Image Printing Based on Halftoning

Cody A. Ray

April 29, 2010

1 Project Goal

This objective of this project was to develop an image printing program based on the concept of halftoning.[1] The following figure shows ten shades of gray approximated by dot patterns. Each gray level is represented by a 3 x 3 pattern of black and white dots. A 3 x 3 area full of black dots is the approximation to gray-level black, or 0. Similarly, a 3 x 3 area of white dots represents gray level 9, or white. The other dot patterns are approximations to gray levels in between these two extremes. A gray-level printing scheme based on dots patterns such as these is called "halftoning." Note that each pixel in an input image will correspond to 3 x 3 pixels on the printed image, so spatial resolution will be reduced to 33% of the original in both the vertical and horizontal direction.

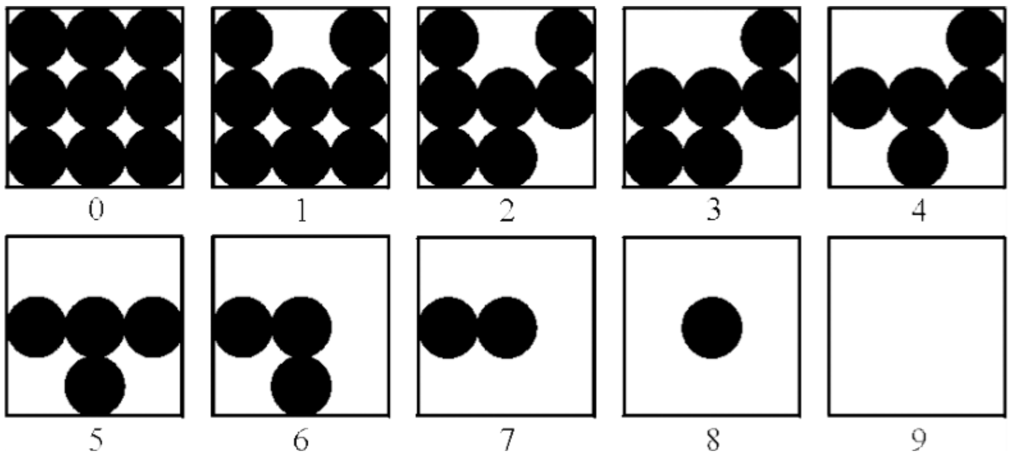

Figure 1: Halftone Dot Pattern Approximating Ten Shades of Gray

[1]Due to differences in printer capabilities, half-tone images were written to file rather than printed.

2 Findings and Conclusions

This halftoning specification, containing only ten shades of gray and having the dots evenly spaced and sized, leaves much to be desired in the quality of the halftoned image. While this specification was simple to implement and has reasonable runtime performance, there are a few clear disadvantages. The multidimensional reduction in spatial resolution is clearly unacceptable, resulting in a very low-quality halftoned image. Furthermore, the resulting three-fold increase in size creates logistical difficulties in scaling images for various applications (e.g., printing, word processing, etc). The disadvantages definitely outweigh the advantages, and this halftoning specification is not recommended to be used.

3 Supporting Data

The system was implemented in MATLAB and tested with images that came with the textbook or were pre-installed with MATLAB.

To ensure the logic was correctly implemented, the first test conducted was with a matrix rather than an actual image. Listing 1 shows the input image `in_img`, the image after scaling to the full grayscale range `mid_img`, and the resulting halftone image `out_img`. In this halftone matrix, each pixel is replaced by the corresponding set of black and white dots, where a black dot is represented by 0 and a white dot by 1.

Listing 1: Testing the Halftoning System

```
in_img =

     1     2     3
     4     5     6
     7     8     9


mid_img =

     0     1     2
     3     5     6
     7     8     9


out_img =

     0     0     0     0     1     0     0     1     0
     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     1
     1        1        0        1        1        1        1        1        1
     0        0        0        0        0        0        0        0        1
     0        0        1        1        0        1        1        0        1
     1        1        1        1        1        1        1        1        1
     0        0        1        1        0        1        1        1        1
     1        1        1        1        1        1        1        1        1
```

A visual comparison of `out_img` and `mid_img` to the dot configurations in Fig. 1 shows that the system is implemented correctly per specification.

