Edges

Project 3

EECS 490

Shaun Howard

Date Due: October 8th

Date Submitted: October 8th

The world is delineated by edges. Many images of contain multiple edges to distinguish between objects depicted within the scene of the image. Determining the edges contained within an image is important in order to recognize the objects within the image. Matlab R2016a provides many tools for detecting the edges within a given image, including Prewitt, Sobel, Canny, and various manual detection utilities including gradient and Laplacian operations. Since many images are different and most images are filtered before they are processed for edges, these edge detection algorithms are robust to image differences given the tuning of several parameters. Each of these algorithms and the necessary parameters to accomplish various edge detection or object segmentation goals are discussed within this paper. The Matlab functions and algorithms used to determine the experiment results are outlined. The results of experiments are displayed and analyzed to determine the success of the edge operations performed. Subsequently, the results are compared amongst other comparable edge detection strategies across and between experiments.

1. **Technical Discussion**

For part 1, IMAGE1 was read in with imread. The 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 grayscale threshold function, graythresh. Zero-crossing detection was performed on the laplace-transformed image using the edge function with ‘zerocross’ option and a tuned threshold value discussed in section 2. The results were displayed with subplot\_tight and imshow in a 2x2 grid. The previous steps were then repeated after a Gaussian filtering of the image using the imgaussfilt method with a 2D kernel size of 3x3.

For part 2, the Lena and IMAGE3 images were read using imread. The row and column-wise numerical gradient operations were performed on both images. The 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 subplot\_tight and imshow.

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 ‘Canny’ operator argument. The edges of the figure were then displayed using figure and imshow.

For part 4, global thresholding was applied to two images in order to detect the objects within them. First, IMAGE1 was thresholded using the graythresh function to obtain the global grayscale intensity threshold image. Then, the image was converted to a binary image using the im2bw function since the goal is to figure out how well objects can be recognized using binary separation with thresholding. Second, IMAGE5 was read with imread and then thresholded similarly using the graythresh function. The output image was then similarly converted to a binary image using im2bw. Both results were then displayed in a 2x1 grid with their original using subplot\_tight and imshow.

For part 5, the edges of the cup in IMAGE7 were found using the Prewitt edge operation. First, the image was read with imread. The edges were detected using the edge function with the ‘Prewitt’ edge operation argument. The edges were then displayed using figure and imshow.

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. 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 3x3 Gaussian kernel via the imgaussfilt function with ‘FilterSize’ set to [3 3], then taking the gradient of the image using the gradient function, and finally thresholding the image using the graythresh function. The third, lower left image was created using the ‘log’ edge operator from the edge function. The log edge operation in Matlab is comparable to the Marr-Hildreth algorithm used in figure 10.25. The fourth, lower right image was constructed using the ‘Canny’ edge operator from the edge function with a lower threshold of 0.04, upper threshold of 0.1, sigma of 4 and mask of 5x5 as described in the book. The figure 10.25 replication results were then displayed using subplot\_tight and imshow in a 2x2 grid.

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 had detected the correct edges in a source image, the POM was 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 edge image was used as the ideal image given the proven success of the Canny operator in the field.

1. **Discussion of Results**

* 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 in 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 that 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. 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. 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 operations are also rotationally symmetric.

* 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. For example, the Prewitt operator uses the maximum directional gradient and when compared to Sobel, it 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) or Marr-Hildreth method. For the canny operation, thresholding can be sensitive as well given that the Canny algorithm entails thresholding hysteresis, which requires both a lower and upper threshold value. 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. Hence, the Canny operation is trying to connect the peaks and valleys of the edges within the specified threshold range. Hence, when using the Canny operator, the minimum and maximum threshold values can be sensitive if there are many intricate details in the image that may easily be confused as edge continuations.

* 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. 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. This comparison using the POM values the distance between the actual and ideal edges and penalizes edges that are displaced in the actual edge image using a tuned constant.

1. **Results**
2. **Appendix**

**4. A. Program Listings**

The subplot\_tight function from the Matlab File Exchange (FEX) was applied to display images in grids without spaces. 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.