Faculty of Engineering

Credit Hours Engineering Programs

Mechatronics Engineering and Automation

Academic Year 2019/2020 – Spring 2019

CSE 489

Machine Vision

Project No. (1)

Image Printing Based on Half-toning

Date due 18-Mar-19
Date handed in 18-Mar-19

Submitted by:

---------- ----------

Contents

List of figures

Abstract

The printing technology uses Halftoning in printing. It reduces the ink use and saves money. We were asked to write two matlab codes to do gray scale Halftoning and colored halftoning.

Technical Discussion

First of all, the technique used here assumes each gray level is represented by 3x3 pattern of black and white dots and each figure shows only ten shades of gray approximated by dot pattern. Implementing an algorithm to convert the input image to ten shades of gray level was necessary "`gray_scale_image_into_10_gray_levels` "

It takes the input image and determines the maximum and the minimum intensity of the image and divide them by the total number of gray level. If the image is a colored image `rgb2gray` function is used.

`"Resolution"` It takes the width, height and dpi of an image and using this formula Pixels/DPI = Inches.

`"Scale_Down_By3".` Depending on the step used it skips the corresponding columns and row resulting an image scaled down by step used. Which is in our case is 3. Scaling the image is required because as aforementioned, each pixel will occupy 3x3 pattern while printing. This function will help us in halftonning as images will be scaled down before used.

During half-tonning process each gray level will map to the corresponding dot pattern.

In non-colored halftoning, the 'rgb2gray' function was used to convert the colored image into a black & white one, this is a very important step before using the non-colored halftoning as we know that colored image's matrix are 3 times bigger than the non-colored one.

The colored images

Dithering is a more general term that refers to randomisation or perturbation of colours values or positions or intensity in order to simulate more tones than are available.

Dithering is a technique used in computer graphics to create the illusion of color depth in images with a limited color palette (color quantization). In a dithered image, colors not available in the palette are approximated by a diffusion of colored pixels from within the available palette. The human eye perceives the diffusion as a mixture of the colors within it

Result Discussion

The 1st Figure shows the resolution of the printed image based on a given length, width and dot per inch quality.

The 2nd Figure shows the original images

The 3rd Figure is the output of "`gray_scale_image_into_10_gray_levels`". It contains then shades of gray level but the naked eye cannot differentiate between them as the level are too close therefore it appears totally black.

The 4th Figure show the image being scaled down by 3

The other results are gray halftonning and colored halftonning.

Results

```
Command Window
New to MATLAB? See resources for Getting Started.
>> main
>> main
>> main
the width of the image is: 360.000000 pixels
the Height of is the image is : 240.000000 pixels
```

