<u>Computer Vision CAP 5415 – Fall 2017 – Programming Assignment I</u>

<u>Girish Kumar Kannan – UCF ID 4196719 – University of Central Florida</u>

Question 1 – Canny Edge Detection

- The three grayscale images chosen are included in the files submitted.
- The Canny Method was executed from Scratch as well as using the built-in Canny function to visually compare the outputs. Since the in-built function is more optimized, the method executed from scratch yielded good but not excellent (in-par) results when compared.
- Sigma (σ) plays a key role in getting the output. Sigma has to be relatively higher than normal assumptions of under 0 to 1. When Sigma increases edges are clearer and when Sigma decreases, the edges are bit noisy. The kernel selection also plays a vital role along with Sigma. Greater the kernel size, dustier/noisier the image or edges. When higher Sigma values are used, time taken also increases. Optimal but not the best Sigma is mentioned in the codes. Sigma less than 0 is still noisy while Sigma greater than 2 or 3 becomes featureless.

Question 2 – Entropy Thresholding

- The three grayscale images chosen are included in the files submitted.
- Threshold value is automatically selected using the algorithms designed.
- Two additional methods to obtain a binary image with good black and white balance is also written along with original method. These two methods are implemented based on image pixel intensity statistics like maximum, minimum, mean and standard deviation.

Question 3 – Corner Detection

- The provided three input images were used, also attached along with the files.
- Hessian Matrix method was implemented and threshold values are included in the code, corners plotted as an overlay over the grayscale image as red dots. A Box Blur filter was applied before the Hessian method was implemented.
- Harris Corner Detection method was implemented and threshold values are included in the code, corners plotted as an overlay over the grayscale image as red dots. As explained earlier, selection of filter size and Sigma value played a tricky yet vital role in getting the corner points. A Sigma of value greater than 2 was chosen. The near-optimal values are mentioned in the code. The two methods of Harris Corner Detection were applied and timed to check which performed better. The method with Determinant and Trace of Harris Matrix was faster (took less time) than the other method with Eigen values. Increasing Alpha (α) increases noise about the corner points – more the alpha, more the corner points, yet the range of alpha lies in the range 0.00 to 0.25. The results obtained from both methods were same but first method was efficient as it took comparatively less time than the second method.

Question 4 – SUSAN Corner Detection

- The SUSAN Corner Detection method was (successfully) implemented!
- The provided two images for SUSAN technique were used and was also tested on some more noisy and non-noisy images to test the performance. SUSAN method works fine on clear images, but not appreciably on noisy images but was still good enough.
- SUSAN method was implemented on susan_input1.png and near optimal values are mentioned in the code.
- SUSAN method was implemented on susan_input2.png and near optimal values are mentioned in the code. The image was not filtered, therefore corner detection was not as best as the clear image.
- SUSAN method was implemented on susan_input2.png and near optimal values are mentioned in the code. The image this time was denoised, smoothed and normalized before executing SUSAN. The corners so found was appreciably good when compared to worst case of three, the previous observation result. A set of filters were used to get the corners which does not include edge detecting filter as it is not mentioned in the assignment tasks.

Question 5 – Histogram Processing

- Since images were not provided nor instructed to retrieve from a webpage, few images from previously mentioned Berkeley Image Dataset Images were taken along with a few random images from internet image search.
- Histogram equalization method was implemented on all the attached images and images are shown. Histogram Equalization method was better than the below two methods as it yielded better output image with better contrast-brightness balance.
- Clipping algorithm was implemented using the given vales for ‘alpha’, ‘beta’ and ‘b’. Since the clipping removes a few black pixel gray levels and suppresses some white pixel gray levels, the output image so obtained was less dark and relatively brighter due to clipping of whites and removal of blacks. Since the mid-range was preserved and clipped in an increasing slope, the image retained a few features. In simpler words, the final image was less “smoky” and with better contrast-brightness balance.
- Range Clipping was implemented on the images with provided C values 1, 10, 100, 1000 and for 10,000 as well. C of 100 was found to be the optimum multiplier as other values of C, gave a bad output image. The value of C plays a vital role. Lesser the C, lesser the brightness of image and darker but when looked closer, the features are preserved. While, greater the C, image gets distorted and featureless but brightness is preserved. This is due to logarithmic scaling. With high multiplier, more black and white pixel values due to the nature of log curve, while low multiplier gives less black and white pixel values.

The programs written are not excessively yet sufficiently commented. All files provided and procured from internet that was used as image data are attached in the final submission file.