EECS 740 DIGITAL IMAGE PROCESSING ASSIGNMENT #2

Dr. Guanghui(Richard) Wang

Lohith Nanuvala

2790468

09/11/2015

Abstract:

In this assignment we are going to look at implementation of edge detection algorithms like Roberts, Sobel, Marr-Hilderth and Canny. We will also work with Hough transform using canny edge detection to find prominent lines in the image. Finally we do some segmentation using i) Global thresholding, ii) Otsu's method and iii) variable thresholding based on local image properties. For input, I have considered a building image, an image with one object, an indoor image. We will show the output and analyse the results.

Technical Discussion:

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness.

Roberts Edge detection:

- Robert's edge detection is one of the first edge detectors.
- Robert's edge detection works by approximating the gradient of the image.
- We first convolve the image with following 2 kernels,
  [1 0; 0 -1] and [0 1; -1 0]. If I(x,y) is the original point, $G_x(x,y)$ be the point in the image formed by convolving with 1st kernel and $G_y(x,y)$ be the point in the image formed by convolving with 2nd kernel, then the gradient is defined as,
  $\Delta I(x,y) = G(x,y) = \sqrt{G_x^2+G_y^2}$

Sobel Edge detection:

- Sobel edge detection is similar to Roberts, but the kernels used are different. Here we compute the 2-D spatial gradient to emphasize the regions of high spatial frequency which correspond to edges.
- The kernels are as shown below, each is a 90 degree rotation of the other.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

- Sobel edge detection also incorporates reduction of noise also.

Marr-Hilderth edge detection:

Marr-Hilderth edge detection is a method of detecting edges in digital images that are continuous curves wherever there are well-built and fast variations in image brightness.

- The step by step procedure for Marr-Hilderth edge detection is,
    1. Filter the input image with n*x*n Gaussian filter, this removes noise.
    2. Compute the laplacian of the resulting image.
    3. Find the zero crossings from the image obtained from step 2.
- Zero crossing can be found by looking for opposite signs of the neighbours.

Canny Edge detection: Canny edge detection technique is one of the standard edge detection techniques. . It was first created by John Canny for his Master's thesis at MIT in 1983, and still outperforms many of the newer algorithms that have been developed.

- The Canny edge detector uses a multi-stage algorithm to detect a wide range of edges in images.

Step 1: The first step is to filter out any noise in the original image before trying to locate and detect any edges. A Gaussian filter is used to perform this operation.

Step 2: The next step is to find the edge strength by taking the gradient of the image. The sobel operator is used to compute this gradient. The magnitude, or EDGE STRENGTH, of the gradient is then approximated using the formula: $|G|=|G_x|+|G_y|$ where G_x , G_y are gradients in x and y direction.

Step 3: In this step we compute the edge direction by using the gradients obtained from previous step by the formula

$$\text{theta} = \arctan(G_y / G_x)$$

Step 4: Once the edge direction is also known, we relate this direction to a direction that can be traced in the image. Now the edge orientation has to be resolved into one of the four directions (Horizontal [0 degrees], along positive diagonal [45 degrees], vertical direction [90 degrees] or negative diagonal[135 degrees]) depending on which direction it is closest to chosen pixel.

Step 5: Now, non-maximum suppression has to be applied. Non-maximum suppression is used to trace along the edge in the edge direction and suppress

any pixel value (set it equal to 0) that is not considered to be an edge. This will give a thin line in the output image.

Step 6: In the final step, hysteresis is used to eliminate broken edge contours. hysteris used 2 thresholds, a high and a low. Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and is marked as such immediately. Then, any pixels that are connected to this edge pixel and that have a value greater than T2 are also selected as edge pixels. If you think of following an edge, you need a gradient of T2 to start but you don't stop till you hit a gradient below T1.

Hough Transform:

- Hough transform can be used to detect lines, circles or other parametric curves in an image.
- Hough transform works by parameterizing the 2D space. Consider the simple line equation: b=-xa+y(rearranged from y=ax+b).
- Consider x,y to be parameters and a,b to be variables. Then every (x,y) values, gives a line in (a,b) space. Let (x1,y1) and (x2,y2) belong to XY space, correspondingly we have two lines in (a,b) space which intersect at (a',b').
- All points on a line defined by (x,y) and (z,w) in XY space will parameterize the lines that intersect at (a',b') in (a,b) space. Points that lie on a line form a cluster of intersections in (a,b) space.
- The parameter space (a,b) is now divided into cells, also called as accumulator cells. Then we count the # of times a line intersects given cell.
    - For each point (x,y) with value 1 in the binary image, find the values of (a,b) in the range $[[a_{min},a_{max}],[b_{min},b_{max}]]$ defining the line corresponding to this point.
    - Increase the value of the accumulator for these [a',b'] point.Then proceed with the next point in the image.
- Cells receiving maximum # of "votes" are assumed to correspond to lines in the (x,y) space.

Advantages:

- Hough transform is conceptually simple, also easy to implement.
- It handles missing and occluded data very gracefully.
- Can be extended to other forms of curves.

