Assignment 4: Hough Transformation and the OpenCV Library

Igli Duro
Department of Computer Science & Engineering
University of South Florida
Tampa, FL

I. Introduction

OpenCV is a C++ library full of functionality for image processing and computer vision. In this lab we learned & utilized OpenCV to test and implement several image processing functions. Some of these processing functions were implemented manually in previous assignments and are built-in to the OpenCV API. We also compared the performance of some of OpenCV's processing functions to our own.

II. Description of Algorithms

In this section, an overview of the implemented algorithms are given and each algorithm will have its own subsection. The following algorithms in this section are: Histogram Stretching, Histogram Equalization, Hough transformations, Histogram Equalization + Sobel, and Histogram Equalization + Canny. Each of the algorithms use existing functions in the OpenCV library for image manipulation.

A. Histogram Stretching (OpenCV)

Built-in functionality to OpenCV, performs normalization on an input image to increase contrast based on a given min and max value.

B. Histogram Equalization (OpenCV)

Built-in functionality to OpenCV, performs equalization on an input image to increase contrast by attempting to equalize the pixel count of each intensity in an image.

C. Hough Transformation (OpenCV)

Using built-in functionality to OpenCV, performs edge detection and blurring on an input image along with Hough Circle detection to identify circles in an image.

D. Histogram Equalization + Sobel (OpenCV)

Using built-in functionality to OpenCv, performs equalization and Sobel edge detection on an image.

E. Histogram Equalization + Canny (OpenCV)

Using built-in functionality to OpenCv, performs equalization and Canny edge detection on an image.

III. Description of Implementation

For this assignment we were given starter code working with the OpenCV library. Preexisting functions were also provided. And the assignment as a whole utilizes existing functions in the OpenCV library. Each of the algorithms described were added to our utility class built off previous assignments. For the implementation of histogram stretching and histogram equalization we used existing functions in the form of normalize and equalizeHist which perform the operations for us. We also compared the histogram stretching algorithm in OpenCV to our own histogram stretching algorithm by measuring the time taken to complete them.

For the Hough Transform we began by using Canny edge detection on the input image and applied a Gaussian blur over the image. We then used the HoughCircles function to find circles in a grayscale image and drew the circles onto the output image.

We last combined operations by performing histogram equalization prior to doing Sobel or Canny edge detection. We then compared results of doing edge detection with and without histogram equalization.

IV. Description of and Analysis of Results

A. Description of Results

This section will go over the results of the algorithms implemented. We'll first be looking at the histogram equalization/stretching algorithm along with performance differences. Followed by the Hough Transformation and lastly our combined operations.

Figures[1-3] shows the results of using histogram stretching/equalization on the 'slope.jpg' file. We see that histogram equalization also brightened the image considerably in addition to increase the contrast. Figure[4] shows the elapsed time of running our own stretching vs OpenCV's stretching and equalization functions. It is seen that OpenCV's stretching function was overall roughly twice as fast, and the equalization algorithm being faster by a large margin. This result can be attributed to using look up tables to avoid repeated calculations.

In Figures[5-10] we see the results of performing Hough Circle Transform on the input images. In Figures[5-6] the input image used had distinct circles far apart from one another and the OpenCV algorithms were able to recognize them. In

Fig. 1. slope.jpg

Fig. 2. slope.jpg slope_CVHS.jpg opencv 1 0 0 800 534 histostretchcv

more complicated scenarios with overlapping figures, such as in Figures[7-10] the algorithm wasn't able to find all the balls. This seems to be largely dependent on much much of the outline of the ball was covered. Additionally objects further in the background of the image had trouble being detected as-well.

For Figures[11-15] we have the results of doing edge detection on its own along with doing histogram equalization beforehand. It can be seen that doing histogram equalization before hand results in greater edge detection. As more details were picked up, such as the grass in the background and the clouds.

V. Conclusion

For this assignment, we implemented a set of image processing algorithms using the OpenCV library. Some of these algorithms were previously implemented manually in past assignments and performance comparisons were done. In this case it was seen that OpenCV's functions were faster in completion time than our own when ran on the same image. We Additionally we gained the functionality to detect circles in an image using Hough Circle Transformation. We also

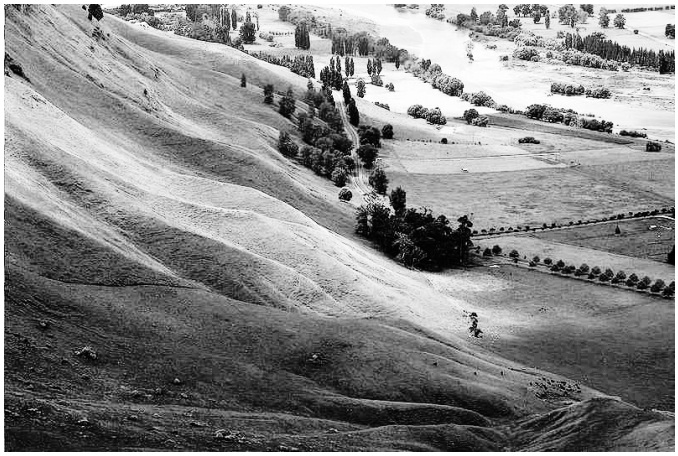

Fig. 3. slope.jpg slope_CVHE.jpg opencv 1 0 0 800 534 histoequalcv

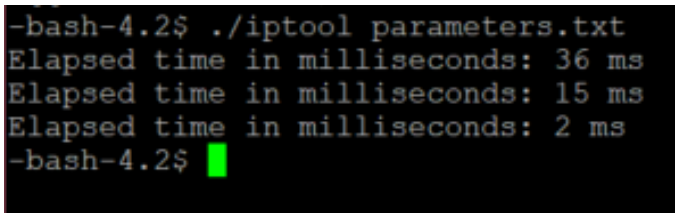

Fig. 4. Comparison of our own HS algorithm (top), OpenCV's normalize function (middle), and OpenCV's histogram equalization (bottom)

compared the results of running equalization on an image before doing edge detection to observe results.

