Digital Image Processing
Project 5
Hough Transform and Intensity Thresholding Methods

Omar Rawashdeh
Harshal Raut
Utsav Shah

November 16, 2016

Contents

List of Figures

1 Introduction

1.1 Hough Transform

The Hough transform is a technique which can be used to isolate features of a shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc. A generalized Hough transform can be employed in applications where a simple analytic description of a feature(s) is not possible. Due to the computational complexity of the generalized Hough algorithm, we restrict the focus of this discussion to the classical Hough transform. Despite its domain restrictions, the classical Hough transform (hereafter referred to without the classical prefix) retains many applications, as most manufactured parts (and many anatomical parts investigated in medical imagery) contain feature boundaries which can be described by regular curves. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise.[1]

The Hough technique is particularly useful for computing a global description of a feature(s) (where the number of solution classes need not be known a priori), given (possibly noisy) local measurements. The motivating idea behind the Hough technique for line detection is that each input measurement (e.g. coordinate point) indicates its contribution to a globally consistent solution (e.g. the physical line which gave rise to that image point).

As a simple example, consider the common problem of fitting a set of line segments to a set of discrete image points (e.g. pixel locations output from an edge detector). Figure shows some possible solutions to this problem. Here the lack of a priori knowledge about the number of desired line segments (and the ambiguity about what constitutes a line segment) render this problem underconstrained.[2]

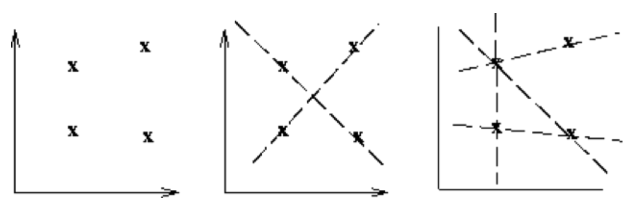

Figure 1: Coordinate Points and Possible Straight Line Fittings

We can analytically describe a line segment in a number of forms. However, a convenient equation for describing a set of lines uses parametric or normal notion: $xcos\theta + ysin\theta = r$

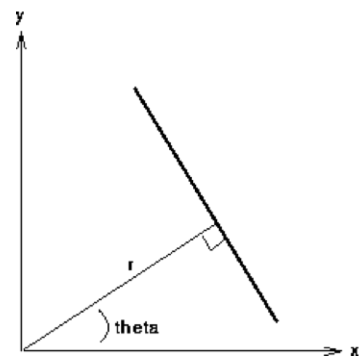

Figure 2: Parametric Description of a Straight Line

We can use this same procedure to detect other features with analytical descriptions. For instance, in the case of circles, the parametric equation is $(x-a)^2 + (y-b)^2 = r^2$

where a and b are the coordinates of the center of the circle and r is the radius. In this case, the computational complexity of the algorithm begins to increase as we now have three coordinates in the parameter space and a 3-D accumulator. (In general, the computation and the size of the accumulator array increase polynomially with the number of parameters. Thus, the basic Hough technique described here is only practical for simple curves.)[3]

The generalized Hough transform is used when the shape of the feature that we wish to isolate does not have a simple analytic equation describing its boundary. In this case, instead of using a parametric equation of the curve, we use a look-up table to define the relationship between the boundary positions and orientations and the Hough parameters. (The look-up table values must be computed during a preliminary phase using a prototype shape.)[4]

1.2 Intensity Thresholding Methods

Thresholding is a process of converting a grayscale input image to a bi-level image by using an optimal threshold,

2 Problem 1

2.1 Moon Image

Description

imfindcircle('sensitivity')

The in-built function in MATLAB imfindcircle('sensitivity') is used to detect and find the circles in an image. As in given image, we should find the circle of the moon. As this function has low sensitivity by default it is not possible to detect the circle of the moon. The sensitivity parameter of in-built function imfindcircle is increased to detect and find the circle more efficiently and effectively. Basically, due to low sensitivity of the function used the circles in an image (image of moon) is not detected.

