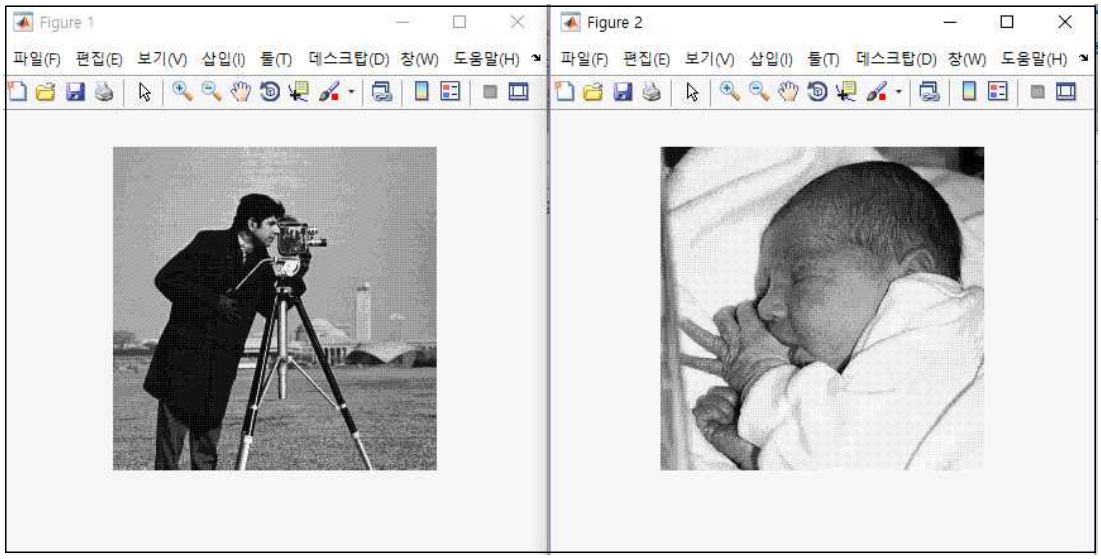
컴퓨터 비젼

[HW02]

학번	201203393
분반	00
이름	김헌겸
과제번호	02

제출일 : 2016년 3월 27일 일요일

1. (MATLAB) For ‘newborn.tif’ and ‘cameraman.tif’, implement the dithering with D, when 8-gray level is used.

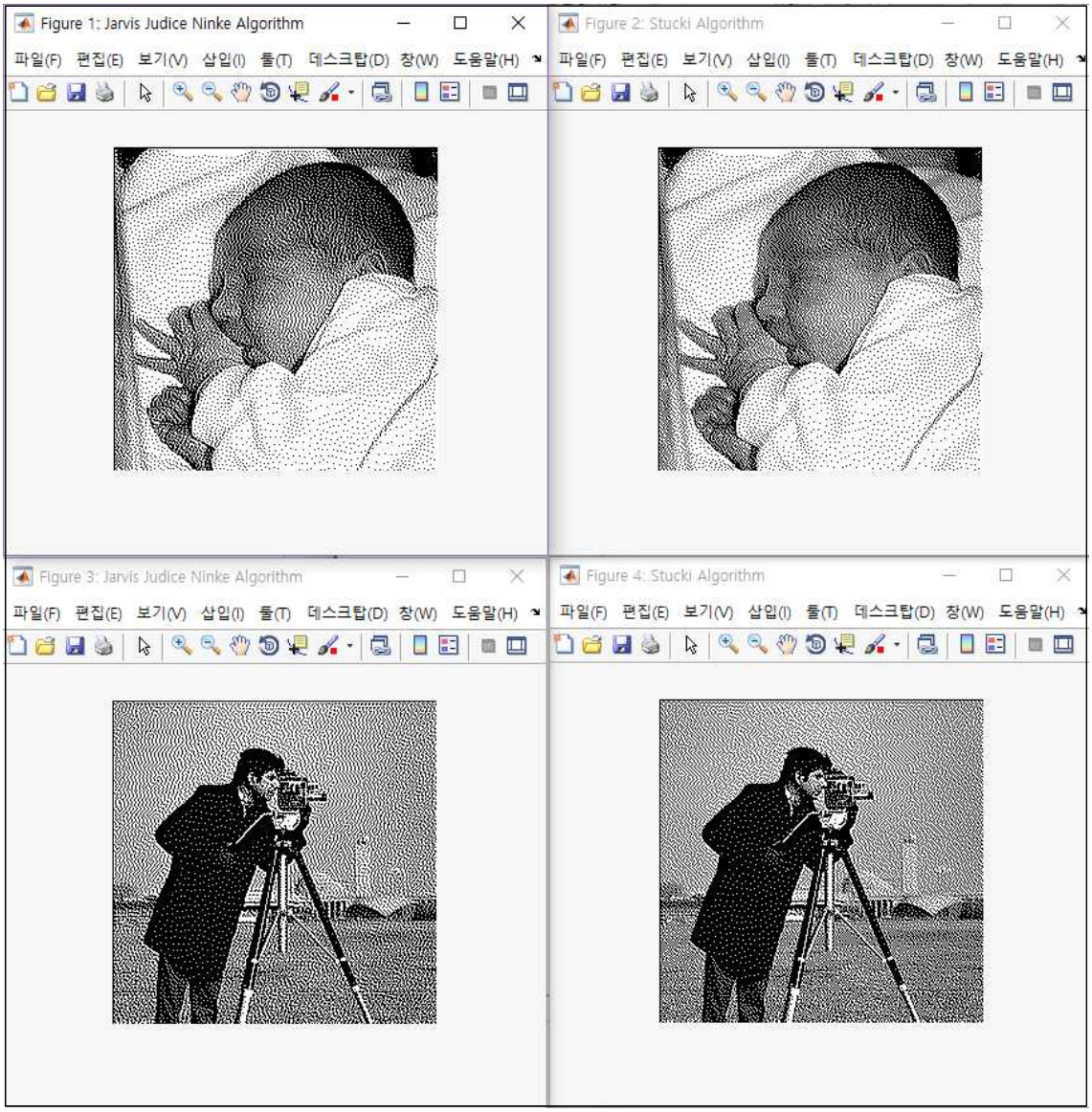



```
function D_8gray_dither(filename)
fid = fopen(filename);
if fid == -1
    error('잘못된 입력값!');
end;

x = imread(filename);

D = [0,24;36,12];
r = repmat(D, 128, 128);
x = double(x);
q = floor(x/37);
x4 = q +(x-37*q>r);
figure, imshow(uint8(x4*37));
```

