بنام خدا

دانشگاه صنعتی اصفهان
دانشکده برق و کامپیوتر

تمرین سری سوم –پردازش تصاویر دیجیتال

استاد: دکتر سعید صدری

پروانه رشوند ۹۴۱۰۱۲۴
رضا سعادتی فرد ۹۴۱۱۳۹۴

تاریخ تحویل ۹۵/۰۷/۲۷

سوال یک-الف) (کد این قسمت در HW3_Q1_A.m موجود می باشد)

در این سوال، می خواهیم الگوریتم Niblack و Sauvola در دوسطحی سازی محلی را بر روی یک تصویر پیاده سازی نماییم. برای این منظور یک پنجره به طول ۷ در نظر می گیریم. در معیار نایبلک، T را به صورت $T = m - k\sigma$ در نظر میگیریم. یعنی معیار نایبلک براساس میانگین کل پنجره و همچنین پراکندگی سطوح روشنایی عمل می کند. K را که ثابت نایبلک می باشد در این سوال ۰,۲ قرار می دهیم و بر مبنای معیار نایبلک، کد مورد نظر را بدین صورت می نویسیم: همچنین برای افزایش سرعت اجرای برنامه از integral image که در متن درس به آن اشاره شده است، استفاده نموده ایم:

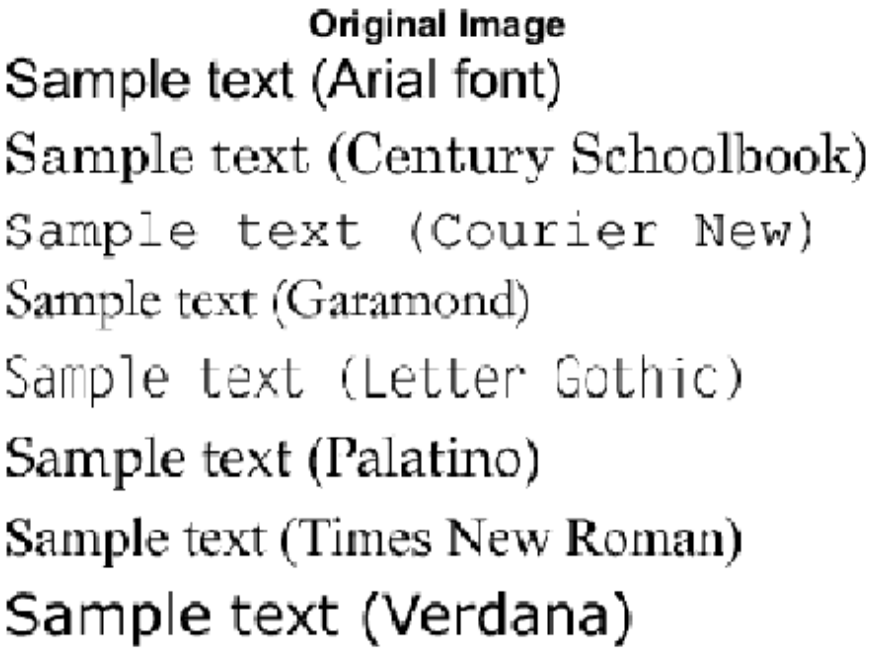
```
I=imread('degarded_text.png');
I = 255 * im2double(I);                %%% conver unit8 to decimal number
figure, imshow(I,[]),title('Original Image');

%%%%%%%%%%%%%%%%%%%%%%%%%%%     NiBlack & Savoula Approach   %%%%%%%%%%%%%%
k=.2;
R = 47;
win_length = 7;

[M,N] = size(I);
step_win = fix(win_length/2);

I_temp = I; %%%%%%%%%%%%%%%%%%%%
[M_temp,N_temp] = size(I_temp);

step_win = fix(win_length/2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Integeral Image  %%%%%%%%%%%%%%%%%%%%%%%%%%

I_Integral = zeros(M+1,N+1);
I_Integral(2:M+1,2:N+1) = I;

I_Integral_2 = zeros(M+1,N+1);
I_Integral_2(2:M+1,2:N+1) = I.^2;

Int_Mat = zeros(M+1,N+1);          %%% Integral matrix
Int_Mat_2 = zeros(M+1,N+1);        %%% Integral matrix power 2

for m=2:M+1
    for n=2:N+1
        Int_Mat(m,n) = Int_Mat(m-1,n) + Int_Mat(m,n-1) - Int_Mat(m-1,n-1) +  I_Integral(m,n);
        Int_Mat_2(m,n) = Int_Mat_2(m-1,n) + Int_Mat_2(m,n-1) - Int_Mat_2(m-1,n-1) +
I_Integral_2(m,n);
    end
end

%%% calculate mean and variance
p=0;
for m=1 + step_win : M - step_win
    p=p+1; q=0;
    for n=1 + step_win :N - step_win
        q=q+1;
        m_1 = m+ step_win+1;  n_1 = n + step_win+1;    coef_1 = m_1*n_1;
        m_2 = m_1 - win_length;   n_2 = n_1;                   coef_2 = m_2*n_2;
        m_3 = m_1;                n_3 = n_1 - win_length;      coef_3 = m_3*n_3;
        m_4 = m_1-win_length;     n_4 = n_1-win_length;        coef_4 = m_4*n_4;
        mu_Int(p,q) = 1/win_length^2 * (Int_Mat(m_1,n_1) - Int_Mat(m_2,n_2) - Int_Mat(m_1,n_3)
+  Int_Mat(m_4,n_4));

        var_Int(p,q) = sqrt((1/win_length^2 * (Int_Mat_2(m_1,n_1) - Int_Mat_2(m_2,n_2) -
Int_Mat_2(m_1,n_3) +  Int_Mat_2(m_4,n_4)))...
            - mu_Int(p,q).^2);

        T_Niblack_Int(p,q) = mu_Int(p,q) - k*var_Int(p,q);            %%% Threshold Niblack
        T_Savoula_Int(p,q) = mu_Int(p,q)*(1+k*(var_Int(p,q)/R - 1));  %%% THreshold Savoula
    end
end

T_temp = zeros(M,N);
```

```
T_temp(1 + step_win : M - step_win,1 + step_win :N - step_win) = T_Niblack_Int;
T_Niblack_Int = T_temp;

T_temp = zeros(M,N);
T_temp(1 + step_win : M - step_win,1 + step_win :N - step_win) = T_Savoula_Int;
T_Savoula_Int = T_temp;


%%%% Niblack
I_Niblack=I;
index_Ni_255 = find(I_Niblack >= T_Niblack_Int);
index_Ni_0 =   find(I_Niblack < T_Niblack_Int);

I_Niblack(index_Ni_255)=255;
I_Niblack(index_Ni_0)=0;
temp=I_Niblack(4:end-4,4:end-4);
figure, imshow(I_Niblack);
str_tit =sprintf('Niblack Thresholding , Window Length =%d ,Parametere K =%.1f',win_length,k);
title(str_tit);


%%%%%% Savoula
I_Savoula = I;
index_Sv_255 = find(I_Savoula >= T_Savoula_Int);
index_Sv_0 =   find(I_Savoula < T_Savoula_Int);

I_Savoula(index_Sv_255)=255;
I_Savoula(index_Sv_0)=0;
figure, imshow(I_Savoula);
str_tit = sprintf('Savoula Thresholding , Window Length =%d ,Parametere K =%.1f , R
=%d',win_length,k,R); title(str_tit);
```

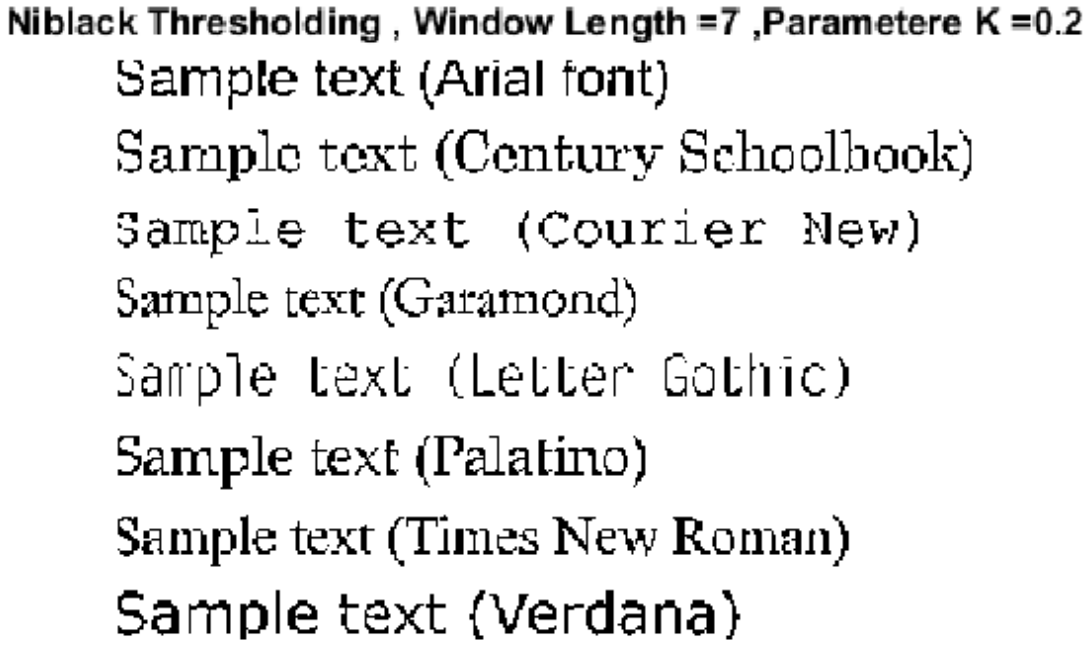
ساوولا در معیار خود حد آستانه را به صورت $T = m\left(1 + k\left(\frac{\sigma}{R} - 1\right)\right)$ در نظر گرفت تا به نوعی الگوریتم نایبلک را بهبود بخشد. یعنی پراکندگی کل تصویر که همان R می باشد را نیز به معیار خود اضافه نمود.