Output

Figure 3: Original Moon Image

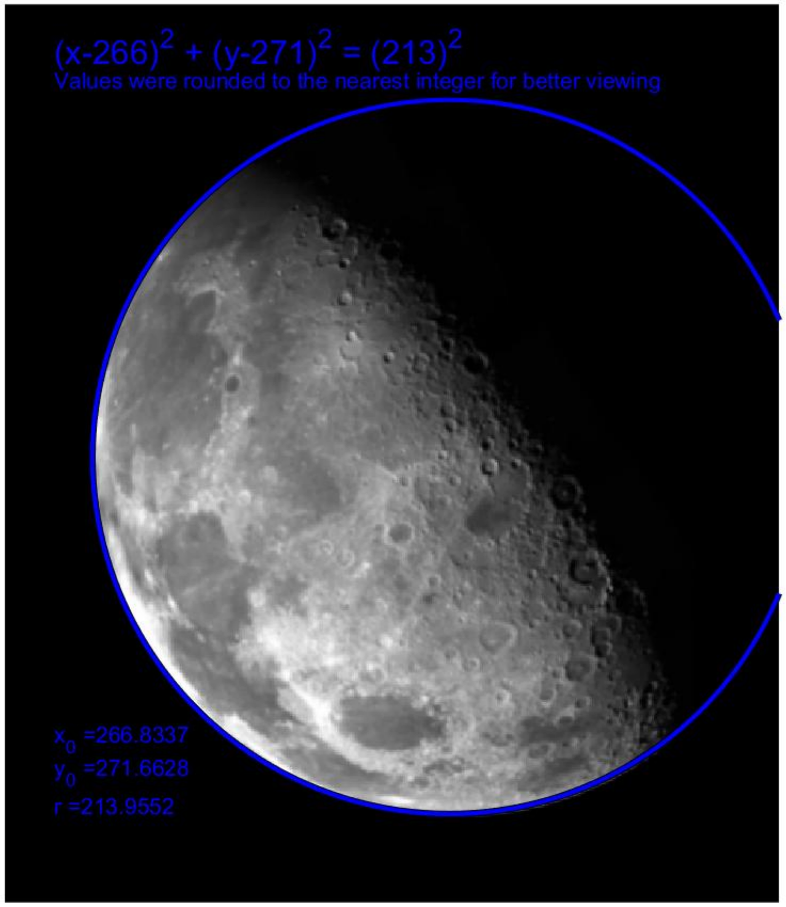

Figure 4: Circles: Moon Image

2.2 Matches Image

Description

houghlines (BW,theta,rho,peaks)

houghlines (BW,theta,rho,peaks)is an inbuild MATLAB function which is used to extract line segments in a black and white image houghlines (BW, theta, rho, peaks) where theta and rho are the vectors returned by function hough. Peaks in matrix is returned by houghpeaks function that contains row and columns co-ordinates of hough transform used to search for line segments in an image.