Drawbacks:

- It looks for only single type of object.
- Can be fooled by false lines.
- Length and position of a line segment cannot be determined.
- Co-linear line segments cannot be separated.

Image Segmentation:

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super pixels). Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images.

1. Global thresholding:
   - In global thresholding we first select an initial estimate for threshold-T and segment using T by producing two groups of pixels.
   - We compute the mean of these two groups and update T by taking average of them. $T=(m_1+m_2)/2$.
   - We repeat the above steps until there is no big difference between the means.

2. Otsu's method:
   - Otsu's method, named after Nobuyuki Otsu, is used to automatically perform clustering-based image thresholding.
   - The algorithm assumes that the image contains two classes of pixels following bi-modal histogram, it then calculates the optimum threshold separating the two classes so that their intra-class variance is minimal, or equivalently so that their inter-class variance is maximal.

   The step by step procedure is as follows,

   1. Compute histogram and probabilities of each intensity level ($P_k=n_k/N$).
   2. Compute the cumulative sums. $P(k)=\sum_{i=0}^{k} P_i$
   3. Compute the cumulative means. $m(k)=\sum_{i=0}^{k} iP_i$
   4. Compute global intensity means.
      $m_G = \sum_{i=0}^{L-1} iP_i$
   5. Compute the between class variance.

$$\sigma_B^2(k) = \frac{\left[m_G P_1(k) - m(k)\right]^2}{P_1(k)\left[1 - P_1(k)\right]}$$

6. The desired threshold is the max of variance.

$$\sigma_B^2(k^*) = \max_{0 \le k \le L-1} \sigma_B^2(k)$$

3. Variable thresholding:
   In variable thresholding we change the threshold value across the image. I have done the variable thresholding by computing local mean and variance. I have used function sigfilt() to generate local variance using a 5x5 window. For mean, I have used mean() function.

Results:

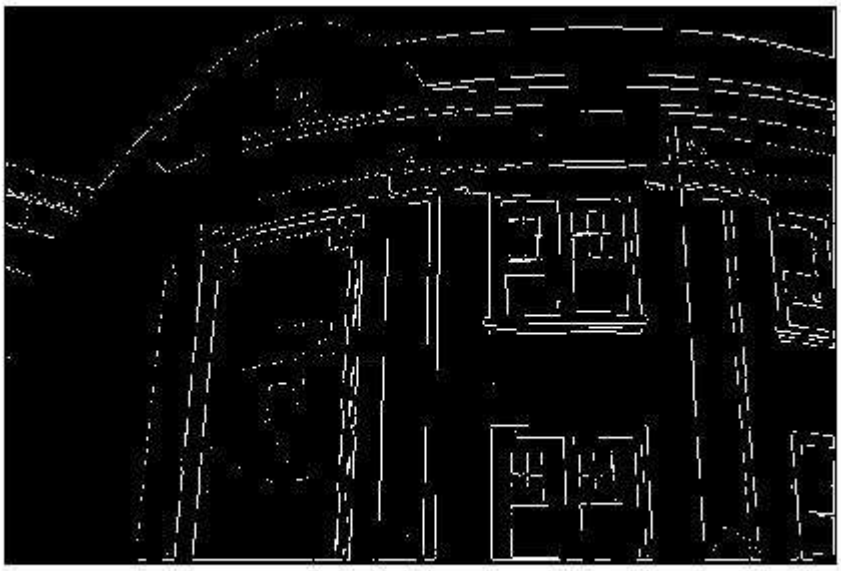
Problem 1:

Building image:

