Edges

Project 3

EECS 490

Shaun Howard

Date Due: October 8th

Date Submitted: October 8th

The world is delineated by edges. Most images contain objects. Each object is delineated by multiple edges to distinguish it from other objects in the image. Edge detection is important for a computer to distinguish the objects in an image. Matlab R2016a provides many edge operators for detecting edges with an image. Some of the operators include Prewitt, Sobel, Laplacian of Gaussian (LoG) and Canny. Matlab also offers great filter, threshold, gradient, and Laplacian operations for images. Given the diversity of images, these edge detection algorithms are applicable to many types of images given their parameters are correctly tuned. Each of these algorithms and its tuned parameters are discussed within this paper, each within the context of a specific task including edge detection and object segmentation. The Matlab functions and algorithms used to determine the edge detection experiment results are outlined and described. The results of experiments are displayed, analyzed and compared in order to determine the success and usefulness of the edge operations performed. Subsequently, edge detection, image thresholding, and edge validation are discussed in response to the results.

1. **Technical Discussion**

**1.** For part 1, IMAGE1 was read in with imread. The image was converted to double precision with im2double. The double gradient operation was performed on the image with the gradient function. The Laplacian operation was performed on the image with the filter fspecial(‘laplacian’) provided as input to the imfilter function. The edges were located in the gradient of the image using Otsu’s method of global thresholding via imbinarize with the ‘global’ threshold argument as described in [1]. Zero-crossing detection was performed on IMAGE1 with the edge function, the ‘zerocross’ operator, an intensity threshold of 0.6, and the Laplace filter created with fspecial. The results were displayed with figure, subplot\_tight and imshow in a 2x2 grid as in section 3, figure 1. The previous steps were then repeated after a Gaussian filtering of the image using the imgaussfilt method with a 2D kernel size of 3x3. The zero-cross threshold in the filtered case was changed to 0.3 for this experiment, half the previous threshold. Results were shown in section 3, figure 2.

**2.** For part 2, the IMAGE2 (Lena) and IMAGE3 (Dart) images were read using imread and converted to double with im2double. The row and column-wise numerical gradient operations were performed on both images. The absolute values of row and column- wise gradient result images were then added to obtain the |Gx| + |Gy| filtering of the image. The row and column-wise Sobel gradient operations were also performed on the images using the imgradientxy function, which defaults to the Sobel gradient operation. The gradient operation results were then displayed using figure, subplot\_tight and imshow, as rendered in section 3, figures 3 and 4.

**3.** For part 3, IMAGE4 was read in using imread. The Canny operator was applied to detect edges in the image using the edge function with the ‘Canny’ operator argument and a threshold range of [0.25, 0.4]. The edges of the figure were then displayed in figure 5 using figure and imshow.

**4.** For part 4, global thresholding was applied to two images in order to detect the objects within them. First, IMAGE1 was thresholded using the imbinarize function to obtain the global grayscale intensity threshold image using Otsu’s method [1]. Second, IMAGE5 was read with imread and then thresholded similarly using the imbinarize function. Both results were then displayed alongside their original in separate 2x1 grids using figure, subplot\_tight and imshow in figures 6 and 7.

**5.** For part 5, the edges of the cup in IMAGE7 were found using the Canny edge operator. First, the image was read with imread. The edges were detected using the edge function with the ‘Canny’ edge operation argument, threshold range [0.18, 0.5] and sigma 2. The original and edges were then displayed in figure 8 using figure, subplot\_tight and imshow.

