Binary Image Rendering using Halftoning

Lab Assignment #1

Ralphilou Tatoy

February 21, 2022

Abstract:

This lab is about performing image processing of shade images into halftone images. Halftone Is a process that selects patterns depending on the various grayscale image. The output of this process is a black and white image. The halftone algorithm finds the average of a 3x3 matrix then must select between 10 3x3 fixed patterns depending on the average value. When the pattern is selected, it will be designated in the same coordinates as the grayscale 3 x 3 matrix. This process repeats until all the images have been assessed. The output image is quite like the original but loses the shade quality.

Technical Directions:

Halftone is transforming a grayscale value image into a binary scale image. Image can be represented in a matrix and is the basis of the halftone process. The function will assess the 3x3 section of the grayscale image. By taking the mean of the matrix section, we can get the average. The tricky part is doing it the second time since the first mean will not output one scalar which is imperative in choosing the pattern in either the dots matrix or during the if-else I have implemented.

The overall number of types of a pattern is ten. When chosen, the dots matrix will be implemented into the output matrix with either pixel of white or black. Technical difficulty is that the use of for loops will not apply to a dimension of a matrix that is smaller than a 3 x 3. Another method must be implemented for it to apply to the column edges, rows edges, or both. Using the mod function, the function can detect matrices that are not 3 x 3. The process of halftoning for these types is the same in terms of taking averages but must account that the size of both grayscale and binary scale when replacing it with the pattern. This process repeats until the image is all transformed.

The script to make a matrix with an increasing number as the row increases is implemented using for loop. It is located on halftoneRun.m file. To show the image in gray scale, I used imshow with uint8.

Results

The test pattern (figure 1) that is created by a script at the start of the halftoneRun.m script showed a gradual change in gray color from dark to light. Using the halftone, we see gradual white spots being spotted as you increase from the row section (figure 2). This makes sense since the graduality from black to light corresponds with the frequency of white spots until the section are completely white.

The face picture (Figures 3 and 4) showed the same big objects such as eyes, nose, etc. The tone or shadow is different. The shadows are still completely visualized but not gradually. In conclusion, close subjects are the best result of this half-tone process.

The cameraman (figure 5 and 6) loses facial qualities. The background is mostly preserved. The background buildings are still intact but other background objects such as the vegetation are not visible.

The crowd (Figures 7 and 8) shows the most decrease in detail. The facial expressions are decreased significantly due to the shadow because of the sun. The bodies of people are still showing.

In conclusion, the transformation of these pictures is preserved depending on the distance and the variability of the grayscale. If there is much variability, the more the image loses its detail. Shadow is more noticeable when dealing with close images.