Building with Canny edge detection

Building with Marry-Hilderth edge detection

Building with Roberts edge detection



Building with Sobel edge detection

Indoor image

Indoor with Canny edge detection

Indoor with Marry-Hilderth edge detection

Indoor with Roberts edge detection

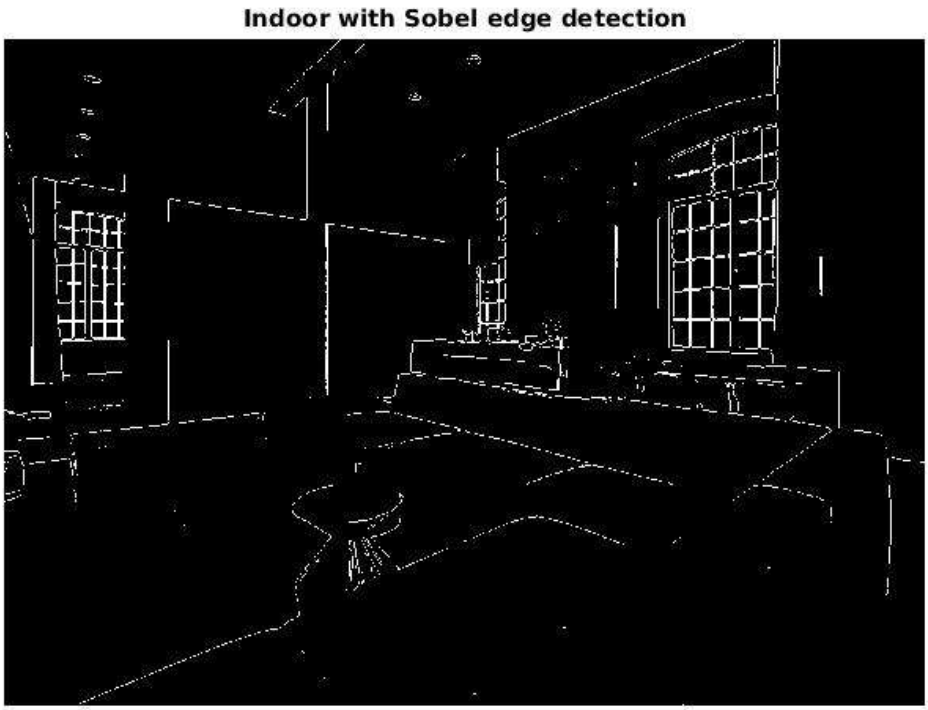
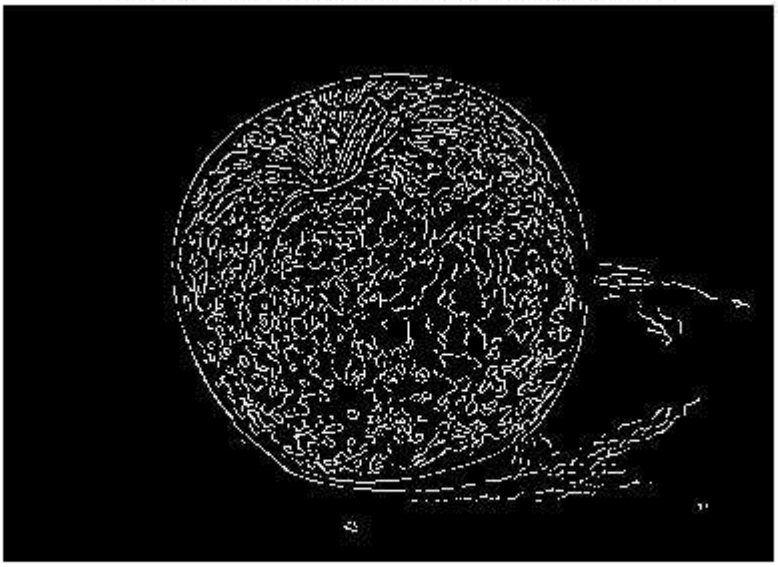
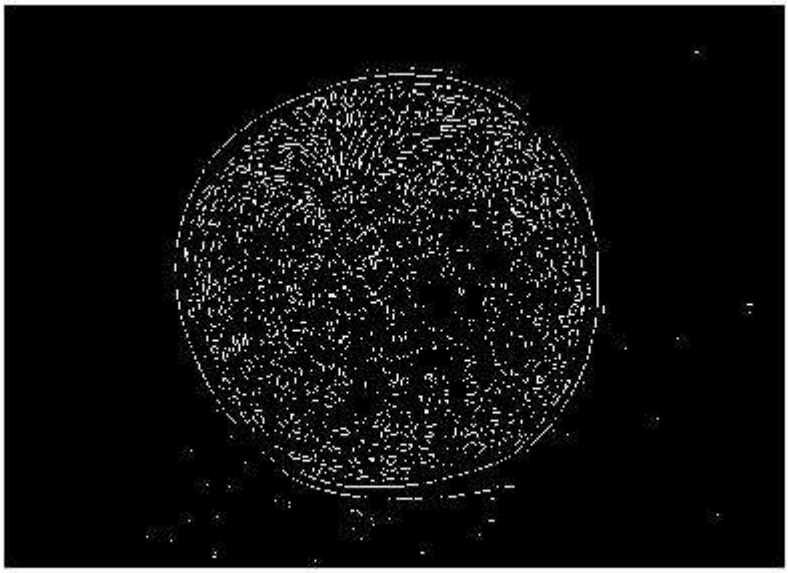


Image with one object.