8-gray level로 dithering하려면 255/7 = 37인 값을 기준으로 매트릭스를 만든다. D = $\begin{pmatrix} 0 & 24 \\ 36 & 12 \end{pmatrix}$ 로 만든 다음, reputation을 하여 256x256의 크기로 맞춘다. 이를 double 형으로 바꿔준 뒤 q 값과 책에 나와 있는 수식을 통해 x4 매트릭스를 만들어 낸다. 그리고 이를 다시 원상태의 크기로 복원하여 int로 바꿔준 뒤 figure를 실행해보면 위와 같은 결과를 얻을 수 있다.

8-gray level은 0,1,2,3,4,5,6,7을 데이터로 사용한다는 말이기 때문에 그에 맞춰서 threshold값을 정해준다. 색의 구분을 8가지로 하기 때문에 좀 더 세분화하기 위해서 255/7 = 37인 값을 사용하게 되고 매트릭스의 각 값들이 작은 범위를 갖기 때문에 색의 경계 구분이 많아져서 결과적으로 사진의 색이 다양하고 깔끔해 지는 효과를 얻을 수 있다.

2. (MATLAB) For ‘newborn.tif’ and ‘cameraman.tif’, implement error diffusion (implement Javis algorithm, Stucki algorithm)



```
function y = jarvis_judice_ninke(filename)
fid = fopen(filename);
if fid == -1
    error('잘못된 입력값!');
end;

x = imread(filename);

height = size(x,1);
width = size(x,2);
y = uint8(zeros(height, width));
z = zeros(height+4, width+4);
z(3:height+2, 3:width+2)=x;
for i = 3 : height+2,
    for j = 3 : width+2,
        if z(i,j) < 128
            y(i-1,j-1) = 0;
            e = z(i,j);
        else
            y(i-1,j-1) = 255;
            e = z(i,j)-255;
        end
        z(i,j+1)=z(i,j+1)+7*e/48;
        z(i,j+2)=z(i,j+2)+5*e/48;
        z(i+1,j-2)=z(i+1,j-2)+3*e/48;
        z(i+1,j-1)=z(i+1,j-1)+5*e/48;
        z(i+1,j)=z(i+1,j)+7*e/48;
        z(i+1,j+1)=z(i+1,j+1)+5*e/48;
        z(i+1,j+2)=z(i+1,j+2)+3*e/48;
        z(i+2,j-2)=z(i+2,j-2)+e/48;
        z(i+2,j-1)=z(i+2,j-1)+3*e/48;
        z(i+2,j)=z(i+2,j)+5*e/48;
        z(i+2,j+1)=z(i+2,j+1)+3*e/48;
        z(i+2,j+2)=z(i+2,j+2)+e/48;
    end
end

figure('Name','Jarvis Judice Ninke Algorithm'), imshow(uint8(y));
```

```
function y = stucki_diffusion(filename)
fid = fopen(filename);
if fid == -1
    error('잘못된 입력값!');
end;

x = imread(filename);

height = size(x,1);
width = size(x,2);
y = uint8(zeros(height, width));
z = zeros(height+4, width+4);
z(3:height+2, 3:width+2)=x;
for i = 3 : height+2,
    for j = 3 : width+2,
        if z(i,j) < 128
            y(i-1,j-1) = 0;
            e = z(i,j);
        else
            y(i-1,j-1) = 255;
            e = z(i,j)-255;
        end
        z(i,j+1)=z(i,j+1)+8*e/42;
        z(i,j+2)=z(i,j+2)+4*e/42;
        z(i+1,j-2)=z(i+1,j-2)+2*e/42;
        z(i+1,j-1)=z(i+1,j-1)+4*e/42;
        z(i+1,j)=z(i+1,j)+8*e/42;
        z(i+1,j+1)=z(i+1,j+1)+4*e/42;
        z(i+1,j+2)=z(i+1,j+2)+2*e/42;
        z(i+2,j-2)=z(i+2,j-2)+e/42;
        z(i+2,j-1)=z(i+2,j-1)+2*e/42;
        z(i+2,j)=z(i+2,j)+4*e/42;
        z(i+2,j+1)=z(i+2,j+1)+2*e/42;
        z(i+2,j+2)=z(i+2,j+2)+e/42;
    end
end

figure('Name','Stucki Algorithm'), imshow(uint8(y));
```

