM,NN);
        temp_4(ind) = temp_3(ind);

        fig=figure;
        imshow(temp_3),title(['Neucles , length =',num2str(N_Length),'  thr =
',num2str(thr_sussan)]);
        name =
29print('sussan_Length_%d__Thr_%s.jpg',N_Length,num2str(thr_sussan));
        saveas(fig,name);
        g = temp_4;
        save('f_Wlennght_23_Thr_.105.mat','g');
    end
end
```

قرار است سلول‌ها را با محیط بیرونیشان نمایش دهیم. نتایج برای ابعاد مختلف هسته و آستانه‌های مختلف به صورت زیر است:

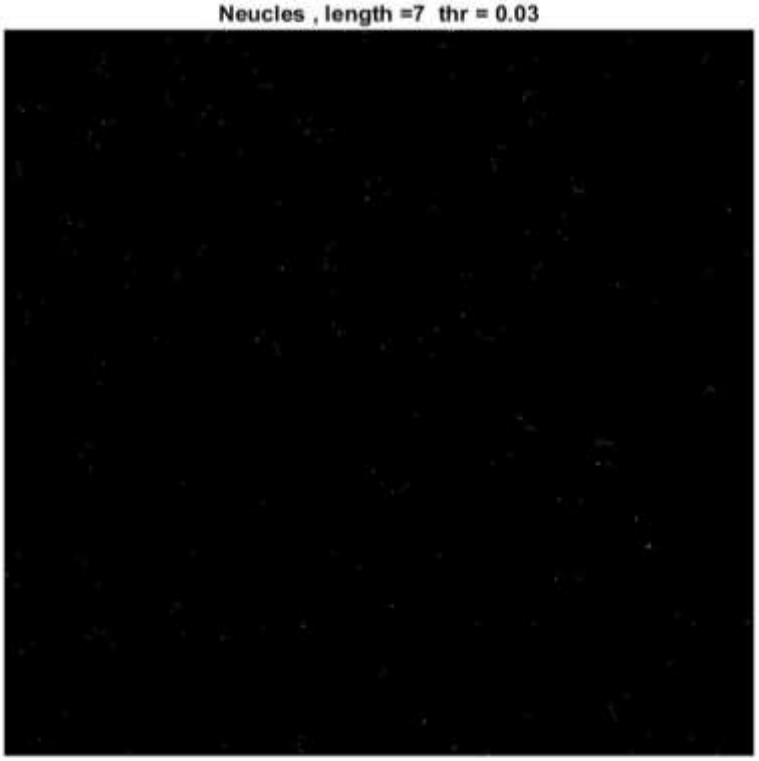
برای هسته با ابعاد 7*7 و آستانه‌های متفاوت نتایج زیر را داریم:



Neucles , length =7 thr = 0.02



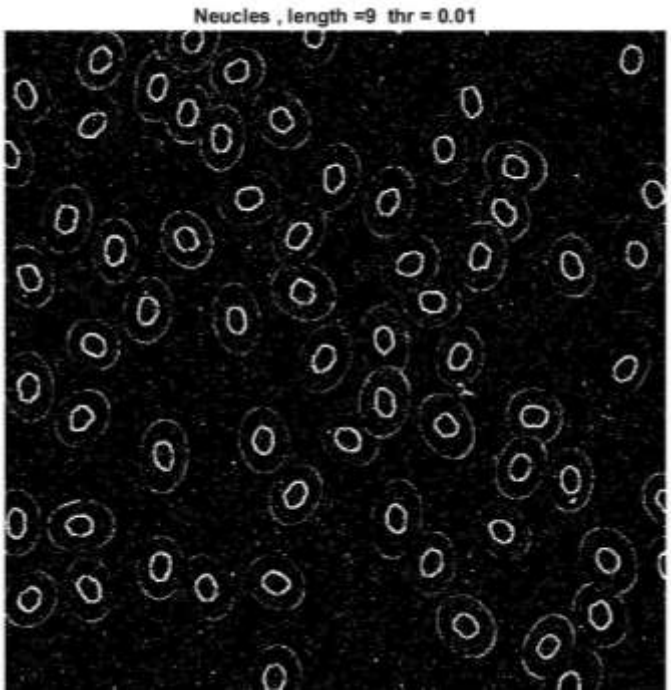
Neucles , length =7 thr = 0.03



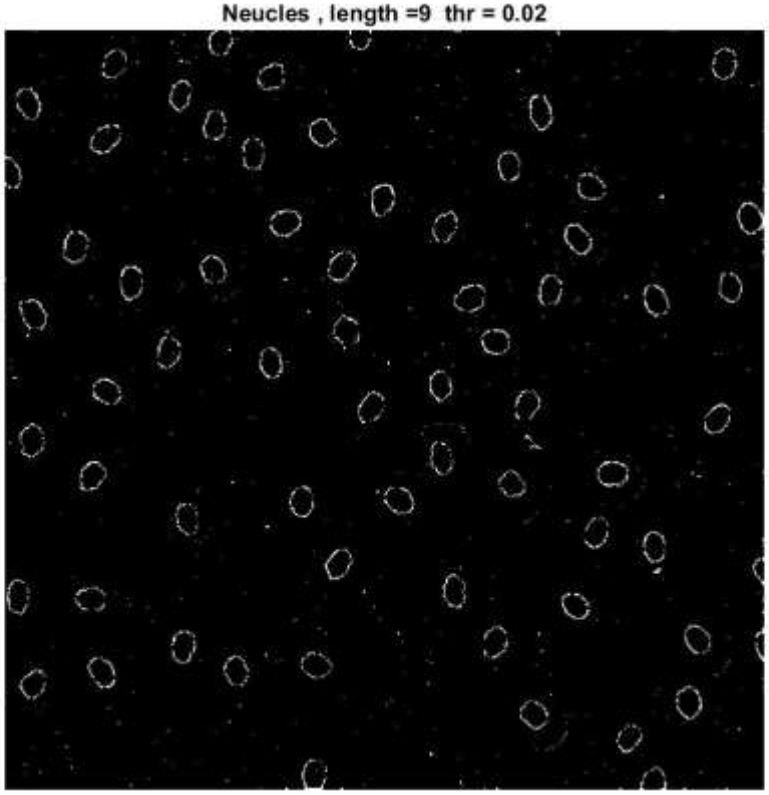
مشاهده می‌شود که با ابعاد هسته ی 7*7 هرچه آستانه افزایش می‌یابد، از آنجایی که الگوریتم به دنبال اختلاف سطح روشنایی بیشتر می‌گردد(به اندازه‌ی آستانه)، لذا لبه‌ها به عنوان لبه‌ی ضعیف در نظر گرفته شده و آشکارسازی نمی شوند.

حال ابعاد پنجره را افزایش می‌دهیم.

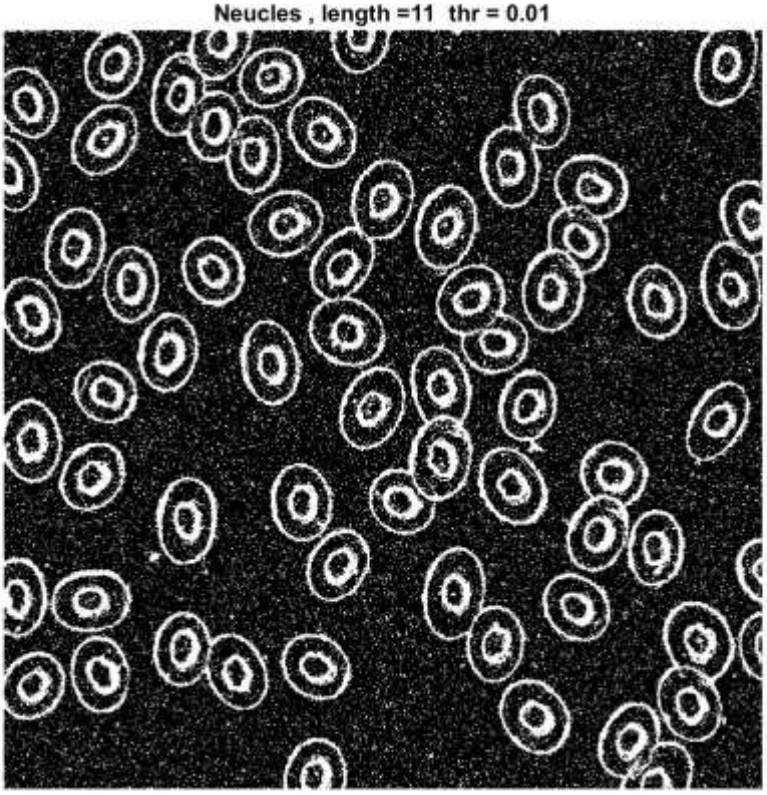
برای پنجره با ابعاد ۹:



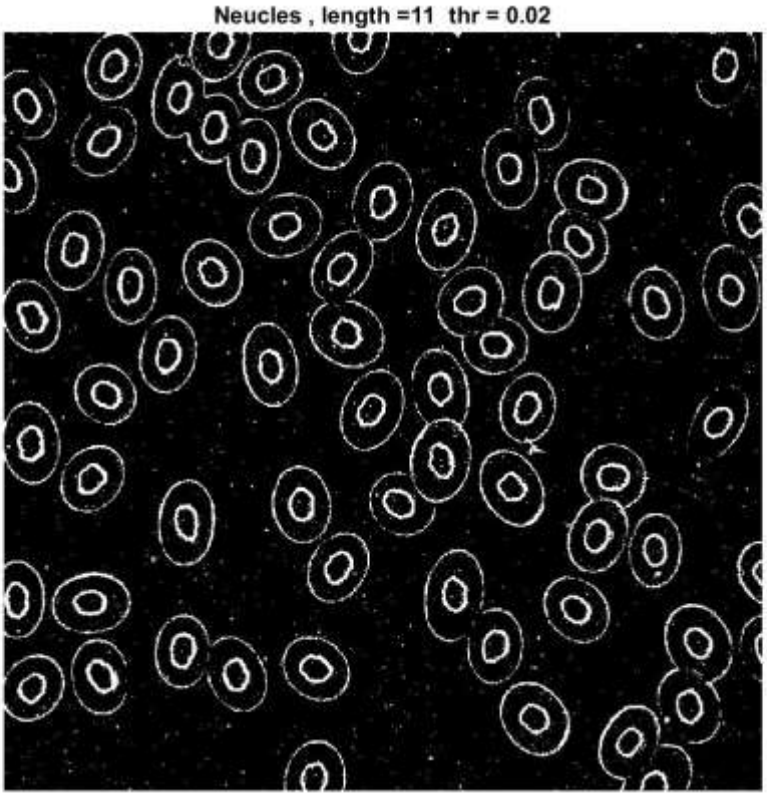
Neucles , length =9 thr = 0.02



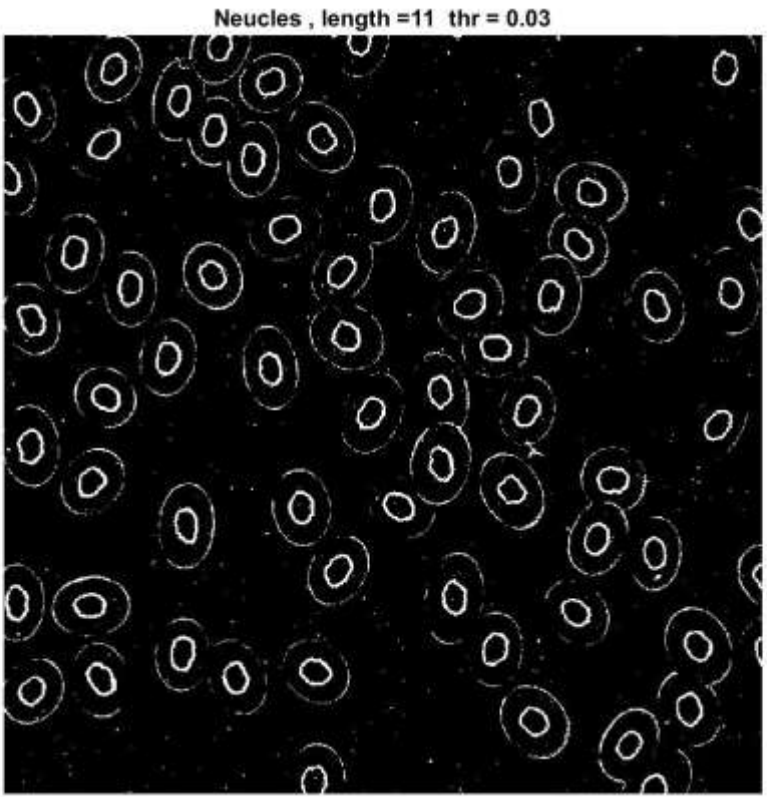
Neucles , length =11 thr = 0.01



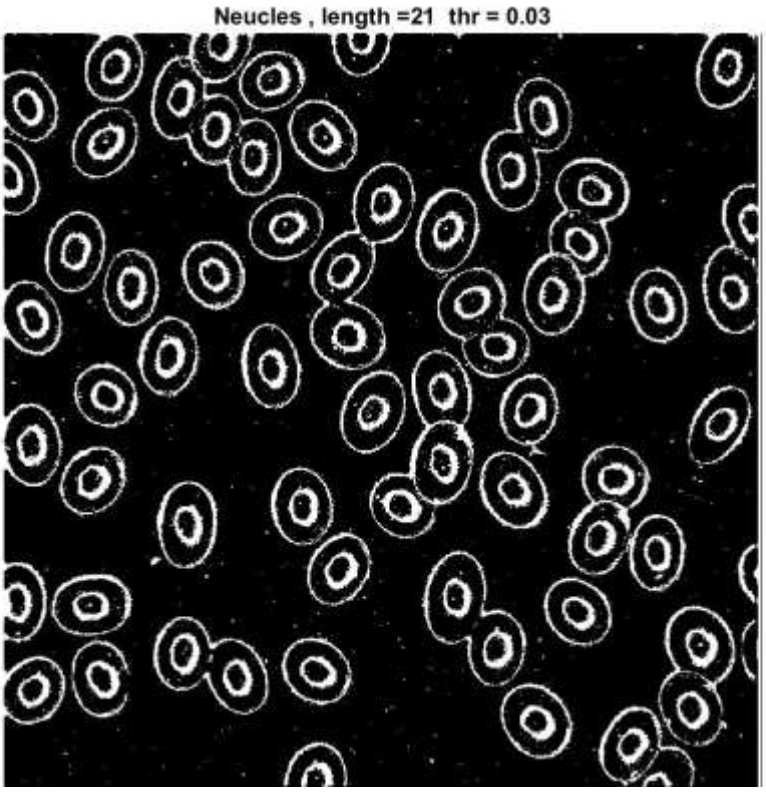
مشاهده می شود که با افزایش ابعاد هسته نتیجه بسیار بهبود می یابد. برای هسته با بعد ۱۱، محیط های سلول‌ها به خوبی مشخص شده اند اما نویزها نیز آشکار سازی شده اند. لذا لازم است آستانه را افزایش دهیم تا حدی که نویزها به نحو قابل قبولی از بین روند.

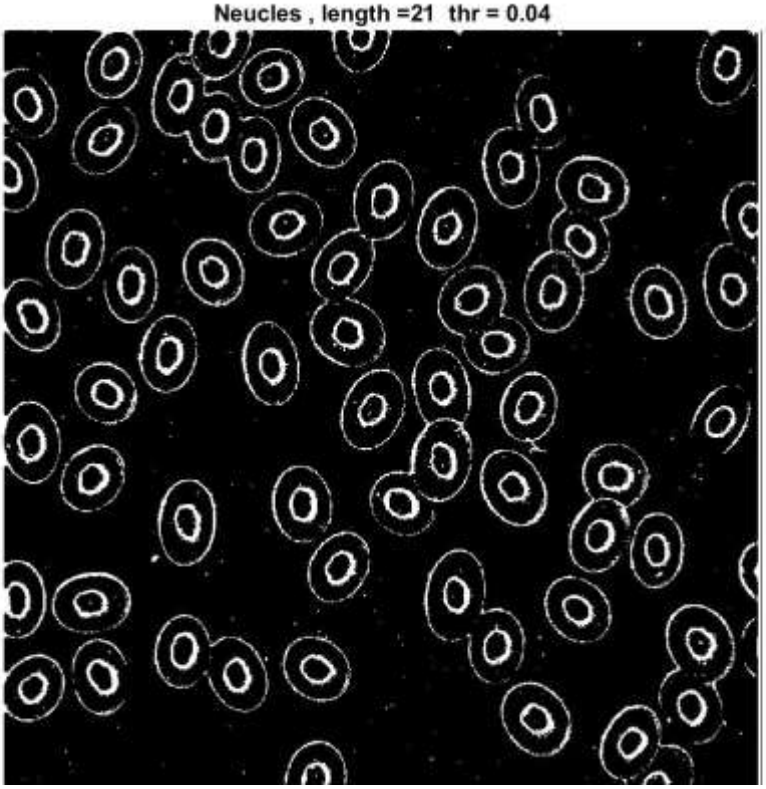


مشاهده می‌شود که با افزایش آستانه به0.02 برای بعد ۱۱، محیط‌های سلول‌ها به خوبی مشخص بوده و از طرفی نویزها نیز تا حد مناسبی حذف شده اند. اگر آستانه را دوباره افزایش دهیم ممکن است لبه‌ها نیز حذف شوند که در تصویر زیر مشاهده می نمایید :



با اضافه کردن ابعاد پنجره تا ۲۱ نتیجه زیر را داریم:





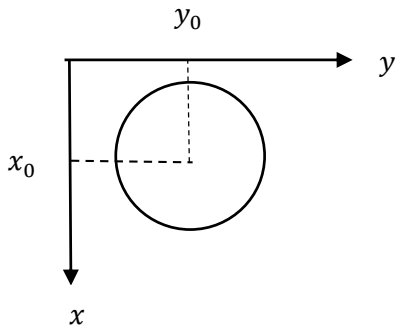
بنابراین با افزایش ابعاد هسته محیط سلول‌ها بسیار بهتر آشکار می‌گردد. همچنین هرچه آستانه افزایش می‌یابد بسیاری از لبه‌ها که شامل محیط سلول‌ها می‌باشد که قرار است آشکارسازی شوند به عنوان لبه‌ی ضعیف قلمداد شده و حذف می‌شوند. لذا باید استانه را در حدی در نظر گرفت که همزمان هم نویزها تا حد قابل قبولی حذف شوند و هم محیط سلول‌ها از بین نرود. جواب برای هسته با بعد ۲۱ و آستانه‌ی ۰,۰۴ قابل قبول می‌باشد که با سعی و خطا و مقایسه‌ی نتایج به دست آمد.

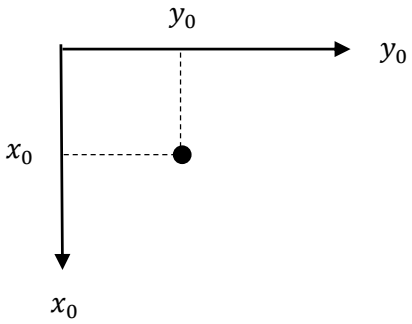
د) (کد این قسمت در HW6_Q2_c قرار دارد)

در این قسمت قرار است از الگوریتم هاف برای آشکارسازی دایره استفاده نموده و هر سلول را با مجموعه ای از پیکسل‌های ۱ به کمک مرکز دوایر مشخص نماییم.

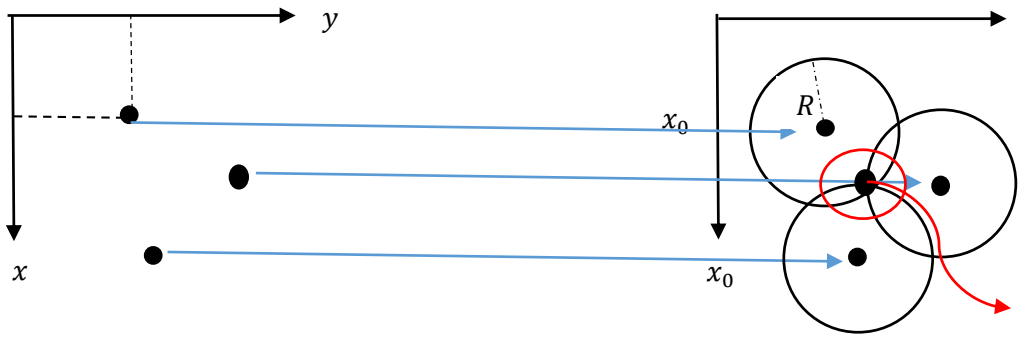
تبدیل هاف برای آشکارسازی دایره یک مپ از صفحه‌ی (x, y) به صفحه ی (x, y) می‌باشد که (x_0, y_0) مرکز دایره‌ای به شعاع R در صفحه‌ی (x, y) است. یعنی در صفحه‌ی (x, y) داریم:

$$(x - x_0)^2 + (y - y_0)^2 = R^2$$





همان‌طور که مشاهده می‌نمایید یک دایره در صفحه‌ی (x, y) به یک نقطه در صفحه‌ی (x_0, y_0) مپ می‌شود. و همین‌طور یک نقطه در صفحه‌ی (x, y) به یک دایره در صفحه‌ی (x_0, y_0) مپ می‌شود. حال فرض کنید چند نقطه در صفحه‌ی (x, y) داریم که همگی روی یک دایره به شعاع R بوده اند. هرکدام از این نقطه ها در صفحه‌ی (x, y) به یک دایره در صفحه ی (x_0, y_0) مپ می‌شود که تمامی این دایره ها در یک نقطه همدیگر را قطع می‌نمایند که ان نقطه مرکز دایره‌ی موجود در صفحه‌ی (x, y) می‌باشد. از آن‌جا که شعاع نیز معلوم است به راحتی می‌توان با داشتن مرکز دایره، دایره را آشکارسازی نمود.

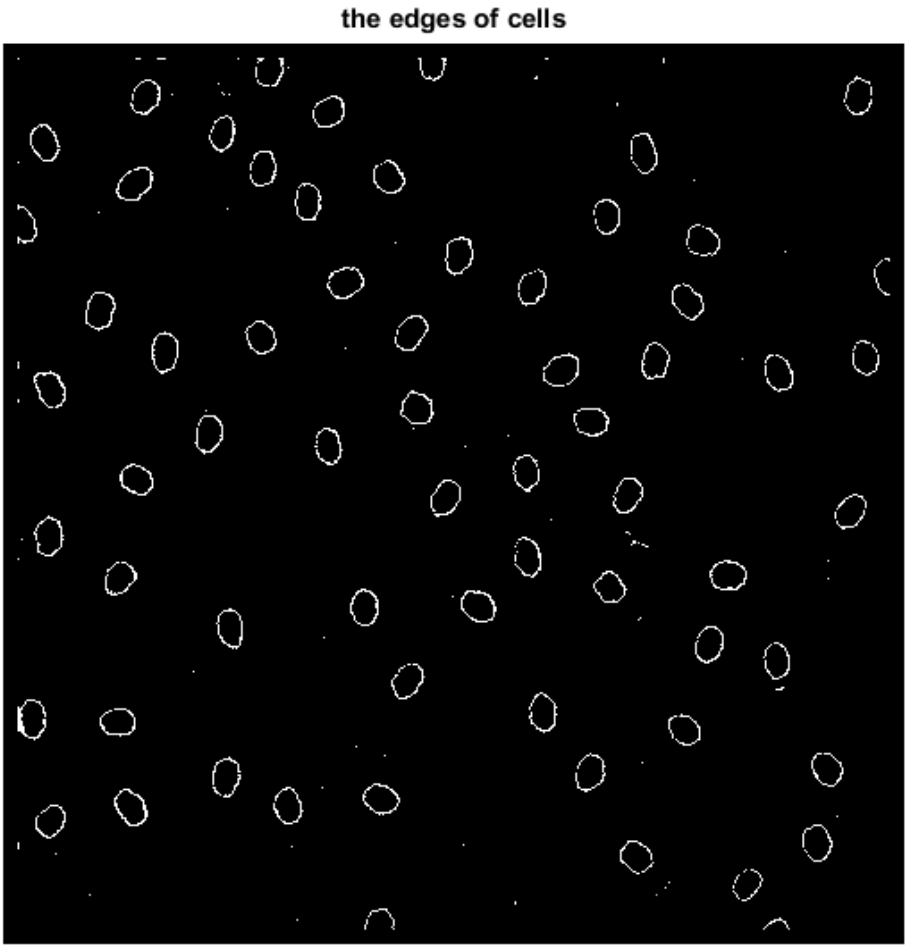


مرکز دایره ای که سه نقطه مورد نظر از ان می‌گذرند

حال میتوان به راحتی با رسم دایره ای به این مرکز یافت شده و با شعاع R دایره را بطور کامل آشکارسازی نمود.

حال قرار است از این تبدیل برای آشکارسازی سلولهای خونی استفاده نماییم.

برای این منظور ابتدا تصویر مربوط به آشکارسازی محیط سلول‌های خونی که در قسمت قبل یافتیم را در نظر گرفته که به صورت زیر می‌باشد:



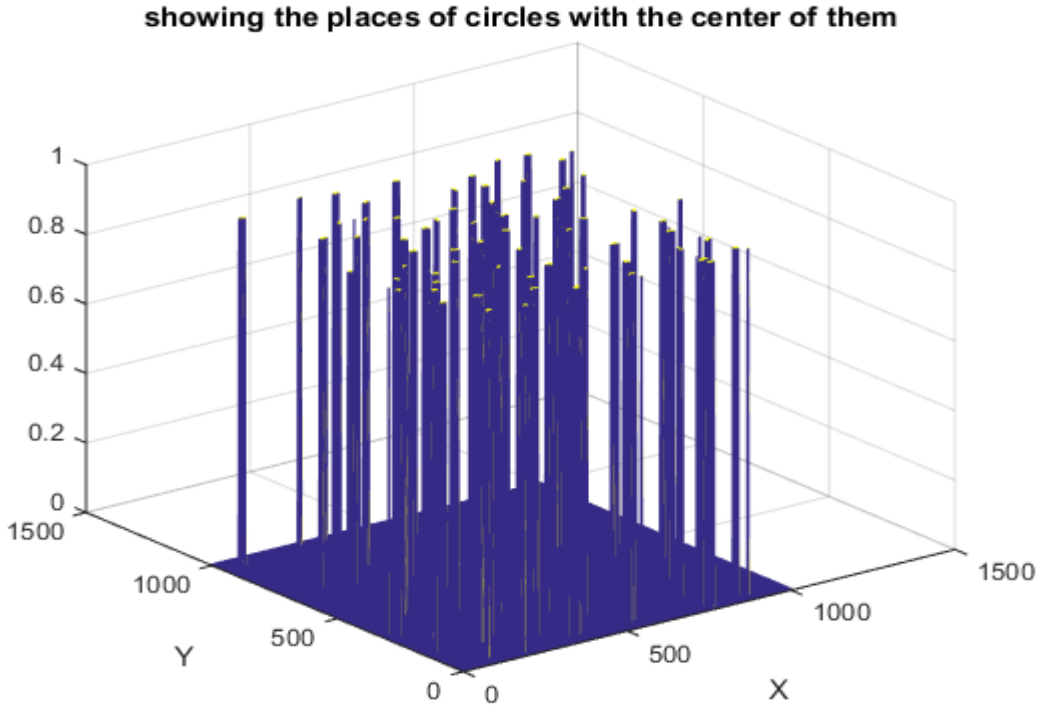
حال باید روی همه‌ی پیکسل‌های ۱ در این تصویر تبدیل هاف دایره ای بزنیم. برای این منظور از کد زیر استفاده می‌نماییم:

```
load('f_WLennght_23_Thr_.105.mat');
[M,N] = size(g);
R=15;
f = zeros(M+2*R , N+2*R);
f(R+1:end-R, R+1:end-R)=g;
figure,imshow(f);
title('the edges of cells')



thr = .5;
temp_R = R.* ones(2*R+1,2*R+1);


fff = zeros(M+2*R,N+2*R);%
[M_1,N_1] = find(f); %%% find indice that are equal to 1

ff= zeros(M+2*R , N+2*R);
for i=1:length(M_1)
    p=0;
   for m=M_1(i) - R : M_1(i) + R
       q=0;p=p+1;
       for n=N_1(i) - R : N_1(i) + R
           q=q+1;
           r(p,q) = sqrt((m-M_1(i))^2 + (n-N_1(i))^2);
       end
   end
  dif_temp = abs(r - temp_R);
  ind_1 = find(dif_temp < thr); %% these inice must be 1
  temp_f = zeros(2*R+1,2*R+1);
```

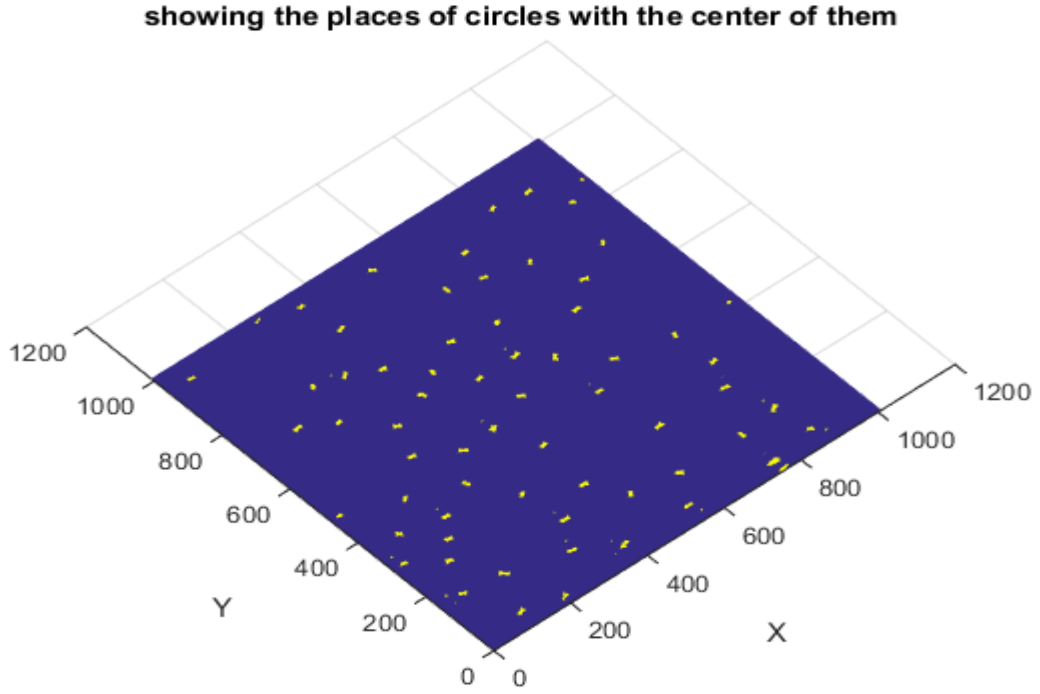
```
  temp_f(ind_1) = 1;
  temp_ff = zeros(M+2*R , N+2*R);
  jj=M_1(i) - R;
  jjj=M_1(i) + R ;
  jjjj = M_1(i);
  temp_ff(M_1(i) - R:M_1(i) + R  , N_1(i) - R:N_1(i) + R ) = temp_f;
  ff = ff + temp_ff;
end

ind = find(ff>15);
fff(ind) = 1;
[NN,MM] = meshgrid(1:N+2*R,1:M+2*R);
figure,mesh(MM,NN,fff),xlabel('X') , ylabel('Y');
title('showing the places of circles with the center of them');
p=0;
cond=0;
g = zeros(M+2*R,N+2*R);
[ind_x,ind_y] = find(fff == 1);

while cond==0
    p=p+1;

    r = sqrt( (ind_x(1:end)-ind_x(1)).^2 + (ind_y(1:end)-ind_y(1)).^2 );
    h = find(r < 4.5*R);
    ind_n_x = round(mean(ind_x(h)));
    ind_n_y = round(mean(ind_y(h)));

    g(ind_n_x,ind_n_y) =1;
    for i=1:length(h)
        fff(ind_x(h(i)),ind_y(h(i))) = 0;
    end

    [ind_x,ind_y] = find(fff == 1);
    if isempty(ind_x)
        cond =1;
    end

end

figure,imshow(g)
title('the center of circles with hoghtoning transform')
[ind_xx , ind_yy] = find(g==1);

%%%% Create Circles
gg=g;
for i=1:length(ind_xx)
   ind_x = ind_xx(i);
   ind_y = ind_yy(i);
   p=0; r=[];
   for m=ind_x - R:ind_x + R
       q=0;p=p+1;
       for n=ind_y - R:ind_y + R
           q=q+1;
           r(p,q) = sqrt((m-ind_x)^2 + (n-ind_y)^2);
       end
   end
  dif_temp = abs(r - temp_R);
%   ind_1 = find(dif_temp <= thr); %% these inice must be 1
  ind_1 = find(r <= R); %% these inice must be 1
  temp_f = zeros(2*R+1,2*R+1);
  temp_f(ind_1) = 1;
  gg(ind_x - R:ind_x + R  , ind_y - R:ind_y + R ) = temp_f;
end
gg = gg(R+1:end-R,R+1:end-R);
figure,imshow(gg),title(['the circles with the R=',num2str(R)]);

[M_ ,N_] = size(gg);
ggg = zeros(M_ , N_);
ind_0 = find(gg ==0);
ggg(ind_0) = 1;
figure,imshow(ggg),title('showing the circles with 1');
```

بعد از این‌که تبدیل هاف دایره ای زده شد، برای هر سلول چند دایره به دست می‌اید که همگی در یک نقطه هم را قطع کرده و لذا در مرکز دوایری که قرار است بسازیم یک پیک زده می‌شود.

مراکز دوایر یه صورت زیر می‌باشد:

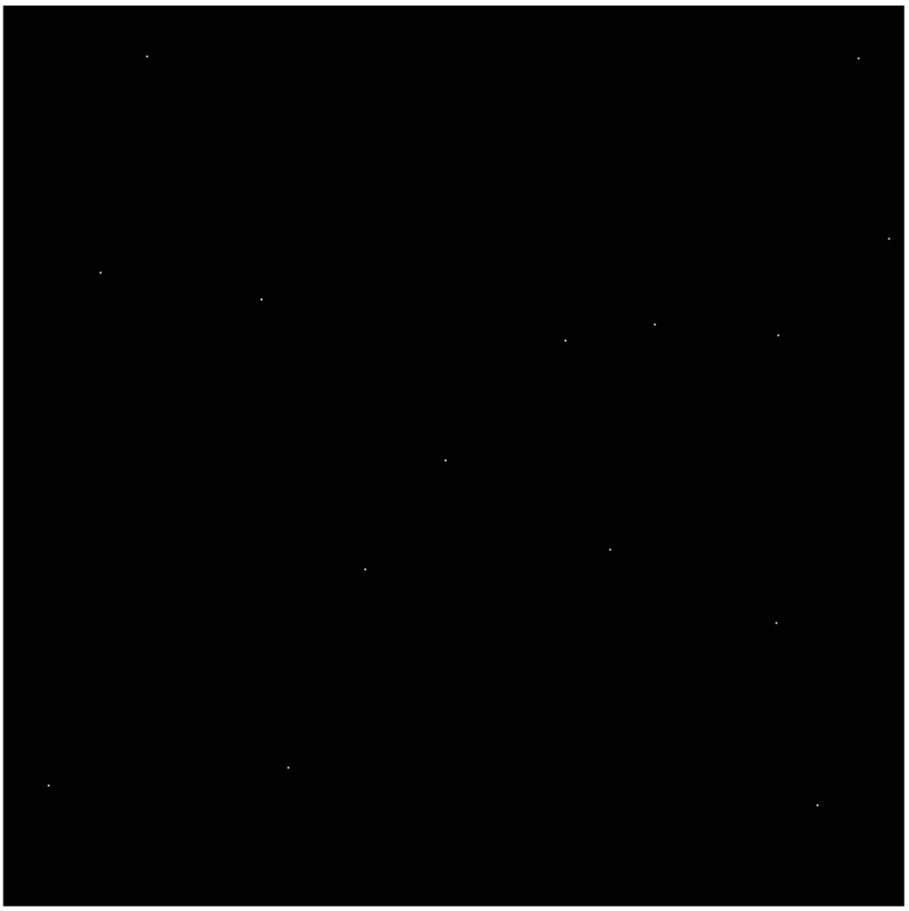


و از بالا به صورت زیر می‌باشد که محل مراکز به وضوح مشخص می‌باشد.

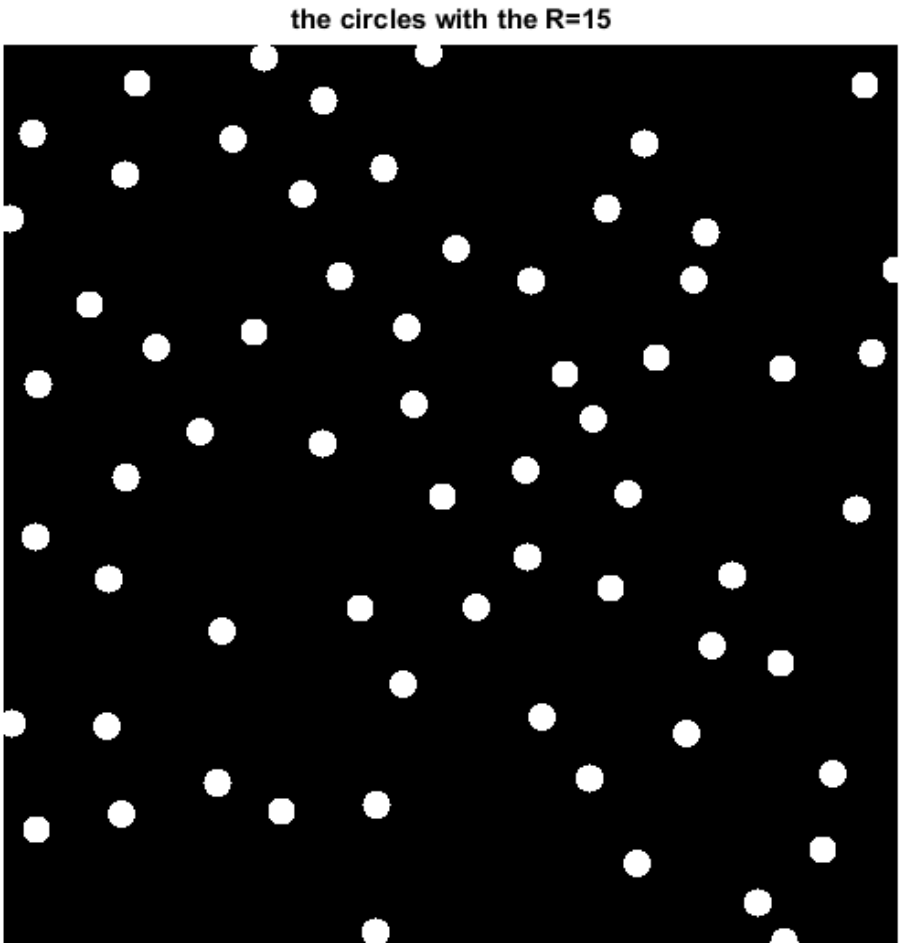


برای هر سلول تعدادی ۱ به عنوان نماینده ایجاد شده است. در ادامه برای هر سلول تنها یک نماینده برمیگزینیم که نتیجه زیر را در بر دارد (با تعریف یک آستانه از میان ۱ های کنار هم تنها یکی را انتخاب و به خروجی منتقل میکنیم).

g



حال باید در محل این مراکز دوایری به شعاع R بزنیم. توجه شود که برای به دست آوردن R از روی همان تصویر اولیه که شامل محیط سلول‌ها بود شعاع ای سلول‌ها را اندازه گیری کرده و یک میانگین گیری از آنها انجام داده تا مقدار یک R واحد به دست آید. اما از آنجا که سلول‌ها بیضوی هستند ما شعاع به دست‌آمده را اندکی کمتر کرده ایم تا بتواند همه سلول‌ها را بیابد و حتی الامکان سلولی را میس ننماید. نتیجه به صورت زیر در می‌آید:



حال به راحتی می‌توان سلول‌ها را به راحتی و با تفکیک خیلی خوبی از هم دید.

ه)

روش اول) در روش اول می‌توان با استفاده از نتیجه‌ی قسمت اول سوال، یعنی استفاده از روش هاف دایره‌ای، از این الگوریتم جهت شمارش سلول‌ها استفاده نمود. برای این منظور کافیست کد را به نحوی اصلاح نماییم که تعداد دایره‌های به دست آمده را بشمارد. برای این کار کافیست با اضافه نمودن یک تغییر کوچک در کد بالا،کافیست تعداد یک‌های تصویری که در ان برای هر سلول به عنوان یک نماینده، یک سطح روشنایی ۱ را در نظر گرفته ایم را بشماریم. این تصوی رهمان تصویر g در کد بالا بود. و لذا خط زیر را به کد اضافه می‌نماییم.