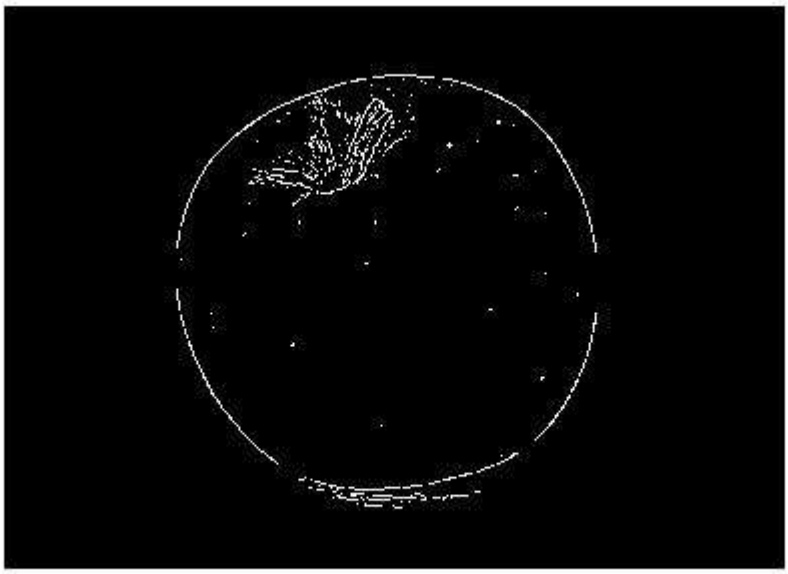
Object with Canny edge detection



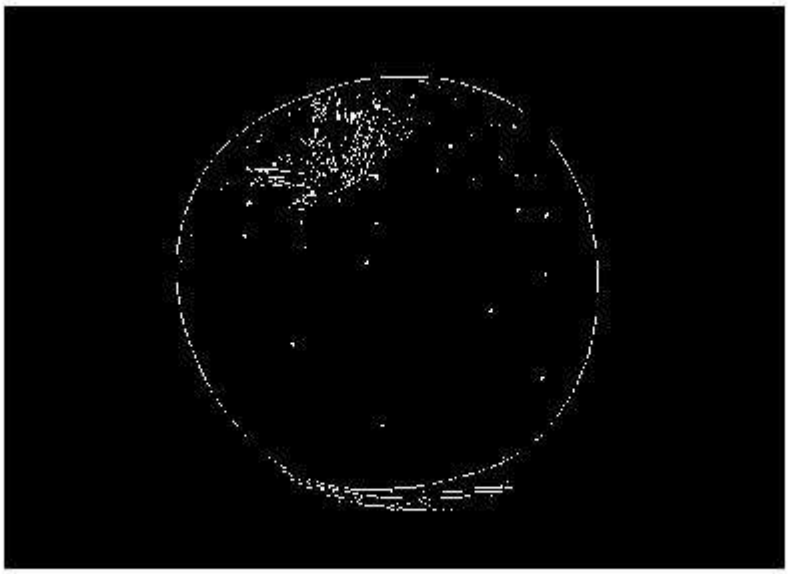
Object with Marry-Hilderth edge detection