두 알고리즘의 구현은 책에 나와 있는 Floyd-Steinberg 알고리즘을 약간 변형시키면 손쉽게 만들어 볼 수 있다. 단, 값의 범위에만 유의하면 된다. 이 두 알고리즘은 에러 확산의 범위가 두 칸씩 되기 때문에 z 매트릭스를 zeros(height+4, width+4) 만큼 주어 양변에 2씩의 여유를 주어야 하고, (3:height+2, 3:width+2)까지 범위를 주게 된다. 반복문 또한 3부터 height+2, 3부터 width+2까지 실행하고, Floyd-Steinberg에서 했던 것과 같이 각 pixel의 위치에 맞춰 알고리즘을 구현하면 처음의 사진과 같은 결과를 얻을 수 있다.

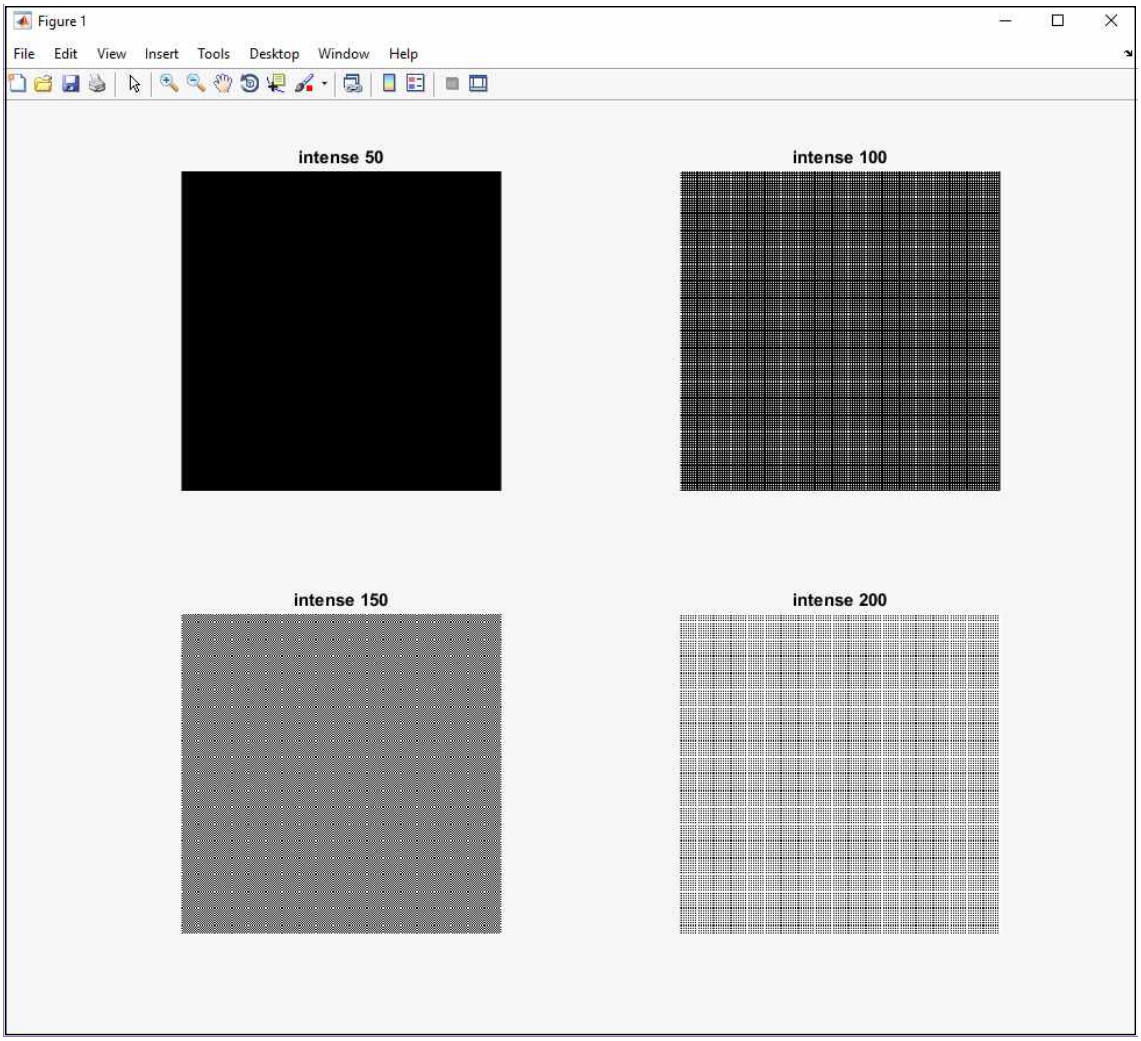
3. (MATLAB) Exercise 6: G is 256x256 grayscale image of value 50, ,100, 150, or 200 only. Namely, G contains a single intensity value. For these 4 cases,
1) Find the proper 2 x 2 dither matrix D for 2-gray level.