کد مربوط به پیاده سازی ساوولا نیز در بالا امده است که ما در این معیار نیز k را ۰,۲ و طول پنجره را ۷ گرفته ایم و R را ۷۴ می گیریم.

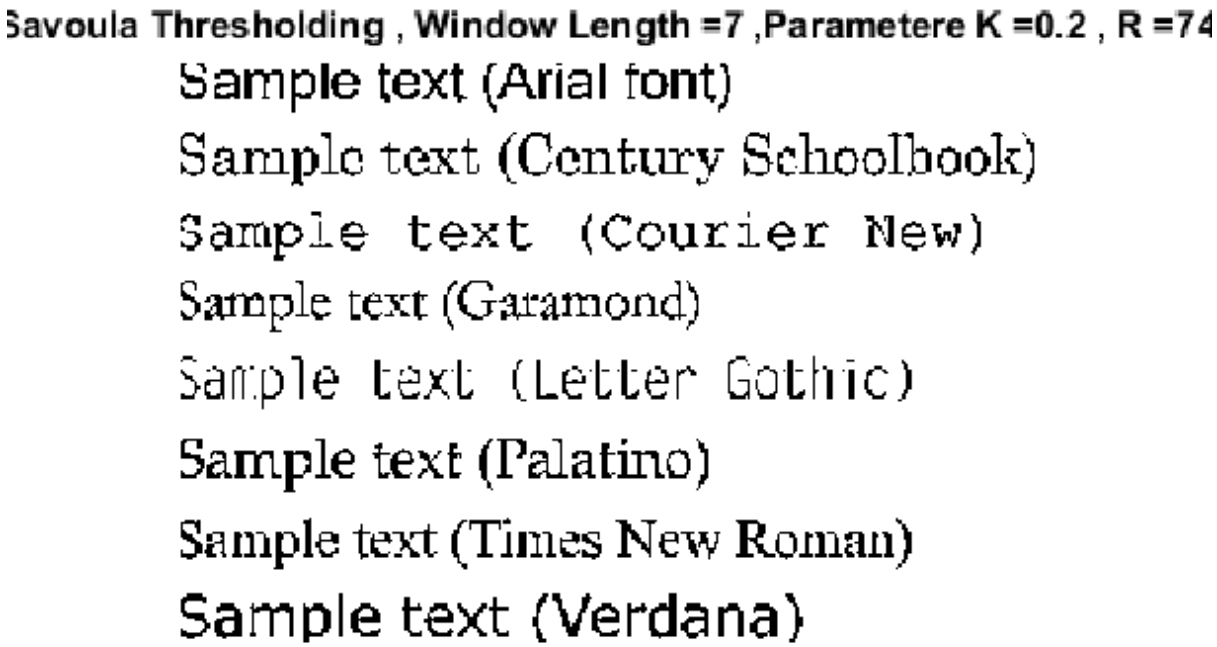
تصویر اصلی به صورت زیر می باشد:



نتیجه ی مشاهده شده از باینریزیشن تصویر توسط الگوریتم ارایه شده توسط Niblack به صورت زیر خواهد بود:



و نتیجه ی حاصل از باینریزیشن تصویر توسط الگوریتم ارایه شده توسطsauovola به صورت زیر خواهد بود:

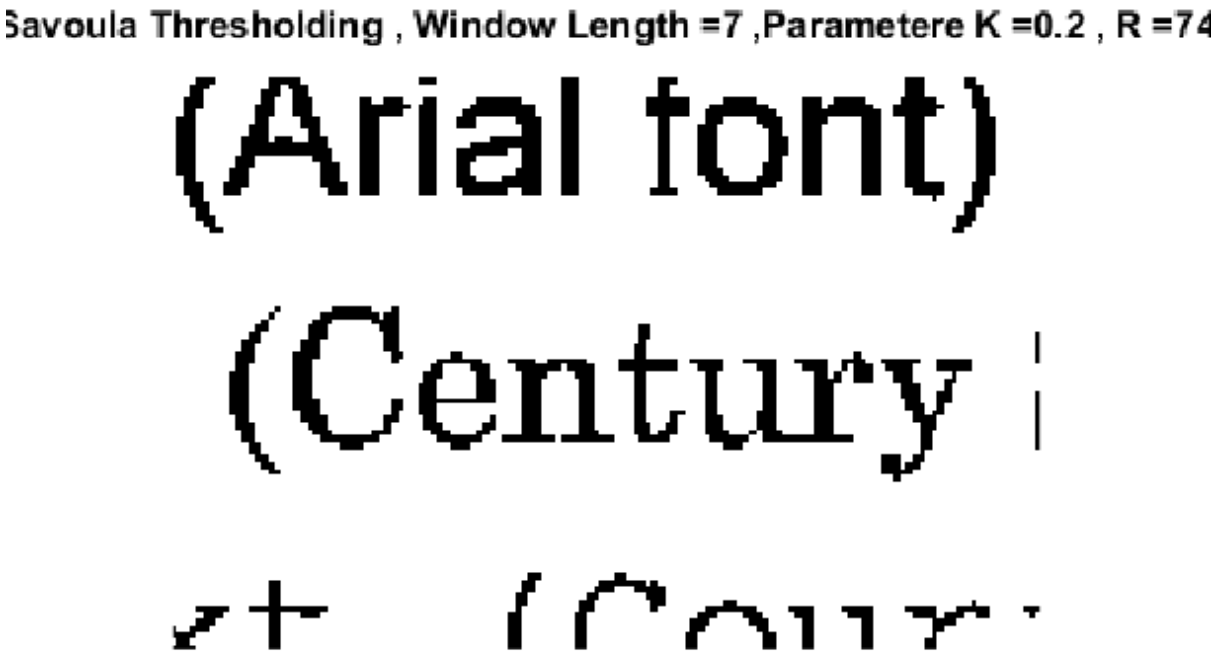


برای مقایسه ی بهتر نتایج حاصل و اینکه کدام بهتر عمل نموده است نتایج را زوم می کنیم تا تفاوت ها بهتر مشخص شوند:

تصویر اصلی به صورت زیر می باشد:

نتیجه ی مشاهده شده توسط الگوریتم ارایه شده توسط Niblack به صورت زیر خواهد بود:

و نتیجه ی مشاهده شده توسط الگوریتم ارایه شده توسط Sauvola به صورت زیر خواهد بود:



با مقایسه ی نتیجه ی حاصل از این دو معیار، مشاهده می کنیم که تا حدودی معیار ساوولا توانسته بهتر پیوستگی شکل را حفظ نماید. می توان علت این امر را بدین شکل توجیه نمود:

همان طور که می دانیم در معیار نایبلک، حد آستانه به صورت زیر تعریف می شوند:

نایبلک $T = m - k\sigma$

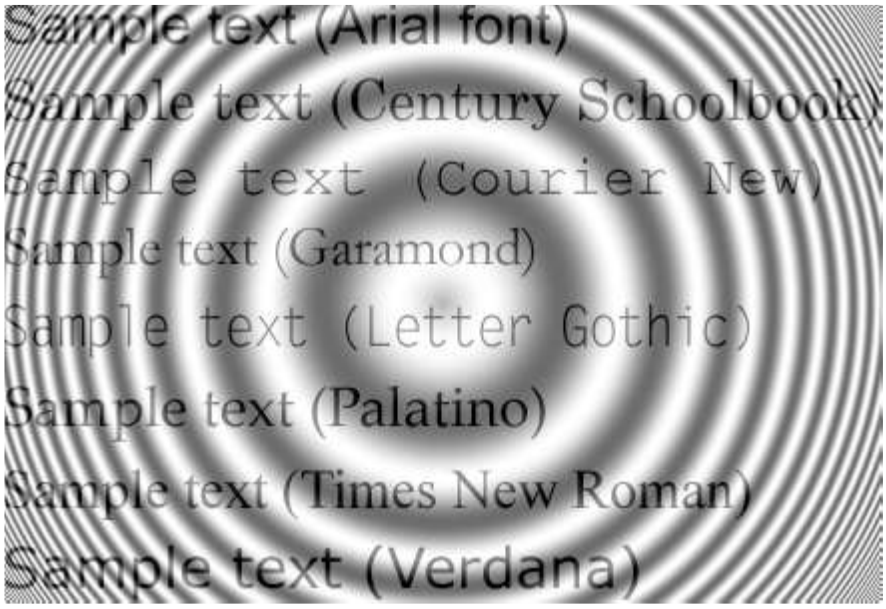
برای پیکسل های یکسان، میانگین و ثابت k یکسان بوده لذا وقتی در معیار ساوولا مقدار $\boldsymbol{\sigma}$ به R تقسیم می شود، مقدار T به نسبت معیار نایبلک افزایش می یابد چرا که $\frac{\sigma}{R}$ عددی کوچک تر از ۱ می شود. بنابراین می توان نتیجه گرفت که در معیار ساوولا، حد آستانه به نسبت معیار نایبلک، مقداری بیشتر خواهدشد و بنابراین، به علت این که تعدا پیکسل های صفر شده در این معیار بیشتر می شود، الگوریتم ساوولا تصویر را سیاه تر نشان می دهد و برعکس تعداد پیکسل هایی که ۱ می شوند در معیار نایبلک بیشتر می شوند و زمانی که در نایبلک، یک سری پیکسل ها سفید نشان داده می شوند، منجر می شود که از دید ما مقداری ناپیوستگی به نسبت ساوولا مشاهده شود.