Object with Roberts edge detection

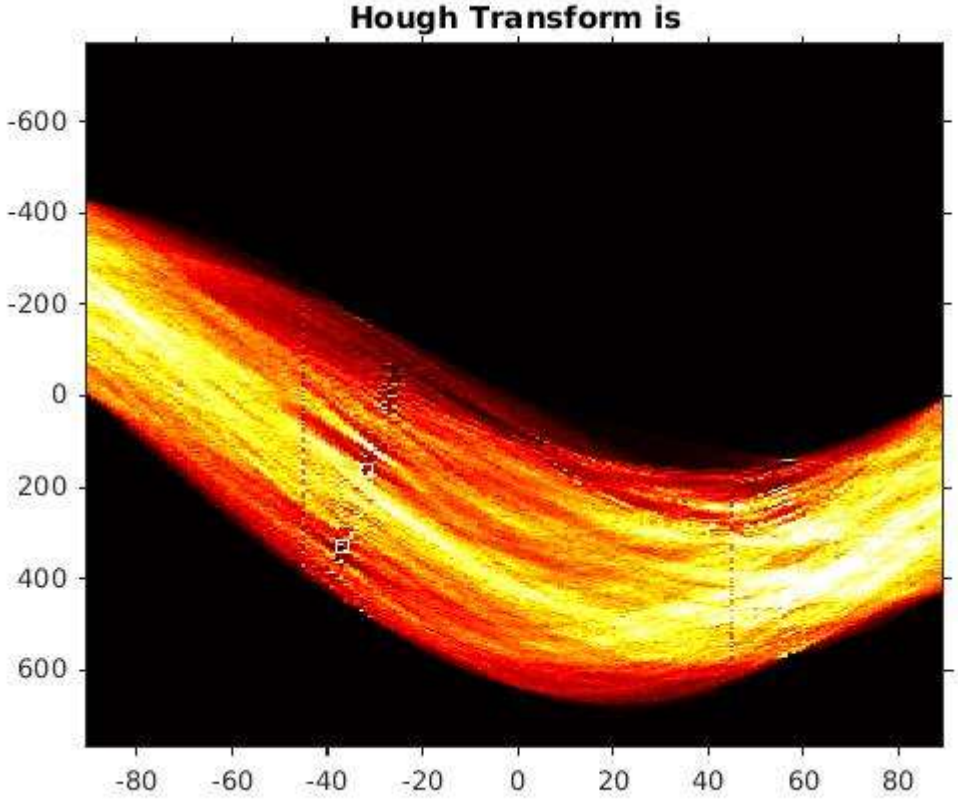


Object with Sobel edge detection



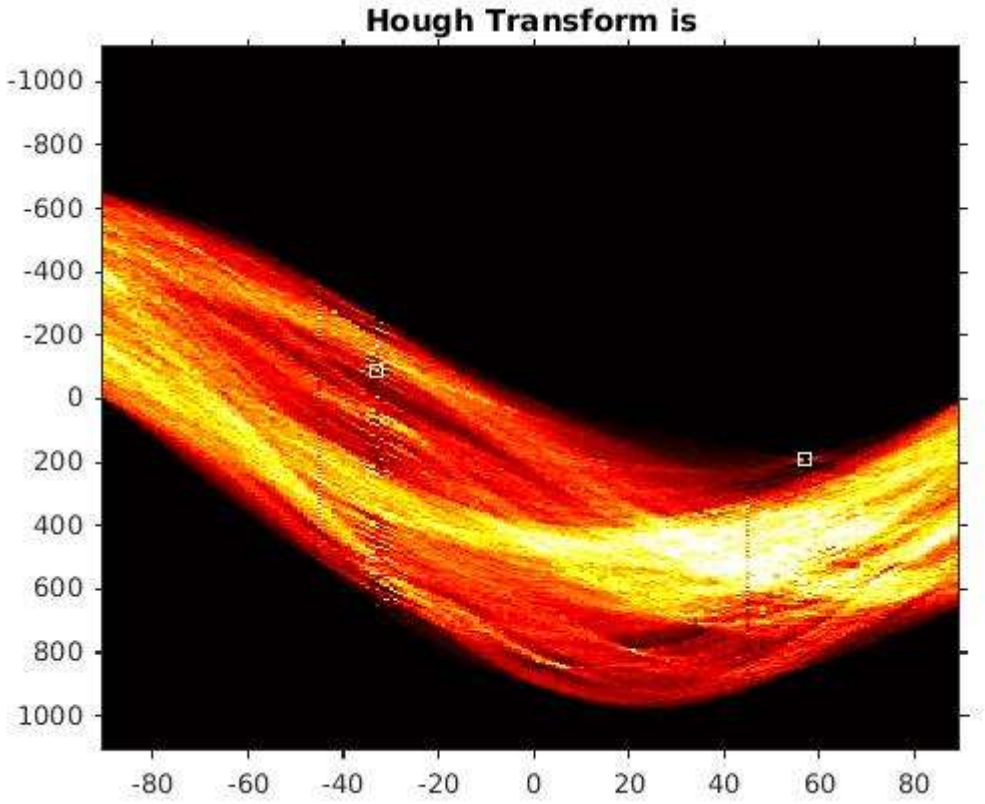
Problem 2:

Hough transform for building image:



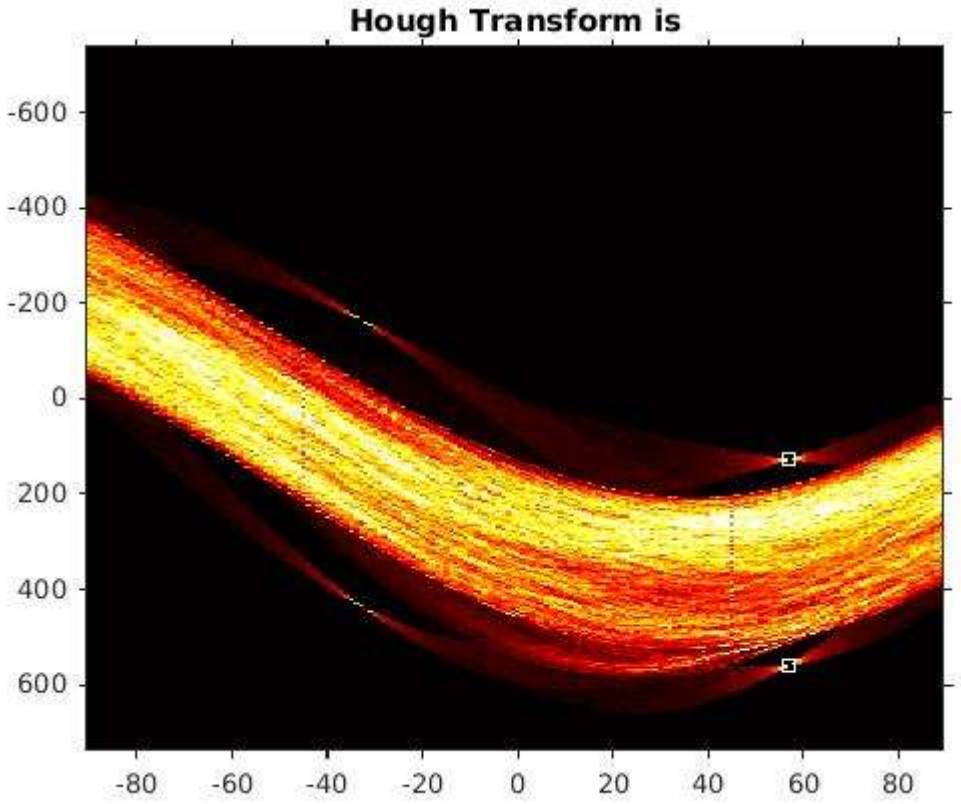
Lines detected in Building image:

Hough transform of indoor image:

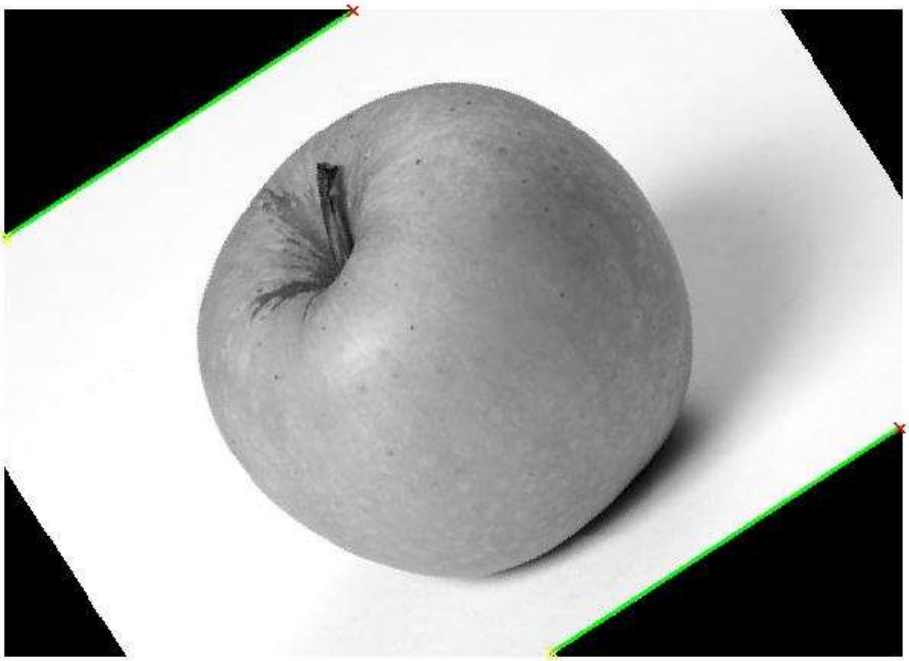


Indoor with lines:

Hough transform of Object image:



Lines in object image:



Problem 3:

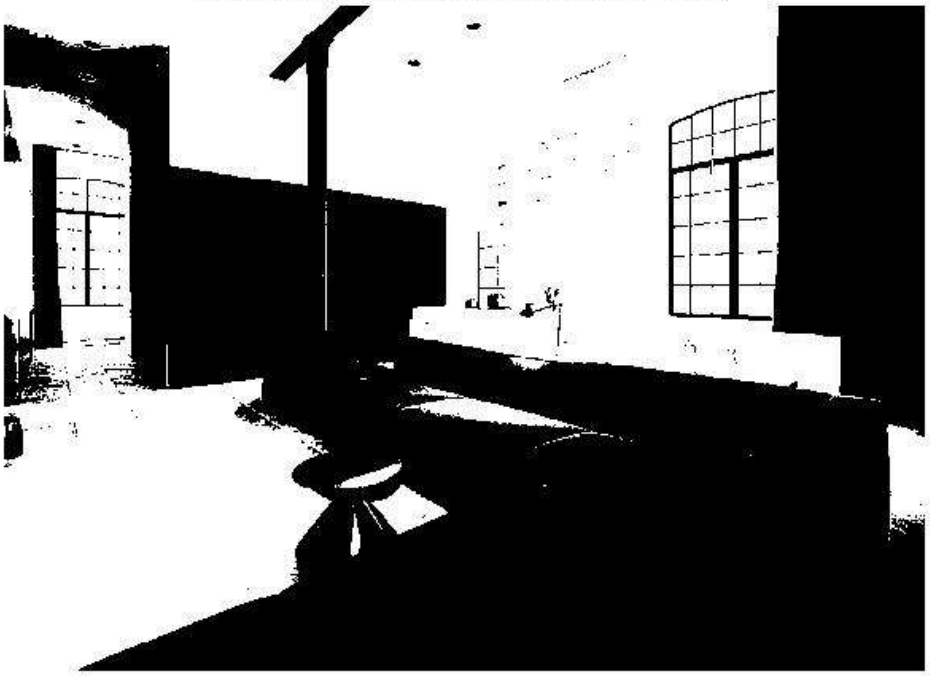
Building segmented using global thresholding

Building segmented using Otsus method

Building segmented using variable thresholding

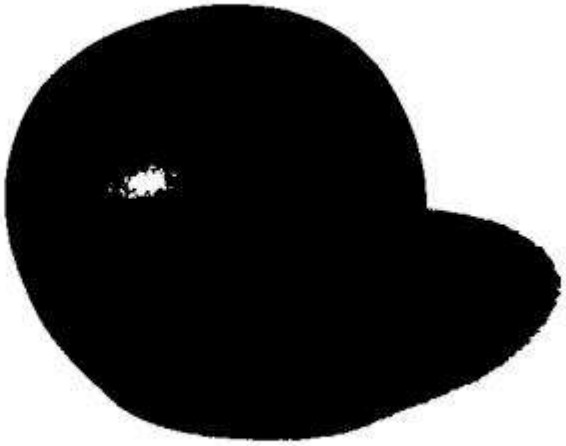
Indoor segmented using global thresholding

