LAB REPORT - 3

IMAGE PROCESSING

January 2, 2019

Submitted to

Desire SIDIBE

ASRAF ALI Abdul Salam Rasmi
Masters in Computer Vision
Centre Universitaire Condorcet
Universite de Bourgogne

Chapter 1

Filtering

1.1 INTRODUCTION

Image Filtering is the process of modifying or enhancing an Image mainly performed for Feature Extraction from the Image. Image filtering can be done both in Spatial domain and Frequency domain based on the Application. Here we are dealing with Image Filters in Spatial domain.

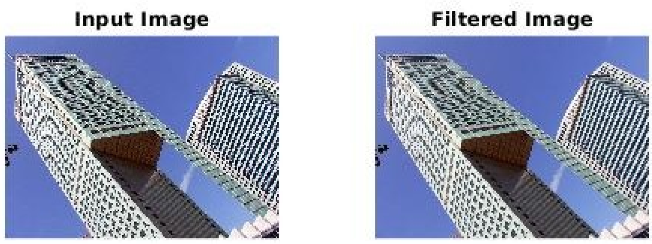

Figure 1.1: Filtering in Spatial Domain

When filters are used, we need to deal with image borders because a filter when applied to border pixel doesn't have all neighborhood pixels. In order to overcome this issue we have border conditions such as 'symmetric' (Input array values outside the bounds of the array are computed by mirror-reflecting the array across the array border), 'replicate' (Input array values outside the bounds of the array are assumed to equal the nearest array border value) and 'circular' (Input array values outside the bounds of the array are computed by implicitly assuming the input array is periodic) [1].

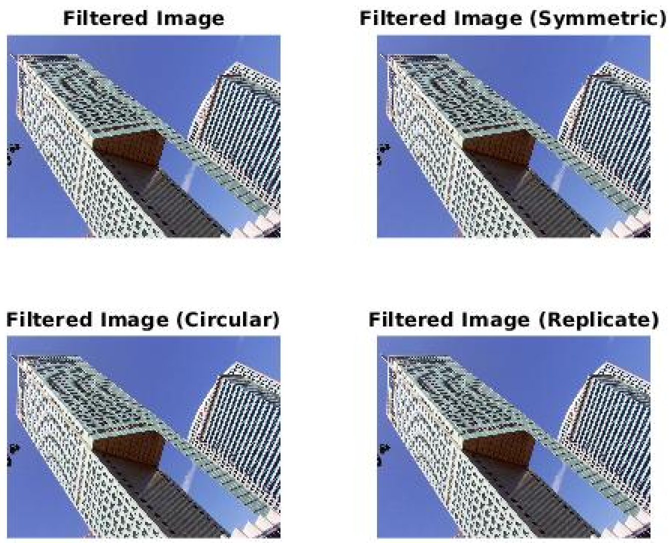

Figure 1.2: Filtering with Border Padding

1.2 LINEAR FILTERING

Linear filtering is the process in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood [2]. Here we consider two Linear Filters, Average Filter and Gaussian Filter and the results are shown in figure 1.3

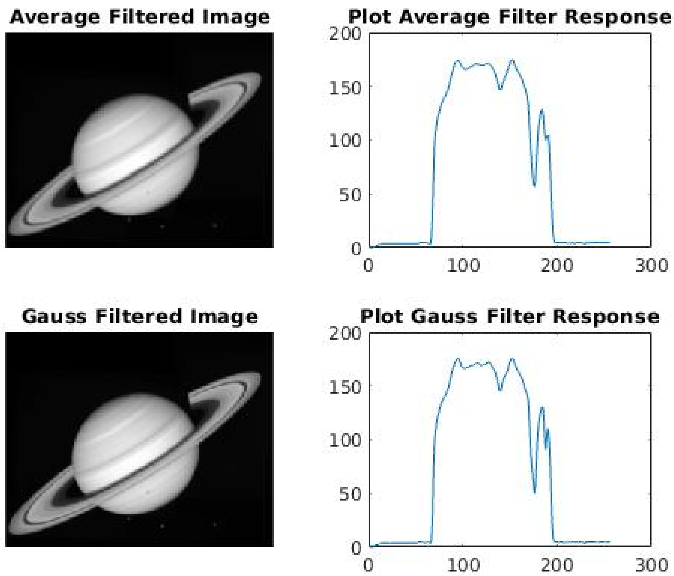

Figure 1.3: Linear Filtering

For the given Noisy Image the Averaging Filter gives good response when compared to Gauusian.