Output

Figure 5: Original Matches Image

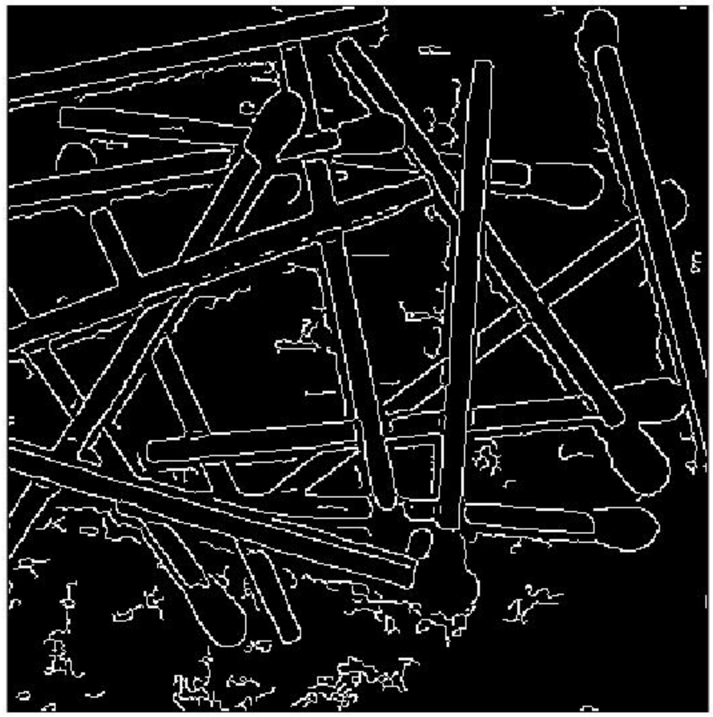

Figure 6: B and W Matches Image

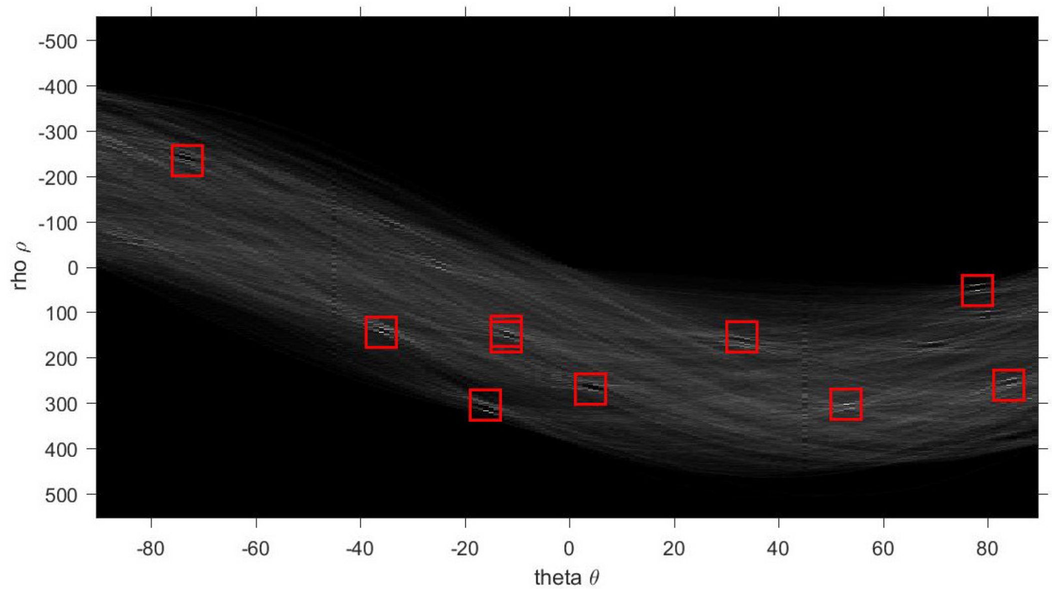

Figure 7: Rho v/s Theta Plot of Matches Image

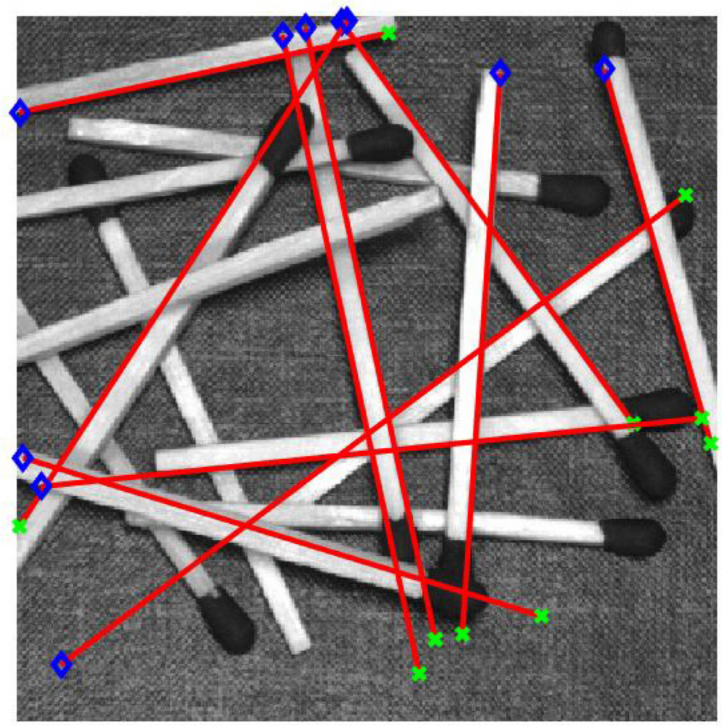

Figure 8: Lines: Matches Image

2.3 Source Code

```
% ###########################################################################
%##                               Students names :                         ##
%##                              Omar Rawashdeh                            ##
%##                               Harshal Raut                             ##
%##                                Utsav Shah                              ##
% ###########################################################################
% ###########################################################################
%##                        Digital Image Processing HW5                    ##
%##                              Hough Transform                           ##
%##                                     &                                  ##
%##                       Intensity Thresholding Methods                   ##
%##                           Matlab R2016a was used                       ##
%##                                v9.0.0.341360                           ##
% ###########################################################################

%close all figures
close all;
clear all;

%prepare folders
warning('off', 'MATLAB:MKDIR:DirectoryExists');

mkdir(fullfile(pwd,'results'));

mkdir(fullfile(pwd, 'results', 'Hough'));
mkdir(fullfile(pwd, 'results', 'Hough', 'Circle'));
```

```matlab
mkdir(fullfile(pwd, 'results', 'Hough', 'lines'));

warning('on', 'MATLAB:MKDIR:DirectoryExists');

%--------------------------%
%      problem 1           %
%--------------------------%
%<<<<<<<<<<     part 1   >>>>>>>>>>
%................................
%load the moon image
MoonImage = imread('Images for Project 5\Problem 1\MoonOriginal.jpg', 'jpg');
r_range = [200, 280];

%find circles in the image using hough method that has radius value in the
%range defined above
[centers, radii] = imfindcircles(MoonImage, r_range, 'ObjectPolarity', 'bright', 'Sensitivity', 0.9);