Figure 1resolution

Figure 2 Original Image

Figure 3 First ten shades of gray levels

Figure 4 scaled down by 3

Figure 5 Orginal Image

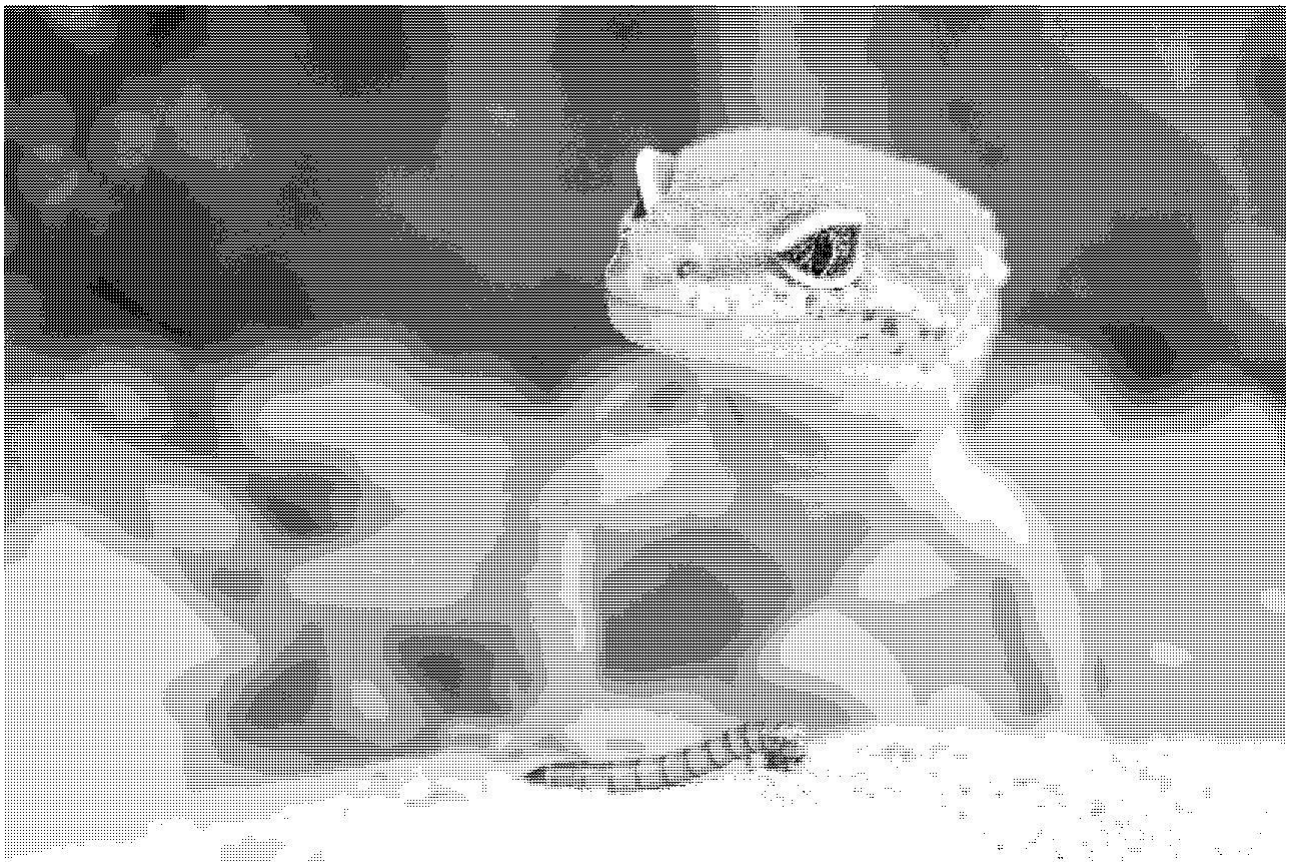

Figure 6 After Gray Half-Tonning

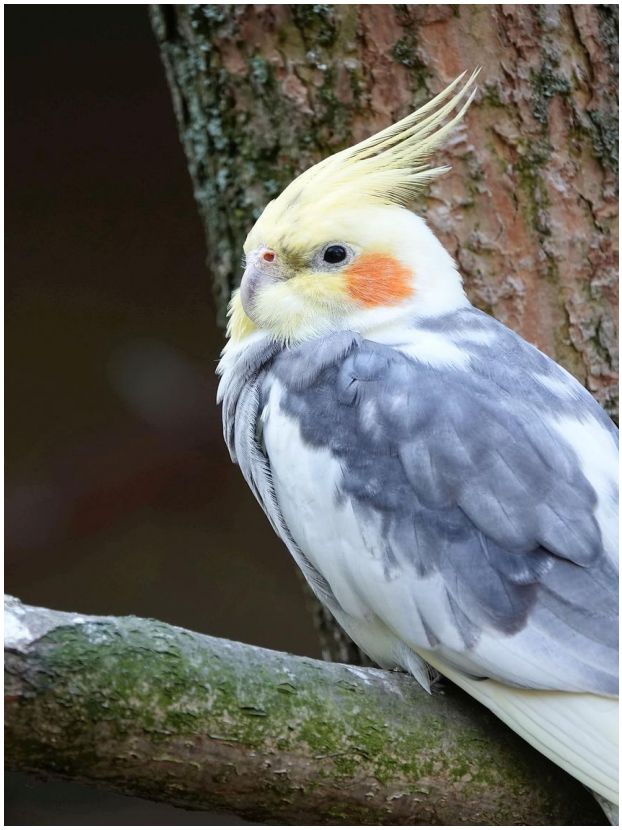

Figure 7Orginal Image

Figure 8 After Gray Half-Tonning

Figure 9 Colored Halftonning

Figure 10Orginal Image

Figure 11 Colored Halftonning

Figure 12 After Gray Half-Tonning

Figure 13Orginal Image

Figure 14 After Gray Half-Tonning

Figure 15 Colored Halftonning

Figure 16Orginal Image

Figure 17 Colored Halftonning

Figure 18 After Gray Half-Tonning

Figure 19 Orginal Image

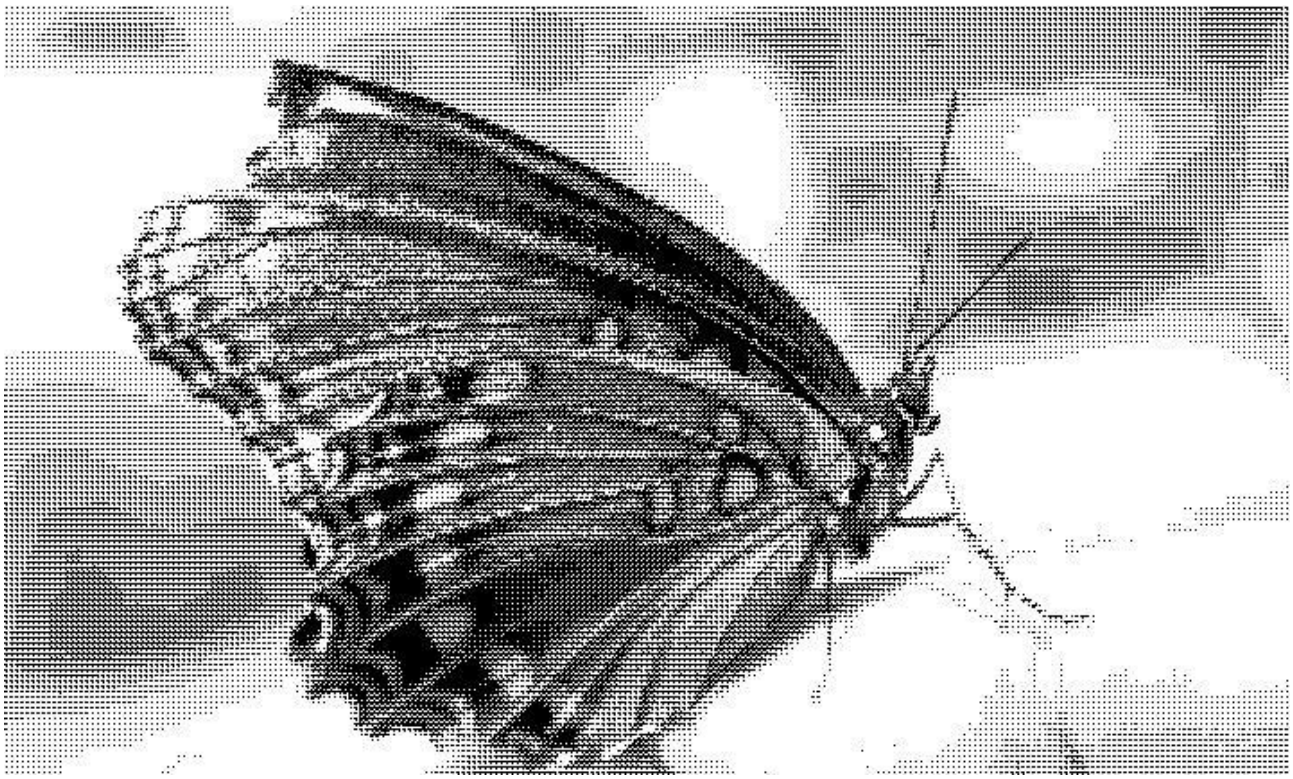

Figure 20 After Gray Half-Tonning

Figure 21 Colored Halftonning

Figure 22Orginal Image

Figure 23 After Gray Half-Tonning

Figure 24 Colored Halftonning

Figure 25Orginal Image

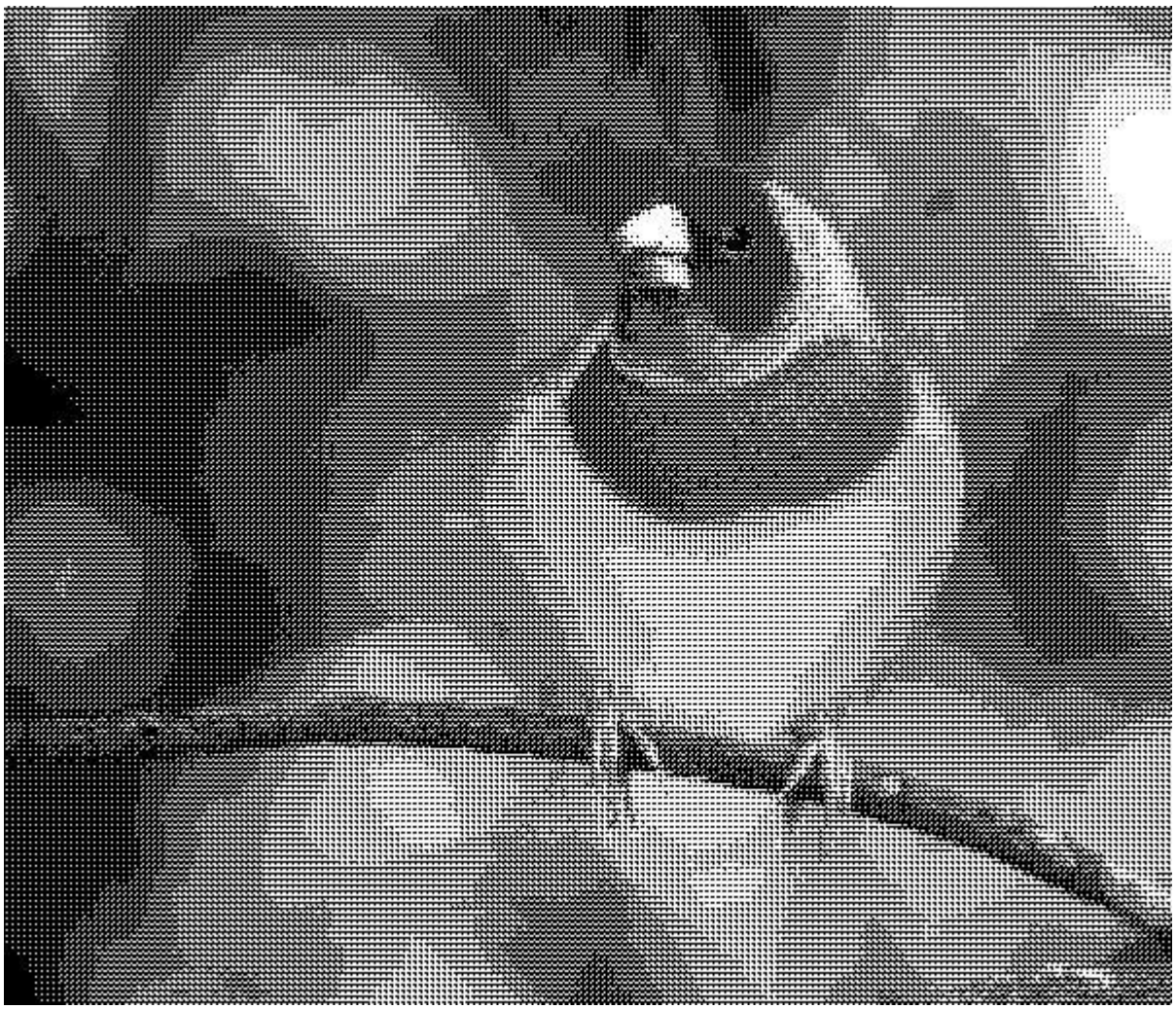

Figure 26 After Gray Half-Tonning

Figure 27 Colored Halftonning

Figure 28Orginal Image

Figure 29 After Gray Half-Tonning

Figure 30 Colored Halftonning

Figure 31Orginal Image

Figure 32 After Gray Half-Tonning

Figure 33 Colored Halftonning

Appendix

MATLAB Code

- Q1)a

```