Indoor segmented using Otsus method

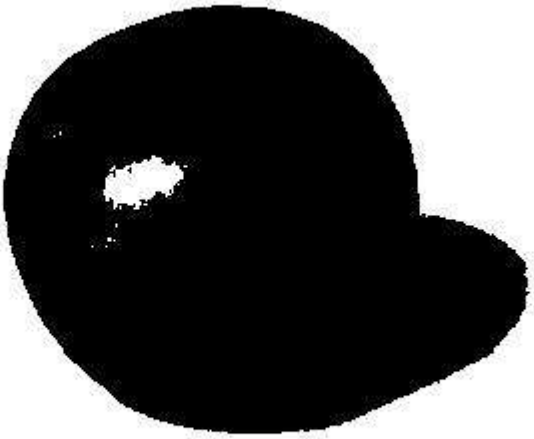


Indoor segmented using variable thresholding

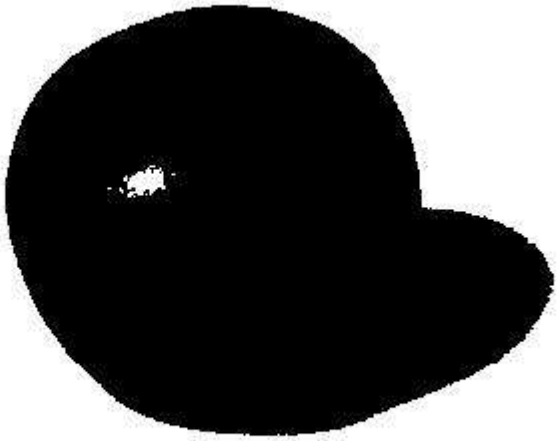
Object segmented using global thresholding



Object segmented using Otsus method



Object segmented using variable thresholding



Analysis:

Edge Detection:

- In theory we have said that Canny edge detector is the best of all the four edge detection method we have considered here. We have observed the same here.
- In all the 3 cases, i.e. a building image, image with one object, indoor image, edges detected using Canny edge detector were clearer than that of the others.
- Robert's edge detector is the basic edge detector with less complex procedure, and the result is also a very low quality edge detection. This can be clearly seen in the output images.
- From the result images, we can conclude the order of accuracy of the edge detection algorithms as Canny, Mary-Hilderth, Sobel, Robert with Canny being highly accurate and Robert least accurate.

Hough Transformation:

- I have tried to implement the Hough transform with the actual image but, I couldn't get expected results, so I have tried a rotation to the original image and the results were good.
- Especially, my image with an object has a curved object in the fore ground. We know that Hough is predominantly used to detect lines, so when I rotated the image, it has detected the border as a line which is acceptable.
- For this assignment I have considered only 2 lines, as an extension to this, I want to detect as many lines as possible, also I will try different rotations and see how the results would look.

Segmentation:

- The first thing I have observed is that the global thresholding algorithm produces a smooth image when compared to Otsu's and variable thresholding (here I have used local mean and variance).
- Further I would like to perform variable thresholding using other methods and see how the results would look.
- The second thing which we can observe is that, the global thresholding and variable thresholding have outperformed Otsu's method as they were able to segment more portion of the image when compared to Otsu in the case of image with one object.
- Where as in the case of building image and indoor image, Otsu's method has outperformed the global thresholding and variable thresholding.

NOTE: Noise has not been taken into consideration in implementing the methods, so the results obtained may vary from expected results.

Appendix:

Problem 1:

```
clear all;
% read the input image
I = imread('building.jpg');
J = imread('object.jpg');
Z = imread('indoor.jpg');
% converting to gray scale
```

```
I1 = rgb2gray(I);
J1 = rgb2gray(J);
Z1 = rgb2gray(Z);
% Roberts edge detection
I2 = edge(I1,'Roberts');
I2 = imresize(I2,0.65);
figure, imshow(I2); title('Building with Roberts edge detection');
J2 = edge(J1,'Roberts');
J2 = imresize(J2,0.65);
figure, imshow(J2); title('Object with Roberts edge detection');
Z2 = edge(Z1,'Roberts');
Z2 = imresize(Z2,0.65);
figure, imshow(Z2); title('Indoor with Roberts edge detection');
% Sobel edge detection
I3 = edge(I1,'Sobel');
I3 = imresize(I3,0.65);
figure, imshow(I3); title('Building with Sobel edge detection');
J3 = edge(J1,'Sobel');
J3 = imresize(J3,0.65);
figure, imshow(J3); title('Object with Sobel edge detection');
Z3 = edge(Z1,'Sobel');
Z3 = imresize(Z3,0.65);
figure, imshow(Z3); title('Indoor with Sobel edge detection');
%Marr-Hildreth edge detection(laplacian of gaussian)
I4 = edge(I1,'log');
I4 = imresize(I4,0.65);
figure, imshow(I4); title('Building with Marry-Hilderth edge detection');
J4 = edge(J1,'log');
J4 = imresize(J4,0.65);
figure, imshow(J4); title('Object with Marry-Hilderth edge detection');
Z4 = edge(Z1,'log');
Z4 = imresize(Z4,0.65);
figure, imshow(Z4); title('Indoor with Marry-Hilderth edge detection');
% Canny edge detection
I5 = edge(I1,'Canny');
I5 = imresize(I5,0.65);
figure, imshow(I5); title('Building with Canny edge detection');
J5 = edge(J1,'Canny');
J5 = imresize(J5,0.65);
figure, imshow(J5); title('Object with Canny edge detection');
```

```
Z5 = edge(Z1,'Canny');
Z5 = imresize(Z5,0.65);
figure, imshow(Z5); title('Indoor with Canny edge detection');
% end of program
```