Fig1 = figure('Name', 'MoonEdges', 'units', 'normalized', 'Visible', 'Off', 'outerposition',[0 0 1 1]);
imshow(MoonImage);
%visualise the circle
viscircles(centers, radii,'EdgeColor','b');
Equation = strcat('(x-', num2str(fix(centers(1,1))), ')^{2} + (y-', num2str(fix(centers(1,2))), ')^{2} = (', num2str(fix(radii(1))), ')^{2}');
text(30, 25, Equation, 'color', 'blue', 'FontSize', 15);
text(30, 45, 'Values were rounded to the nearest integer for better viewing', 'color', 'blue', 'FontSize', 10);
text(30, 440,strcat('x_{0} = ', num2str(centers(1,1))), 'color', 'blue', 'FontSize', 10);
text(30, 460,strcat('y_{0} = ', num2str(centers(1,2))), 'color', 'blue', 'FontSize', 10);
text(30, 480,strcat('r = ', num2str(radii(1))), 'color', 'blue', 'FontSize', 10);

saveas(Fig1, fullfile(pwd, 'results', 'Hough', 'Circle', 'MoonCircle'), 'jpg');

%<<<<<<<<<<     part 2   >>>>>>>>>>
%................................
Fig2 = figure('Name', 'MatchesLines', 'units', 'normalized', 'Visible', 'Off', 'outerposition',[0 0 1 1]);
```

```
MatchesImage = imread('Images for Project 5\Problem 1\Matches_Image_No._4_for_project_3.bmp', 'bmp');

Or_Im = subplot(2, 2, 1);
%convert the Image to a single channel
MatchesImage = rgb2gray(MatchesImage);
MatchesImage = im2uint8(MatchesImage);
imshow(MatchesImage);

BW_Im = subplot(2, 2, 2);
%get the edge extraction binary image
BW_Matches = edge(MatchesImage, 'canny');
%show the Black and white image of the edges
imshow(BW_Matches);

HT_Im = subplot(2, 2, 3);
%do the hough transform
[H,theta,rho] = hough(BW_Matches);
%show the transform
imshow(H,[],'XData',theta,'YData',rho,'InitialMagnification','fit');
xlabel('theta \theta'), ylabel('rho \rho');
axis on, axis normal, hold on;

%find the top 10 peaks of the hough transform
Peaks = houghpeaks(H, 10, 'threshold', ceil(0.3*max(H(:))));
%show them on the image of the hough transform
PlotedSquares = plot(theta(Peaks(:,2)),rho(Peaks(:,1)),'s','color','red', 'markersize', 20);
set(PlotedSquares, 'linewidth', 2);

%get the lines from the peaks
%connect lines with gap less than 100 to form 1 strait line
lines = houghlines(BW_Matches,theta,rho,Peaks, 'FillGap',100);

LD_Im = subplot(2, 2, 4);
imshow(MatchesImage);
hold on;
max_len = 0;
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','red');