function [ Output_Image ] =  gray_scale_image_into_10_gray_levels( Input_Image )
%gray_scale_image_into_10_gray_levels Summary of this function goes here
%Ahmed Mohammed Abdullah 14P8024
%Write a function to decompose the intensities of the input gray scale image
into 10 gray levels
%only to make it suitable for the printing process.
copy_Input_Image=Input_Image;
Gray_Image=rgb2gray(copy_Input_Image);
[height,width]=size(Gray_Image);
max_gray_level= max(Gray_Image(:));
min_gray_level= min(Gray_Image(:));
T=uint8(floor(max_gray_level - min_gray_level)/10);
Output_Image =uint8( zeros(height,width));%prelocating for computational speed
for i=1:height
    for j=1:width
        if (min_gray_level+0*T)<= Gray_Image(i,j) &&
Gray_Image(i,j)<(min_gray_level+T)
            Output_Image(i,j)=0;
        elseif (min_gray_level+1*T)<= Gray_Image(i,j) &&
Gray_Image(i,j)<(min_gray_level+2*T)
            Output_Image(i,j)=1;
        elseif (min_gray_level+2*T)<= Gray_Image(i,j) &&
Gray_Image(i,j)<(min_gray_level+3*T)
            Output_Image(i,j)=2;
        elseif (min_gray_level+3*T)<= Gray_Image(i,j) &&
Gray_Image(i,j)<(min_gray_level+4*T)
            Output_Image(i,j)=3;
        elseif (min_gray_level+4*T)<= Gray_Image(i,j) &&