Figure 1: Gray Scale Gradual

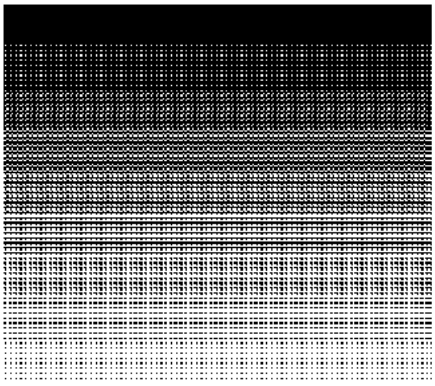

Figure 2: Halftone Grayscale Gradual

Figure 3: Face

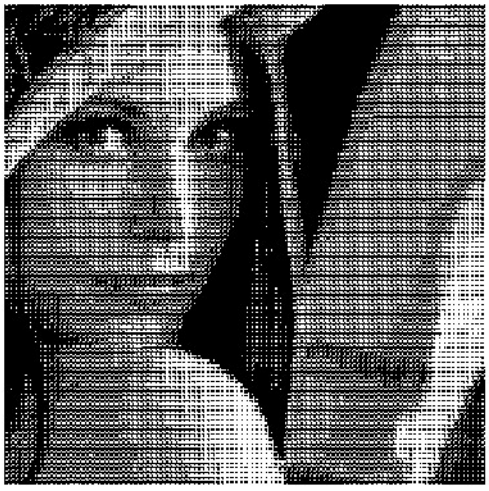

Figure 4: Halftone Face

Figure 5:Camera

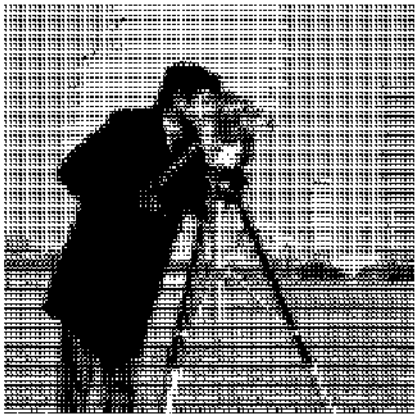

Figure 6:Halftone Camera

Figure 7:Crowd

Figure 8:Halftone Crowd

```
function out = halftone(in)
% Inputting an image in matrix form and outputting a halftone image.
% A halftone image is an image where different shades of gray that
% range from 0-255 are converted to either binary numbers.
% These binary numbers are 0 and 255 which represent black/white.
%
%
% Syntax:
% out = halftone(in)
%
%
%
% in - the gray scale matrix input that range from 0-255.
% out - the halftone output of uint8 of values of 0 or 255.
%

%Size of argument
[row ,col] = size(in);
%out = in;

%Make an image
out = uint8(zeros(row,col));

%Dot Pattern
dots = uint8(zeros(3,3,10));

%Initialize the dot pattern
dots(:,:,1) = [0,0,0 ; 0,0,0; 0,0,0];
dots(:,:,2) = [0,255,0 ; 0,0,0; 0,0,0];
dots(:,:,3) = [0,255,0 ; 0,0,0; 0,0,255];
dots(:,:,4) = [255,255,0 ; 0,0,0; 0,0,255];
dots(:,:,5) = [255,255,0 ; 0,0,0; 255,0,255];
dots(:,:,6) = [255,255,255 ; 0,0,0; 255,0,255];
dots(:,:,7) = [255,255,255 ; 0,0,255; 255,0,255];
dots(:,:,8) = [255,255,255 ; 0,0,255; 255,255,255];
dots(:,:,9) = [255,255,255 ; 255,0,255; 255,255,255];
dots(:,:,10) = [255,255,255 ; 255,255,255; 255,255,255];

%Calculate average to pick pattern
r_edge = mod(row,3);
c_edge = mod(col,3);

for r = 1:3:row-r_edge
    for c = 1:3:col-c_edge
        %Calculating Average
        average = mean(in(r:r+2,c:c+2));
        average = mean(average);

        %Picking the pattern
```

```
if average <25
    pick = 1;
elseif average >25 && average < 50
    pick = 2;
elseif average > 50 && average <75
    pick = 3;
elseif average >75 && average <100
    pick = 4;
elseif average > 100 && average < 125
    pick =5;
elseif average >125 && average <150
    pick =6;
elseif average >150 && average < 175
    pick = 7;
elseif average >175 && average <200
    pick = 8;
elseif average >200 && average < 225
    pick = 9;
elseif average >225 && average < 250
    pick = 10;
end

%Assign Pattern to output. Starting position r or c all the way up
%to 2
out(r:r+2,c:c+2) = dots(:,:,pick);

%clean up edges
if c_edge>0
%Taking average of matrix with specified col size
average = mean(in(r:r+2,col-c_edge:col));
average = mean(average);
if average <25
    pick = 1;
elseif average >25 && average < 50
    pick = 2;
elseif average > 50 && average <75
    pick = 3;
elseif average >75 && average <100
    pick = 4;
elseif average > 100 && average < 125
    pick =5;
elseif average >125 && average <150
    pick =6;
elseif average >150 && average < 175
    pick = 7;
elseif average >175 && average <200
    pick = 8;
elseif average >200 && average < 225
    pick = 9;
```

```
elseif average >225 && average < 250
    pick = 10;
end

%Assign pattern to row of 3 and column depending on the edge.
%Therefore col-c_edge:col will correspond to the right number of size
out(r:r+2,col-c_edge:col) = dots(:,1:c_edge+1,pick);
end

%If row edge are not a 3 x 3 comparison
if r_edge>0
%Taking average of matrix with specified row size
average = mean(in(row-r_edge:row,c:c+2));
average = mean(average);
if average <25
    pick = 1;
elseif average >25 && average < 50
    pick = 2;
elseif average > 50 && average <75
    pick = 3;
elseif average >75 && average <100
    pick = 4;
elseif average > 100 && average < 125
    pick =5;
elseif average >125 && average <150
    pick =6;
elseif average >150 && average < 175
    pick = 7;
elseif average >175 && average <200
    pick = 8;
elseif average >200 && average < 225
    pick = 9;
elseif average >225 && average < 250
    pick = 10;
end
%Assign pattern to col of 3 and row depending on the edge.
%Therefore row-r_edge:row will correspond to the right number of size
out(row-r_edge:row,c:c+2) = dots(1:r_edge+1,:,pick);


end

%Taking care of edges that both row and col size are not 3 x 3
if c_edge > 0 && r_edge > 0
    %Average of the matrix with specified size
    average = mean(in(row:r_edge:row,col:c_edge:col));
    average = mean(average);
```

```
            if average <25
                pick = 1;
            elseif average >25 && average < 50
                pick = 2;
            elseif average > 50 && average <75
                pick = 3;
            elseif average >75 && average <100
                pick = 4;
            elseif average > 100 && average < 125
                pick =5;
            elseif average >125 && average <150
                pick =6;
            elseif average >150 && average < 175
                pick = 7;
            elseif average >175 && average <200
                pick = 8;
            elseif average >200 && average < 225
                pick = 9;
            elseif average >225 && average < 250
                pick = 10;
            end

            %The very corner of picture. Where either bottom left or right
            out(row-r_edge:row,col-c_edge:col) = dots(1:r_edge+1,1:c_edge+1,pick);
        end
    end
end
end
```

```
%Script to input different number as the row increases
for a = 1:256
    TestPattern(a,1:256) = a;
end
imshow(uint8(TestPattern));
figure;
output = halftone(TestPattern);
imshow(output);
figure;

%image 1 (face)
image = imread('Fig0225(a)(face).tif');
imshow(image);
size(image);
output = halftone(image);
figure;
size(output);
imshow(output);
%imwrite(out,"faceHalftone.tif");

%image 2 (camera man)
figure;
image2 = imread("Fig0225(b)(cameraman).tif");
imshow(image2);
size(image2);
output2 = halftone(image2);
figure;
size(output2);
imshow(output2);
figure;


%image 3 (crowd)
image3 = imread("Fig0225(c)(crowd).tif");
imshow(image3);
size(image3);
output3 = halftone(image3);
figure;
size(output3);
imshow(output3);
```