The first image with which the program was tested was the football photograph bundled with MATLAB. However, as the program only accepts monochromatic images, the photograph was first converted to grayscale with the command `rgb2gray`. For this report, the original image was scaled using photoshop prior to halftone processing so that the halftoned image (which is three times as large) can fit on the page. The original (unscaled) image appears in Fig. 2, and the halftoned image is shown in Fig. 3.

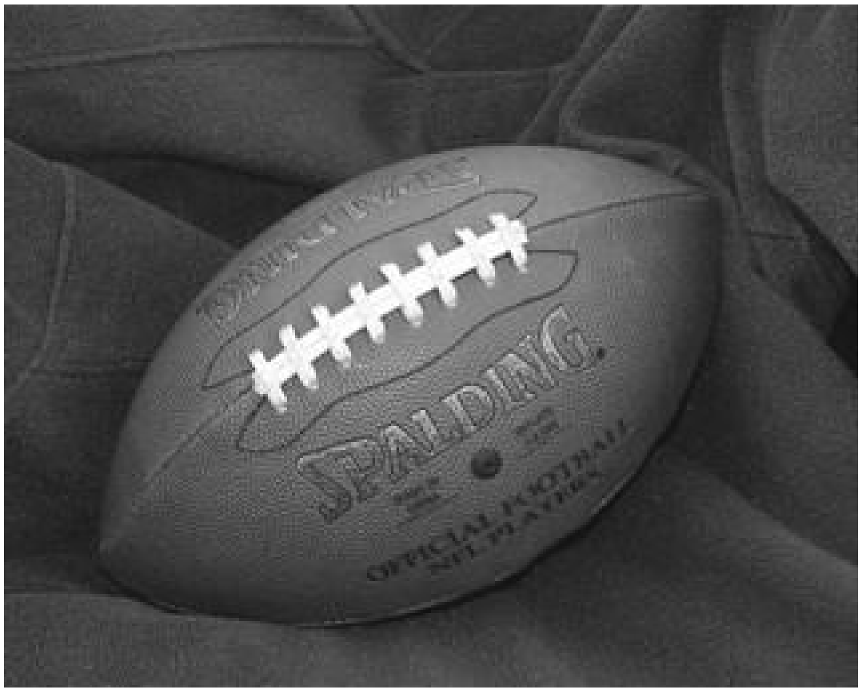

Figure 2: Original Football Image (Unscaled)

The second image tested was the near-famous image of “Lena” distributed with the textbook *Digital Image Processing* by Gonzales and Woods. The note from above regarding scaling for the report applies to this image as well. Additionally, the image was first converted from BMP format to JPG format using the ImageMagick `convert` command.