**6.** Finally, for part 6, figure 10.25 of the book was reproduced using the steps outlined in the assignment. First, the image used in figure 10.25 was read using imread and convert to double using im2double. The first, upper left image was constructed by simply scaling the original image to the range [0, 1] using the following feature scaling equation:

The second, upper right image was constructed by first smoothing the image using a 10x10 Gaussian kernel with sigma of 5 via the fspecial and imfilter functions, then taking the gradient of the image using the imgradient function, and finally globally thresholding the image using the imbinarize function with ‘global’ argument. The third, lower left image was created using global thresholding via imbinarize on the output of the ‘log’ edge operator from the edge function with a 14x14 kernel and sigma of 0.46. The LoG (Laplacian of Gaussian) edge operation in Matlab is comparable to the Marr-Hildreth algorithm applied by the authors of figure 10.25. The fourth, lower right image was constructed using the ‘Canny’ edge operator from the edge function with a lower threshold of 0.07, upper threshold of 0.1, sigma of 4 and mask of 24x24, very close to the parameters as described in the book. The overall figure 10.25 replication results were then displayed in figure 9 using figure, subplot\_tight and imshow in a 2x2 grid. All figures were exported using export\_fig from the Matlab File Exchange (FEX).

1. **Discussion of Results**

**2. A. Results Comparison**

**1.** For part one, figures 1 and 2 were produced. In figure 1, the gradient and Laplacian filtered images were sharp. The gradient provided a good segmentation from the background. The Laplacian method was somewhat noisy around the wrench edges but produced highly detailed results. The global thresholding of the image provided a decent binary segmentation of the objects from the background. The zero-crossing detection with Laplacian filter required an intensity threshold of 0.6 to segment the objects similar to the global thresholding method, however, it included more pixels within the wrenches than the global thresholding method. After applying the 3x3 Gaussian filter, there was a decrease in brightness and sharpness in the image results overall. Much of the noise that required high thresholds in the edge detectors was eliminated by the filter, which allowed figure 2 to be produced with half the zero-crossing threshold (0.3) of that applied in figure 1. The detail in the zero-crossing image increased within the wrench while lowering the threshold, providing a suitable quality increase in the Laplacian segmentation after Gaussian filtering.

**2.** For part two, figures 3 and 4 were produced. The horizontal and vertical absolute gradients show notable edge details in both of the top right subplots, but the edges have weaker intensity in the bright dart image (IMAGE3) than in the darker and more detailed Lena image. The Sobel horizontal detector produced edges as if a camera scanned from top to bottom of the image, as in the bottom left subplots, whereas the Sobel vertical detector produced edges as if a camera had scanned from left to right of the image. For instance, in Lena, the horizontal detector found more interest in the feathers and background, connecting most of the feather edges together, but her face and the front of her hat were dark. With the vertical Sobel detector, her silhouette was better defined, with a quite continuous edge defining her outline but hardly including the background. In the dart figure, both the horizontal and vertical Sobel detectors perform well, with horizontal catching the edges in darker areas and vertical catching the edges in lighter areas.

**3.** The edges of the connecting rod in IMAGE 4 were perfectly detected after parameter tuning, as seen in figure 5. The tuned hysteresisthreshold range was [0.25, 0.4]. The edges were all well-connected and had no gaps with this tuned range. The bright setting of the image clearly worked well with the Canny edge operator and this tuned range. Evidently the Canny operator works well when the object(s) in the image are visually raised and distinguished from the other object(s), if any.

**4.** Global thresholding worked well to segment the wrenches in IMAGE1 as displayed in figure 6. The wrenches had higher intensity values than the surrounding background, but the pixels did not overlap as there was a clear distinguishment between each of the wrenches in the image. The segmentation of characters in IMAGE5 is shown in figure 7, but this is simply just an echo of the original image contents since the characters are in fact the edges given the black on white background. This echoed result was not surprising given that each character is a line, i.e. edge.

**5.** In order to determine the edges of the cup, many edge operators and custom configurations of those operators were tried. Horizontal, vertical and bi-directional detector variants of Canny, Sobel, Prewitt, Roberts and Laplacian were examined. Summations and differences of these operators were taken along with gradients and thresholds. Given the similarity in the background wallpaper and the color of the cup, the edge detectors in general were confused in distinguishing the cup from the background given the confusing and intricate nature of the wallpaper design. The design can easily flow into the edges of the cup under detection using most of the algorithms outlined. After numerous experiments, a simple Canny operator was applied with a honed threshold range of [0.18, 5] and a sigma of 2. The results of this operation are shown in figure 8.