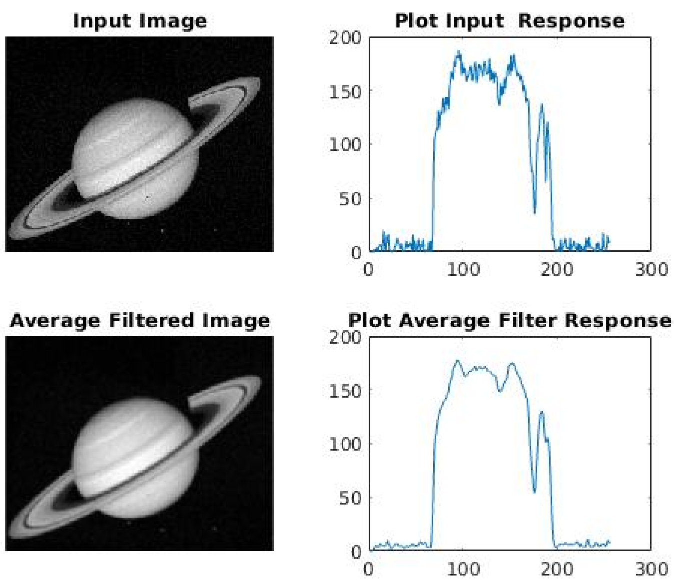

Figure 1.4: Averaging Filtering

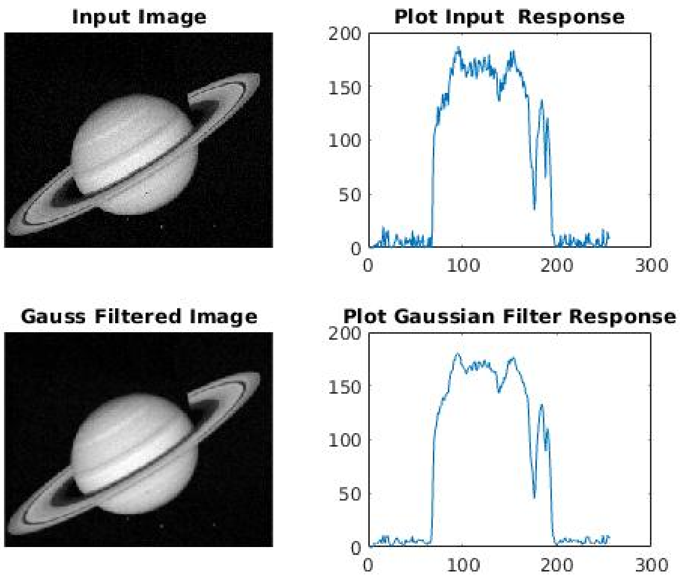

Figure 1.5: Gaussian Filtering

1.3 Nonlinear Filtering

Linear filtering is the process in which the value of an output pixel is not a linear combination of the values of the pixels in the input pixel's neighborhood. Median filter is one such filter. It is good for removing Salt & Pepper Noise. A Comparison between Linear and Nonlinear Filters are shown below.

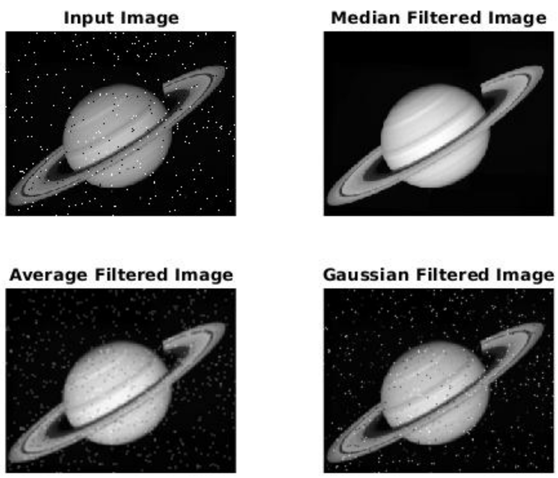

Figure 1.6: Comparison Between Median, Averaging and Gaussian Filters

Chapter 2

Edge Detection

2.1 Introduction

Edge Detection Algorithm finds the points at which the pixel intensity is changing sharply. There are many Edge detection Algorithms and we have performed some of those here.