```

```
    % beginning and end of each line
     plot(xy(1,1),xy(1,2),'d','LineWidth',2,'Color','blue')
         ;
     plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','green'
         );
end

saveas(Fig2, fullfile(pwd, 'results', 'Hough', 'lines', '
    MatchesLines'), 'jpg');
```

3 Problem 2

3.1 Thresholding

Basic Global Thresholding Method

Global thresholding approach is usually considered a one stage thresholding approach, the Integral Ratio method is a first stage in a multi-stage thresholding approach. In the first stage, the image is divided into three subimages (instead of two): foreground, background, and a fuzzy subimage where it is hard to determine whether a pixel actually belongs to the foreground or the background.

If a pixel's intensity is less than or equal to A, the pixel belongs to the foreground and if a pixel's intensity is greater than or equal to C, the pixel belongs to the background. If, however, a pixel has an intensity greater than A but less than C, it belongs to the fuzzy subimage and more information from the image is needed to decide whether it actually belongs to the foreground or the background.[5]

Otsu Intensity Thresholding Method

Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.

The approach for calculating Otsu's threshold is useful for explaining the theory, but it is computationally intensive, especially if you have a full 8-bit greyscale. The next section shows a faster method of performing the calculations which is much more appropriate for implementations.[5]

3.2 Output for Basic Global Thresholding Method

Figure 9: Basic Global Thresholding Original Image

Figure 10: Basic Global Thresholding Image

3.3 Output for Otsu Intensity Thresholding Method

Figure 11: Otsu Thresholding Orignal Image

Figure 12: Otsu Thresholding Image

3.4 Comparison

In global thresholding, one arbitrary value is used as threshold. So, to get a good result image, we need to find the right threshold value which is basically a trial and error process. Since we want an automated thresholding algorithm, we need a better method to find the right threshold.

A good threshold value for an image would be a value in the middle of the peaks of histogram, which is exactly what the Otsu method does. It automatically calculates a threshold value from the image histogram of a bi-modal image. Otsu's algorithm searches for the threshold that minimizes the intra-class variance. To do so, it considers all possible thresholds and compute the variance for each of the two classes of pixels.

Hence, Otsu method is more effective and efficient as compared to global thresholding as it computes right threshold values by taking into consideration all the thresholds within an image whereas global thresholding just considers one thresholding value just by trial and error method which isn't as effective and optimal as compared to Otsu's threshold .