**6.** Reproducing figure 10.25 from the book was successful, but with alternate parameters than those provided in the book. After a while of tuning each of the algorithms to produce similar subplots, figure 9 was generated. Each of the detectors had a different thickness and detail in edges. In order to reproduce the top-right subplot, a Gaussian kernel with size 10x10 and a standard deviation of 5 was applied. This kernel size seemed to provide the desired thicker edges of figure 10.25 with coarse detail when coupled with the larger-than-average sigma and global thresholding. The bottom-left image was rendered with a Laplacian of Gaussian kernel of size 14 and a 0.46 standard deviation, which had to be manually determined using visual comparison in results. The bottom-right image was initially generated using the book-specified parameters, but the lower threshold was raised to 0.07 from 0.04 because the results were more erroneous at the lower value.

**2. B. Question and Answer**

* Compare the edge operators to each other.

There are three classes of edge operators mentioned and used in this assignment. The first operators are the classical operators like Sobel and Prewitt. These classical approaches have the advantages of being simple in design and being able to detect edges and edge orientations. However, these methods also have downfalls such as being sensitive to noise and inaccurate while detecting edges in highly detailed scenarios. The second class of operators are those using zero-crossing, second directional derivative methods. The advantages to these approaches are that they can detect edges and edge orientation as well as have fixed characteristics in all directions. The drawbacks to these approaches are that they tend to respond to already existing edges and are more sensitive to noise than the classical approaches.

There is a third class of edge operators, the Gaussian/Canny operators. This class has a different design pattern than the previous classes. The advantages are that this class uses probability to determine error rate, localization and response of edge pixels via various techniques like non-maximal suppression. These methods improve upon signal to noise ratio and provide better detection in noisier conditions. These methods are more complex to implement, they may result in false zero-crossing and they consume more time than the other methods.

* What is the rotational symmetry of each operator?

The operators utilizing the Gaussian derivative are rotationally invariant and thus, rotationally symmetric. The gradient and Laplacian thresholding techniques for edge detection are thus rotationally symmetric. The Prewitt operator is also rotationally symmetric. The Sobel operator, however, does not have complete rotational symmetry given its non-symmetric kernel. The Canny operator has various implementations; some of which are rotationally symmetric. Specifically, the regular, recursive implementation of the Canny algorithm biases vertical and horizontal edges, so it is not rotationally symmetric. The global thresholding operation, i.e. Otsu’s method, is also rotationally symmetric given its use of a clustering-based approach to optimally threshold the image.

* What is the effect of Gaussian or other filtering on the edge image?

Gaussian or other filtering squashes the noisy pixel values in the input image to allow better edge detection. On the edge image, filtering thus allows the edge detector to detect the real edges of the objects rather than be confused by noise. The edge image will also have smoother edges provided pre-filtering before detection. Given that the edge operators take at least the first derivative of the input pixel values in order to detect the gradient of pixels, a differentiation noise is introduced by every derivative taken on the values. Given that operators like the Laplacian of Gaussian (LoG) utilize the second derivative to determine edges, much noise is amplified by the LoG operator itself, making it harder to detect edges in the presence of strong noise. Hence, filtering on the original image before applying an edge operator will provide a better edge operator performance in general and greatly enhance the quality of second-order edge operator edges.

* How sensitive is the actual value of the threshold for segmentation?

The actual values for segmentation thresholding become more sensitive as more objects and/or noise are present in an image. For instance, the first order derivative edge detection methods are very sensitive to noise and produce thicker edges. In example, the Prewitt operator uses the maximum directional gradient that when compared to Sobel, is much more sensitive to noise. The second order derivative edge detection methods are more advanced for edge localization but still very noise sensitive. Given that differentiation increases noise, smoothing the image is typically necessary prior to applying a Laplacian filter like the Laplacian of Gaussian (LoG). For the Canny operation, thresholding is generally hard given that it entails thresholding hysteresis, which requires both a lower and upper threshold value that can essentially double the intricacy, and thus sensitivity, of tuning the parameters. With the Canny operation, any pixels with a gradient above the upper threshold are considered edge points. Any pixels above the lower threshold that are connected to those valid edge points are also classified as edge points. Given that the Canny operation thus tries to connect the peaks and valleys of the edges within the specified threshold range, the dual threshold values can be easily confused if there are many similar and intricate details or lighting deficiencies in the image.