سوال ۱ ج) (کد مربوط به این قسمت در فایل HW3_Q1_c.m قرار دارد.)

در این سوال از ما خواسته شده است که به تصویر داده شده در قسمت قبل یک نویز chirp را اضافه کنیم و سپس توسط دو روش عنوان شده و با استفاده از مقادیر داده شده برای پارامترها عملیات دینویزینگ را انجام دهیم

در زیر تصویر به همراه نویز را شاهد هستیم:

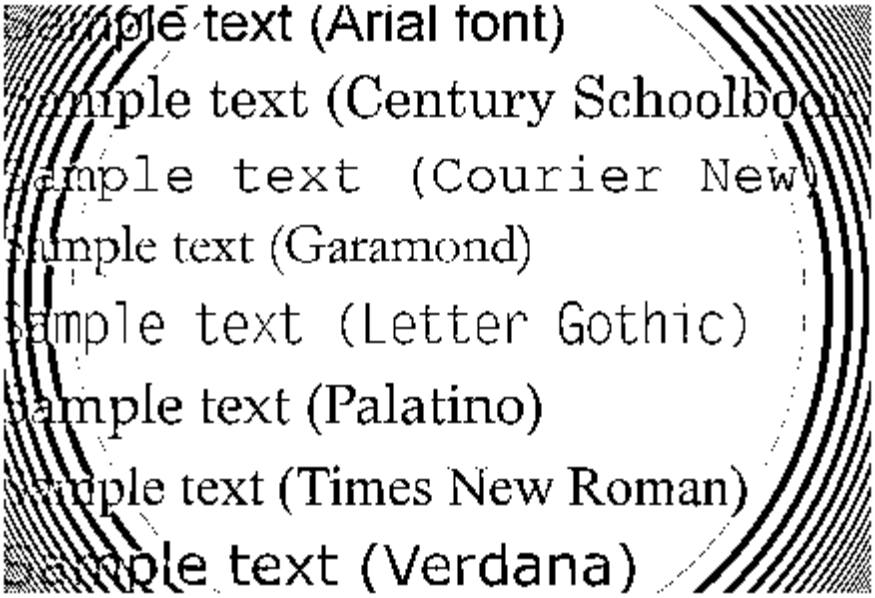
f + noise (chirp)



با استفاده از مقادیر پارامتر های عنوان شده در قسمت قبل، و استفاده از روش های Sauvola و Niblack نتایج زیر بدست می‌آید:

Niblack Thresholding , Window Length =7 ,Parametere K =0.20

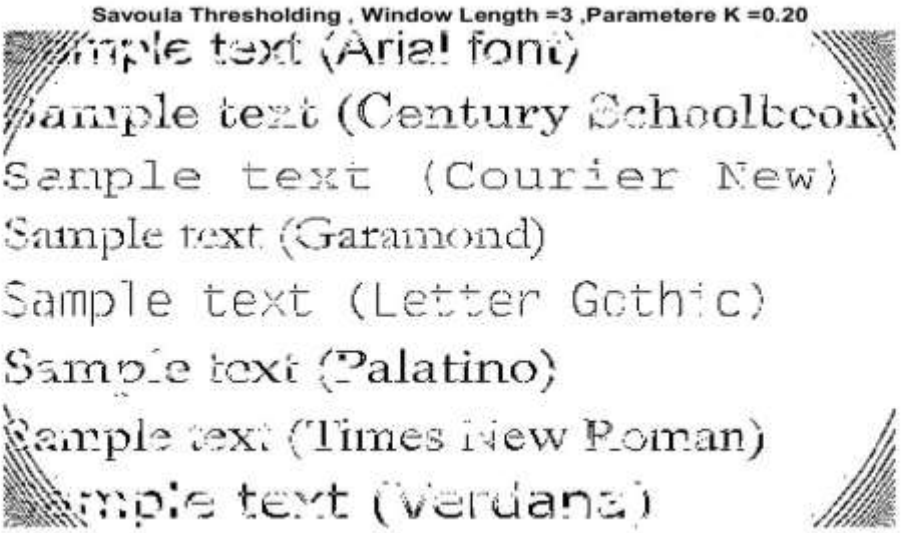
Savoula Thresholding , Window Length =7 ,Parametere K =0.20 , R=74



در سوال پرسیده شده است چرا Sauvola در مکان هایی که متن تیره می‌شود عملکرد بهتری نسبت به Niblack دارد، در مکان‌هایی مانند وسط تصویر که سینگال chirp ما کمترین فرکانس را دارد را دارد در این مکان ها نویز به طور کامل در پنجره ۷*۷ ما قرار می‌گیرد و در Niblack طبق رابطه $T = mu - k\sigma$ و با توجه به کم بودن مقدار k می توان نوشت $T \simeq mu$ و در نتیجه نویز می‌تواند در مکان هایی که اندکی از mu عبور کند باعث سیاه شدن تصویر شود و ما در اثر این شاهد حلقه های سیاه در میانه تصویر هستیم.

اما در Sauvola به علت تقسیم شدن σ به R نویز یه شدت تضعیف می‌شود وباعث کاهش T می‌شود که این امر باعث می‌شود نویز را به عنوان متن تشخیص ندهد و در وسط تصویر شاهد عملکرد بهتر آن هستیم.

اما در نزدیک قطر با توجه به اینکه سیگنال چرپ ما دارای فرکانس بالایی می‌باشد و تغییرات سریع تری دارد ممکن است در پنجره ۷*۷ ما نویز به طور کامل قرار نگیرد و درواقع به مانند یک نوشته در نظر گرفته می‌شود، در نتیجه دیگر به عنوان نویز شناخته نمی‌شود و طبیعی است که به رنگ سیاه نسبت داده شود.
برای مثال انتظار داریم که اگر طول پنجره را کاهش دهیم به طوری که کل نویز در پنجره قرار گیرد شاهد اشکار سازی نویز نباشیم که در زیر ما از یک پنجره ۳*۳ استفاده کردیم و مشاهده کردیم تا حد زیادی در نواحی نزدیک به قطر شاهد بهبود تصویر بودیم:

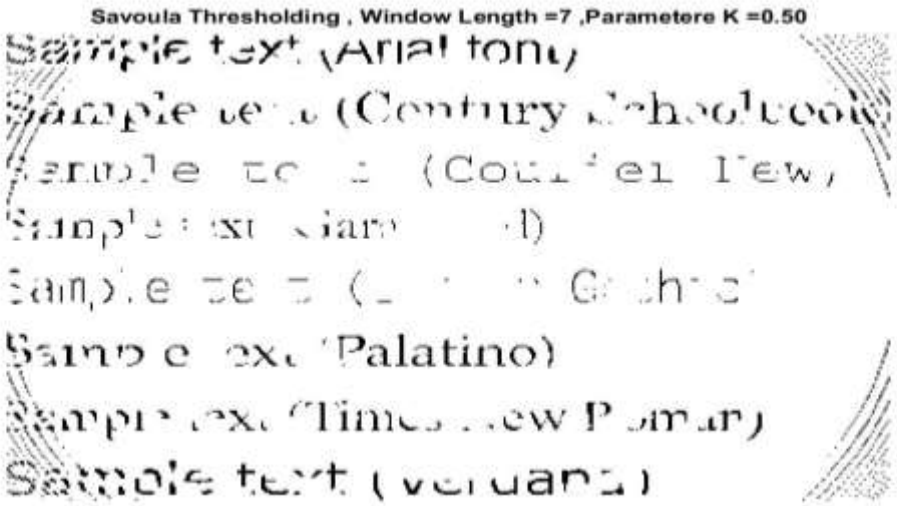


در زیر اثر تغییر پارامترهای k و طول پنجره را برای معیار Niblack و همچنین برای روش Sauvoula اثر تغییر پارمتر R را مورد بررسی قرار می دهیم. و انتخاب بهینه هر کدام از پارامتر ها برای رسیدن به جواب بهتر را ارزیابی می‌کنیم.