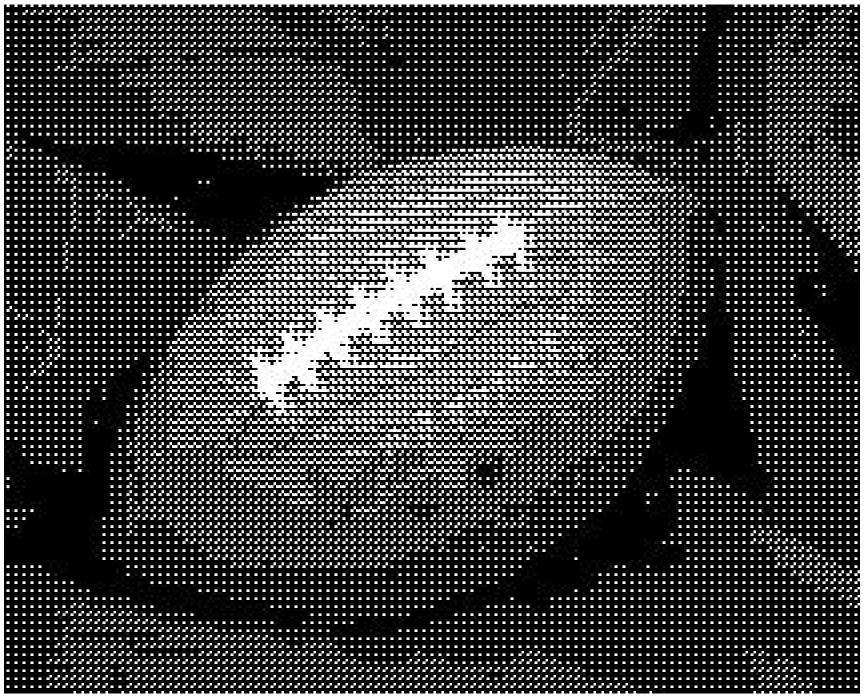

Figure 3: Halftoned Football Image

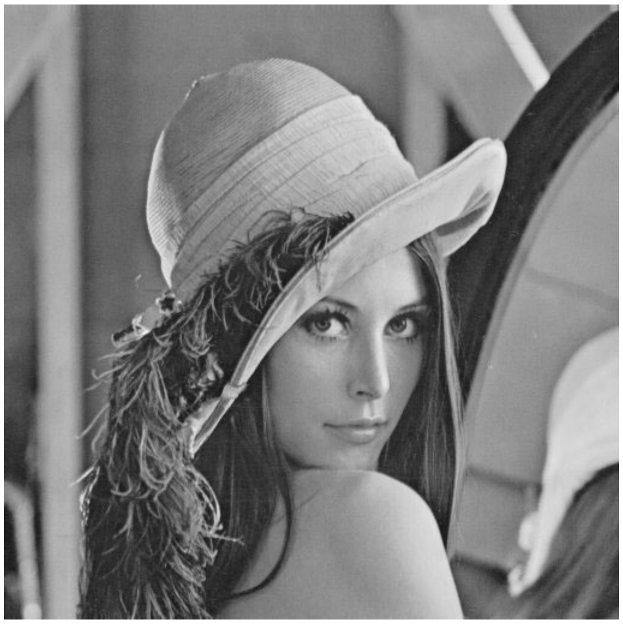

Figure 4: Original Lena Image (Scaled)