```
 [ind_xx , ind_yy] = find(g==1);
clc;
number_estimation_cell = length(ind_xx)
```

و لذا با اجرای برنامه تعداد سلولها را 64 عدد محاسبه می‌نماید.

number_estimation_cell=

روش دوم) کد این قسمت در HW6_Q2_h قرار دارد:

ایده‌ی مربوط به الگوریتم این قسمت از مقاله‌ی زیر گرفته شده است:

Sanpanich, Arthorn, et al. *"White blood cell segmentation by distance mapping active contour." Communications and Information Technologies, 2008. ISCIT 2008. International Symposium on*. IEEE, 2008.

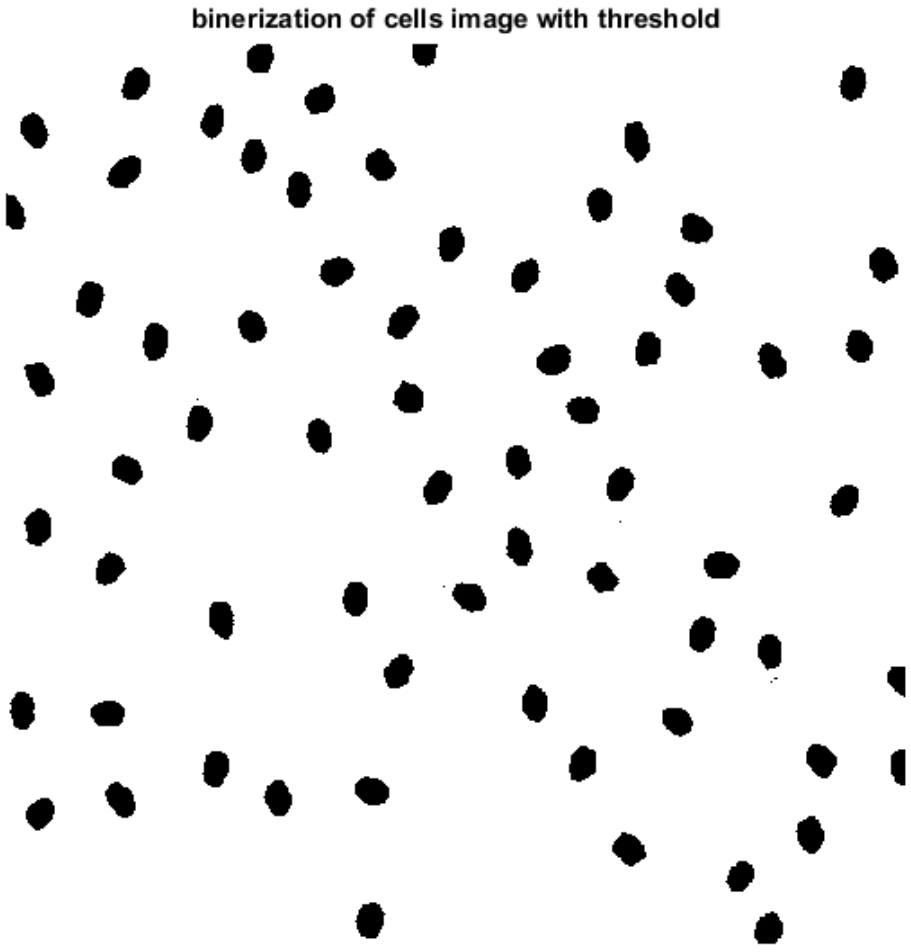
برای شمارش سلول‌ها از hit and miss در این مقاله استفاده شده است.(توضیحات بیشتر مربوط به هیت اند میس در سوال ششم آورده شده است توجه شود که در این سوال هدف پیاده سازی هیت اند میس نیست و پیاده سازی و توضیحات کامل آن در سوال ۶ اورده شده است.).

برای پیاده سازی این روش از کد زیر استفاده نموده ایم:

```
clc
clear
close all;

g=imread('bloodcel_95.jpg');
f=rgb2gray(g);
f = double(f);
[M,N] = size(f);
ff = ones(M,N);
ind_x = find(f<100);
ff(ind_x) = 0;    %image bineriztion
figure, imshow(ff);
title('binerization of cells image with threshold')

% making strel and using it for hit and miss
r=13;
W = 2*r+1;
strl = zeros(W , W);
for m=-r:r
    for n=-r:r
        d = sqrt(m.^2 + n.^2);
        if ceil(d) <= r
            strl(m+r+1,n+r+1) = -1;
        end
    end
end
figure,imshow(strl,[]),title(' disk strel with r=13');
BBW = strl;     %%% using strel


g = ones(M+W , N+W);
g(1+r:M+r , 1+r:N+r) = ff;
[MM,NN] = size(g);



gr = bwhitmiss(ff,BBW);
gr = double(gr);
figure,imshow(gr)
title('hit and miss binary image with disk strel');



p=0;
cond=0;
grr = zeros(M,N);
[ind_x,ind_y] = find(gr == 1);
thr_r = 13;
```

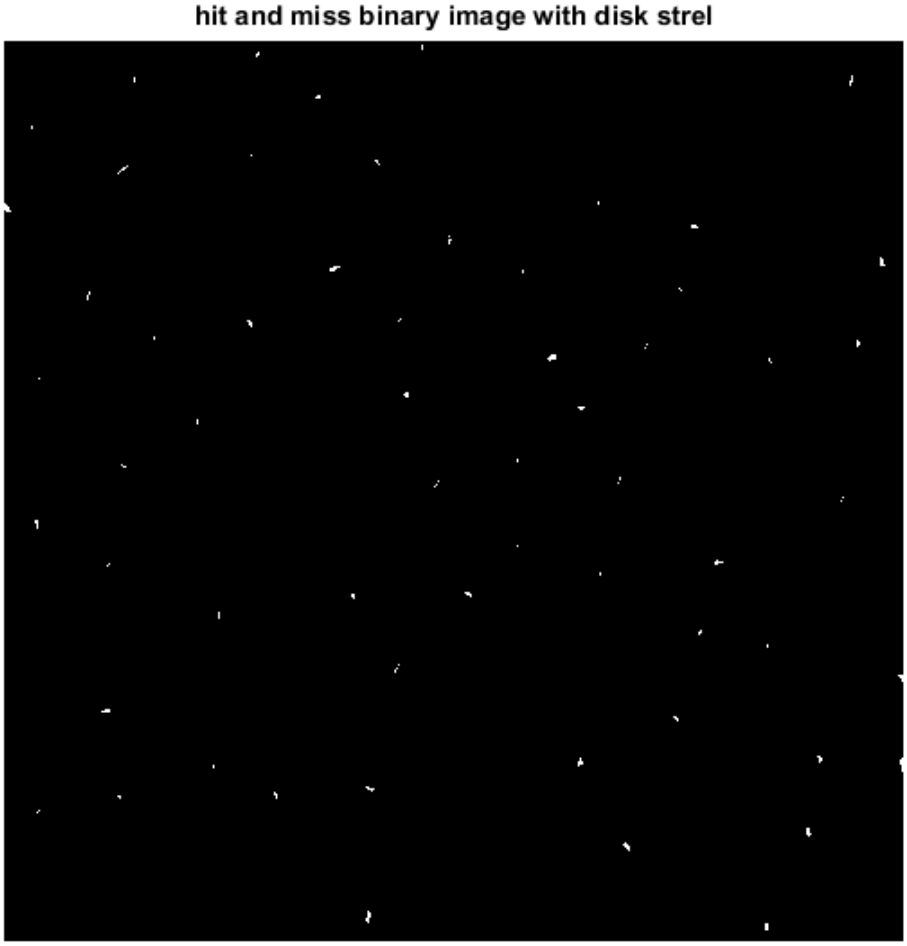
```
%this loop is for choosing one 1 for every cell
while cond==0
    p=p+1;

    r = sqrt( (ind_x(1:end)-ind_x(1)).^2 + (ind_y(1:end)-ind_y(1)).^2 );
    h = find(r < thr_r);
    ind_n_x = round(mean(ind_x(h)));
    ind_n_y = round(mean(ind_y(h)));

    grr(ind_n_x,ind_n_y) =1;
    for i=1:length(h)
        gr(ind_x(h(i)),ind_y(h(i))) = 0;
    end

    [ind_x,ind_y] = find(gr == 1);
    if isempty(ind_x)
        cond =1;
    end

end

figure,imshow(grr)
[ind_xx , ind_yy] = find(grr==1);

number_cell = length(ind_xx)     %calculating the numer of ones as cell numbers
```

در ابتدا با یک آستانه گذار تصویر را باینری می‌نماییم. برای این منظور ما از یک آستانه‌ی ۱۰۰ استفاده نموده ایم و تصویر سلول‌های خونی را به صورت زیر باینری نموده ایم:

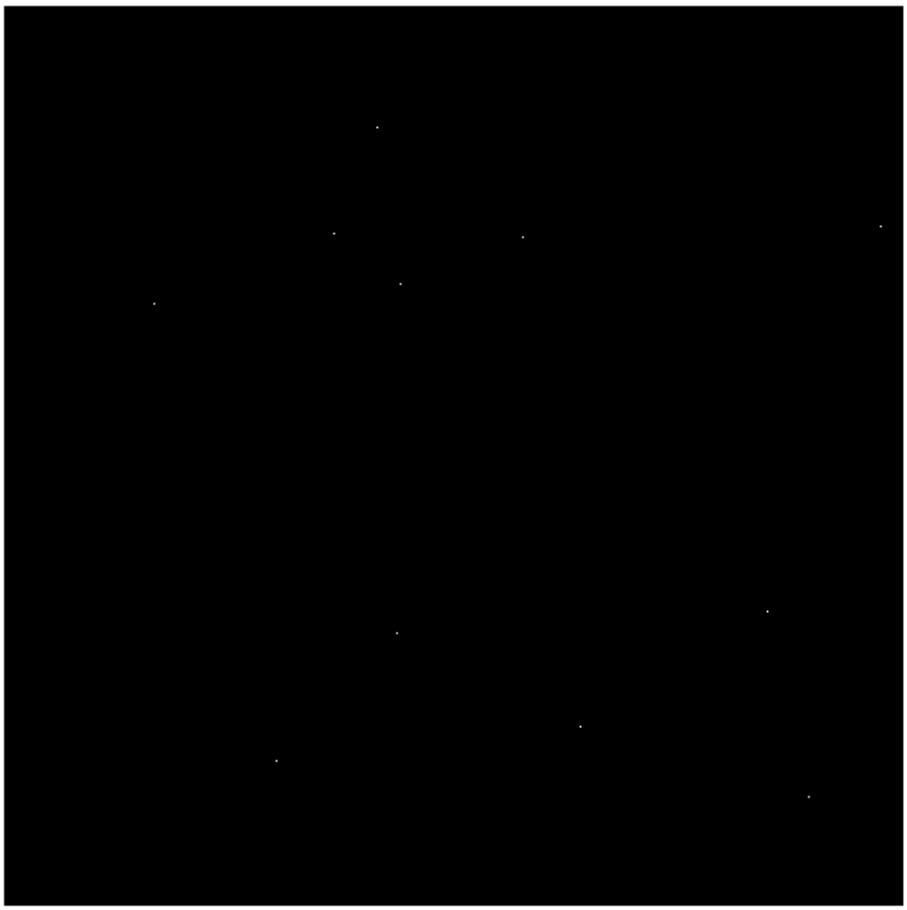


سپس از الگوریتم hit and miss استفاده می‌نماییم. برای این منظور strel را یک دایره با شعاع ۱۳ در نظر گرفته و آن را با تصویر باینری شده hit and miss می‌نماییم. Strel مورد نظرمان به صورت زیر می‌باشد:

نتیجه‌ی حاصل از هیت اند میس کردن تصویر باینری و strl به صورت زیر خواهد بود:



حال باید تعداد ۱ ها در تصویر بالا را به عنوان تعداد سلول‌ها در نظر بگیریم. از آنجا که برای هر سلول چند ۱ به عنوان نماینده قرار داده شده در ادامه برای هر سلول فقط یک نماینده از ۱ را در نظر می‌گیریم که نتیجه زیر حاصل می‌شود:



حال کافیست تعداد ۱ ها در این تصویر را به عنوان تعداد سلول‌ها برشماریم که بعد از شبیه سازی نتیجه ۶۵ را نشان داد.

number_cell=

۶۵

مشاهده می‌نماییم که در روش اول یعنی استفاده از تبدیل هاف دایره ای تعداد سلولها ۶۴ تا تخمین زده شد و در روش دوم یعنی استفاده از هیت اند میس تعداد آنها ۶۵ تا تخمن زده شد حال آنکه با مشاهده تصویر اصلی، تعداد واقعی سلول‌ها ۶۷ تاست. لذا جدول زیر را می‌توان به عنوان جدول خطا ترسیم نمود:

تعداد واقعی سلولها	۶۷	خطا
تعداد سلولها در روش هاف	۶۴	$\frac{(67-64)}{67} * 100 = 4.48٪$
تعداد سلولها در روش هیت اند میس	۶۵	$\frac{(67-65)}{67} * 100 = 2.99٪$

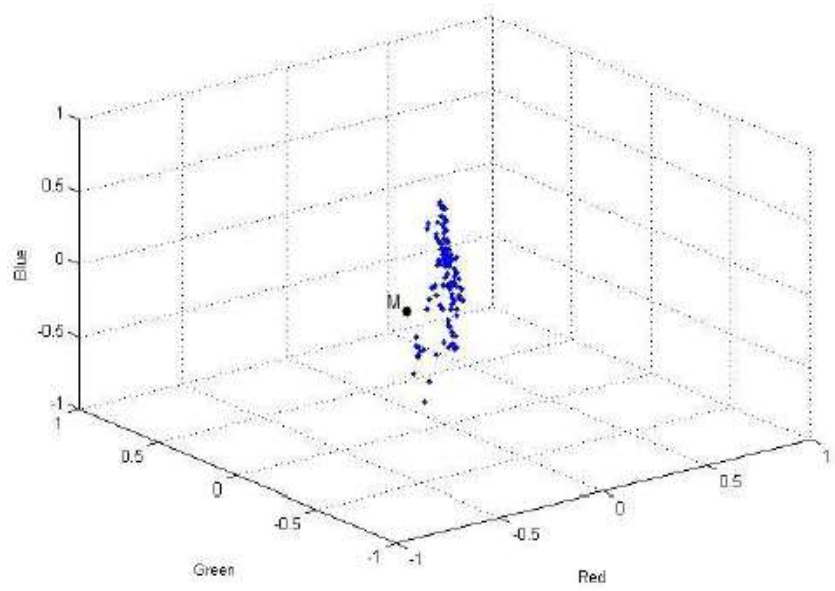
مشاهده می‌نماییم روش هیت اند میس در عین سرعت بالاتر خطای کمتری دارد و لذا مناسب تر می‌باشد.

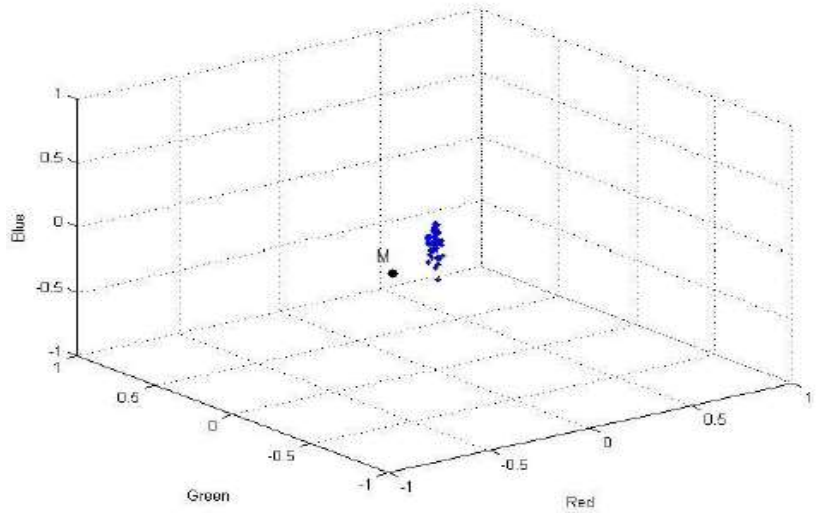
سوال سوم) (کد این قسمت در فایل HW6_Q3 قرار دارد)

در این سوال قرار است از روشی استفاده کنیم تا ۱۰ تصویر زرافه از گوگل را گرفته و براساس نمونه های گرفته شده از لکه های قهوه ای رنگ و سفیدرنگ بدن آنها و براساس فاصله ماهالانوبیس بتوانیم لکه های قهوه ای و سفیدرنگ بدن زرافه خواسته شده در سوال را جداسازی نماییم.

برای این منظور باید از یک معیاری در جهت جداسازی لکه های سفید رنگ و قهوه ای رنگ استفاده نماییم.

معیار اول) معیار فاصله $||x-\mu||$: می توان بر اساس این معیار ابتدا بانک داده ای از رنگ های مورد نظر که قرار است جداسازی شوند را به دست آورده ومیانگین آن‌ها را محاسبه نماییم که با μ نمایش می‌دهیم. مثلا دو بانک داده زیر را در نظر بگیرید که در واقع نمونه های موجود در پوست هستند:





اگر از معیار میانگین استفاده نماییم در دو تصویر بالا فاصله پیکسل M از میانگین هر دو نوع بانک داده یکسان است و لذا نمی‌توان از این معیار استفاده نمود. بنابراین پراکندگی داده ها را نیز باید در معیارمان دخالت دهیم.

معیار دوم) معیار ماهالانوبیس: این معیار که معیار مناسب و مد نظر ما می‌باشد پراکندگی داده ها را در نظر گرفته و مقدار $\frac{\|x-\mu\|}{\sigma}$ را حساب می‌نماید و براساس آن تصمیم می‌گیرد یعنی اگر پراکندگی داده ها زیاد باشد، این فاصله کمتر و احتمال تعلق بیشتر می‌شود و اگر پراکندگی داده ها کم باشد این فاصله زیاد می‌شود و احتمال تعلق بیشتر می‌شود.

این فاصله در دستگاه RGB به صورت زیر است:

$$D = \sqrt{\frac{(r-\mu_r)^2}{\sigma_r^2} + \frac{(g-\mu_g)^2}{\sigma_g^2} + \frac{(b-\mu_b)^2}{\sigma_b^2}}$$

$$D^2 = \begin{bmatrix} r-\mu_r & g-\mu_g & b-\mu_b \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_r^2} & 0 & 0 \\ 0 & \frac{1}{\sigma_g^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_b^2} \end{bmatrix} \begin{bmatrix} r-\mu_r \\ g-\mu_g \\ b-\mu_b \end{bmatrix}$$

$$\rightarrow D^2 = \begin{bmatrix} r-\mu_r & g-\mu_g & b-\mu_b \end{bmatrix} \Sigma^{-1} \begin{bmatrix} r-\mu_r \\ g-\mu_g \\ b-\mu_b \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} E(rr^*) & E(rg^*) & E(rb^*) \\ E(gr^*) & E(gg^*) & E(gb^*) \\ E(br^*) & E(bg^*) & E(bb^*) \end{bmatrix}$$

توجه شود که ما زمانی می‌توانیم از این عبارت برای ماتریس کواریانس استفاده کنیم که اولا تعداد داده ها زیاد باشد و ثانیا میانگین آنها صفر باشد. برای همین در عبارت مربوط به ماریس کواریانس، $r = r - \mu_r$ و $g = g - \mu_g$ و $b = b - \mu_b$ می‌باشد.

در این سوال ابتدا تصویر ۱۰ زرافه را از گوگل انتخاب نموده و بر روی بدن آنها نمونه های قهوه ای و سفید را مشخص نموده تا براساس این نمونه ها یک ماتریس میانگین و کواریانس مربوط به رنگ قهوه ای و سفید را به دست آوریم و در نهایت با استفاده از میانگین و واریانس رنگ قهوه ای و همچنین مشخص کردن یک سطح آستانه، رنگ های قهوه ای و سفید بدن زرافه در تصویر داده شده در سوال را جدا سازی می‌نماییم.

برای این منظور از کد زیر استفاده می‌نماییم:

```
clc;
close all;
clear
thr_brown = .13;
thr_white = .2;

% load('brown_mean.mat');
% load('brown_cov.mat');
%
% load('white_mean.mat');
% load('white_cov.mat');

N0_photo = 10;  %%% Number of phto that we want to use them to extract mean and
covariance


%%%%%% Load Images
for i=1:N0_photo
    var_name = sprintf('f%d.jpg',i);
    f{i} = imread(var_name);
end



%%%%% Select Brown points
figure, imshow('select_brown.jpg'),pause(1.5);

for i=1 : N0_photo
    temp =[]; x=[]; y =[];
    while isempty(x)
        figure,imshow(f{i});
        [x,y] = getpts();
        close;
    end
        temp(1,:) = x;
        temp(2,:) = y;
        brown_xy{i} = temp;
end
close;

%%%%% Select White points
figure, imshow('select_white.jpg'),pause(1.5);

for i=1 : N0_photo
temp =[]; x=[]; y =[];
    while isempty(x)
        figure,imshow(f{i});
        [x,y] = getpts();
        close;
    end
    temp(1,:) = x;
    temp(2,:) = y;
    white_xy{i} = temp;
end
close;

%%%%% Calculate mean and covariance matrix
brown_value=[];
white_value=[];

for i=1 : N0_photo
    %%% Extract pixel value of Brown pixels
    temp_brown =  brown_xy{i};
    x_brown = temp_brown(1,:).';
    y_brown = temp_brown(2,:).';
    brown_value = [brown_value; impixel(f{i},x_brown,y_brown)];

    %%% Extract pixel value of White pixels
```

```
    temp_white =  white_xy{i};
    x_white = temp_white(1,:).';
    y_white = temp_white(2,:).';
    white_value = [white_value; impixel(f{i},x_white,y_white)];
end

%%% Extract White and Brown mean and covariance

brown_mean = mean(brown_value); %%% mean
white_mean = mean(white_value); %%% mean
for i=1:3
    X(:,:,i) = brown_value(:,i) - brown_mean(1,i);
    Y(:,:,i) = white_value(:,i) - white_value(1,i);
end
brown_cov = [X(:,:,1)' ; X(:,:,2)' ; X(:,:,3)'] * [X(:,:,1) X(:,:,2) X(:,:,3)];
white_cov = [Y(:,:,1)' ; Y(:,:,2)' ; Y(:,:,3)'] * [Y(:,:,1) Y(:,:,2) Y(:,:,3)];



%%%%%%%% we use this code to choose threshode
% ff = imread('f10.jpg');
% s=1;
% while s==1
%     figure,imshow(ff);
%     [x,y] = getpts();
%     x = impixel(ff,x,y);
%     Mahal_Brown = sqrt((x - brown_mean) * inv(brown_cov) * (x - brown_mean)');
%     Mahal_White = sqrt((x - white_mean) * inv(white_cov) * (x - white_mean)');
%
%     clc;
%
%     if Mahal_Brown < Mahal_White && Mahal_Brown < thr_brown
%         disp('chosed pixel is BROWN');
%     elseif Mahal_White < Mahal_Brown && Mahal_White < thr_white
%         disp('chosed pixel is WHITE');
%     else
%         disp('we can''t segment this pixel')
%     end
% end



ff = imread('giraf3_2.jpg');
[M,N] = size(ff(:,:,1));
g_Brown = zeros(M,N);
g_White = zeros(M,N);

for m=1:M
    for n=1:N
        x = impixel(ff,n,m);
        Mahal_Brown = sqrt((x - brown_mean) / (brown_cov) * (x - brown_mean)');
        Mahal_White = sqrt((x - white_mean) / (white_cov) * (x - white_mean)');

        if Mahal_Brown < Mahal_White && Mahal_Brown < thr_brown
            g_Brown(m,n) = 1;
        elseif Mahal_White < Mahal_Brown && Mahal_White < thr_white
            g_White(m,n) = 1;
        end
    end
end


% %%%%%%%%%% Black and White

figure,subplot(121),imshow(ff),title('Original Image');
subplot(122),imshow(g_White),title('White Pixel of Giraffe');
```

```
suptitle(['Threshold White = ',num2str(thr_white)])


figure,subplot(121),imshow(ff),title('Original Image');
subplot(122),imshow(g_Brown),title('Brown Pixel of Giraffe');
suptitle(['Threshold Brown = ',num2str(thr_brown)])

%%%%%%%%% Cloured
[m_b , n_b] = find(g_Brown == 1);
[m_w , n_w] = find(g_White == 1);

Brown_1= zeros(M,N);Brown_2= zeros(M,N);Brown_3= zeros(M,N);
for i=1:length(m_b)
    r = ff(m_b(i),n_b(i),1);
    g = ff(m_b(i),n_b(i),2);
    b = ff(m_b(i),n_b(i),3);
    Brown_1(m_b(i),n_b(i)) = r;
    Brown_2(m_b(i),n_b(i)) = g;
    Brown_3(m_b(i),n_b(i)) = b;
end


White_1= zeros(M,N); White_2= zeros(M,N); White_3= zeros(M,N);
for i=1:length(m_w)
    r = ff(m_w(i),n_w(i),1);
    g = ff(m_w(i),n_w(i),2);
    b = ff(m_w(i),n_w(i),3);

    White_1(m_w(i),n_w(i)) = r;
    White_2(m_w(i),n_w(i)) = g;
    White_3(m_w(i),n_w(i)) = b;
end

g_Borwn_Colourd = uint8(cat(3,Brown_1,Brown_2,Brown_3));
g_White_Colourd = uint8(cat(3,White_1,White_2,White_3));


figure,subplot(121),imshow(ff),title('Original Image');
subplot(122),imshow(g_White_Colourd),title('White Pixel of Giraffe');
suptitle(['Threshold White = ',num2str(thr_white)])


figure,subplot(121),imshow(ff),title('Original Image');
subplot(122),imshow(g_Borwn_Colourd),title('Brown Pixel of Giraffe');
suptitle(['Threshold Brown = ',num2str(thr_brown)])
```

پس از انتخاب پیکسل‌های مورد نظر قهوه ای و سفید در ۱۰ تصویر انتخاب شده، ماتریس میانگین و کواریانس برای رنگ‌های سفید و قهموه‌ای به صورت زیر در می‌آید:

```
white_mean =

 207.8926   192.5124   164.0661
```

```
brown_mean =

 120.6477   69.6839   39.6373
```

white_cov =

223156	264958	144513
264958	344735	173129
144513	173129	176058

brown_cov =

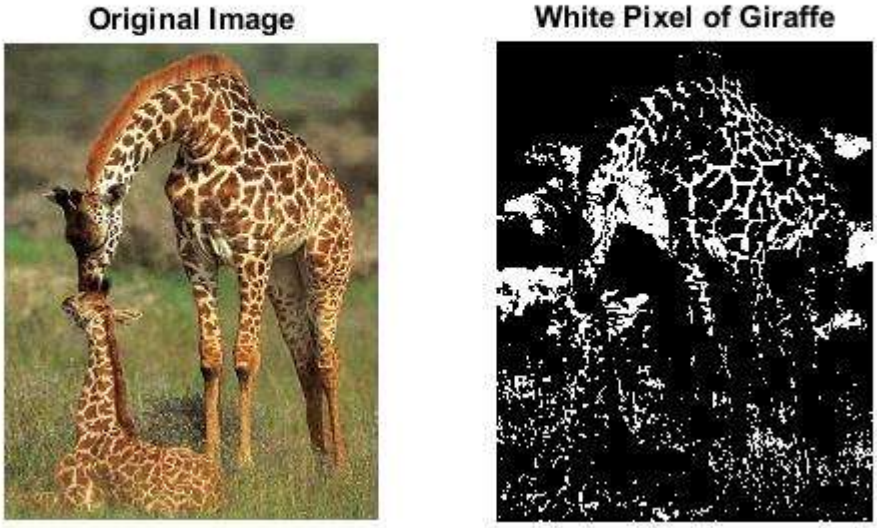
1.0e+05 *

4.3244	2.7535	1.5932
2.7535	1.9558	1.2035
1.5932	1.2035	0.9310

حال باید با توجه به ماتریس‌های میانگین و کواریانس بدست‌آمده قسمت‌های قهوه ای و سفیدرنگ بدن زرافه در تصویر زیر را جداسازی نماییم:

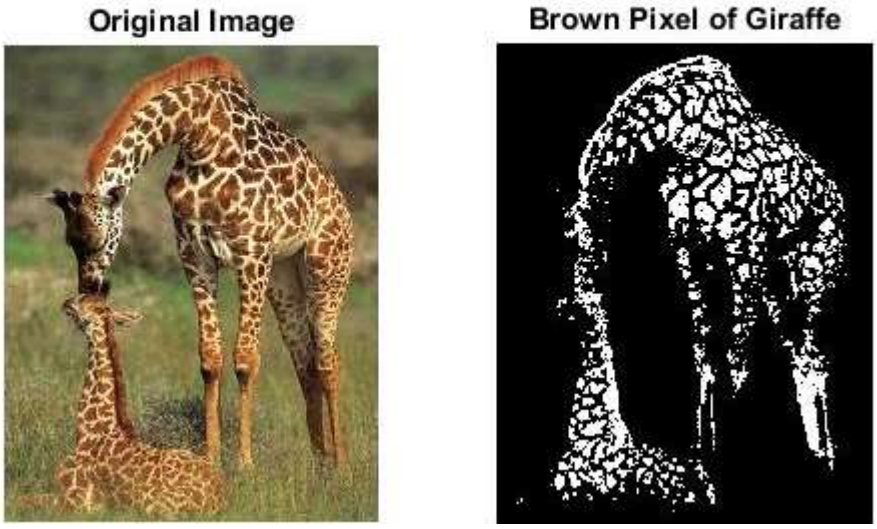
در ابتدا یک آستانه نیز با انتخاب چند پیکسل قهوه ای و سفید از بدن زرافه در این عکس می‌یابیم. که برای رنگ سفید آستانه را ۰,۲ در نظر گرفته و برای رنگ قهوه ای آستانه را ۰,۱۳ و نتیجه به صورت زیر خواهد شد.:

Threshold White = 0.2



با قرارداد ترشولد ۰,۲ برای رنگ سفید، قسمتهای سفید بدن زرافه را با ۱ و بقیه را با ۰ نشان می‌دهد. و با قرار دادن ترشولد ۰,۱۳ برای قهوه ای، قسمتهای قهوه ای رنگ بدن زرافه را با ۱ و بقیه را با ۰ نشان می‌دهد که به صورت زیر خواهد بود:

Threshold Brown = 0.13



همچنین می‌توان این نتیجه را در فضای RGB نیز به صورت زیر مشاهده نمود:

Threshold White = 0.2

Original Image

White Pixel of Giraffe

Threshold Brown = 0.13

Original Image

Brown Pixel of Giraffe

نتیجه برای رنگ سفید خیلی رضایت بخش نبود زیرا تا حدی محیط را هم به عنوان رنگ سفید تشخیص داده است و به همین دلیل آستانه را تغییر داده و نتیجه را در فضای باینری و RGB بری آستانه ۰٫۱۸و۰٫۱۵ برای رنگ سفید مشاهده می‌نماییم ، همانطور که در تصاویر زیر مشاهده میکنیم این کار باعث شده در برخی از قسمت های بدن زرافه که واقعا رنگ ما سفید است تشخیص داده نشود به عنوان سفید، که این امر باعث میشود محیط هم که سفید رنگ قبلا تشخیص داده میشد الان سفید تشخیص داده نشود، در واقع حذف تشخیص محیط به عنوان رنگ سفید ، باعث حذف قسمتی از پیکسل های تصویر اصلی هم میشود که این به علت نزدیک بودن رنگ سفید بدن زرافه با رنگ محیط است

Threshold White = 0.15

Original Image

White Pixel of Giraffe

Threshold White = 0.15

Original Image

White Pixel of Giraffe

Threshold White = 0.18

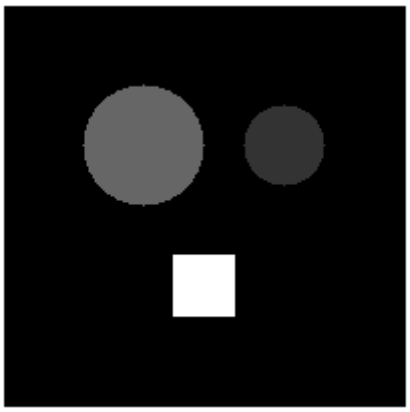
Original Image

White Pixel of Giraffe

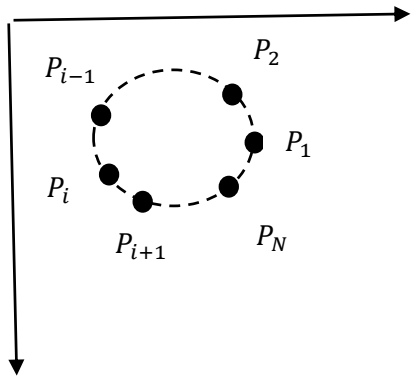
سوال چهارم)(کد این قسمت در فایل HW6_Q4 قرار دارد)

الف) کد مربوط به این قسمت در تابع Kass_Method.m نوشته شده است:. ابتدا تصویر زیر را ساخته و قرار است که از برنامه کانتور kass استفاده نموده تا بتوانیم اشکال موجود در تصویر را جداسازی نماییم.

شکل مورد نظر به صورت زیر است که از دو دایره یکی با شعاع ۳۰ و مرکز (۷۰و۷۰) با سطح روشنایی۳۵ و یکی با شعاع ۲۰ و مرکز (۷۰و۱۴۰) با سطح روشنایی ۳۰ و یک مربع با طول ضلع ۳۰ و مرکز (۱۴۰و۱۰۰) و با سطح روشنایی ۵۰ تشکیل شده است.



الگوریتم kass در دستگاه محورهای مختصات یک کانتور C را در نظر می‌گیرد. روی این کانتور نقاط مختلفی شامل $p_1, p_2, p_3, \dots \dots, p_{i-1}, p_i, p_{i+1}, \dots ., p_N$ به صورت زیر قرار گرفته اند:



با در نظر گرفتن سه نقطه‌ی p^i و p^{i-1} و p^{i+1} برای این کانتور مقادیر زیر مطرح می‌باشند:

$$E_{cont}^{i} = \left\| p^i - p^{i-1} \right\|^2$$

که E_{cont}^{i} میزان continuity در p^i را نمایش می‌دهد. هرچه این مقدار کوچکتر باشد، فواصل به سمت مساوی بودن میل می‌کنند.

$$E_{curv}^{i} = \left\| p^{i-1} - 2p^i + p^{i+1} \right\|^2$$

که E^i_{curv} میزان smoothبودن را نشان میدهد و هرچه کوچکتر بشد محیط کانتور smooth تر بوده و تضاریس کمتری دارد.

E^i_{cont} را می‌توان به صورت زیر تغییر داد:

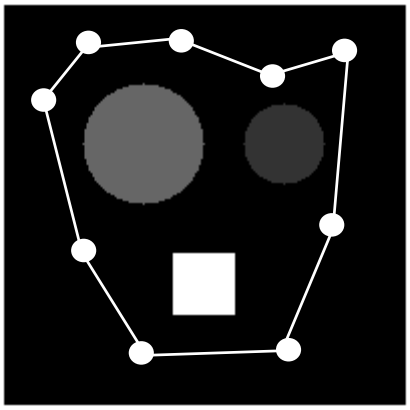
$$E^i_{cont} = abs(d - \|p^i - p^{i-1}\|^2)$$

که در آن d مقدار متوسط طول کانتور بوده و به صورت زیر محاسبه می‌شود:

$$d = \frac{1}{N}\sum_{i=1}^{N}\|p^i - p^{i-1}\|^2$$

$$E_{contour} = \alpha E^i_{cont} + \beta E^i_{curv} - \gamma E^i_{grad}$$

برای انجام الگوریتم، برای نمونه همین شکل صورت سوال را در نظر می‌گیریم.



همانطور که مشاهده می‌شود با دستور getpts در متلب نقاطی را روی شکل ایجاد می‌کنیم تا کانتور اولیه تشکیل شود. یک کانتور خوبی که بخواهد حول شی را بگیرد باید طوری باشد که در پروسه‌ی develop کانتور، فاصله ها تمایل پیدا کنند که مساوی شوند.

بعد از انتخاب کانتور اولیه برای هر نقطه یک پنجره با ابعاد w در نظر گرفته و E^i_{cont} و E^i_{curv} و E^i_{grad} را برای تمام نقاط داخل پنجره حساب نموده و سپس عبارت :

$$E_{contour} = \alpha E^i_{cont} + \beta E^i_{curv} - \gamma E^i_{grad}$$

را برای هر پیکسل داخل پنجره محاسبه نموده که در نتیجه‌ی آن یک ماتریس تشکیل می‌شود. در این ماتریس باید ببینیم که مقدار مینیمم این عبارت در کدام پیکسل قرار دارد اگر مختصات آن پیکسل را با (m,n) نمایش دهیم در این صورت کانتور در مرحله بعد به نقطه (m,n) خواهد رفت. اما توجه شود برای اینکه این سه انرژی با هم جمع شده و عبارت بالا را تشکیل دهند باید آنها را نرمالیزه نماییم تا قابل جمع شدن باشند.

همچنین قبل از جمع کردن این ماتریس مربوط به این سه انرژی و پس از نرمالیزه کردن مقادیرآنها یک تست دیگر هم باید برای تعیین β انجام دهیم.

اگر کانتور برای یک بار تکرار به یک نقطه‌ی تیز برسد ناگهان E_{curv}^{i} بسیار بزرگ می‌شود. و بنابراین مقدار $\alpha E_{cont}^{i} + \beta E_{curv}^{i} - \gamma E_{grad}^{i}$ خیلی بزرگتر شده و در نتیجه کانتور از گوشه عبور کرده و به جستجوی نقطه ای میگردد که حاصل جمع حداقل شود. باید کاری کنیم که وقتی به گوشه رسیدیم β کوچک شود و یا حتی صفر شود. بنابراین به یک تست نیاز داریم که ببینیم آیا کانتور به گوشه رسیده است یا خیر.

در گوشه خواص زیر برقرار است:

$$E_{curv}^{i} > E_{curv}^{i-1}$$

$$E_{curv}^{i} > E_{curv}^{i+1}$$

$$E_{curv}^{i} > curvature\ threshold$$

$$Grad^{i} > Grad\ threshold$$

.بنابراین باید چهار شرط زیر با هم برقرار باشند تا گوشه باشد

اما یک تخمین خوب هم برای threshold:

$$Gradthreshold = 0.7maxgrad - 0.4mingrad$$

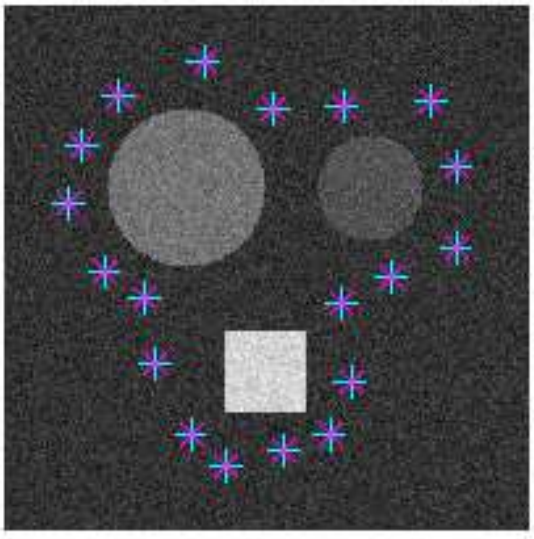
اما برای پیاده سازی این برنامه در متلب به صورت زیر عمل می‌نماییم:

```
function Kass_Method(f , y , x, alpha , beta , gamma , win_length)
a=alpha;
b=beta;
c=gamma;
w_2 = floor(win_length/2);

%%%% filetr Image with a Gaussian
l_win = 15; sigma = l_win/6;
h=fspecial('gaussian', l_win , sigma);
f=imfilter(f,h);


N=length(x);
aa = beta;
%%%%% Coeficients to caculate E
alpha=alpha*ones(1,N);
beta=beta*ones(1,N);
beta2=aa*ones(1,N);
gamma=gamma*ones(1,N);

%%%%%%% Calculate Gradient of Image
[grad_x , grad_y] =  gradient(f);
Grad = grad_x.^2 + grad_y.^2;

%%%%% Threshold for corner detection citeria
max_G = max(Grad(:)); min_G = min(Grad(:)); d = max_G - min_G;
```

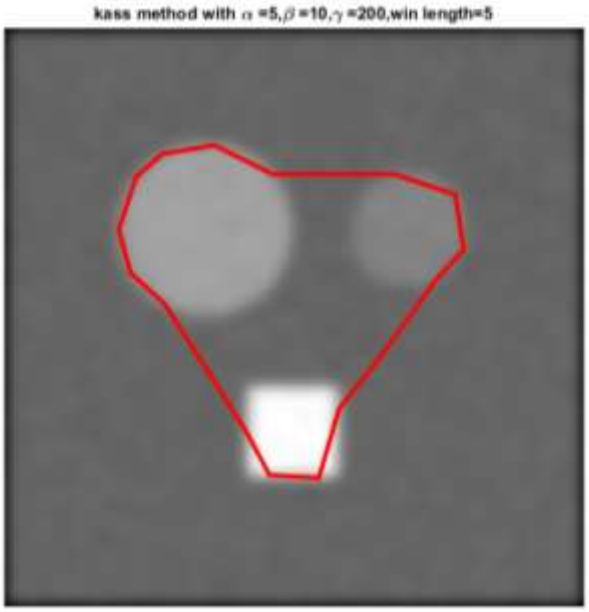
```
grad_thr = (.7*max(Grad(:)) - .4*min(Grad(:)) - min_G)/d;

curv_thr = .5;

%%%%%%%%%%%%%%%%%
x_1 = x;
y_1 = y;
for t=1:130

    %     while(1)
    %%%%%%%%%%%%%%%%%%%%%%%%%

    x_aug = [x_1(N) x_1.' x_1(1)];
    y_aug = [y_1(N) y_1.' y_1(1)];
    d = 0;
    p1=0;
    for i=2:N+1
        p1=p1+1;
        d = d + sqrt((x_aug(i) - x_aug(i-1)).^2 + (y_aug(i) - y_aug(i-1)).^2);
    end
    d = d/N;

    for i=2:N+1
        x_aug = [x_1(N) x_1.' x_1(1)];
        y_aug = [y_1(N) y_1.' y_1(1)];


        for m=-w_2:w_2
            for n=-w_2:w_2
                X = x_aug(i) + m;
                Y = y_aug(i) + n;
                E_Count(m+w_2+1,n+w_2+1) = (d-sqrt((X-x_aug(i-1)).^2 + (Y-y_aug(i-
1)).^2)).^2;
                E_Curv(m+w_2+1,n+w_2+1) = (x_aug(i+1) - 2*X + x_aug(i-1)).^2 +
(y_aug(i+1) - 2*Y + y_aug(i-1)).^2;
            end
        end
        E_Grad= Grad(x_aug(i)-w_2:x_aug(i)+w_2 , y_aug(i)-w_2:y_aug(i)+w_2);

        %%%% NOrmlized
        E_Count_norm = normlize(E_Count,1);
        E_Curv_norm  = normlize(E_Curv,1);
        E_Grad_norm = normlize(E_Grad,2);



        temp = alpha(1,i-1).*E_Count_norm + beta(1,i-1).*E_Curv_norm - gamma(1,i-
1).*E_Grad_norm;



        [xx,yy] = (find(temp == min(temp(:))));
        x_n = x_1(i-1) + xx - (w_2+1);
        y_n = y_1(i-1) + yy - (w_2+1);



        %%%%%%%%%% Update Points
        x_1(i-1) = x_n;
        y_1(i-1) = y_n;
                E_Curv_final(i-1) = E_Curv_norm(xx , yy);
                E_Grad_final(i-1) = E_Curv_norm(xx , yy);
    end

    E_Curv_final = [ E_Curv_final(end) E_Curv_final E_Curv_final(1) ];

    for i=2:N+1
        if E_Curv_final(i) > E_Curv_final(i+1)  && ...
```

```
                E_Curv_final(i) > E_Curv_final(i-1)&&...
                E_Curv_final(i) > curv_thr    && E_Grad_final(i-1)>grad_thr

            beta(i)=0;
        end
    end

    %%%%%% Plot
    hold off
    imshow(f,[]),hold on,plot([y_1;y_1(1)],[x_1;x_1(1)],'r','LineWidth',3)
    title(['\alpha = ',num2str(a),'  \beta = ',num2str(b),'  \gamma =
',num2str(c),'  win Length = ', num2str(win_length)])
    pause(.1)
end