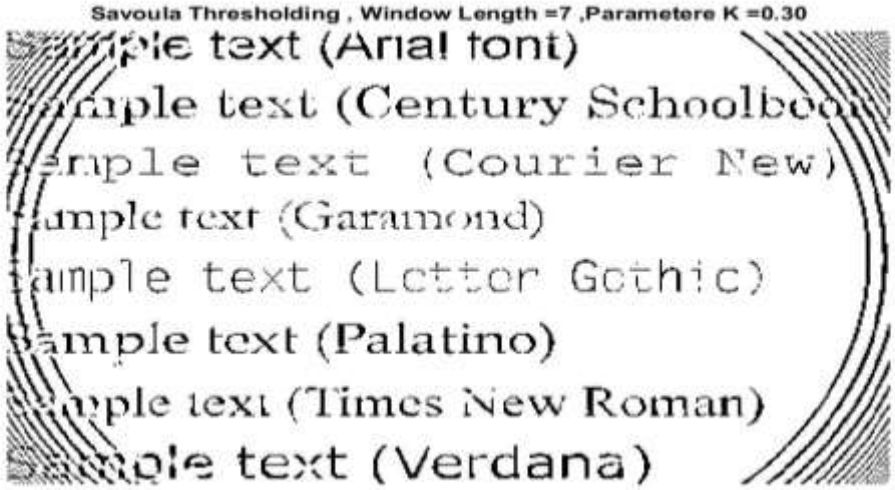
روش Sauvola

بررسی پارامتر k :

با توجه به رابطه موجود در روش Niblack و Sauvola ، با افزایش k مقدار T بیشتر کاهش می یابد و از مقدار میانگین دور تر می‌شود و اثر وارینس را بیشتر شاهد هستیم طبیعتا با کاهش سطح آستانه شاهد، باعث می‌شود پیکسل های بیشتری به ۲۵۵ داده شوند که این امر باعث می‌شود تصویر حاصله در k های بزرگ روشن تر شود. در زیر برای k=.2 , k=.5 نتایج را شاهد هستیم.

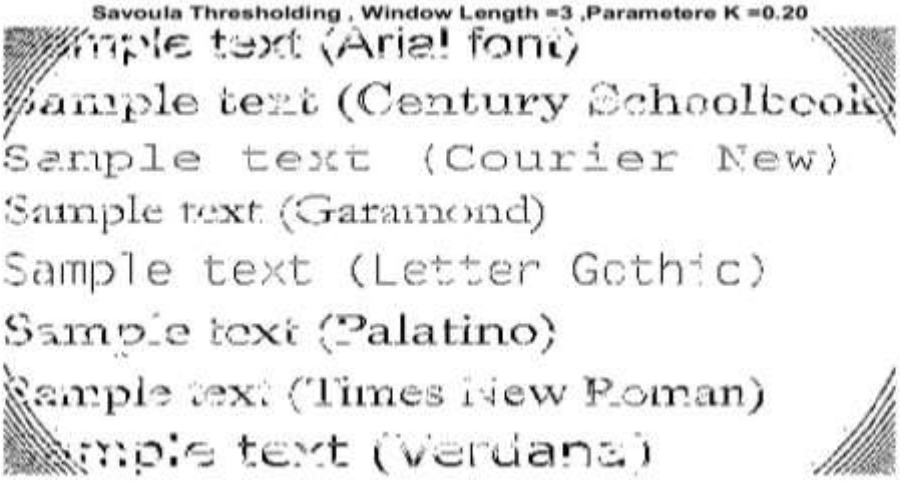


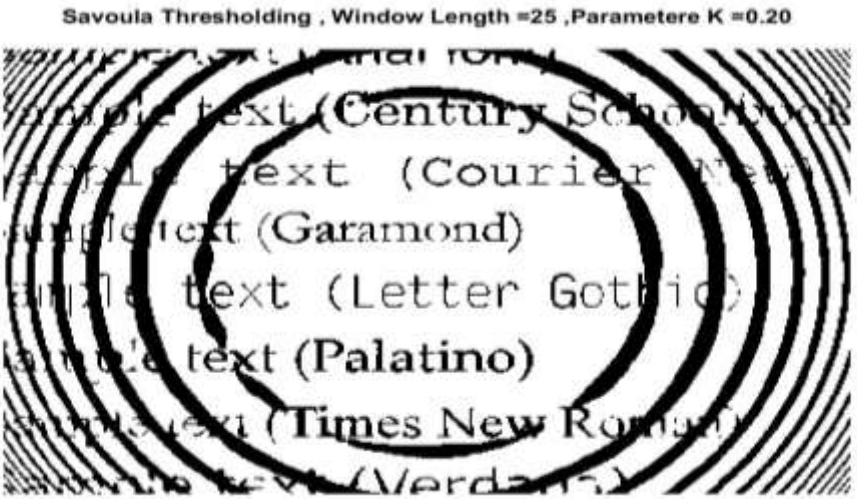
با مشاهده نتایج برای سایر k ها مشاهده شد بهترین k که هم باعث می‌شود نوشته های سیاه رنگ در مرکز به خوبی دیده شوند و هم اینکه دایره ها تا حد مناسبی کم رنگ شوند، ۰,۳ می‌باشد.



بررسی طول پنجره :

در پیکسل های نزدیک به قطر که تغییر رنگ زیادی را شاهد هستیم، اگر طول پنجره کوچک باشد می‌تواند به خوبی این اثر این تغییرات را (با محاسبه میانگین و واریانس پنجره و دخالت دادن آن در محاسبه استانه) در انتخاب آستانه دخالت دهد و به افزایش طول پنجره میزان حساسیت به تغییرات منطقه ای تصویر را کاهش می‌دهد در زیر برای دوحالت پنجره با طول ۳ و طول ۲۵ نتایج را شاهد هستیم:





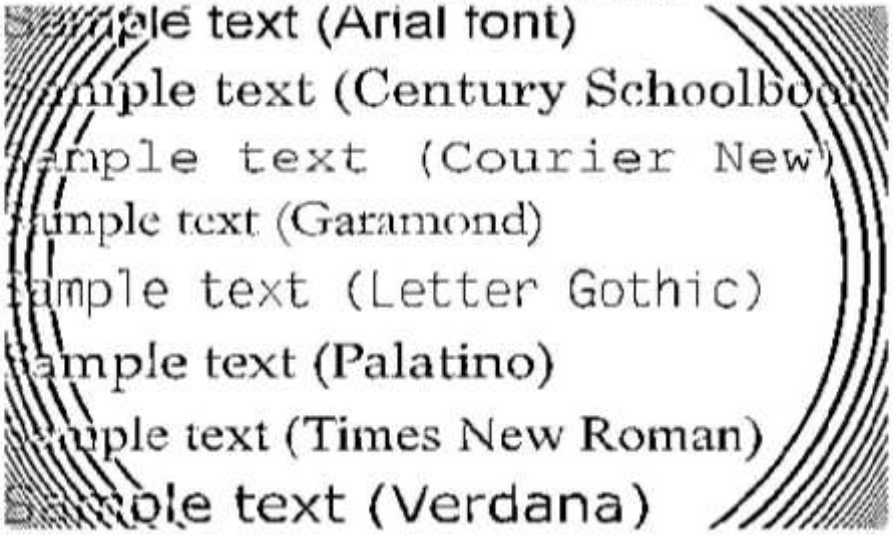
با مشاهده نتایج مختلف می‌توان گفت که برای طول پنجره ۵ و یا ۷ نتیجه از بقیه حالات بهتر است در زیر نتیجه مربوط به طول پنجره ۵ را شاهد هستیم:

بررسی پارامتر R :

R وظیفه نرمالیزه کردن واریانس را برعهده دارد، و مشاهده می‌شود با افزایش مقدار R تا مقدار حداکثری ۱۲۸ به علت کاهش آستانه مقدار تصویر بدست آمده روشن تر می‌شود. در زیر برای دوحالت حداقل و حداکثری ۲۰ و ۱۳۰ نتایج را شاهده هستیم:

Savoula Thresholding , Window Length =7 ,Parametere K =0.20 , R=20

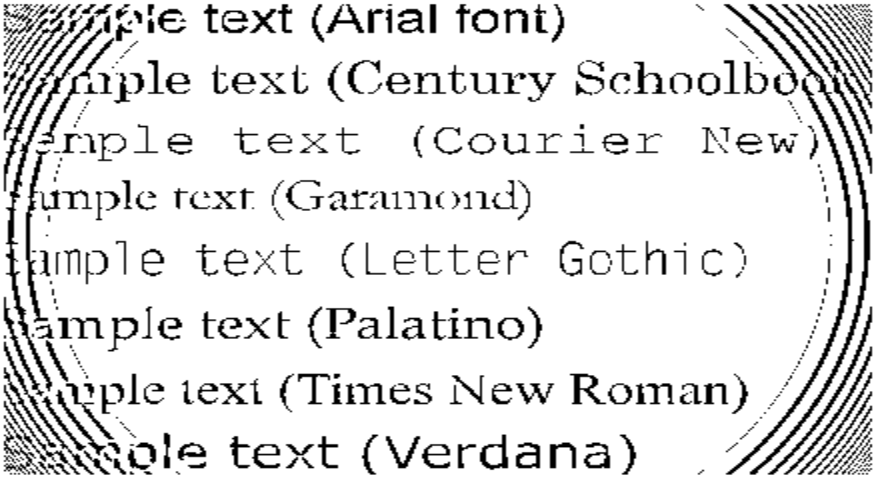
Savoula Thresholding , Window Length =7 ,Parametere K =0.20 , R=130



با بررسی نتایج مربوط به سایر R ها مشاهده شد بهترین نتیجه مربوط به R=128 است.