2.2 Sobel Filters

Sobel is a well known gradient filter whose mask is given by,

$$M_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \quad M_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

The Algorithm is simple.

- Find the derivative I_x and I_y of the Input Image w.r.t. Masks M_x and M_y.
- Compute Norm of the gradient vector $\begin{pmatrix} I_x \\ I_y \end{pmatrix}$.
- Threshold the image of the gradientâĂŹs norm to get an image of contours.

The output of Sobel Edge Detection Algorithm is shown below.

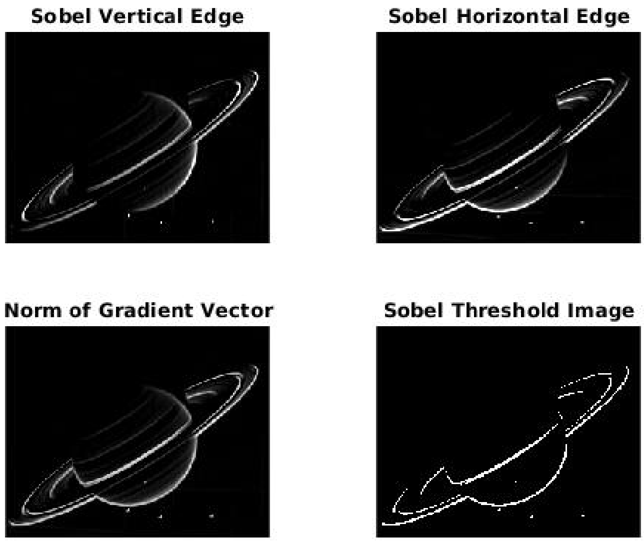

Figure 2.1: Sobel Filter

2.3 COMPARISON

The Comparison between Sobel, Prewitt and Canny Edge Detectors for different images are shown in figures 2.2, 2.3, 2.4. We can see that Canny Edge detector is more sensitive to variation in Pixel Intensity than other two.

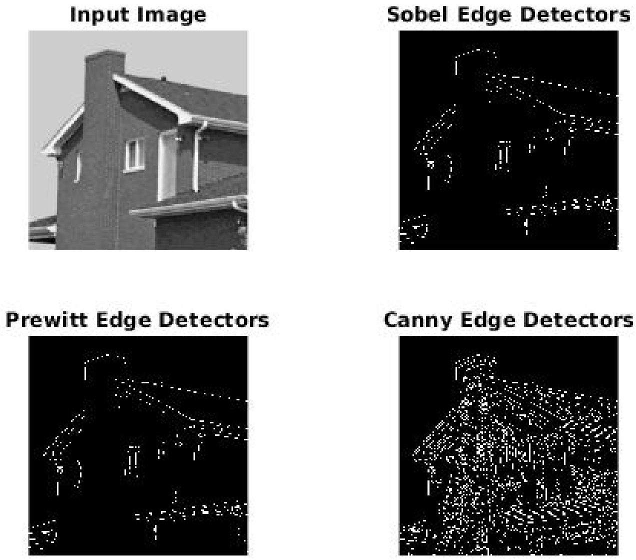

Figure 2.2: Comparison for 'house.jpg'

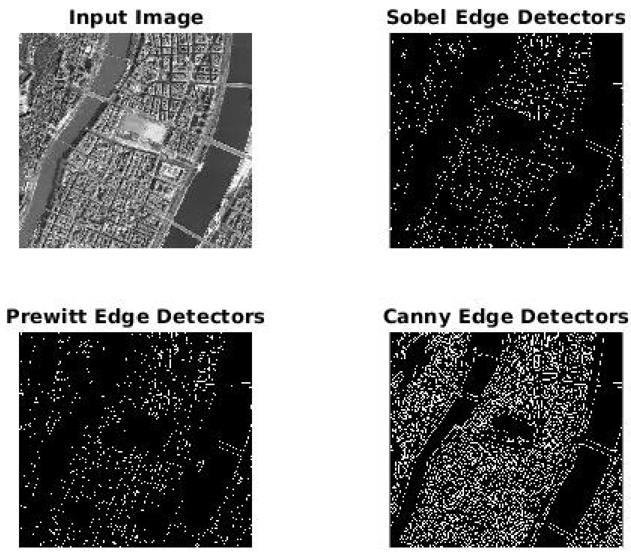

Figure 2.3: Comparison for 'satellite.jpg'