Problem 2:

```
% function to plot the lines detected by hough transform
clear all;
% reading the images
I = imread('building.jpg');
J = imread('object.jpg');
Z = imread('indoor.jpg');
% convert to gray scale
% b = 'building';
% o = 'object';
% i = 'indoor';
I1 = rgb2gray(I);
J1 = rgb2gray(J);
Z1 = rgb2gray(Z);
Hough_lines(I1);
%
Hough_lines(J1);
%
Hough_lines(Z1);
% end of program
```

Hough_lines function: [1]

```
function [] = Hough_lines(In)
% I = imread('building.jpg');
% I1 = rgb2gray(I);
rot = imrotate(In,33,'crop');
E1 = edge(rot,'Canny');
[H, T, R] = hough(E1, 'RhoResolution',1.0,'ThetaResolution',1.0);
peaks = houghpeaks(H,2);
imshow(imadjust(mat2gray(H)),'XData',T,'YData',R,...
   'InitialMagnification','fit');
title('Hough Transform is');
axis on, axis normal; hold on;
colormap(hot);
```

```
x = T(peaks(:,2)); y = R(peaks(:,1));
plot(x,y,'s','color','white');
lines = houghlines(E1,T,R,peaks,'FillGap',5,'MinLength',7);
figure, imshow(rot), 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','green');
  % Plot beginnings and ends of lines
  plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
  plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
  % Determine the endpoints of the longest line segment
  len = norm(lines(k).point1 - lines(k).point2);
  if ( len > max_len)
    max_len = len;
    xy_long = xy;
  end
end
hold off;
end
%lines = houghlines(E1,T,R,peaks);
% end
```

Problem 3:

```
% matlab code for problem 3, segmentation through various methods.
% read all the input files
I = imread('building.jpg');
J = imread('object.jpg');
Z = imread('indoor.jpg');
% converting all to gray
I1 = rgb2gray(I);
J1 = rgb2gray(J);
Z1 = rgb2gray(Z);
figure, imshow(I); title('Building image');
% global thresholding
I2 = global_Thresholding(I1);
```

```
I2 = imresize(I2,0.65);
figure, imshow(I2); title('Building segmented using global thresholding');
J2 = global_Thresholding(J1);
J2 = imresize(J2,0.65);
figure, imshow(J2); title('Object segmented using global thresholding');
Z2 = global_Thresholding(Z1);
Z2 = imresize(Z2,0.65);
figure, imshow(Z2); title('Indoor segmented using global thresholding');
% end of global thresholding.
% Otsu's method of segmentation
level = graythresh(I1);
I3 = im2bw(I1,level);
I3 = imresize(I3,0.65);
figure, imshow(I3); title('Building segmented using Otsus method');
level = graythresh(J1);
J3 = im2bw(J1,level);
J3 = imresize(J3,0.65);
figure, imshow(J3); title('Object segmented using Otsus method');
level = graythresh(Z1);
Z3 = im2bw(Z1,level);
Z3 = imresize(Z3,0.65);
figure, imshow(Z3); title('Indoor segmented using Otsus method');
% end of otsu's method segmentation
% variable thresholding based on local mean and variance
I4 = variable_thresholding(I1);
I4 = imresize(I4,0.65);
figure, imshow(I4); title('Building segmented using variable thresholding');
J4 = variable_thresholding(J1);
J4 = imresize(J4,0.65);
figure, imshow(J4); title('Object segmented using variable thresholding');
Z4 = variable_thresholding(Z1);
Z4 = imresize(Z4,0.65);
figure, imshow(Z4); title('Indoor segmented using variable thresholding');
% end of program.
```

Global thresholding function:

```
function [ Op ] = global_Thresholding( I )
```

```
% global_thresholding function
[m,n]=size(I);
I1 = zeros(m,n);
T = sum(sum(I))/(m*n);
flag = true;
while flag
for i=1:m;
m1 = 0;
m2 = 0;
c1 = 0;
c2 = 0;
   for j=1:n;
      if(I(i,j)<=T)
         I1(i,j)=0;
         m1=m1+I(i,j);
         c1=c1+1;
      else
         I1(i,j)=1;
         m2=m2+I(i,j);
         c2=c2+1;
      end
   end
end
%figure, imshow(I1); title('Segmented image using global thresholding');
% computing mean of groups
m1=m1/c1;
m2=m2/c2;
if(abs(m1-m2)==0)
   flag = false;
else
   T = (m1+m2)/2;
end
end
Op=I1;
%end
```

Variable thresholding function:

```
function [ Iout ] = variable_thresholding( In )
% computing mean
m = mean(mean(In));
% local standard deviation of image
variance = stdfilt(In,ones(5));
%size(variance)
Iout = (In > variance) & (In > m);
% figure, imshow(I2); title('variable thresholding');
end
```

References:

[1] Code is generated from examples available on matlab website.
http://www.mathworks.com/help/images/ref/houghlines.html?s_tid=srchtitle