با در نظر گرفتن مطالب گفته شده برای هر پارامتر به صورت مجزا و ترکیب کردن این پارمتر ها، برای حصول بهترین نتیجه می توان عنوان کرد نتیجه مطلوب Savoula برای .R=128۲۵k= , و طول پنجره ۷ را بدست می‌آید که در زیر نتیجه آن را شاهد هستیم:

Savoula Thresholding , Window Length =7 ,Parametere K =0.25 , R=128



با مقایسه تصویر فوق که از پارمترهای پیشنهادی استفاده کرده با تصویر اولیه شاهد بهبود کاهش اثر نویز در گوشه ها هستیم هر چند که میتوانستیم همچنان در گوشه ها اثر نویز را کاهش دهیم اما این کاهش نویز باعث کم رنگ شدن نوشته ی اصلی می‌شد و ما باید یک مصالحه بین کم رنگ شدن نوشته اصلی و از بین رفتن اثر نویز برقرار کنیم که حاصل آن تصویر فوق است. به عنوان پیشنهاد می‌توان در گوشه های تصویر از پارامترهای متفاوت استفاده کرد و در مرکز تصویر نیز از پارمتر های دیگر و پس از باینری کردن گوشه ها و مرکز تصویر آنها را به هم اتصال داد.

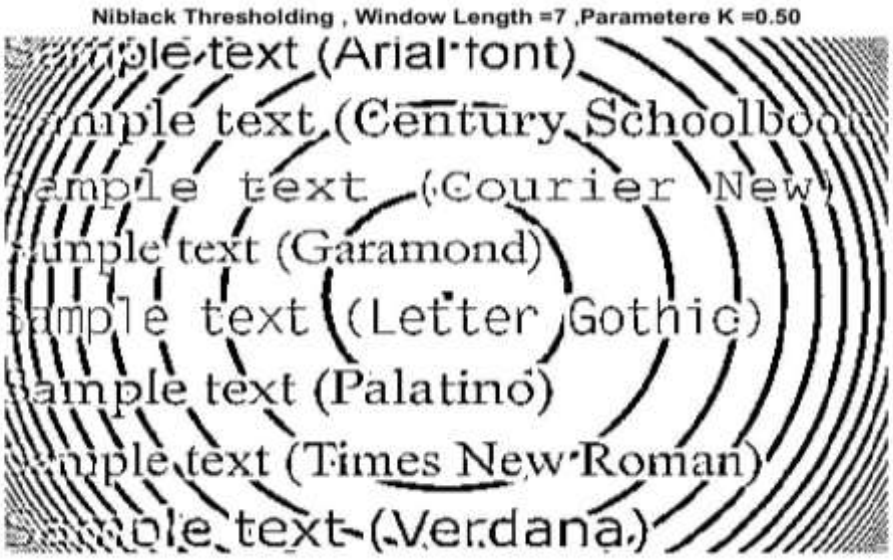
به طور کلی شاهد تاثیر مهم انتخاب پارامتر ها بر نویززدایی هستیم.

روش Niblack :

پارامتر k :

با استدلالی مشابه مطالب مربوط به Sauvola می‌توان مشاهده کرد بهترین جواب براب k=0.5 بدست می‌آید.

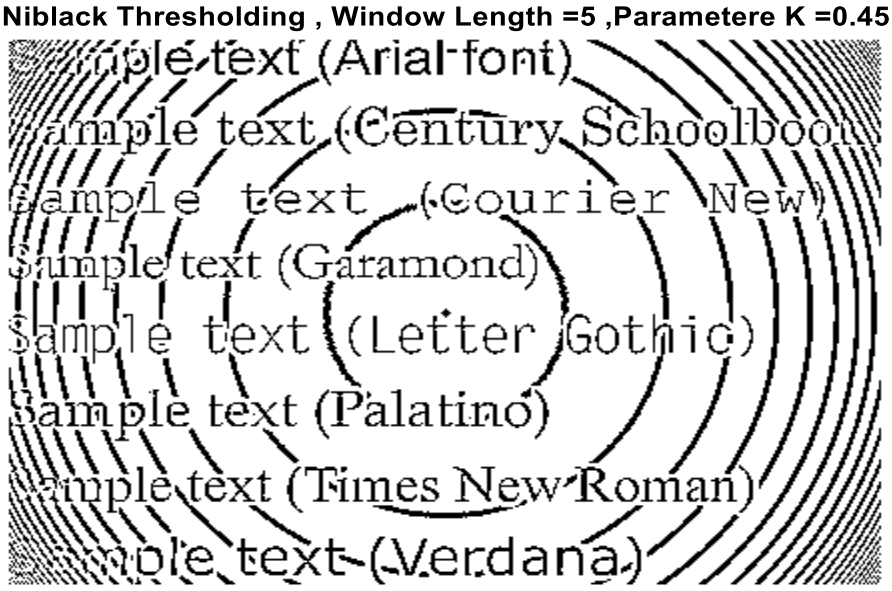
Niblack Thresholding , Window Length =7 ,Parametere K =0.20



بررسی طول پنجره :

در اینجا مشاهده شد بهترین جواب در حالتی رخ می‌دهد که طول پنجره را ۳ یا ۵ در نظر بگیریم. در زیر جواب های مربوط به طول پنجره ۳ و همچنین طول پنجره ۲۵ را شاهد هستیم:

با توجه به دو پارمتر بررسی شده برای روش Niblack می توان گفت برای طول پنجره ۵ و k=.45 جواب بدست آمده از بقیه حالات بهتر می باشد.



با مقایسه جواب بدست آمده با استفاده از پارامترهای پیشنهادی با جواب بدست آمده اولیه به خوبی شاهد بهبود کیفیت تصویر و کاهش اثر نویز هستیم هر چند که این کار باعث شده است تا مقداری از پیکسل های نوشته اصلی نیز به سفید تبدیل شده(تصویر کم رنگ تر شود) و شاهد تاثیرگذاری انتخاب پارامتر بر نتایج بدست آمده هستیم.

در زیر کد مربوط به این قسمت را شاهد هستیم:

```
clc
close all
clear

f=imread('degarded_text.png');
f = 255 * im2double(f);
figure, imshow(f),title('Original Image');
[M,N] = size(f);
```

```
r_max = sqrt(220^2 + 150^2);  r_min = 0;
T_max = 100;                  T_min = 10;

p=0;
for m=-150:149
    p=p+1; q=0;
    for n=-220:219
        q=q+1;
        r= sqrt(m^2 + n^2);
        T = ((T_min -T_max)/(r_max - r_min))*r + T_max;
        chirp(p,q) = sin(2*pi*r/T);
    end
end
chirp=fix(128*(chirp+1));  %%% Normlized noise between 0 - 255

g= .5*(f+chirp);
figure,imshow(g,[]),title('f + noise (chirp)')


%%%%%%%%%%%%%%%%%%%%%%%%%%%     NiBlack & Savoula Approach   %%%%%%%%%%%%%%%
I = g;

k=.45;              %%% k for Niblack adn Savoula Method
R = 128;            %%% R for Savoula Method
win_length = 5;   %%% window length

[M,N] = size(I);

I_temp = I;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Integeral Image  %%%%%%%%%%%%%%%%%%%%%%%%%%
% for R=20 : 10 : 250
step_win = fix(win_length/2);

I_Integral = zeros(M+1,N+1);
I_Integral(2:M+1,2:N+1) = I;

I_Integral_2 = zeros(M+1,N+1);
I_Integral_2(2:M+1,2:N+1) = I.^2;

Int_Mat = zeros(M+1,N+1);          %%% Integral matrix
Int_Mat_2 = zeros(M+1,N+1);        %%% Integral matrix power 2

for m=2:M+1
    for n=2:N+1
        Int_Mat(m,n) = Int_Mat(m-1,n) + Int_Mat(m,n-1) - Int_Mat(m-1,n-1) +  I_Integral(m,n);
        Int_Mat_2(m,n) = Int_Mat_2(m-1,n) + Int_Mat_2(m,n-1) - Int_Mat_2(m-1,n-1) +
I_Integral_2(m,n);
    end
end

%%% calculate mean and variance
p=0; T_Niblack =[]; T_Savoula = [];
for m=1 + step_win : M - step_win
    p=p+1; q=0;
    for n=1 + step_win :N - step_win
        q=q+1;
        m_1 = m+ step_win+1;  n_1 = n + step_win+1;    coef_1 = m_1*n_1;
        m_2 = m_1 - win_length;   n_2 = n_1;                   coef_2 = m_2*n_2;
        m_3 = m_1;                n_3 = n_1 - win_length;      coef_3 = m_3*n_3;
        m_4 = m_1-win_length;     n_4 = n_1-win_length;        coef_4 = m_4*n_4;
        mu = 1/win_length^2 * (Int_Mat(m_1,n_1) - Int_Mat(m_2,n_2) - Int_Mat(m_1,n_3) +
Int_Mat(m_4,n_4));

        var = sqrt((1/win_length^2 * (Int_Mat_2(m_1,n_1) - Int_Mat_2(m_2,n_2) -
Int_Mat_2(m_1,n_3) +  Int_Mat_2(m_4,n_4)))...
            - mu.^2);

        T_Niblack(p,q) = mu - k*var;            %%% Threshold Niblack
        T_Savoula(p,q) = mu*(1+k*(var/R - 1));  %%% THreshold Savoula
    end
end

T_temp = zeros(M,N);
T_temp(1 + step_win : M - step_win ,1 + step_win :N - step_win ) = T_Niblack;
```

```
T_Niblack = T_temp;

T_temp = zeros(M,N);
T_temp(1 + step_win : M - step_win,1 + step_win :N - step_win) = T_Savoula;
T_Savoula = T_temp;


%% Niblack
I_Niblack=I;
index_Ni_255 = find(I_Niblack >= T_Niblack);
index_Ni_0 =   find(I_Niblack < T_Niblack);

I_Niblack(index_Ni_255)=255;
I_Niblack(index_Ni_0)=0;
temp=I_Niblack(4:end-4,4:end-4);
figure, imshow(I_Niblack);
str_tit =sprintf('Niblack Thresholding , Window Length =%d ,Parametere K =%.2f',win_length,k);
title(str_tit);

%%%%%%% Savoula
I_Savoula = I;
index_Sv_255 = find(I_Savoula >= T_Savoula);
index_Sv_0 =   find(I_Savoula < T_Savoula);

I_Savoula(index_Sv_255)=255;
I_Savoula(index_Sv_0)=0;
figure, imshow(I_Savoula);
str_tit = sprintf('Savoula Thresholding , Window Length =%d ,Parametere K =%.2f ,
R=%d',win_length,k,R); title(str_tit);
```