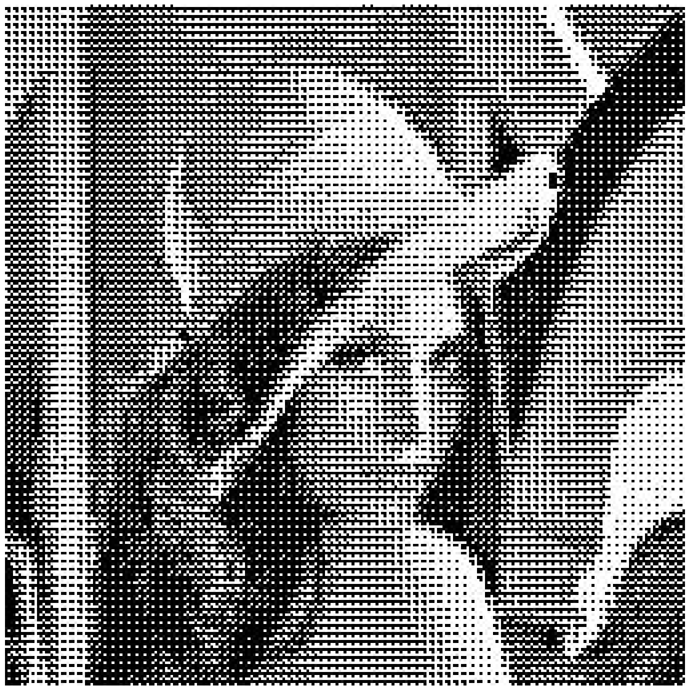

Figure 5: Halftoned Lena Image

A Computer Programs

Listing 2: Source Code of halftone.m

```
% Image Printing Based on Halftoning
% Cody A. Ray, codyaray@drexel.edu
% April 29, 2010
%
% This program rewrites an image based on halftoning.
%
% For halftoning, we use ten shades of gray approximated by dot patterns
% (see below).  Each gray level is represented by a 3 x 3 pattern of black
% and white dots.  A 3 x 3 area full of black dots is the approximation to
% gray-level black, or 0.  Similarly, a 3 x 3 area of white dots represents
% gray level 9, or white.  The other dot patterns are approximations to
% gray levels in between these two extremes.  A gray-level printing scheme
% based on dots patterns such as these is called  "halftoning."  Note that
% each pixel in an input image will correspond to 3 x 3 pixels on the
% printed image, so spatial resolution will be reduced to 33% of the
% original in both the vertical and horizontal direction.

% Algorithm:
%  (1) Read in monochromatic image
%  (2) Scale gray levels to span full half-toning range
%  (3) Map gray levels to halftones
%  (4) Store corresponding half-tone for each pixel
%  (5) Return halftone image

function out_img = halftone(in_img)

% Scale gray levels to span full half-toning range

mini = min(min(in_img));
maxi = max(max(in_img));
step = (maxi-mini)/10;

mid_img = zeros(size(in_img));
for k = 0:9
    lmin = mini+step*k;
    lmax = lmin+step;
    [i,j] = ind2sub(size(in_img),find(in_img>=lmin&in_img<=lmax));
    for l = 1:length(i)
        mid_img(i(l),j(l)) = k;
    end
end

mid_img

% Map gray levels to halftones
```

```

w = 1;
b = 0;

gray0 = [b,b,b;b,b,b;b,b,b];
gray1 = [b,w,b;b,b,b;b,b,b];
gray2 = [b,w,b;b,b,b;b,b,w];
gray3 = [w,w,b;b,b,b;b,b,w];
gray4 = [w,w,b;b,b,b;w,b,w];
gray5 = [w,w,w;b,b,b;w,b,w];
gray6 = [w,w,w;b,b,w;w,b,w];
gray7 = [w,w,w;b,b,w;w,w,w];
gray8 = [w,w,w;w,b,w;w,w,w];
gray9 = [w,w,w;w,w,w;w,w,w];

% Store corresponding half-tone for each pixel

out_img = zeros(size(mid_img)*3);
for i = 1:size(mid_img,1)
    for j = 1:size(mid_img,2)
        switch mid_img(i,j)
            case 0
                level = gray0;
            case 1
                level = gray1;
            case 2
                level = gray2;
            case 3
                level = gray3;
            case 4
                level = gray4;
            case 5
                level = gray5;
            case 6
                level = gray6;
            case 7
                level = gray7;
            case 8
                level = gray8;
            case 9
                level = gray9;
        end

        new_i = i + 2*(i-1);
        new_j = j + 2*(j-1);
        out_img(new_i:new_i+2,new_j:new_j+2) = level;
    end
end
```

Listing 3: Source Code of usage.m

```
% Image Printing Based on Halftoning
% Cody A. Ray, codyaray@drexel.edu
% April 29, 2010
%
% Sample usage of the halftone command

% clearing memory and command window
clc; clear all; close all;

% input image
file_in   = 'football.jpg';
path_in   = ['../imgin/' file_in];
in_img    = imread(path_in);
figure, imshow(in_img),  title('Input Image');

% output halftone
file_out  = ['halftone-' file_in];
path_out  = ['../imgout/' file_out];
out_img   = halftone(in_img);
figure, imshow(out_img), title('Halftoned Image');
imwrite(out_img, path_out);
```