3.5 Source Code

```matlab
%##########################################################################
%##                         Students names :                              ##
%##                          Omar Rawashdeh                               ##
%##                           Harshal Raut                                ##
%##                            Utsav Shah                                 ##
%##########################################################################
%##########################################################################
%##                  Digital Image Processing HW5                         ##
%##                          Hough Transform                              ##
%##                                 &                                     ##
%##                  Intensity Thresholding Methods                       ##
%##                      Matlab R2016a was used                           ##
%##                           v9.0.0.341360                               ##
%##########################################################################

%close all figures
close all;
clear all;

%prepare folders
warning('off', 'MATLAB:MKDIR:DirectoryExists');

mkdir(fullfile(pwd,'results'));

mkdir(fullfile(pwd, 'results', 'Thresholding'));
mkdir(fullfile(pwd, 'results', 'Thresholding', 'Basic'));
```

```
mkdir(fullfile(pwd, 'results', 'Thresholding', 'OTSU'));

warning('on', 'MATLAB:MKDIR:DirectoryExists');

%--------------------------%
%      problem 2           %
%--------------------------%
%<<<<<<<<<<     part 1   >>>>>>>>>>
%................................
RoseImage = imread('Images for Project 5\Problem 2\Fig2-19a.jpg', 'jpg');

%initial value for the threshold
Threshold = 125;

%accepted differencce between the old threshold and the new one
Delta_Threshold = 0.001;

%boolean variable that control when to stop iterating
loop = true;

%do the iterations to select the threshold
while (loop)
    %seperate pixels into 2 groups based on the threshold value
    Group1 = RoseImage(RoseImage <= Threshold);
    Group2 = RoseImage(RoseImage > Threshold);

    %calculate the mean
    m1 = mean(Group1);
    m2 = mean(Group2);

    %calculate the new threshold value
    newT = (m1 + m2)/2;

    %check if the new value does not differ from the old value
    if abs(newT-Threshold) < Delta_Threshold
        %if the new threshold values does not differ by much then stop the
        %loop and accept the calculated new threshold as the correct
        %threshold for segmenting
        loop = false;
    end
```

```matlab

    %assgin the new calculated value to be the threshold
     Threshold = newT;
end

% %make a picture of the same size as the original image filled with only
% %ones 1 1 1 1 ...
% SegmentedBasic = ones(size(RoseImage));
%
% %change ones to zeros for any value that is less than the threshold
% SegmentedBasic(RoseImage <= Threshold) = 0;
% %multiply by 255 to convert from [0 1] to [0 255]
% SegmentedBasic = SegmentedBasic*255;

SegmentedBasic = im2bw(RoseImage, Threshold/255);
%output the result
Fig1 = figure('Name', 'Basic Global', 'units', 'normalized', 'Visible', 'Off', 'outerposition',[0 0 1 1]);
subplot(1, 2, 1);
imshow(RoseImage);

subplot(1, 2, 2);
imshow(SegmentedBasic, []);

saveas(Fig1, fullfile(pwd, 'results', 'Thresholding', 'Basic', 'BasicGlobalThreshold'), 'jpg');


%<<<<<<<<<<     part 2  >>>>>>>>>>
%...............................
%use the same image

%use ostu method to get a threshold
ThresholdLevel = graythresh(RoseImage);

%get the segmented image
SegmentedOtsu = im2bw(RoseImage, ThresholdLevel);

%output the result
Fig2 = figure('Name', 'Basic Global', 'units', 'normalized', 'Visible', 'Off', 'outerposition',[0 0 1 1]);
subplot(1, 2, 1);
```

```
imshow(RoseImage);

subplot(1, 2, 2);
imshow(SegmentedOtsu, []);

saveas(Fig2, fullfile(pwd, 'results', 'Thresholding', '
    OTSU', 'OtsuThreshold'), 'jpg');
```

Bibliography

[1] D. Ballard and C. Brown. *Computer Vision*. Chapter 4. Prentice-Hall, 1982.

[2] R. Boyle and R. Thomas. *Computer Vision: A First Course*. Chapter 5. Blackwell Scientific Publications, 1988.

[3] A. Jain. *Fundamentals of Digital Image Processing*. Chapter 9. Prentice-Hall, 1989.

[4] Dr. Vernon. *Machine Vision*. Chapter 6. Prentice-Hall, 1991.

[5] A. Patankar G. Leedham S. Varma. *Separating text and background in degraded document images - a comparison of global thresholding techniques for multi-stage thresholding*. IEEE. Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on, 2002.