سوال دو-الف) (کد این قسمت در HW3_Q2_A.m موجود می باشد)

در این سوال ابتدا یک تصویر ۳۰۰*۳۰۰ که روشنایی پیکسل آن در همه جا ۰,۴۸ است را تولید می کنیم که تقریبا یک تصویری که همه جا خاکستری است را به ما می دهد. در ادامه قرار است این تصویر را به دو روش error diffusion و dot diffusion، halftone نماییم و نتایج را با هم مقایسه نماییم:

روش اول: error diffusion :

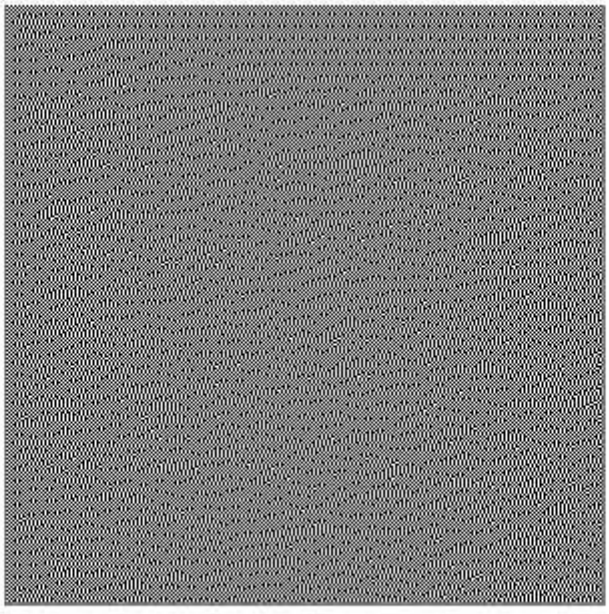
برای مشاهده ی نتیجه ی halftoning به این روش از کد زیر استفاده می نماییم:

```
%%%%%%%%% original image

f = 0.48*ones(300,300);
figure
imshow(f,[])
[M,N] = size(f);
thr = .5;

%%%%%%%%    error diffusion

h_error = [0 0 7/16;3/16 5/16 1/16];

ff = f;
for i=1 : M -2
    for j=2: N -2

        temp_1 = ff(i:i+1 , j-1:j+1);
        temp_2 = ff(i,j);

        if temp_2<= thr
            ff(i,j) = 0;
        else
            ff(i,j) = 1;
        end
        e = ff(i,j) - temp_2;
        ff(i:i+1 , j-1:j+1) = ff(i:i+1 , j-1:j+1) - e.*h_error;
    end
end

figure, imshow(ff) , title('Error Difusion')
```

که نتیجه به صورت زیر مشاهده می شود:

original image

Error Difusion



روش دوم: dot diffusion :

برای استفاده از این روش یک بار از ماتریس C استفاده شده در روش Knuth استفاده نموده ایم و یک بار از ماتریس C برای New method استفاده نموده ایم. کد برنامه به صورت زیر است:

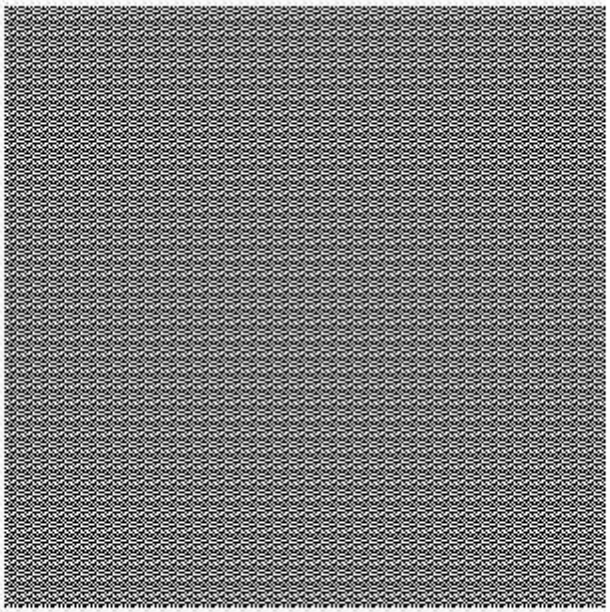
```
%%%%%%%%%%  Dot Fussion new Method

C_Mat = load('C_Matrix.txt');

fff= zeros(8*ceil(M/8),8*ceil(N/8));
fff(1:M,1:N) = f ;
[M1 , N1] = size(fff);
temp_1 = zeros(10,10);
C_Mat_1 = zeros(10,10);
C_Mat_1(2:9,2:9) = C_Mat;
h_dot = zeros(3,3);
p=0;
for i=1:8: M1
    for j=1:8: N1
        p=p+1;
        temp_1(2:9,2:9)=fff(i:i+7,j:j+7);
        for k=1:64
            [row , col] = find(C_Mat_1 ==k);

            if temp_1(row,col) > thr
                e = 1 - temp_1(row,col);
                temp_1(row,col) = 1;
            else
                e = 0 - temp_1(row,col);
                temp_1(row,col) = 0;
            end

            temp_2 = temp_1(row-1:row+1 , col-1: col+1);
            temp_3 = C_Mat_1(row-1:row+1 , col-1: col+1);

            index_1 = find(temp_3 > k);           %%% indeices that greater than k
            if isempty(index_1) == 0
                index_1_e = index_1(mod(index_1,2)==0);
                index_1_o = index_1(mod(index_1,2)==1);
                h_dot(index_1_e) = 2;
                h_dot(index_1_o) = 1;

                index_2 = find(temp_3 <= k);   %%% indeices that greater than k
                h_dot(index_2) = 0;
                h_dot = (1/sum(h_dot(:))).*h_dot;
                temp_4 = temp_2 - e.*h_dot;
                temp_1(row-1:row+1 , col-1: col+1) = temp_4;
            end

        end
        fff(i:i+7,j:j+7) = temp_1(2:9,2:9);


    end
end
fff = fff(1:M,1:N);
figure, imshow(fff),title('dot difussion New Method')

%%%%%%%%%%  Dot Fussion Knuth's Method

Knuth_Mat = load('C_Matrix.txt');
```

```
fff= zeros(8*ceil(M/8),8*ceil(N/8));
fff(1:M,1:N) = f ;
[M1 , N1] = size(fff);
temp_1 = zeros(10,10);
Knuth_Mat_1 = zeros(10,10);
Knuth_Mat_1(2:9,2:9) = Knuth_Mat;
h_dot = zeros(3,3);
p=0;
for i=1:8: M1
    for j=1:8: N1
        p=p+1;
        temp_1(2:9,2:9)=fff(i:i+7,j:j+7);
        for k=1:64
            [row , col] = find(Knuth_Mat_1 ==k);

            if temp_1(row,col) > thr
                e = 1 - temp_1(row,col);
                temp_1(row,col) = 1;
            else
                e = 0 - temp_1(row,col);
                temp_1(row,col) = 0;
            end

            temp_2 = temp_1(row-1:row+1 , col-1: col+1);
            temp_3 = Knuth_Mat_1(row-1:row+1 , col-1: col+1);

            index_1 = find(temp_3 > k);           %%% indeices that greater than k
            if isempty(index_1) == 0
                index_1_e = index_1(mod(index_1,2)==0);
                index_1_o = index_1(mod(index_1,2)==1);
                h_dot(index_1_e) = 2;
                h_dot(index_1_o) = 1;

                index_2 = find(temp_3 <= k);    %%% indeices that greater than k
                h_dot(index_2) = 0;
                h_dot = (1/sum(h_dot(:))).*h_dot;
                temp_4 = temp_2 - e.*h_dot;
                temp_1(row-1:row+1 , col-1: col+1) = temp_4;
            end

        end
        fff(i:i+7,j:j+7) = temp_1(2:9,2:9);
    end
end
fff = fff(1:M,1:N);
figure, imshow(fff),title('dot difussion Knuth''s Method')
```