end
```

همانطور که اشاره شد برای کانتور اولیه ابتدا با دستور getpts نقاط مد نظر را به صورت دلخواه انتخاب می‌نماییم مثلا ما این نقاط را بدین صورت در نظر گرفتیم:

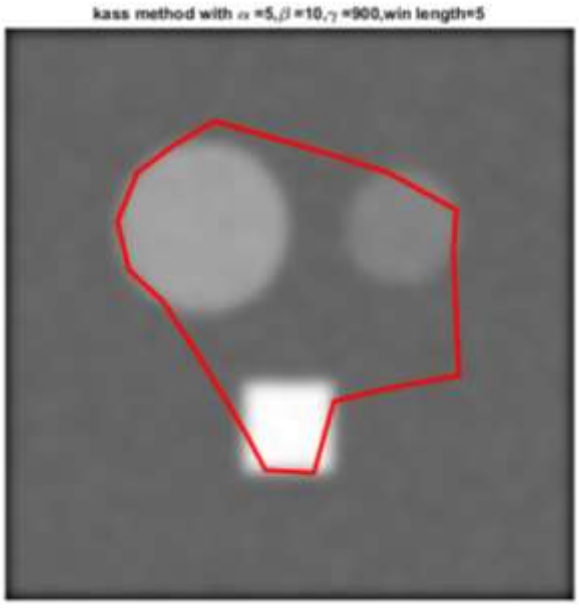


سپس با استفاده از کد بالا برنامه را اجرا نموده تا عملیات سگمنتیشن را برای طول پنجره ۵ و α=۵، β =۲۰ و γ=۲۰۰ که پارامترهای الگوریتم هستند به صورت زیر به دست آورد:



حال مقادیر را تغییر می‌دهیم تا ببینیم چه تغییری در نتیجه حاصل می‌شود.

در ابتدا با ثابت بودن بقیه پارامترها مقدار گاما را افزایش میدهیم:

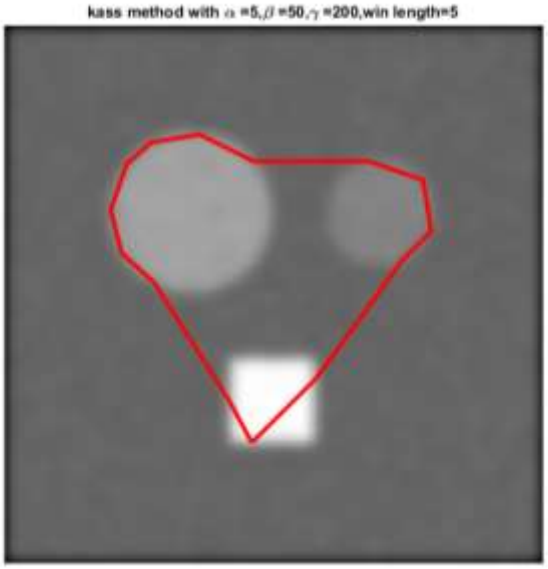


با مقایسه‌ی این دو نتیجه که ناشی از افزایش گاما می‌باشد می‌توان به این نتیجه رسید که با افزایش گاما طبق روابط بیان شده در ابتدا، یعنی رابطه انرژی کانتور:

$$E_{contour} = \alpha E^i_{cont} + \beta E^i_{curv} - \gamma E^i_{grad}$$

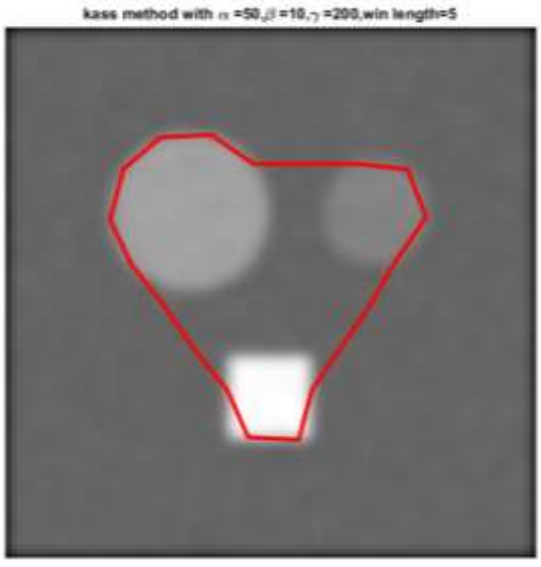
با افزایش گاما اثر گرادیان زیاد شده و به همین دلیل کانتور نویز را هم گوشه در نظر گرفته و می‌ایستد و به همین علت مشاهده می‌نمایید که کانتور نسبت به جواب بالا در یک جا متوقف شده که اصلا گوشه نیست بلکه نویز را به جای گوشه در نظر گرفته است.

حال با ثابت نگه داشتن بقیه پارامترها، اثر تغییر بتا را بررسی می‌نماییم:



مشاهده می‌شود با افزایش بتا از ۱۰ به ۵۰ از آن‌جا که طبق معادله بیان شده مربوط به انرژی کانتور، اثر E_{curv} زیاد می‌شود لذا کانتور نسبت به حالت اول بیشتر به سمت داخل می‌رود.

در مرحله بعد اثر تغییر آلفا را بررسی میکنیم:



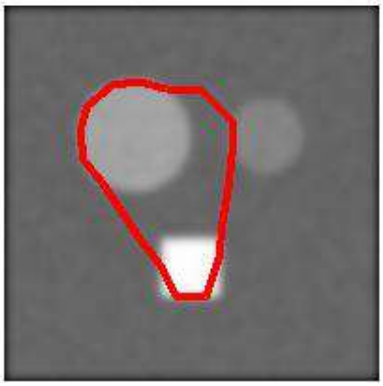
مشاهده می‌شود افزایش آلفا تغییر زیادی را در جواب نسبت به حلت اول ایجاد نکرده.

اما در انتها اثر تغییر ابعاد پنجره را بررسی می‌نماییم:

برای این منظور کانتور اولیه را دارای تضاریس در نظر می‌گیریم تا اثر افزایش ابعاد پنجره مشخص شود.

برای پنجره با طول۳ داریم:

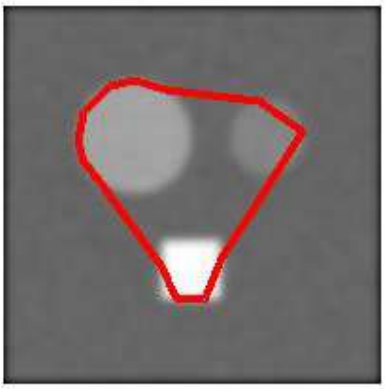
$\alpha = 5 \ \beta = 25 \ \gamma = 250$ win Length = 3



مشاهده می‌شود چون تضاریس زیاد بوده کانتور برای اینکه بتواند تضاریس را برطرف کند باید مقدار زیادی جابجا شود اما در پنجره با طول ۳ تنها می‌تواند یک پیکسل جابجا شود و لذا جواب قابل قبول نیست.

اما با افزایش طول پنجره به ۷ داریم:

$\alpha = 5 \ \beta = 25 \ \gamma = 250$ win Length = 7

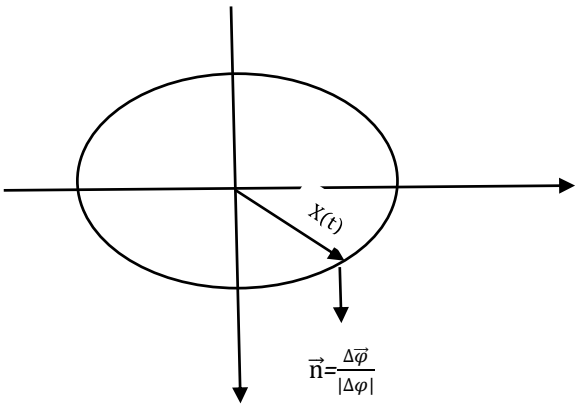


مشاهده می‌شود چون برای پنجره با طول ۷ کانتور می‌تواند تا ۳ پیکسل جابجا شود نتیجه بسیار بهتر شده است. بنابراین افزایش ابعاد پنجره می‌تواند موجب بهبود جواب شود.

ب)کد مربوط به این قسمت در تابع Level_Set_Method.m نوشته شده است:.

در این قسمت قرار است که از الگوریتم level set برای جداسازی تصویر بخش قبل استفاده نماییم.

این روش شامل سطوحی است که از سطح صفر بالا آمده و می‌توانند هر شکلی به خود بگیرند. هر نقطه روی کانتور در فضا را با φ(x(t),t) نمایش می‌دهیم. لذا سطح صفر برابر با φ(x(t),t) =0 خواهد بود. اگر برای سطح صفر و با دید از بالا به آن نگاه کنیم داریم:



این سطح می‌تواند هر شکلی باشد و حتما لازم نیست به صورت روبرو باشد.

که n بردار یکه عمود بر سطح کانتور و x(t) مکان می‌باشد.

لذا داریم: $\vec{n}.\frac{\partial x(t)}{\partial t}$=F

که F اندازه سرعت نقطه x در صفحه صفر در جهت عمود بر کانتور می‌باشد.

طبق معادله‌ی جکوویچ برای حرکت داریم: $|\Delta\varphi|.F+\frac{\partial\varphi}{\partial t}=0$

معادله را دیسکریت مینماییم: $\frac{\varphi^{t+\Delta t}-\varphi^{t}}{\Delta t}=-|\nabla\varphi|.F$

$\rightarrow \varphi^{t+\Delta t}=\varphi(t)-\Delta t\,|\nabla\varphi|.F$

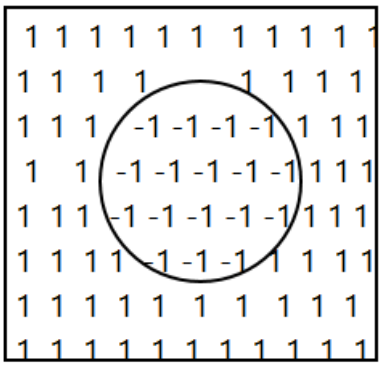
در حالت کلی حل معادله بالا ساده نمی‌باشد و آن بدلیل تغییر کانتور سطح صفر می‌باشد. می‌توانیم مثلا φ را طوری طراحی نماییم که همواره $|\nabla\varphi|$=1 باشد.

تابع روبرو همواره دارای $|\nabla\varphi|=1$ می‌باشد.

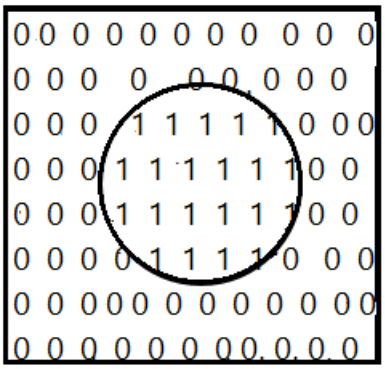
$$\varphi(x,y)=\begin{cases}+d & \text{برای خارج کانتور}\\ -d & \text{برای داخل کانتور}\end{cases}$$

که منجر به کانتوری مخروطی می‌شوند.

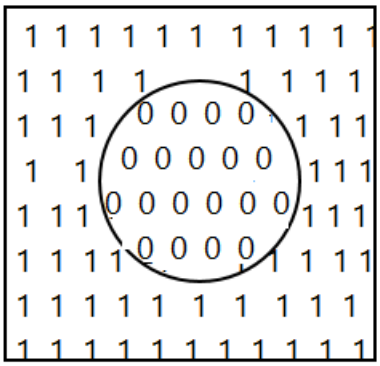
اما برای ساختن یک φ در برنامه متلب مثلا کانتور سطح صفر را به صورت زیر در نظر می‌گیریم:



سپس z را به دو صورت زیر تعریف می‌نماییم:



$Z=\varphi<0$



$Z=\varphi<0$

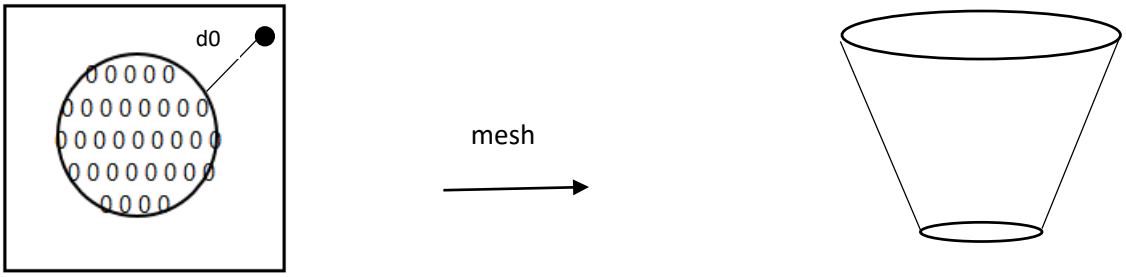
مقدار شعاع این دوایر با توجه به ابعاد تصویر تغییر میکند و باید حداقل محیط کل اشکالی که قرار است سگمنت شوند را در بر بگیرد.

و در نهایت φ به صورت زیر به دست می‌آید:

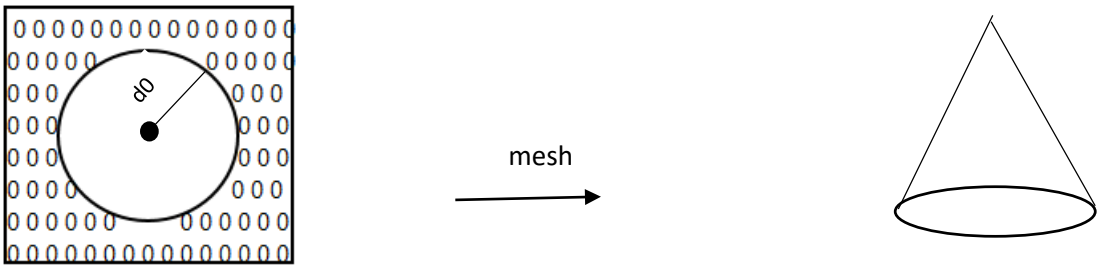
$$\varphi = double\left((\varphi < 0) * bwdis(\varphi < 0) - 0.5\right) - (\varphi < 0) * \text{bwdis}(\varphi > 0) - 0.5))$$

که bwdis روی ماتریس‌های باینری عمل نموده و خروجی آن برای هر پیکسل، فاصله پیکسل تصویر مورد نظر از اولین ۱ است که برای z در بالا به صورت زیر درمی‌آیند(d0 را برای نمونه فاصله یک پیکسل از اولین ۱ فرض کرده ایم):

Bwdis(φ<0)



Bwdis(φ>0)



اما برای انتخاب F و یا اندازه سرعت عمود بر کانتور در هر نقطه یک بحث وجود دارد. اگر F ثابت باشد در این صورت کانتور صفحه صفر حساسیتی به ویژگی‌های تصویر f نخواهد داشت و درواقع کانتور در هیچ جا نمی‌ایستد.

اما باید طوری F را در نظر بگیریم که در جایی که گرادیان تصویر زیاد است، سرعت کانتور کم شود و لذا باید یک رابطه عکس بین آن‌ها وجود داشته باشد و بنابراین F را بصورت زیر در نظر می‌گیریم:

$$F = \frac{1}{1 + \gamma|\nabla I|}$$

γ نیز یک ضریب است که با سعی و خطا به دست می‌آید.

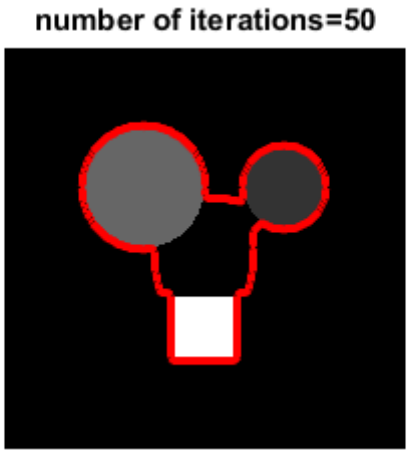
و بنابراین هرجا که گرادیان تصویر بزرگتر باشد سرعت کانتور کمتر خواهد شد.

حال این الگوریتم را می‌خواهیم روی تصویر داده شده پیاده سازی نماییم. برای این منظور با توجه به توضیحات گفته شده از کد زیر استفاده می‌نماییم:

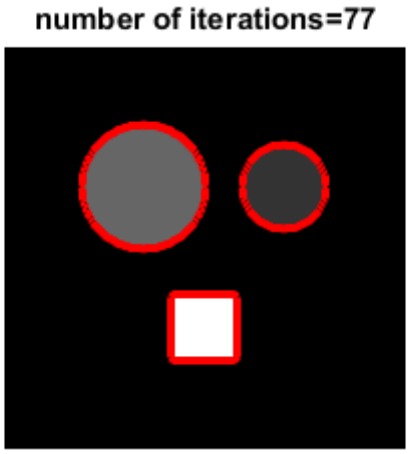
```
function Level_Set_Method(f)

[M N]=size(f);
[fx,fy]=gradient(f);
absgrad=sqrt(fx.^2+fy.^2);
gama=2;
g_m=1+gama*absgrad;
p=1./g_m;
p=double(p);
phi=ones(M,N);
for m=1:M
    for n=1:N
        D=sqrt((m-100)^2+(n-100)^2);
        if D<=75,phi(m,n)=-1;
        end
    end
end

phi = double((phi > 0).*(bwdist(phi < 0)-0.5) - (phi < 0).*(bwdist(phi > 0)-0.5));
figure;surf(phi);

e=.51;
figure,
for t=1:77
phi=phi+e.*p;
    phi = double((phi > 0).*(bwdist(phi < 0)-0.5) - (phi < 0).*(bwdist(phi > 0)-
0.5));

    imshow(f,[]);hold on
    z=contour(phi,[0,0],'r','LineWidth',3);title(['number of
iterations=',num2str(t)]);
    pause(0.00001)
end


end
```

برای اجرای سریع تر برنامه بهتر است شعاع دایره کانتور را خیلی بزرگ نگیریم و طوری باشد که فقط اندکی از محیط اشکالی که قرار است سگمنت شوند بزرگ تر باشد. ما با توجه به تصویری که داریم شعاع ۷۵ را انتخاب کردیم که در تکرار اول بدین شکل است و همانطور که مشاهده میشود کانتور را کوچکترین حالت ممکن در نظر گرفته ایم:

و سپس برنامه را برای چند تکرار اجرا کرده تا ۳ شکل را بطور کامل جدا نماید. مثلا برای ۵۰ تکرار نتیجه زیر را داریم:



مشاهده می‌شود هنوز اشکال به طور کامل سگمنت نشده اند . لذا تکرار را افزایش میدهیم. با افزایش تکرارها به صورت سعی و خطا تا زمانی که سه شکل کاملا سگمنت شوند به این نتیجه رسیدیم که اولین جایی که هر سه شکل سگمنت میشوند بعد از ۷۷ تکرار می‌باشد که نتیجه زیر را داریم:



همچنین مقدار γ نیز با سعی و خطا انجام می‌شود که ما در اینجا مقدار ۲ را گرفته ایم و نتیجه مناسب بود.

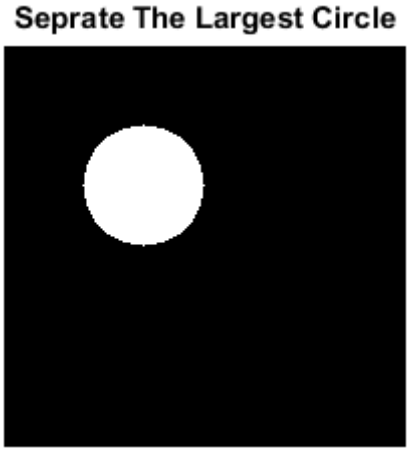
بنابراین برای حل این قسمت، شعاع کانتور سطح صفر را ۷۵ در نظر گرفتیم و برای ۷۷ تکرار توانستیم به نتیجه بالا دست یابیم.

ج) کد مربوط به این قسمت در تابع Morphology_Method.m نوشته شده است:.

در این قسمت، با فرض اینکه نمیدانیم دو دایره و مربع در کجای تصویرند، براساس مورفولوژی مکان آنها را مشخص نماییم. برای این کار ابتدا از یک strel دایروی برای آشکارسازی دایره‌ها استفاده می‌کنیم. دو دایره در تصویر داریم یکی به شعاع ۳۰ و دیگری ۲۰. برای آشکارسازی دایره‌ی بزرگتر، باید شعاع strel را طوری تنظیم نمود که از دایره کوچک بزرگتر بوده و از دایره بزرگ، کوچکتر باشد و پس از آن از دستور imopen استفاده نماییم. دستور imopen در ابتدا یک erosion انجام می‌دهد و لذا strel به دنبال دایره‌ای میگردد که بتواند کاملا در ان قرار گیرد. بنابراین دایره کوچکتر حذف می‌شود و دایره یزرگتر از آنجا که ابعادی بیشتر از strel دارد آشکار می‌گردد. و سپس با یک بار delation بعد از erosion ابعاد تصویر آشکار شده را به ابعاد اولیه باز میگردانیم.

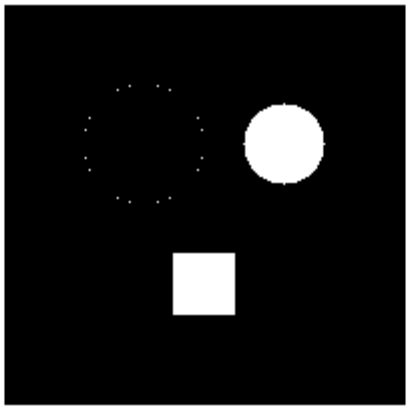
بهتر است در ابتدا تصویر را باینری کنیم تا به صورت زیر دراید:

سپس با وجه به توضیحات گفته شده و با انتخاب یک strel دایروی و با شعاع ۲۵ (از دایره کوچک بزرگتر و از دایره بزرگ کوچکتر)، نتیجه‌ی زیر را داریم:

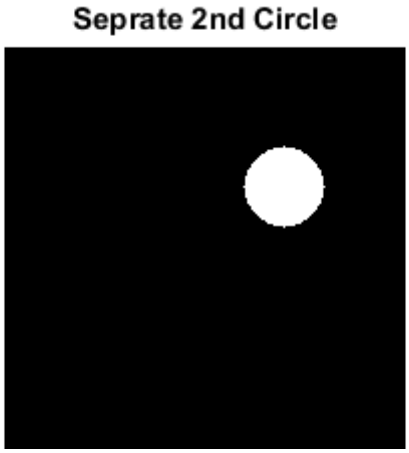


مشاهده می‌شود که مکان دایره بزرگ به خوبی مشخص شد.

حال برای آشکارسازی دایره کوچکتر ابتدا باید تصویر اصلی را از این تصویر به دست آمده کم نماییم چرا که اگر ما strel را برای آشکارسازی دایره کوچک، یک دایره با شعاعی کمتر از دایره کوچک انتخاب نماییم و مجددا از تصویر اصلی استفاده کنیم دوباره دایره بزرگ تر نیز آشکار می‌گردد. لذا با این کار آن را حذف می‌نماییم که نتیجه زیر مشاهده می‌شود:

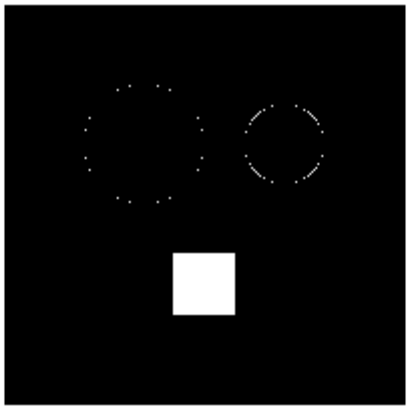


حالا در این قسمت از این تصویر استفاده می‌نماییم. از آنجا که شعاع دایره کوچک تر ۲۰ میباشد لذا strel را یک دایره با شعاع ۱۸ در نظر گرفته و مجددا از دستورimopen برای تصویر بالا استفاده می‌نماییم که نتیجه زیر حاصل می‌شود:

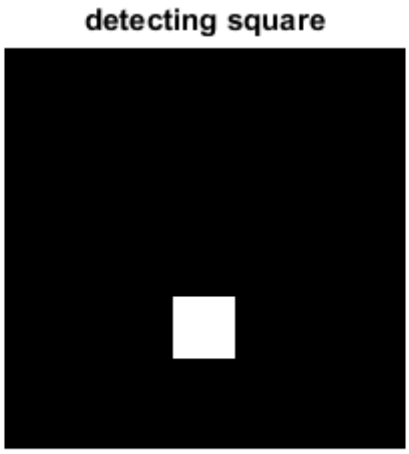


مشاهده می‌شود که دایره کوچکتر نیز آشکار می‌شود.

حال مجددا تصویر قبلی را از این تصویر کم می‌کنیم تا دو دایره حذف شوند و فقط مربع باقی بماند که نتیجه زیر مشاهده می‌شود:



در نهایت نوبت به آشکارسازی مربع میرسد. برای این منظور از تصویر بالا استفاده موده و strel را یک مربع با طول ضلع کمتر از مربع تصویر انتخاب می‌نماییم .مثلا مربعی با ضلع ۱۰ و مجددا از دستورimopen استفاده مینماییم. و نتیجه‌ی زیر مشاهده می‌شود:



بنابراین توانستیم هر دو دایره و مربع را بطور جداگانه و به واسطه دانش مورفولوژی آشکارسازی نماییم.

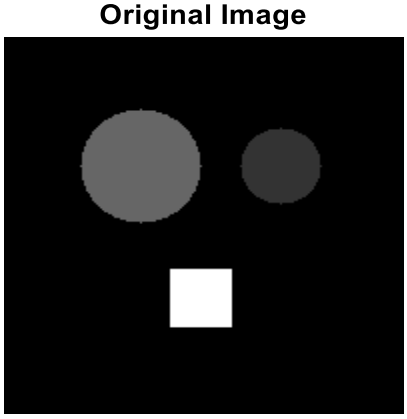
کد استفاده شده برای انجام این مراحل در ادامه آورده شده است:

```
f = 25.*ones(200,200);
%%%%%% Initial values
RR=[30 20 15];
xx_center = [70 70 140];
yy_center = [70 140 100];
incriment = [10 5 25];
%%%% Create Image

for i=1:3
    R = RR(i);
    x_center = xx_center(i);
```

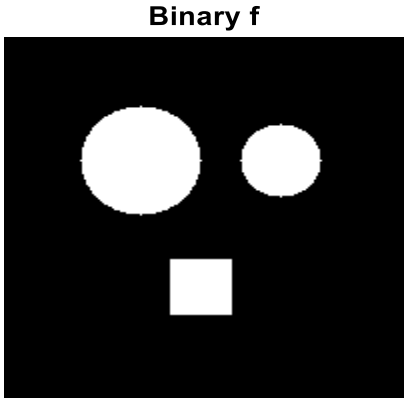
```
    y_center = yy_center(i);


    %%% Create circles
    if i==1 || i==2
        temp_R = zeros(2*R+1,2*R+1);
        for m=-R:R
            for n=-R:R
                if sqrt(m^2 + n^2) <= R
                    temp_R(m+R+1,n+R+1) = incriment(i);
                end
            end
        end
        ff = zeros(200,200);
        ff(x_center-R:x_center+R,y_center-R:y_center+R) = temp_R;
        f = ff + f;


        %%%%%% Create Square
    else
        temp_R = incriment(i) .* ones(2*R+1,2*R+1);
        ff = zeros(200,200);
        ff(x_center-R:x_center+R,y_center-R:y_center+R) = temp_R;
        f = ff + f;
    end
end
figure,imshow(f,[]);
title('original image')

ind_1 = find(f==25);
ind_2 = find(f>25);
f(ind_1) = 0;
f(ind_2) = 1;
 figure,imshow(f,[]);

% %%%%%  j
se_A = strel('disk',25,0);
A = imopen(f,se_A);
figure, imshow(A);


ff= f-A;
 figure,imshow(ff);
se_B = strel('disk',18,0);
B = imopen(ff,se_B);
figure,imshow(B)


fff = ff - B;
 figure,imshow(fff);
se_C = strel('square',10,0);
C = imopen(fff,se_C);
figure,imshow(C)
```

د) کد مربوط به این قسمت در تابع square_Detection.m نوشته شده است.

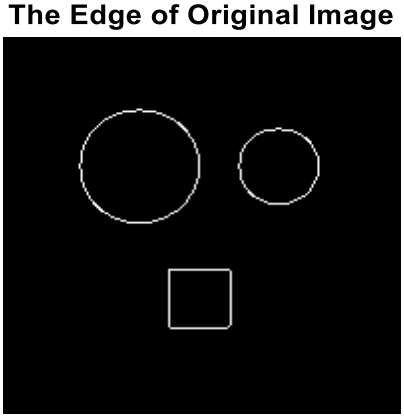
در این قسمت از ما خواسته شده است تا بتوانیم مربع شکل زیر را توسط تبدیل هاف خط جدا کنیم.



برای استفاده از تبدیل هاف، ابتدا شکل را باینری می‌کنیم



حال برای استفاده از تبدیل هاف ابتدا باید توسط یک تشخیص دهنده لبه، نظیر کنی، لبه ها را جدا کنیم که حاصل را در زیر شاهد هستیم:

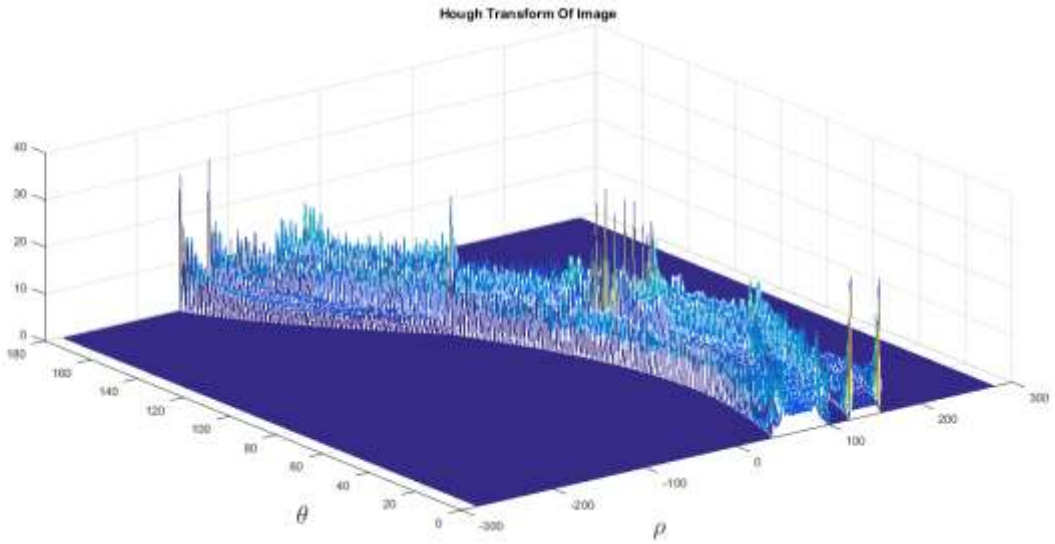


با توجه به اینکه تشخیص دهنده کنی که در سوال ۲ استفاده کردیم بدلیل اینکه از گرادیان در راستای محور x و y استفاده میکند تنها برای تشخیص لبه های افقی و عمودی مناسب است و برای مربع های دوران داده شده (که در این سوال این حالت

را نیز مورد بررسی قرار می‌دهیم) و همچنین تشخیص دوایر، مشکل دارد، به همین علت از تشخیص دهنده لبه متلب استفاده میکنیم که دقت بسیار بهتری برای تشخیص لبه ها دارد (با توجه به اینکه در این سوال موضوع اصلی استفاده از تبدیل هاف است، مجاز به استفاده از کد آماده برای تشخیص دهنده لبه هستیم به این علت که پس از استفاده از canny که کد آن را نوشته بودیم نتایج خوبی بدست نیامد که علت این امر هم ضعف کنی در لبه های زاویه دار است)

حال از تصویر که پس از استفاده از تشخیص دهنده لبه بدست آورده ایم، تبدیل هاف میگیریم.

نتیجه را در زیر شاهد هستیم:



۲ شرط برای اینکه ۴ نقطه (در فضای تبدیل هاف) متعلق به ۴ ضلع یک مربع باشد، وجود دارند

۱- ۴ نقطه ، را بتوان به دو دسته، دو نقطه ای دسته بندی کرد به طوریکه نقاط هر دسته، دارای زاویه یکسان و ρ متفاوت باشند

۲- دو دسته دارای اختلاف زاویه ۹۰ باشد.

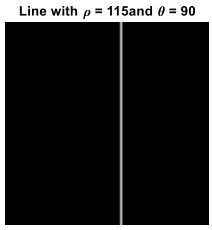
حال کد را بگونه ای مینویسیم که بتواند این نقاط را جدا سازی کند توجه شود که هنگامی خط به صورت افقی باشد هم زاویه آن در ۰ می باشد هم در ۱۸۰ که این یک حالت استثنا است که این مسئله را هم در کد دخالت داده‌ایم.

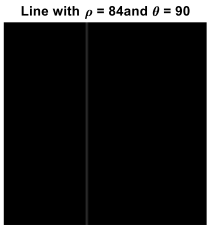
همچنین کد باید به گونه ای باشد که بتواند ماکسیوم بین چند نقطه مجاور را بدست آورد و اگر چند پیک مجاور هم داشته باشیم، تنها یک نقطه را به عنوان پیک انتخاب کند

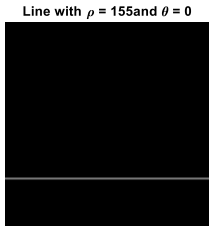
در زیر شاهد ۴ نقاطی که به عنوان ماکسیوم شناسایی شده هستیم (این نقاط باید دارای دو شرط گفته شده فوق باشند)

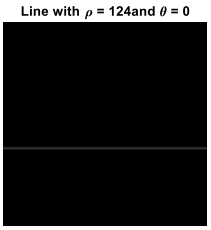
ρ	۱۲۴	۱۵۵	۸۴	۱۱۵
θ	۰	۰	۹۰	۹۰

حال میتوان با استفاده از ρ , θ های ۴ گانه ۴ خط رسم کرد که حاصل آن به صورت شکل زیر خواهد بود:

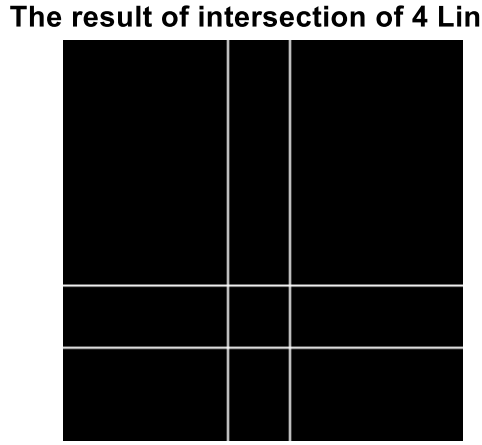








شکل حاصل از تلاقی این ۴ نقطه بایکدیگر به صورت زیر می‌باشد :



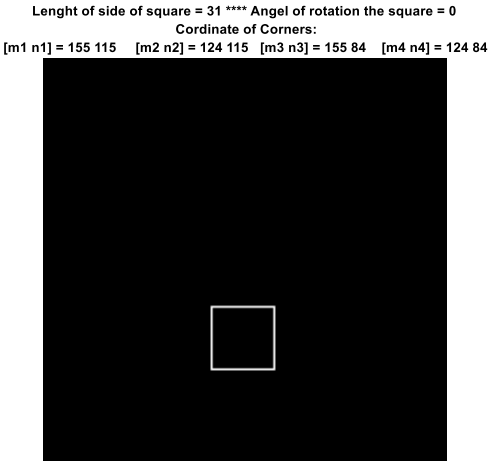
حال باید کد را بگونه ای بنویسیم که ۴ نقطه فوق را که تشکیل یک مربع داده اند را بدست آوریم برای این کار محل تقاطع هر دو خط را (در صورت) وجود بدست می‌اوریم حاصل نهایی آن چهار نقطه می‌شود.

n	۱۱۵	۱۱۵	۸۴	۸۴
m	۱۵۵	۱۲۴	۱۵۵	۱۲۴

مشاهده می‌شود به خوبی توانسته ایم این ۴ نقطه را بدست آوریم. حال در مرحله اخر باید این ۴ نقطه را به یکدیگر وصل کنیم بگونه ای که مربع بدست آید. توجه شود که برخی نقاط را نمیتوان به یکدیگیر وصل کرد (دو نقطه ای که به صورت

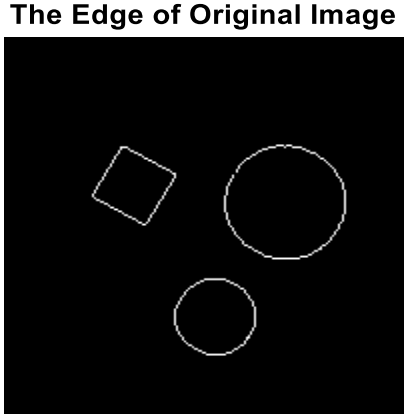
قطری با یکدیگر در ارتباط هستند) و باید در کد شرطی بگذاریم که این حالت را نیز شامل شود و ۴ نقطه صحیح به یکدیگر متصل شوند.

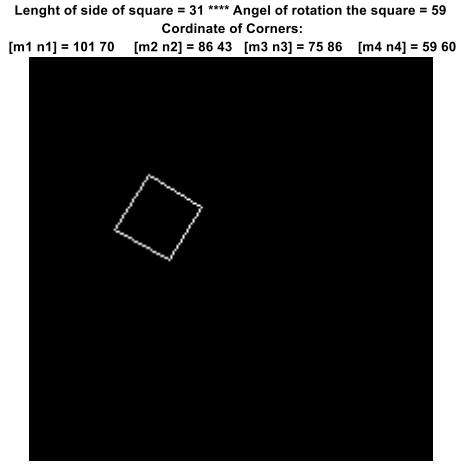
در زیر شاهد جواب نهایی هستیم:



مشاهده می‌شود که بخوبی توانسته ایم این مربع را توسط تبدیل هاف حداسازی کنیم.

توجه شود کد بگونه ای نوشته شده است که حتی در صورتی که مربع چرخانده شود بخوبی میتواند مربع را جداسازی کند برای مثال پس از دوران شکل به اندازه ۲۴۰ درجه شاهد نتایج الگوریتم پیشنهادی هستیم:





در زیر شاهد کد های مربوط به این قسمت هستیم:

```
function squre_Detection_Hough(f)

f = edge(f);
figure,imshow(f),title('The Edge of Original Image');

[M,N] = size(f);
[ind_x,ind_y] = find(f); %%% find indices that are equal to 1


%%%%%% Calculate Hogh Transform
step_theta = pi/180;
theta = 0:step_theta:pi-step_theta;
l_theta = length(theta);
M_2 = floor(M*sqrt(2));
ro = zeros(2*M_2 , l_theta);

for i=1:length(ind_x)

    temp_ro = round(ind_x(i)* cos(theta) + ind_y(i).*sin(theta));
    ff = zeros(2*floor(M*sqrt(2)) , l_theta);
    for j=1:1:length(temp_ro)
        ff(temp_ro(j)+ M_2,j) = 1;
    end
    ro = ro + ff;
end

%%%%%%% Plot Hough Transform
[NN,MM] = meshgrid(0:179,-M_2:M_2-1);
figure,mesh(MM,NN,ro),xlabel('\rho','FontSize', 20) , ylabel('\theta','FontSize', 20);
title('Hough Transform Of Image')


%%%%%% Extract Maximum Value of Hough and Delete Noisy Peaks
ro_sort = sort(ro(:),'descend');
max_ro = ro_sort(1:10);

p=0;
for i=1:length(max_ro)
    [mm,nn] = find(ro == max_ro(i));
    for k=1:length(mm)

        if i==1 && k==1 && abs(nn(k)-180)>5 %%% Ignore theta = 180
            p=p+1;
            ind_x_max(p) =  mm(k);
            ind_y_max(p) = nn(k);
        else
            counter =0;
            for j=1:p
                dd = sqrt((ind_x_max(j)-mm(k)).^2 + (ind_y_max(j)-nn(k)).^2);
                if dd>15
                    counter = counter +1;
                end
```

```
            end
            if counter == p && abs(nn(k)-180)>5
                p=p+1;
                ind_x_max(p) =  mm(k);
                ind_y_max(p) = nn(k);
            end
        end
    end

end

%%%%% each 2 poits of a square must have the similar Theta
for i=1:length(ind_y_max)
    for j=1:length(ind_y_max)
        if abs(ind_y_max(i) - ind_y_max(j)) < 3
            ind_y_max(i) = ind_y_max(j);
        end
    end
end

%%%%% Delete Noisy Peaks
p=0;
while length(ind_x_max) > 0
    aa = find(ind_y_max(1) == ind_y_max);
    if length(aa) > 1
        for i=length(aa) : -1 : 1
            p=p+1;
            rho_max(p) = round(ind_x_max(aa(i))-M*sqrt(2));
            theta_max(p) = ind_y_max(aa(i))-1;
            ind_x_max(aa(i)) =[];
            ind_y_max(aa(i)) =[];
        end
    else
        ind_x_max(1) = [];
        ind_y_max(1) = [];
    end
end

%%%% Plot Line Between each 2 point with RHO and THETA
gg = zeros(M,N,length(rho_max));
ggg = zeros(M,N);
ggg_1 = zeros(M,N);
figure,
if abs(abs(theta_max(3)-theta_max(1))-90) < 3
    for i=1:length(rho_max)

        for m=1:M
            for n=1:N
                if abs(rho_max(i) -( m*cosd(theta_max(i)) + n*sind(theta_max(i)))) < 1
                    gg(m,n,i) = 1;
                    ggg(m,n) = 1;
                    ggg_1(m,n) =1;
                end
            end
        end
        L_gg(i)  = length(find(ggg));
        subplot(2,2,i);
        imshow(ggg),title(['Line with \rho = ',num2str(rho_max(i)),'and \theta =
',num2str(theta_max(i))])
        ggg = zeros(M,N);
    end
end
figure,imshow(ggg_1),title('The result of intersection of 4 Lines');

G  = zeros(M,N,length(rho_max));
for i=1:length(L_gg)
    aa = find(L_gg == max(L_gg));
    G(:,:,i)= gg(:,:,i);
    L_gg(aa) = 0;
end



%%%%%%%%%% Extract 4 Points that obtained from square of 4 Lines
p=0;
for i=1:length(rho_max)
```

```
    [m_1 n_1] = find(G(:,:,i)==1);
    for j=i+1: length(rho_max)
        [m_2 n_2] = find(G(:,:,j)==1);

        k=0;
        for r=1:length(m_1)
            temp_ind = find(m_1(r) == m_2);
            if length(temp_ind) ~= 0
                for j=1:length(temp_ind)
                    if n_1(r) == n_2(temp_ind(j))
                        p=p+1;
                        pixel(p,:) =[m_1(r) n_1(r)]; %%%%% Cordinate of Corner Points
                        k=1;
                        break
                    end
                end
                if k==1
                    break
                end
            end
        end

    end
end

%%%%%% Chack which 2 points can connect to each other
p=0;
for i=1:length(pixel)
    point_1 = pixel(i,:);
    for j=i+1:length(pixel)
        point_2 = pixel(j,:);
        p=p+1;
        mm(p) = round(sqrt( (point_2(2) - point_1(2)).^2  + (point_2(1) - point_1(1)).^2));
        data(p,:) = [point_1 point_2 mm(p)];
    end
end

%%%%%%%%%%%%%% connect 2 points ans make square
g_1=zeros(M,N);
min_d = min(mm);
for i=1:6
    if abs(data(i,5)-min_d)<5
        % distances according to both axes
        y1 = data(i,1);  y2 = data(i,3); x1 = data(i,2);  x2 = data(i,4);
        xn = abs(x2-x1);
        yn = abs(y2-y1);

        % interpolate against axis with greater distance between points;
        % this guarantees statement in the under the first point!
        if (xn > yn)
            xc = x1 : sign(x2-x1) : x2;
            yc = round( interp1([x1 x2], [y1 y2], xc, 'linear') );
        else
            yc = y1 : sign(y2-y1) : y2;
            xc = round( interp1([y1 y2], [x1 x2], yc, 'linear') );
        end

        % 2-D indexes of line are saved in (xc, yc), and
        % 1-D indexes are calculated here:
        ind = sub2ind( size(g_1), yc, xc );

        % draw line on the image (change value of '255' to one that you need)
        g_1(ind) = 1;
%         figure,imshow(g_1)
    end
end

figure,imshow(g_1)
tit = sprintf(' Lenght of side of square = %d **** Angel of rotation the square = %d \n
Cordinate of Corners:\n [m1 n1] = %d %d     [m2 n2] = %d %d   [m3 n3] = %d %d    [m4 n4] = %d
%d'...