* How can you measure the success of an edge finding algorithm?

The success of an edge finding algorithm can be measured by applying the Pratt Figure of Merit (PFOM) algorithm between both the ideal edge image and the actual edge image. The PFOM algorithm provides a quantitative comparison between the actual and the ideal edge images using the distances between actual and ideal edge images. The ideal edge image can be generated, if not by hand, by applying the Canny edge operator on the original image given its proven success rate in the field. The POM metric is described further in appendix, section 5. A.

1. **Results**

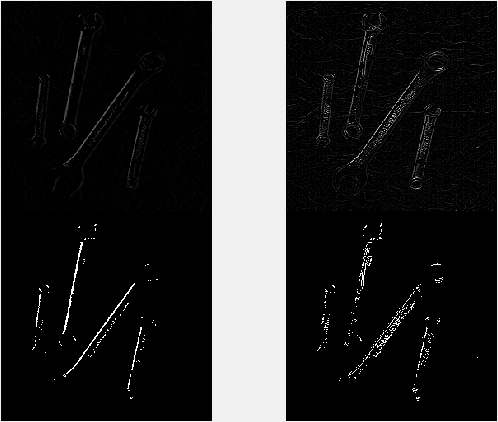
****

Figure : All of the following operations were performed on IMAGE1: Top-left: Gradient operation; Top-right: Laplacian Operation; Bottom-left: Edge Detection using Global Threshold; Bottom-right: Edge detection via zero-crossing detection.



Figure : All of the following operations were performed on a 3x3 Gaussian-filtered IMAGE1: Top-left: Gradient operation; Top-right: Laplacian Operation; Bottom-left: Edge Detection using Global Threshold; Bottom-right: Edge detection via zero-crossing detection.



Figure : All of the following operations were performed on the Lena image: Top-left: original image; Top-right: |Gx| + |Gy| gradient operation; Bottom-left: X-directed Sobel; Bottom-right: Y-directed Sobel.

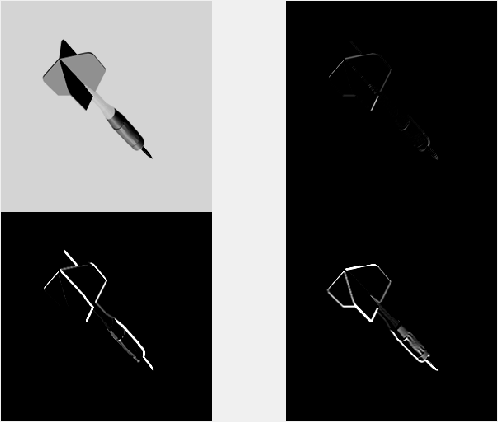


Figure : All of the following operations were performed on IMAGE3: Top-left: original image; Top-right: |Gx| + |Gy| gradient operation; Bottom-left: X-directed Sobel; Bottom-right: Y-directed Sobel.

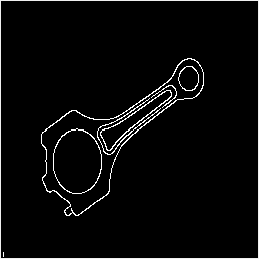


Figure : Canny edge operation performed on IMAGE4.

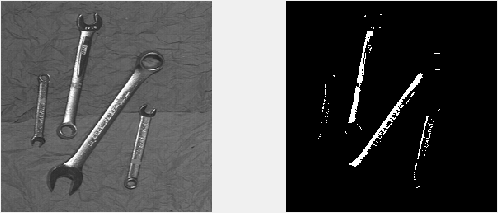


Figure : Segmented wrenches from IMAGE1 using Otsu's method of global thresholding.