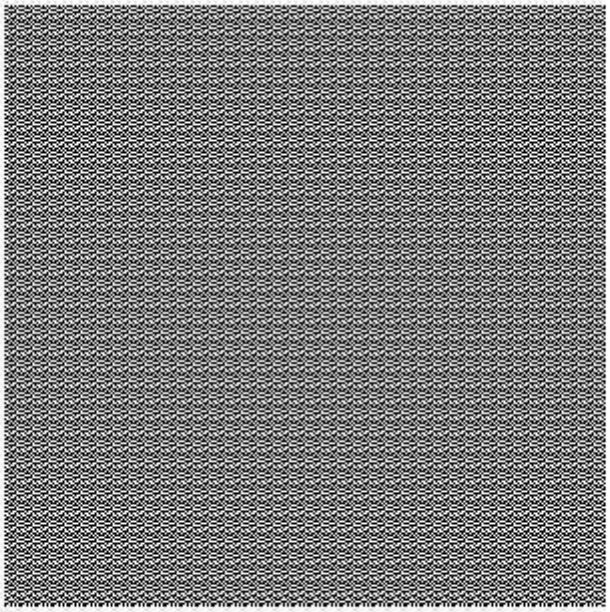
توجه شود که ماتریس های ضرایب برای New Method و Knuth در فایل ضمیمه به نام های C_Matrix.txt و Knuth_Mat.txt آورده شده است.

نتایج مشاهده شده به صورت زیر خواهد بود:

dot difussion New Method



dot difussion Knuth's Method



همانطور که مشاهده می شود، در روش dot diffusion به دلیل اینکه ماتریس ضرایبی که استفاده می شود ثابت بوده و تصویر اصلی ما نیز دارای پیکسل هایی با سطح روشنایی ثابت می باشند هربار که ما این ماتریس ضرایب ۸ در ۸ را روی تصویر قرار می دهیم،به صورت یکنواخت عمل نموده و میتوان گفت برای کل پیکسل ها این روش به صورت یک روش تکراری و ثابت عمل می نماید و لذا انتظار تولید worm نیز وجود ندارد

اما در روش error diffusion ، اگر ماتریس h را به مانند یک فیلتر در نظر بگیریم و با توجه به اینکه درایه اپدیت شده برابر است با $f_{new} = f_{old} - e*h$ و $e = 0or1 - f_{old}(i,j)$

0	0	7/16
3/16	5/16	1/16

اگر مقدار e مثبت باشد یعنی درایه جدید انتخابی برای عنصر i,j ۱ است و باتوجه به اینکه هنگام اپدیت کردن درایه های جدید تاثیر معکوس دارد مقدار ارور (به علت منفی استفاده شده در فرمول اپدیت) و همچنین با توجه به اینکه در ماتریس h بیشترین تاثیر ضرایب را در عنصر (i+1,j),(i,j+1) شاهد هستیم مقدار این دو درایه تا حد زیادی کاهش می‌یابد و به صفر نزدیک می‌شود در گام بعدی با توجه به اینکه (i,j+1) در مرحله قبل کاهش قابل توجه داشته باعث می‌شود احتمالا در این مرحله مقدار ۰ را بگیرد در نتیجه ارور در این گام مثبت می‌شود و در هنگام اپدیت کردن در این مرحله شاهد افزایش مقادیر هستیم که این بار در مرحله بعد باز بیشترین افزایش را در درایه (i,j+2) , (i+1,j+1) شاهدیم در نتیجه باز به احتمال زیاد در گام بعد به هنگام کوانتیزاسیون به علت افزایش در مرحله قبل احتمالا این بار به ۱ نسبت داده می‌شود و باز این دومرحله گفته شده تکرار می‌شود. البته به این نکته توجه شود که ممکن است گاهی این افزایش یا کاهش اعمال شده در اثر اپدیت کردن به حدی نباشد که بتواند حتما باعث تغییر علامت نسبت به سلول قبل شود(برای مثال ممکن است به قدری عدد به صفر نزدیک باشد که حتی در درایه (l,j+1) که میزان افزایش برابر است با e*h مقدار اپدیت شده از ۱ گذر نکند و دوباره ۰ تولید کند اما در چنین حالتی انتظار می‌رود بعد از چند تکرار دوباره روال تولید ۰و۱ از سر گرفته شود)

هنگامی هم که به سطر پایین می‌رویم باز با توجه به اینکه در اینجا هم میزان اپدیت نسیتا زیاد بوده(5/16) باز علامت مخالف با عنصر بالا سری بوده اما دقیقا مشابه استدلال گفته شده در فوق ممکن است در حالتی این تغییر علامت رخ ندهد اما معمولا بعد از چند سلول دوباره تغییر علامت حاصل می‌گردد.

درنتیجه انتظار می‌رود که به صورت قطری ۰و۱ تولید شود (البته با توجه به توضیحات این تولید قطری منظم نمی باشد و در برخی حالات منقطع می‌شود) که این تولید ۰و۱ قطری و گاها منقطع باعث تولید کرم می‌شود.

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

سوال دو-ب) (کد این قسمت در HW3_Q2_B.m موجود می باشد)

در این قسمت، تصویر Baby2.jpg را با دو روش دکر شده در بالا halftoning نموده و نتایج را مشاهده می نماییم تا مقایسه ای بین آنها انجام دهیم:

کد برنامه به صورت زیر است:

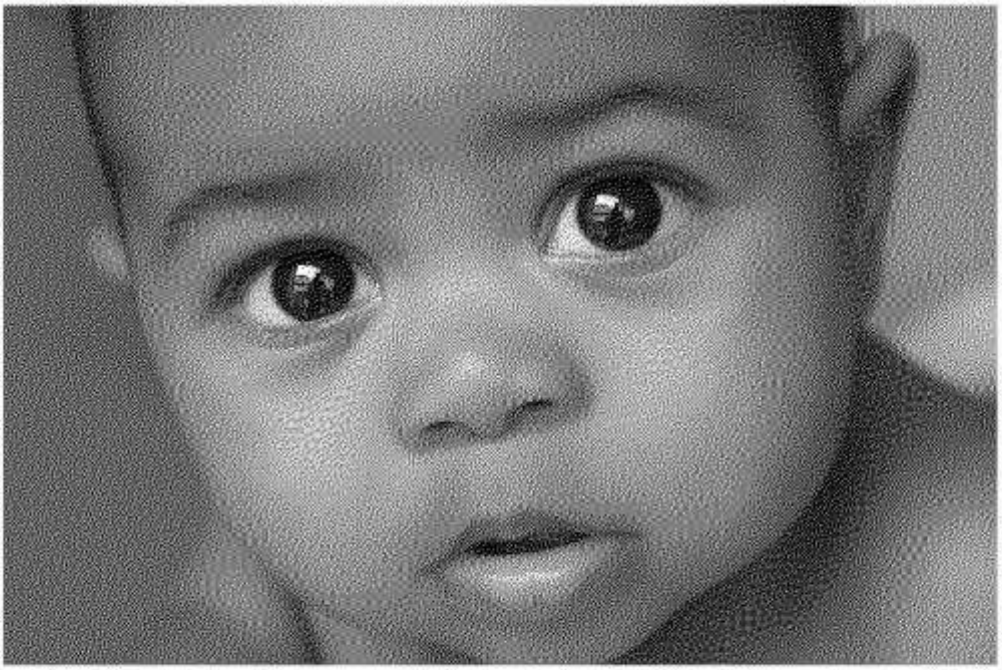
```
clc;
clear;
% close all;

%%%%%%%%%
f = imread('baby2.jpg');
f= rgb2gray(f);
f = im2double(f);
figure, imshow(f) , title('Oroginal Image');
[M,N] = size(f);
thr = .5;
%%%%%%%    error difusion
h_error = [0 0 7/16;3/16 5/16 1/16];

f_error = f;
for i=1 : M -2
    for j=2: N -2

        temp_1 = f_error(i:i+1 , j-1:j+1);
        temp_2 = f_error(i,j);

        if temp_2<= thr
            f_error(i,j) = 0;
        else
            f_error(i,j) = 1;
        end
        e = f_error(i,j) - temp_2;
        temp_3 = e.*h_error;
        f_error(i:i+1 , j-1:j+1) = f_error(i:i+1 , j-1:j+1) - temp_3;
    end
end

figure, imshow(f_error) , title('Error Difusion')
)
%%%%%%%%%  Dot Fussion New Method

C_Mat = load('C_Matrix.txt');

f_dot= zeros(8*ceil(M/8),8*ceil(N/8));
f_dot(1:M,1:N) = f ;
[M1 , N1] = size(f_dot);
temp_1 = zeros(10,10);
C_Mat_1 = zeros(10,10);
C_Mat_1(2:9,2:9) = C_Mat;
h_dot = zeros(3,3);
p=0;
for i=1:8: M1
    for j=1:8: N1
        p=p+1;
        temp_1(2:9,2:9)=f_dot(i:i+7,j:j+7);
        for k=1:64
            [row , col] = find(C_Mat_1 ==k);
```

```
            if temp_1(row,col) > thr
                e = 1 - temp_1(row,col);
                temp_1(row,col) = 1;
            else
                e = 0 - temp_1(row,col);
                temp_1(row,col) = 0;
            end

            temp_2 = temp_1(row-1:row+1 , col-1: col+1);
            temp_3 = C_Mat_1(row-1:row+1 , col-1: col+1);

            index_1 = find(temp_3 > k);           %%% indeices that greater than k
            if isempty(index_1) == 0
                index_1_e = index_1(mod(index_1,2)==0);
                index_1_o = index_1(mod(index_1,2)==1);
                h_dot(index_1_e) = 2;
                h_dot(index_1_o) = 1;

                index_2 = find(temp_3 <= k);   %%% indeices that greater than k
                h_dot(index_2) = 0;
                h_dot = (1/sum(h_dot(:))).*h_dot;
                temp_4 = temp_2 - e.*h_dot;
                temp_1(row-1:row+1 , col-1: col+1) = temp_4;
            end

        end
        f_dot(i:i+7,j:j+7) = temp_1(2:9,2:9);


    end
end
f_dot = f_dot(1:M,1:N);
figure, imshow(f_dot),title('dot difussion New Method')

%%%%%%%%%%  Dot Fussion Knuth Method

C_Mat = load('C_Matrix.txt');
Knuth_Mat = load('Knuth_Mat.txt');

f_dot= zeros(8*ceil(M/8),8*ceil(N/8));
f_dot(1:M,1:N) = f ;
[M1 , N1] = size(f_dot);
temp_1 = zeros(10,10);
Knuth_Mat_1 = zeros(10,10);
Knuth_Mat_1(2:9,2:9) = Knuth_Mat;
h_dot = zeros(3,3);
p=0;
for i=1:8: M1
    for j=1:8: N1
        p=p+1;
        temp_1(2:9,2:9)=f_dot(i:i+7,j:j+7);
        for k=1:64
            [row , col] = find(Knuth_Mat_1 ==k);

            if temp_1(row,col) > thr
                e = 1 - temp_1(row,col);
                temp_1(row,col) = 1;
            else
                e = 0 - temp_1(row,col);

            end

            temp_2 = temp_1(row-1:row+1 , col-1: col+1);
            temp_3 = Knuth_Mat_1(row-1:row+1 , col-1: col+1);

            index_1 = find(temp_3 > k);           %%% indeices that greater than k
            if isempty(index_1) == 0
                index_1_e = index_1(mod(index_1,2)==0);
                index_1_o = index_1(mod(index_1,2)==1);
                h_dot(index_1_e) = 2;
                h_dot(index_1_o) = 1
                index_2 = find(temp_3 <= k);   %%% indeices that greater than k
                h_dot(index_2) = 0;
                h_dot = (1/sum(h_dot(:))).*h_dot;
                temp_4 = temp_2 - e.*h_dot;
                temp_1(row-1:row+1 , col-1: col+1) = temp_4;
            end
```

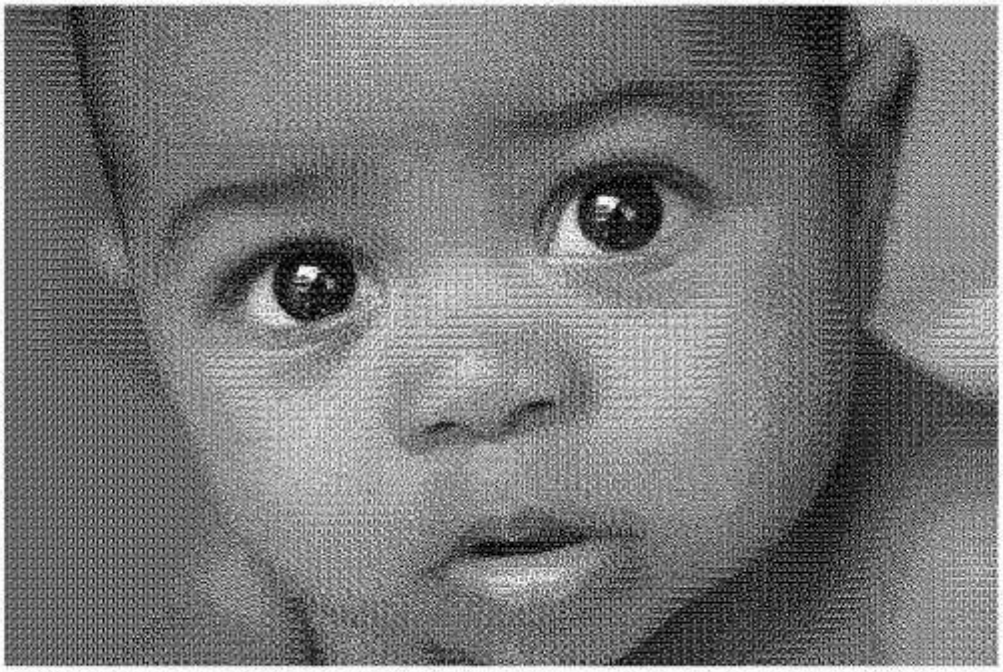
```
        end

        f_dot(i:i+7,j:j+7) = temp_1(2:9,2:9);

    end
end

f_dot = f_dot(1:M,1:N);
figure, imshow(f_dot),title('dot diffusion Knuth Method')
```

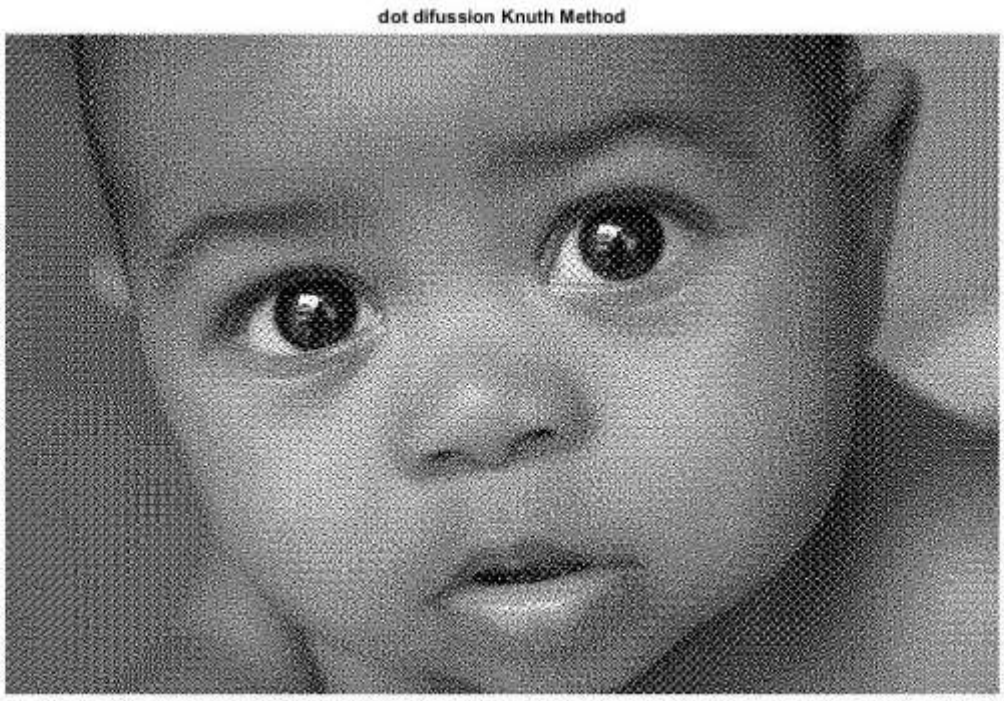
و نتایج، به صورت زیر خواهد بود:

Error Difusion



dot difussion new Method





با مشاهده ی نتایج حاصل، و به همان دلایلی که در قسمت اول سوال بیان شد، مشاهده ی کنیم که در روش error diffusion کرم تولید شده است که محل هایی که در آنها این اتفاق افتاده است را در تصویر زیر مشخص کرده ایم، الیته دلیل رخداد این پدیده را میتوان این گونه توصیف کرد این کرم ها در محل هایی ایجاد می‌شوند که در یک ناحیه پیکسل های ما دارای روشنایی نزدیک به هم و حدودا برابر ۱۲۸ (در تصویر اصلی) هستیم، بوجود می‌آیند.