,min_d,min(abs(theta_max)),pixel(1,1),pixel(1,2),pixel(2,1),pixel(2,2),pixel(3,1),pixel(3,2),p
ixel(4,1),pixel(4,2));
title(tit)
                                                                                    end
```

سوال پنجم) کد این قسمت در HW6_Q5 قرار دارد:

در این سوال تصویر اثر انگشت طیر داده شده است و قرار است از طریق تکنیک های موجود در مورفولوژی آن را حتی الامکان بهبود بخشیم:

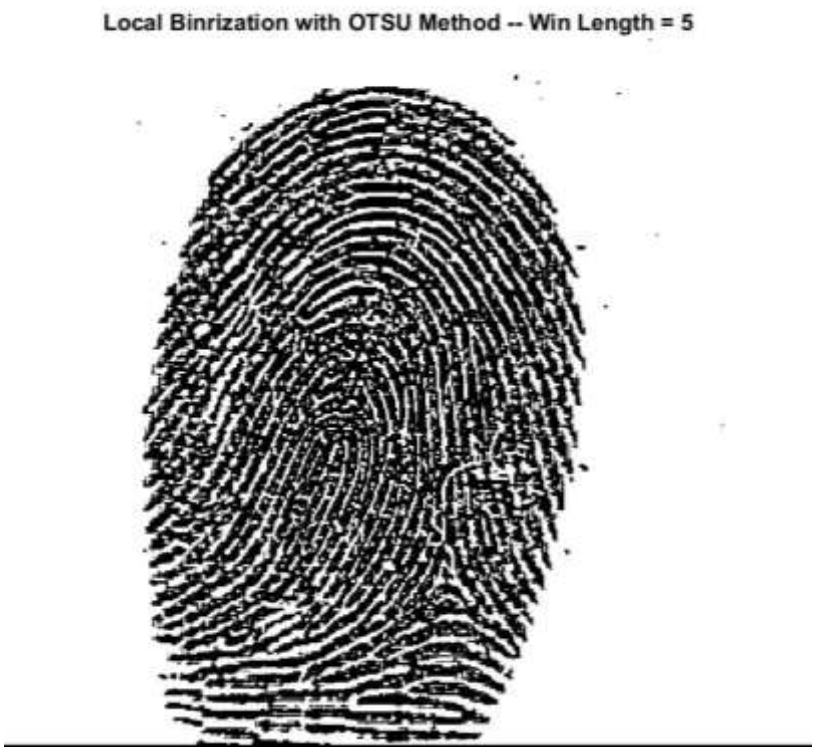
ابتدا نیاز است که تصویر را باینری نماییم. برای این منظور باید از یک ترشولد استفاده نماییم. که این ترشولد را می‌توان به دو روش گلوبال و لوکال تعیین نمود. در روش گلوبال که از Tی ثابت استفاده میشود نتیجه به صورت زیر خواهد بود:

نتیجه حاصل از باینری سازی گلوبال اصلا خوب نبوده و در این مورد مناسب نیست لذا از روش لوکال استفاده می‌نماییم برای این منظور از کد زیر استفاده می‌نماییم:

```
clc;
clearvars -except f_local_b;
close all;

f = imread('finger.tif');
ff = im2double(f);
% figure,imshow(f),title('Oroginal Image');

[M,N] = size(ff);

W = 9; w_2 = floor(W/2);
f_local_b = zeros(M,N);
for m=(w_2+1) :5: M - (w_2+1)
    for n= (w_2+1) :5: N - (w_2+1)
        temp = ff(m-w_2 : m+w_2, n-w_2 : n+w_2);
        thr = graythresh(temp);
        if thr ~=0
           k=0;
        end
        temp_2 = zeros(W,W);
        for mm=1:W
            for nn=1:W
               if   temp(mm,nn) > thr
                    temp_2(mm,nn) = 1;
               end
            end
        end
        f_local_b(m-w_2 : m+w_2, n-w_2 : n+w_2) = temp_2;
    end
end
```

```
h=figure,imshow(f_local_b),title(['Local Binrization with OTSU Method -- Win Length
= ' , num2str(W)]);
tit = sprintf('Binrization_W_%d.jpg',W);
saveas(h,tit);
```

و بنابراین نتیجه حاصل از باینریزیشن لوکال و از روش OTSU برای طول پنجره های مختلف به صورت زیر خواهد بود:



Local Binrization with OTSU Method -- Win Length = 7

Local Binrization with OTSU Method -- Win Length = 9

نتیجه برای پنجره با طول ۹ نسبتا مناسب می‌باشد. در ادامه قرار است از روش‌های مورفولوژی تصویر را ارتقا دهیم. برای این منظور از کد زیر استفاده می‌نماییم:

```
r1 =2;
strl = strel('disk',r1,0);
f1 = imclose(f_local_b , strl);
h1 = figure,imshow(f1),title(['Image after imclose with a disk with radius =
',num2str(r1)]);
tit_1 = sprintf('W_%d_imclose_Disk_%d.jpg',W,r1);
saveas(h1,tit_1);


r2 = 2;
strl = strel('square',r2);
f2 = imopen(f1 , strl);
h2 = figure,imshow(f2),title(['Image after imopen with a square with side =
',num2str(r2)]);
tit_1 = sprintf('W_%d_imopen_Square_%d.jpg',W,r2);
saveas(h1,tit_1);
```

یعنی ابتدا از دستور imclose استفاده می‌نماییم با strel دیسک و برای شعاع های مختلف strel نتایج زیر را داریم:

Image after imclose with a disk with radius = 2

همانطور که می‌دانیم دستور imclose ایندا از یک dilation و سپس از یک erosion استفاده می‌نماید و لذا در آن تمام اجزا نزدیک به هم پیوسته می‌شوند.

حال تصویر imclose شده را imopen مینماییم که نتیجه به صورت زیر است:

Image after imopen with a square with side = 1

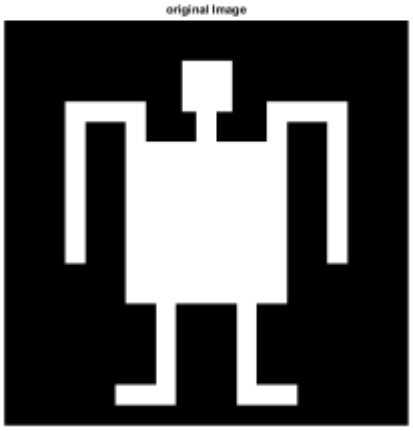
Image after imopen with a square with side = 2

همانطور که می‌دانید دستور imopen از یک erosion وسپس یک dilation استفاده می‌نماید و لذا محیط کلی را نگه داشته و گوشه های محدب را گرد می‌کند و گوشه های مقعر را حفظ می‌نماید.

بنابراین از طریق یک local binerization و پس از آن یک imclose و در نهایت یک imopen توانستیم تصویر را حتی الامکان بهبود بخشیم.

سوال شش)(کد این قسمت در فایل HW6_Q6 قرار دارد)

در این سوال قرار است الگوریتمی را بر مبنای هشت strel داده شده بنویسیم تا اسکلت تصویر زیر نمایش داده شود.(این تصویر در f_skeleton.mat ذخیره شده است):



برای این منظور باید از الگوریتم hit&miss استفاده نماییم. ۸ تا strel داده شده بصورت زیر می‌باشند: (این strel ها در BW.mat ذخیره شده اند).

B_1

۰	۰	۰
	۱	
۱	۱	۱

$B_2 =$

	۰	۰
۱	۱	۰
۱	۱	

B_3

۱		۰
۱	۱	۰
۱		۰

$B_4 =$

۱	۱	
۱	۱	۰
	۰	۰

B_5

۱	۱	۱
	۱	
۰	۰	۰

$B_6 =$

	۱	۱
۰	۱	۱
۰	۰	

B_7

۰		۱
۰	۱	۱
۰		۱

$B_8 =$

۰	۰	
۰	۱	۱
	۱	۱

الگوریتم hit & miss به این طریق عمل می‌کند که برای یک تصویر و یک strel مشخص هرجایی از تصویر که strel در آن کاملا فیت می‌شود، پیکسل مرکزی را ۱ قرار می‌دهیم. مثلا فرض کنیم که ماتریس زیر مربوط به یک تصویر می‌باشد و B strel موردنظر باشد:

۰	۰	۰	۰	۰	۰
۰	f = ۱	۱	۰	۰	۰
۰	۱	۱	۱	۱	۰
۰	۱	۱	۱	۱	۰
۰	۰	۱	۱	۰	۰
۰	۰	۱	۰	۰	۰

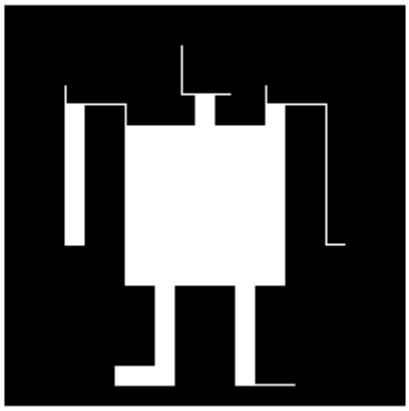
B =

	۰	۰
۱	۱	۰
	۱	

توجه شود که پیکسل‌های خالی در strel مهم نیستند که ۰ باشند یا ۱ فقط کافی است سایر پیکسل‌های B به طور کامل درf قرار گیرد. نتیجه حاصل از هیت اند میس این دو به صورت زیر خواهد بود:

۰	۰	۰	۰	۰	۰
۰	۰	۰	۱	۰	۰
۰	۰	۰	۰	۱	۰
۰	۰	۰	۰	۰	۰
۰	۰	۰	۰	۰	۰
۰	۰	۰	۰	۰	۰

حال الگوریتم را برای تصویر مورد نظر خودمان در سوال انجام داده و در هرجای تصویر که hit&miss اتفاق بیفتد آن پیکسل را ۱ قرار می‌دهیم و سپس تصویر اصلی را از تصویر hit miss شده کم می‌نماییم. برای نمونه اگر مثلا این الگوریتم را برای B_2 انجام دهیم نتیجه‌ی حاصل به صورت زیر خواهد بود:

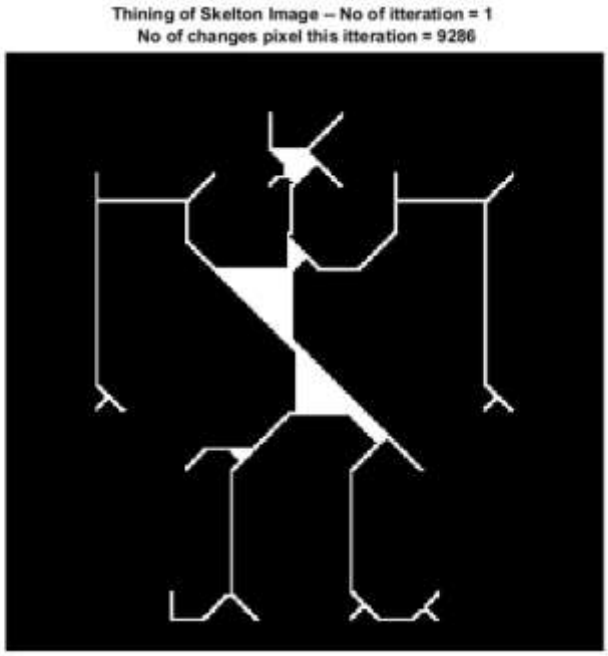


برای اجرای این الگوریتم از کد زیر استفاده نموده ایم:

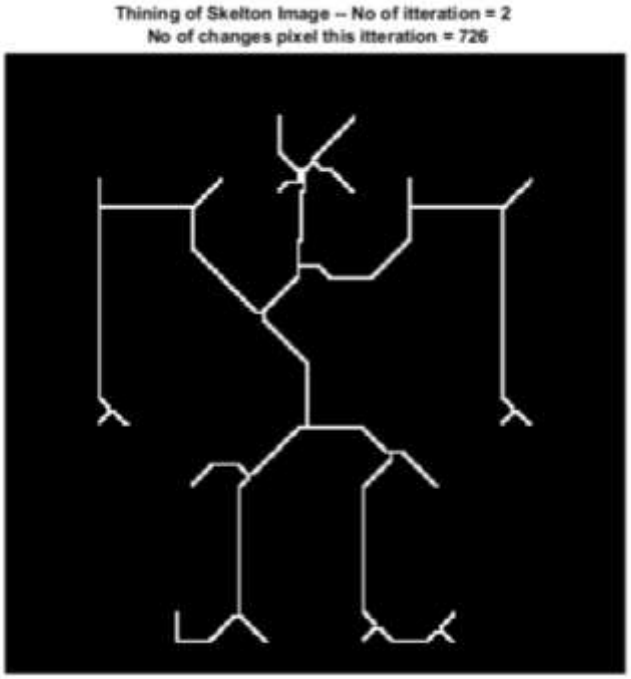
```
load('f_skeleton.mat');
load('BW.mat','BW');    %%% Load Strels
[M,N] = size(f);

figure,imshow(f),title('original Image');

non_zero_skeletion = length(find(f)); %%%  # of non_zero elements of skeletion (the
stopping criteria is based on this)

counter = 0; %%%  # of itteration of applying strels on Image
g=f;
exit_1 =0; %%% stopping cirteria of Main Loop
while exit_1 == 0
    counter = counter + 1;
    for i=1:8
        BBW= BW(:,:,i); %%% using strek No i
        w = length(BBW);
        w_2 = floor(w/2);
        exit_2 = 0;

        while exit_2 == 0;

            ff = zeros(200,200);

            ind_neg = find(BBW == -1); l_ind_neg = length(ind_neg); %%% indices
that must be zero
            ind_pos = find(BBW ==  1); l_ind_pos = length(ind_pos); %%% indices
that must be one

            for m = 1+w_2 : M-w_2
                for n = 1+w_2 : N-w_2

                    temp = f(m-1:m+1,n-1:n+1); %%% window that we want compare it
with strel
                    temp_neg = find(temp == 0);  l_temp_neg = length(temp_neg); %%%
indices that r 1 of window (center pixel is m,n)
                    temp_pos = find(temp ==  1);  l_temp_pos = length(temp_pos);%%%
indices that r 0 of window (center pixel is m,n)

                    %%%% Assessing weather 1 indices are exit or not
                    c1=0;
                    for j =1:l_ind_neg
                        c1 = c1 + length(find(ind_neg(j) == temp_neg));
                    end

                    c2=0;
```

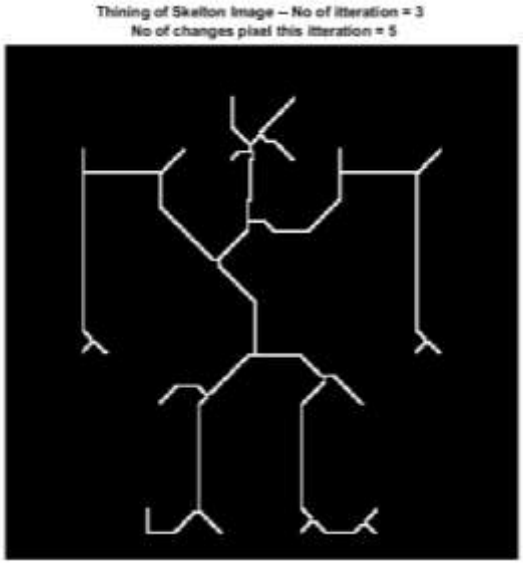
```
                    if c1 == l_ind_neg
                        %%%% Assessing weather 1 indices are exit or not
                        for j=1:l_ind_pos
                            c2 = c2 + length(find(ind_pos(j) == temp_pos));
                        end
                        %%%% if 0 & 1 indices are exist, Hit-Miss ocures
                        if c2 == l_ind_pos
                            ff(m,n) =1;
                        end
                    end

                end
            end

            f = f - ff;
            aa_2 = find(ff == 1);
            if isempty(aa_2) %%%% if this condition is True , we have No change
with this strel and must change or strel
                exit_2 =1;
            end
        end
    end

    f = f-ff;
    l_f = length(find(f==1));
    l_g = length(find(g==1));
    if  l_f == l_g %%%% if this condition is True , we have No change and we must
exit
        exit_1 =1;
    end
    g=f;
    tit = sprintf('Thining of Skelton Image -- No of itteration = %d \n No of
changes pixel this itteration = %d',counter,l_g-l_f);
    figure,imshow(f),title(tit)
end
```

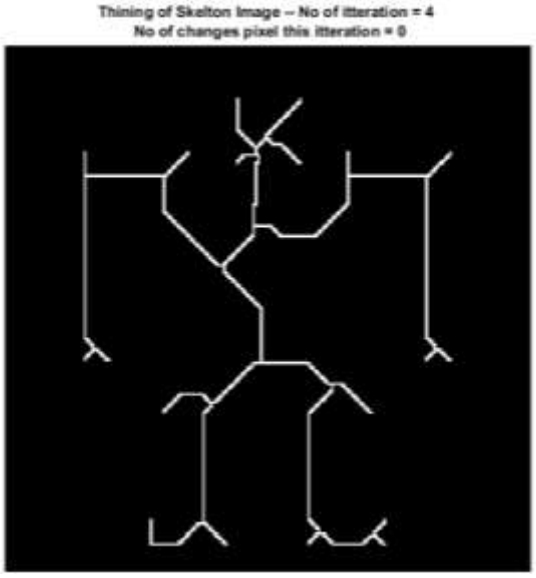
حال الگوریتم را یک بار برای B_1 تا B_8 برای تصویر انجام داده و نتیجه زیر را مشاهده می‌نماییم:



مشاهده می‌کنیم که با یک بار اعمال الگوریتم برای ۸ تا strel نتیجه‌ی بالا به دست آمد. اما مشاهده می‌نماییم که اسکلت به طور کامل مشخص نشده است. لذا الگوریتم را مجددا تکرار می‌کنیم و آنقدر الگوریتم را تکرار می‌نماییم که دیگر نیاز به هیت اند میس نداشته باشد.







مشاهده می‌شود که با ۴ بار تکرار الگوریتم دیگر تغییری در تصویر ایجاد نمی‌شود و بنابراین اسکلت تصویر مورد نظر به ازای ۴ بار اعمال strel های B_1 تا B_8 به صورت تصویر بالا در می‌آید.