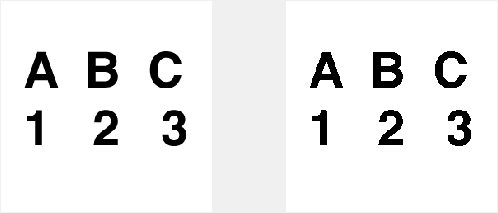


Figure : Segmented characters from IMAGE5 using Otsu's method of global thresholding.

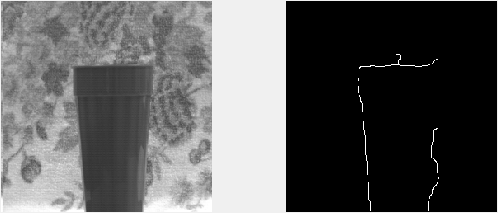


Figure : Left: original IMAGE7; Right: Detected edges of cup in IMAGE7 using Canny edge operator. The rendering of edges on the right does not quite represent the true output of the algorithm given compression upon output, but the entire left edge of the cup was detected. Any edges with gaps except for the top right side were also connected in the actual Matlab rendering.

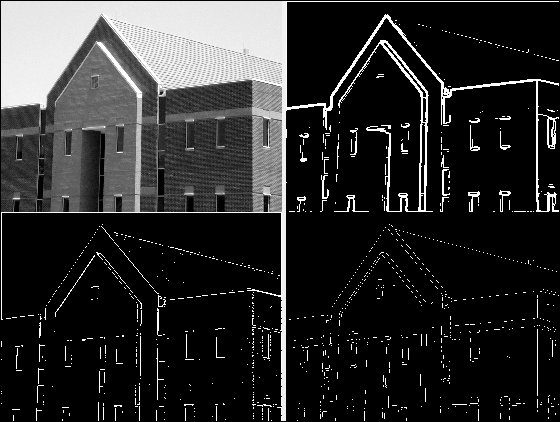


Figure : A reproduced figure 10.25 as presented in the assignment. Top-left: original image with intensity scaled inclusively to range [0, 1]; Top-right: Thresholded gradient of smoothed image; Bottom-left: Image obtained using globally thresholded Laplacian of Gaussian, a.k.a. Marr-Hildreth, algorithm; Bottom-left: Image obtained using Canny algorithm with Tl=0.07, Th=0.1, sigma=4, 24x24 mask. Edges in bottom images were distorted by Matlab image compression.

1. **References**

[1] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, 1979, pp. 62-66.

1. **Appendix**

**5. A. Pratt Figure of Merit**

The Pratt Figure of Merit (POM) is a metric designed to analyze and balance the errors associated with the edge detection process. The POM equation is the following:

where =max(, ) and and are the number of ideal and actual edge map points. d is the distance of separation between a line of ideal edge points and the normal of an actual edge point, while is the scaling factor. In Pratt’s work, is usually 1/9. In order to determine if an edge operator has detected the correct edges in a source image, the POM can be applied with the actual image as an edge-operator generated image and the ideal image as the edge image produced by a tuned Canny edge operator via the edge function with ‘Canny’ operator. The Canny operator would be chosen as the gold standard image given its proven success in the field and experiments.

**5. B. Program Listings**

The subplot\_tight function from the Matlab File Exchange (FEX) was applied to display images in grids without spaces. The export\_fig function, also from FEX, was used to export figure images. The program written for the project is included below:

**Project3.m**: the program to perform the edge detection steps for problems 1 through 6 of project 3.

% EECS 490 Project 3: Edges

% author: Shaun Howard (smh150@case.edu)

%% 1.a. perform gradient operation on IMAGE1

% convert image to double for higher precision

IMAGE1 = im2double(imread('pics/IMAGE1.tif'));

img1\_gradient = gradient(IMAGE1);

%% 1.b. apply laplacian operation to IMAGE1

H = fspecial('laplacian');

% apply laplacian filter.

img1\_laplace = imfilter(IMAGE1,H);

%% 1.c. locate the edges in both images using thresholding for (a) and

% zero-crossing detection for (b)

% thresholding for (a)