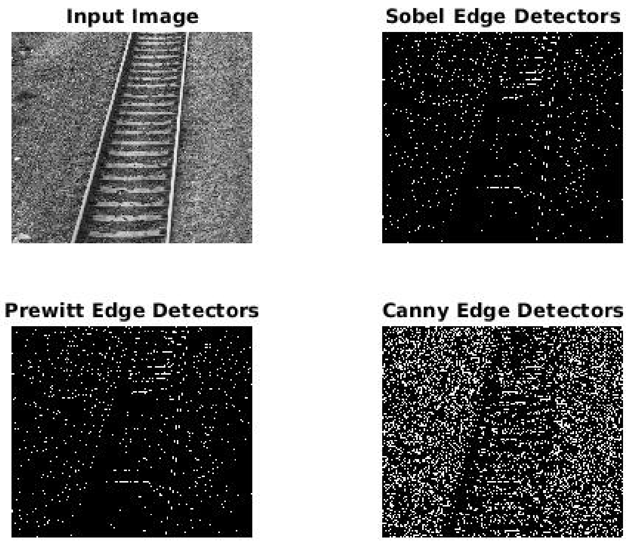

Figure 2.4: Comparison for 'railway.jpg'

Chapter 3

Hough Transform

3.1 INTRODUCTION

The Hough transform is a method that allows the detection of regular shapes (lines, circles, ellipses & etc.)in a binary image. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform [3].

3.2 HOUGH LINES

The well known application of Hough Transform is to detect lines in any given Image. The Algorithm is as follows.

1. Read an Image and convert it to gray scale.
2. Perform Edge Detection to get Binary Image with Edges.
3. Set θ_{range} as $-\theta_{max} : \theta_{max}$ where $\theta_{max} = 90$.
4. Set ρ_{range} as $-\rho_{max} : \rho_{max}$ where $\rho_{max} = \sqrt{(row)^2 + (column)^2}$.
5. Transform each pixel $(x, y)$from Image space to Parametric Space using the formula,

$$\rho = x.cos(\theta) + y.sin(\theta)$$

6. Store those values in a Hough Matrix $H(\rho,\theta)$

7. Find the First N maxima in the Hough Matrix and for Each Maxima draw a line with x as length of the column and y as

$$y = \frac{\rho - x.cos(\theta)}{sin(\theta)}$$

3.3 Results

The Matlab Function for detecting N longest lines in an image using Hough Transform is as follows.

```
function [Hough, theta_range, rho_range] = myHoughTransform(Img, N)
% MYHOUGHTRANSFORM takes an image as input and detects N longest ...
    lines using
% Hough Transform
%
%   Input
%       I - Image
%       N - No. of lines to be drawn
%
% Output
%       Hough - Hough Transform of an Image

%% Function starts here

% Convert the Image to GrayScale
if size(Img,3) == 3
    Img = rgb2gray(Img);
end

% Extract the Edges
I = edge(Img, 'canny');
% figure, imshow(BW), title('Canny Edge Detection ');

[r, c] = size(I);

% Set Rho theta Range
theta_maximum = 90;
rho_maximum = floor(sqrt(r^2 + c^2)) - 1;
```

```
theta_range = -theta_maximum:theta_maximum - 1;
rho_range = -rho_maximum:rho_maximum;

% Initialize Hough Matrix
Hough = zeros(length(rho_range), length(theta_range));

wb = waitbar(0, 'Finding Hough Transform');

% Finding the Hough Transform
for row = 1:r
    waitbar(row/r, wb);
    for col = 1:c
        if I(row, col) > 0
            x = col - 1;
            y = row - 1;
            for theta = theta_range
                % Conversion from Image Space to Feature ...
                    (Parametric) Space
                rho = round((x * cosd(theta)) + (y * sind(theta)));
                rho_index = rho + rho_maximum + 1;
                theta_index = theta + theta_maximum + 1;
                Hough(rho_index, theta_index) = Hough(rho_index, ...
                    theta_index) + 1;
            end
        end
    end
end

close(wb);
%   figure, imagesc(theta_range,rho_range,Hough),title('Hough ...
    Transform');

% To Draw N longest lines on the Image
[temp, ¬] = sort (Hough, 'descend');
figure, imshow(Img), title('Hough Lines'), hold on;
Er =0;

for i = 1:N
    [r, c] = find(Hough==temp(i));
    for j= 1: numel(r)
        rho = rho_range(r(j));
        theta = theta_range(c(j));
```

```
        x = 1:size(I, 2);
        % Conversion from Feature Space to Image Space
        y = round((rho - x * cosd(theta)) / sind(theta));
        plot(x, y, 'g-');
        Er = Er+1;
        if Er == N
            break;
        end
    end
    if Er == N
        break;
    end
end
hold off;

end
```