50, 100, 150, 200의 값으로 채워진 각각의 사진을 dithering 할 수 있는 matrix는 각 값의 사이에 위치한 값으로 만들어 주면 된다. 따라서 그 어느 매트릭스도 적절한 값이 될 수 있다. $\begin{pmatrix} 51 & 151 \\ 201 & 101 \end{pmatrix}$ 뿐만 아니라 $\begin{pmatrix} 75 & 175 \\ 225 & 125 \end{pmatrix}$를 사용해도 똑같은 결과를 얻을 수 있는 것이다.

```
function gray_dither()

intense_50 = 50 * ones(256);
intense_100 = 100 * ones(256);
intense_150 = 150 * ones(256);
intense_200 = 200 * ones(256);

uint8_50 = uint8(intense_50);
uint8_100 = uint8(intense_100);
uint8_150 = uint8(intense_150);
uint8_200 = uint8(intense_200);

D = [51,151;201,101];
r = repmat(D, 128, 128);
uint8_50 = uint8_50>r;
uint8_100 = uint8_100>r;
uint8_150 = uint8_150>r;
uint8_200 = uint8_200>r;

figure,
subplot(2,2,1), imshow(uint8_50), title('intense 50')
subplot(2,2,2), imshow(uint8_100), title('intense 100')
subplot(2,2,3), imshow(uint8_150), title('intense 150')
subplot(2,2,4), imshow(uint8_200), title('intense 200')
```

1로 채워진 256x256 매트릭스를 만들고 이 값들을 각각 50, 100, 150, 200으로 채운다. 여기에서 이 값들을 dithering하여 2-gray level로 만들기 위해서는 딱 이보다 1 높은 값으로 매트릭스를 만들어 검정색이 칠해지는 양을 다르게 할 수 있다. 따라서 위의 코드처럼 D = $\begin{pmatrix} 51 & 151 \\ 201 & 101 \end{pmatrix}$ 매트릭스를 만들고 난 뒤, 사진의 크기에 맞추어 확장하고 이전의 1번 문제와 비슷하게 코드를 적용하면 아래의 결과를 얻을 수 있게 된다.

2) Display the dithered results.



4. (Report) Exercise 8: Explain the necessary properties of 2×2 dither matrix D when 2-Gray or 4-Gray levels are used.

2-Gray level이라면 0, 1로만 값을 표현하게 된다. 그렇다면 최대한 밝은(값이 큰) 값은 1에 맞춰지도록 하고 어두운 값은 0에 맞춰지도록 한다. 따라서 적절한 기준점을 잡고서 나누기 위해 threshold를 잘 잡는 것이 중요하다. 2-Gray level은 255/2 = 128정도로 잡고 $\begin{bmatrix} 0 & 128 \\ 192 & 64 \end{bmatrix}$ 처럼 해당 기준에 맞추어 나머지 사이값을 채워주면 된다.

4-Gray level이면 0, 1, 2, 3 이므로 255/3 = 85로 잡고 $\begin{bmatrix} 0 & 28 \\ 56 & 84 \end{bmatrix}$ 정도의 매트릭스를 정해주면 이에 맞추어 적절히 값을 골라낼 수 있을 것이다.