% binarize image using global image threshold with Otsu's method

img1\_grad\_thresh = imbinarize(img1\_gradient, 'global');

% zero-crossing detection with laplacian for (b)

zx\_thresh = .6;

img1\_laplace\_zeroxdetect = edge(IMAGE1,'zerocross',zx\_thresh, H);

%% display results for parts 1 a. - c.

margins=[0 0];

figure('name', 'Gradient and Laplacian with Thresholding and Zero-Crossing');

subplot\_tight(2,2,1,margins);

imshow(img1\_gradient);

subplot\_tight(2,2,2,margins);

imshow(img1\_laplace);

subplot\_tight(2,2,3,margins);

imshow(img1\_grad\_thresh);

subplot\_tight(2,2,4,margins);

imshow(img1\_laplace\_zeroxdetect);

export\_fig('output/figure1\_1.bmp');

%% 1.d. Repeat (a) - (c), but apply Gaussian 3x3 filter to image before doing

% anything

img1\_gauss = imgaussfilt(IMAGE1, 'FilterSize', [3 3]);

%% 1.d.a. perform gradient operation on Gauss filtered IMAGE1

img1\_gauss\_gradient = gradient(img1\_gauss);

%% 1.d.b. apply laplacian operation to Gauss filtered IMAGE1

img1\_gauss\_laplace = imfilter(img1\_gauss,H);

%% 1.d.c. locate the edges in both images using thresholding for (d.a) and

% zero-crossing detection for (d.b)

% thresholding for (d.a)

% binarize image using global image threshold with Otsu's method

img1\_gauss\_grad\_thresh = imbinarize(img1\_gauss\_gradient, 'global');

% zero-crossing detection with Laplacian for (d.b)

zx\_thresh = .3;

img1\_gauss\_laplace\_zeroxdetect = edge(img1\_gauss,'zerocross',zx\_thresh, H);

%% display results for parts 1 d.a. - d.c.

margins=[0 0];

figure('name', 'Filtered Gradient and Laplacian with Thresholding and Zero-Crossing');

subplot\_tight(2,2,1,margins);

imshow(img1\_gauss\_gradient);

subplot\_tight(2,2,2,margins);

imshow(img1\_gauss\_laplace);

subplot\_tight(2,2,3,margins);

imshow(img1\_gauss\_grad\_thresh);

subplot\_tight(2,2,4,margins);

imshow(img1\_gauss\_laplace\_zeroxdetect);

export\_fig('output/figure1\_2.bmp');

%% 2. Process the Lena (IMAGE2) and Dart (IMAGE3) images using the following

% operators:

IMAGE2 = im2double(imread('pics/Lena.tif'));

IMAGE3 = im2double(imread('pics/IMAGE3.tif'));

% 2.a. |Gx| + |Gy|, the x- and y-directed gradient operations

[img2\_grad\_x,img2\_grad\_y] = gradient(IMAGE2);

% sum |x| and |y| gradients

img2\_grad\_sum = abs(img2\_grad\_x) + abs(img2\_grad\_y);

[img3\_grad\_x,img3\_grad\_y] = gradient(IMAGE3);

% sum x and y gradients

img3\_grad\_sum = img3\_grad\_x + img3\_grad\_y;

% 2.b. X and Y-directed Sobel

[img2\_sobel\_x,img2\_sobel\_y] = imgradientxy(IMAGE2);

[img3\_sobel\_x,img3\_sobel\_y] = imgradientxy(IMAGE3);

%% display results for parts 2. a. and b. for IMAGE2

margins=[0 0];

figure('name', 'Absolute Gradient Addition and X- and Y-direction Sobel');

subplot\_tight(2,2,1,margins);

imshow(IMAGE2);

subplot\_tight(2,2,2,margins);

imshow(img2\_grad\_sum);

subplot\_tight(2,2,3,margins);

imshow(img2\_sobel\_x);

subplot\_tight(2,2,4,margins);

imshow(img2\_sobel\_y);

export\_fig('output/figure2\_1.bmp');

%% display results for parts 2. a. and b. for IMAGE3

margins=[0 0];

figure('name', 'Absolute Gradient Addition and X- and Y-direction Sobel');

subplot\_tight(2,2,1,margins);

imshow(IMAGE3);

subplot\_tight(2,2,2,margins);

imshow(img3\_grad\_sum);

subplot\_tight(2,2,3,margins);

imshow(img3\_sobel\_x);

subplot\_tight(2,2,4,margins);

imshow(img3\_sobel\_y);

export\_fig('output/figure2\_2.bmp');

%% 3. Use the Canny edge operator to locate the edges in the connecting rod

% image, IMAGE4.