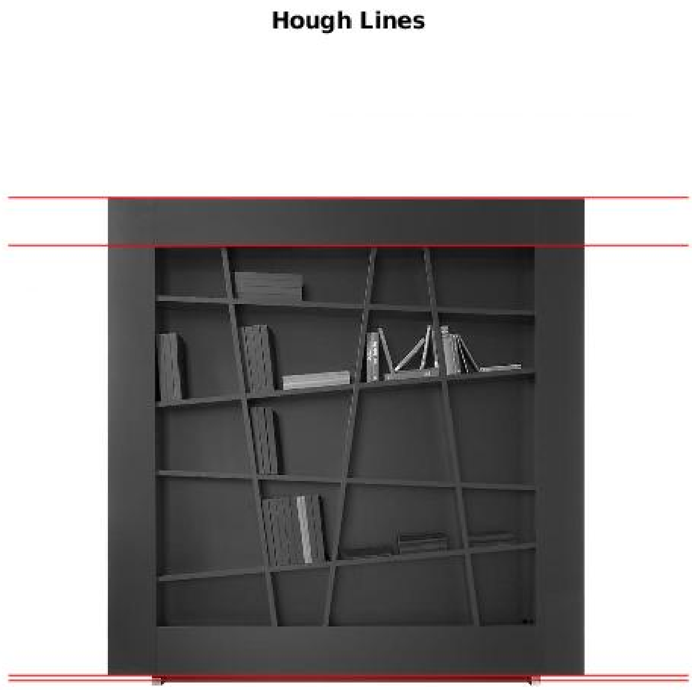

Figure 3.1: Hough Lines for $N = 4$

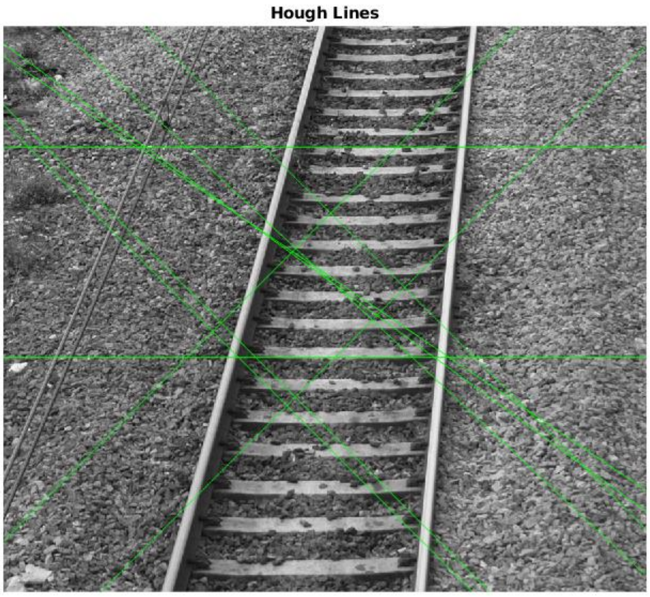

Figure 3.2: Hough Lines for $N = 10$

Bibliography

[1] "MATLAB Documentation : N-D filtering of multidimensional images." https://fr.mathworks.com/help/images/ref/imfilter.html. Accessed: 2019-01-01.

[2] "Linear Filtering and Filter Design (Image Processing Toolbox)." https://edoras.sdsu.edu/doc/matlab/toolbox/images/linfilt3.html. Accessed: 2019-01-01.

[3] L. Shapiro and G. Stockman, *Computer Vision.* Prentice Hall, 2001.