Gray_Image(i,j)<(min_gray_level+5*T)
            Output_Image(i,j)=4;
        elseif (min_gray_level+5*T)<= Gray_Image(i,j) &&
Gray_Image(i,j)<(min_gray_level+6*T)
            Output_Image(i,j)=5;
        elseif (min_gray_level+6*T)<= Gray_Image(i,j) &&
Gray_Image(i,j)<(min_gray_level+7*T)
            Output_Image(i,j)=6;
        elseif (min_gray_level+7*T)<= Gray_Image(i,j) &&
Gray_Image(i,j)<(min_gray_level+8*T)
            Output_Image(i,j)=7;
        elseif (min_gray_level+8*T)<= Gray_Image(i,j) &&
Gray_Image(i,j)<(min_gray_level+9*T)
            Output_Image(i,j)=8;
        else
            Output_Image(i,j)=9;
        end
    end
end
```

Q1)B Write a function to find the resolution of the printed image (assume H x W). The arguments of
this function will be: a) the dimensions in cm/inch of the printed image and b) the image quality
(dpi).). After computing H and W, your program carries out the operations: H=3*round(H/3) and
W=3*round(W/3) to make sure that the resulting height and width divisible by 3

The Answer is:

```
function [H,W ] = resolution( h,w,dpi )
%resolution Summary of this function goes here
%Ahmed Mohammed Abdullah 14P8024
%   Detailed explanation goes here