IMAGE4 = imread('pics/IMAGE4.tif');

zx\_thresh = [0.25 0.4];

img4\_edges = edge(IMAGE4,'Canny', zx\_thresh);

figure;

imshow(img4\_edges);

export\_fig('output/figure3.bmp');

%% 4. Using global thresholding, segment the objects in the wrench image,

% IMAGE1, and the letters image, IMAGE5.

% binarize image using global image threshold with Otsu's method

img1\_thresh = imbinarize(IMAGE1, 'global');

% display figure comparison

figure('name', 'Wrenches segmented');

subplot\_tight(2,2,1,margins);

imshow(IMAGE1);

subplot\_tight(2,2,2,margins);

imshow(img1\_thresh);

export\_fig('output/figure4\_1.bmp');

% read image 5 and threshold

IMAGE5 = imread('pics/IMAGE5.tif');

% binarize image using global image threshold with Otsu's method

img5\_thresh = imbinarize(IMAGE5, 'global');

% display figure comparison

figure('name', 'Characters segmented');

subplot\_tight(2,2,1,margins);

imshow(IMAGE5);

subplot\_tight(2,2,2,margins);

imshow(img5\_thresh);

export\_fig('output/figure4\_2.bmp');

%% 5. Determine the edges of the cup in image IMAGE7 using any method you

% wish. Describe your algorithm.

IMAGE7 = imread('pics/IMAGE7.tif');

img7\_edges = edge(IMAGE7, 'Canny', [0.18 0.5], 2);

% display cup edges

figure('name', 'Cup Edges');

subplot\_tight(2,2,1,margins);

imshow(IMAGE7);

subplot\_tight(2,2,2,margins);

imshow(img7\_edges);

export\_fig('output/figure5.bmp');

%% 6. Attempt to duplicate as well as you can the results shown in Figure

% 10.25 of the textbook.

% load figure 10.25

fig1025 = im2double(imread('pics/Fig1025 original.tif'));

% 6.a. scale original image to range [0,1]

fig1025\_scaled = (fig1025 - min(fig1025(:))) / (max(fig1025(:)) - min(fig1025(:)));

% 6.b. Smooth image

H=fspecial('gauss', 10, 5);

fig1025\_gauss = imfilter(fig1025, H);

% perform gradient and threshold operation on Gauss filtered image

fig1025\_gauss\_grad\_thresh = imbinarize(imgradient(fig1025\_gauss), 'global');

% 6.c Apply Marr-Hildreth algorithm

% detect edges using the Laplacian of Gaussian ('log') method

H=fspecial('log', 14, .46);

fig1025\_edges = imbinarize(imfilter(fig1025, H));

% 6.d. Apply Canny algorithm, book says Tl=0.04, TH=0.1, sigma=4, mask=25x25

% In matlab, filterLength = 8\*ceil(sigma);

% mask with sigma = 4 will be ~24x24, very close to 25x25

fig1025\_canny\_edges = edge(fig1025,'Canny',[0.07 0.1],4);

%% display fig1025 results

figure('name', 'Figure 10.25 Replication');

subplot\_tight(2,2,1,margins);

imshow(fig1025\_scaled);

subplot\_tight(2,2,2,margins);

imshow(fig1025\_gauss\_grad\_thresh);

subplot\_tight(2,2,3,margins);

imshow(fig1025\_edges);

subplot\_tight(2,2,4,margins);

imshow(fig1025\_canny\_edges);

export\_fig('output/figure6.bmp');