% Write a function to find the resolution of the printed image (assume H x W).
%The arguments of
% this function will be: a) the dimensions in cm/inch of the printed image and
b)
%the image quality
%(dpi).). After computing H and W, your program carries out the operations:
%H=3*round(H/3) and
% W=3*round(W/3) to make sure that the resulting height and width divisible by 3
%h,w are in inches
H=dpi*h;
W=dpi*w;
H=3*round(H/3);
W=3*round(W/3);
fprintf('the width of the image is: %f pixels \n',W)
fprintf('the Height of is the image is : %f pixels \n',H)
% Convert pixels to inches (Output to Monitors/Printers)
% Formula: Pixels ÷ DPI = Inches
end
```

Q1)C Write a function to scale the input image to h x w where h=H/3 and w=W/3.

The Answer is:

```
function [ Output_Image ] = Scale_Down_By3( Input_Image )
%Scale_Down_By3 Summary of this function goes here
%   Detailed explanation goes here
% Ahmed mohammed Abdullah 14P8024
Input_Image=rgb2gray(Input_Image);
[height,width]=size(Input_Image);
Output_Image=uint8(zeros(round(height/3),round(width/3)));
x=1;y=1;
for i=1:3:height%step size for scaling
    for j=1:3:width
        Output_Image(x,y)=Input_Image(i,j);
        y=y+1;
    end
x=x+1;
y=1;
end
end
```

Q1)D

```
function Output=HalftoningGray(Input_Image)
%Converting each pixel into 3x3 pixels
% Write a computer program for halftoning: the input will be the gray-scale
image (resulting from
% step (c)); process will map the image intensity to the dot patterns as
discussed above.
Input_Image=gray_scale_image_into_10_gray_levels(Input_Image);
[height,width,~]=size(Input_Image);
resize_img=Scale_Down_By3(Input_Image);
[h,w]=size(resize_img);
Output=uint8(zeros(h,w));
x=1;
y=1;
for i=1:3:height
    for j=1:3:width
        switch resize_img(x,y)
            case 0
                Output(i,j)=0;
                Output(i,j+1)=0;
                Output(i,j+2)=0;
                Output(i+1,j)=0;
                Output(i+1,j+1)=0;
                Output(i+1,j+2)=0;
                Output(i+2,j)=0;
                Output(i+2,j+1)=0;
                Output(i+2,j+2)=0;
                y=y+1;
            case 1
                Output(i,j)=0;
                Output(i,j+1)=255;
                Output(i,j+2)=0;
                Output(i+1,j)=0;
                Output(i+1,j+1)=0;
                Output(i+1,j+2)=0;
                Output(i+2,j)=0;
                Output(i+2,j+1)=0;
                Output(i+2,j+2)=0;
                y=y+1;
            case 2
                Output(i,j)=0;
                Output(i,j+1)=255;
                Output(i,j+2)=0;
                Output(i+1,j)=0;
                Output(i+1,j+1)=0;
                Output(i+1,j+2)=0;
                Output(i+2,j)=0;
                Output(i+2,j+1)=0;
                Output(i+2,j+2)=255;
                y=y+1;
            case 3
                Output(i,j)=255;
                Output(i,j+1)=255;
                Output(i,j+2)=0;
                Output(i+1,j)=0;
                Output(i+1,j+1)=0;
                Output(i+1,j+2)=0;
```

```
        Output(i+2,j)=0;
        Output(i+2,j+1)=0;
        Output(i+2,j+2)=255;
        y=y+1;
    case 4
        Output(i,j)=255;
        Output(i,j+1)=255;
        Output(i,j+2)=0;
        Output(i+1,j)=0;
        Output(i+1,j+1)=0;
        Output(i+1,j+2)=0;
        Output(i+2,j)=255;
        Output(i+2,j+1)=0;
        Output(i+2,j+2)=255;
        y=y+1;
    case 5
        Output(i,j)=255;
        Output(i,j+1)=255;
        Output(i,j+2)=255;
        Output(i+1,j)=0;
        Output(i+1,j+1)=0;
        Output(i+1,j+2)=0;
        Output(i+2,j)=255;
        Output(i+2,j+1)=0;
        Output(i+2,j+2)=255;
        y=y+1;
    case 6
        Output(i,j)=255;
        Output(i,j+1)=255;
        Output(i,j+2)=255;
        Output(i+1,j)=0;
        Output(i+1,j+1)=0;
        Output(i+1,j+2)=255;
        Output(i+2,j)=255;
        Output(i+2,j+1)=0;
        Output(i+2,j+2)=255;
        y=y+1;
    case 7
        Output(i,j)=255;
        Output(i,j+1)=255;
        Output(i,j+2)=255;
        Output(i+1,j)=0;
        Output(i+1,j+1)=0;
        Output(i+1,j+2)=255;
        Output(i+2,j)=255;
        Output(i+2,j+1)=255;
        Output(i+2,j+2)=255;
        y=y+1;
    case 8
        Output(i,j)=255;
        Output(i,j+1)=255;
        Output(i,j+2)=255;
        Output(i+1,j)=255;
        Output(i+1,j+1)=0;
        Output(i+1,j+2)=255;
        Output(i+2,j)=255;
        Output(i+2,j+1)=255;
        Output(i+2,j+2)=255;
        y=y+1;
    case 9
        Output(i,j)=255;
        Output(i,j+1)=255;
        Output(i,j+2)=255;
        Output(i+1,j)=255;
```

```
                Output(i+1,j+1)=255;
                Output(i+1,j+2)=255;
                Output(i+2,j)=255;
                Output(i+2,j+1)=255;
                Output(i+2,j+2)=255;
                y=y+1;

        end
    end
    x=x+1;
    y=1;
end
end
```

Q1)E

```
%Test your program using 10 images at least
for i=1:10
input_img=imread(sprintf('%d.jpg',i));
output_img=HalftoningGray(input_img);
imwrite(output_img,sprintf('after%d.jpg',i));
end
```

Q2)

Write a function to make the Color Halftoning process, describe this function, and test it on 10
images.

```
for i=1:10
Colored_Halftonning = imread(sprintf('%d.jpg',i));
[dither,map]=rgb2ind(Colored_Halftonning,10,'dither');
imwrite(dither,map,sprintf('%d_colored.tif',i));
end
```