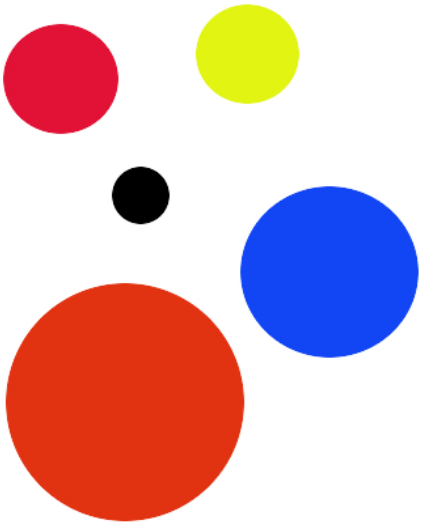

Fig. 5. circles.jpg

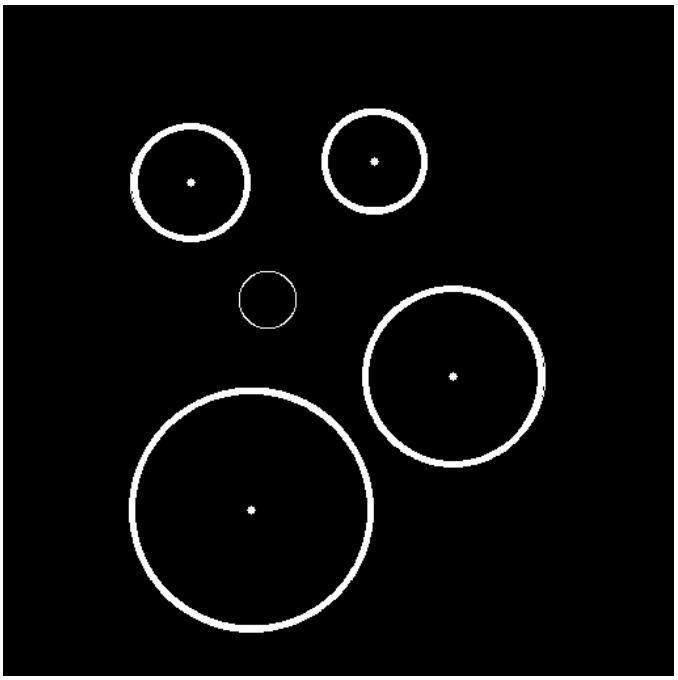

Fig. 6. circles.jpg circles_HT.jpg opencv 1 0 0 512 512 houghtrans

Fig. 7. ball.jpg

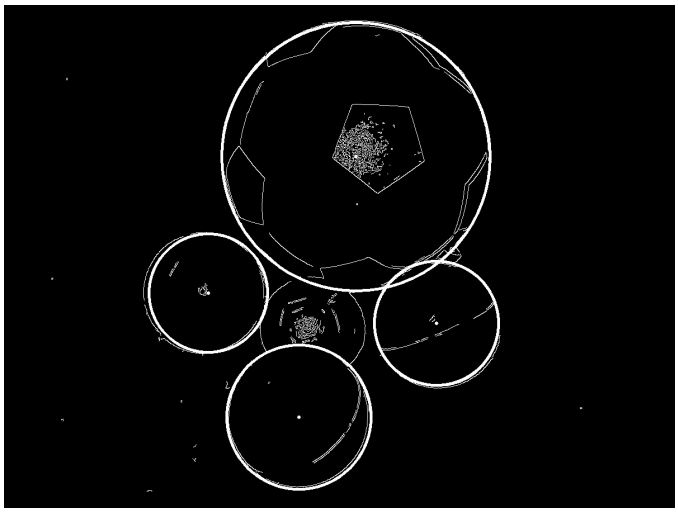

Fig. 8. ball.jpg ball_HT.jpg opencv 1 0 0 1200 900 houghtrans

Fig. 9. ball2.jpg

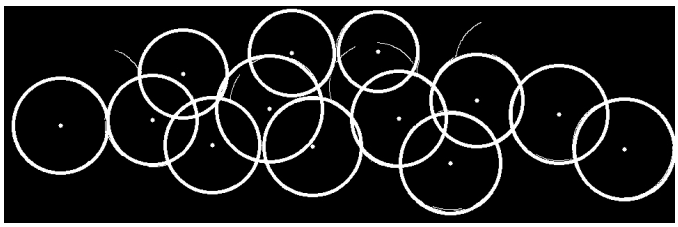

Fig. 10. ball2.jpg ball2_HT.jpg opencv 1 0 0 964 311 houghtrans

Fig. 11. fence.jpg

Fig. 12. fence.jpg fence_HEcanny.jpg opencv 1 0 0 1200 800 hecanny

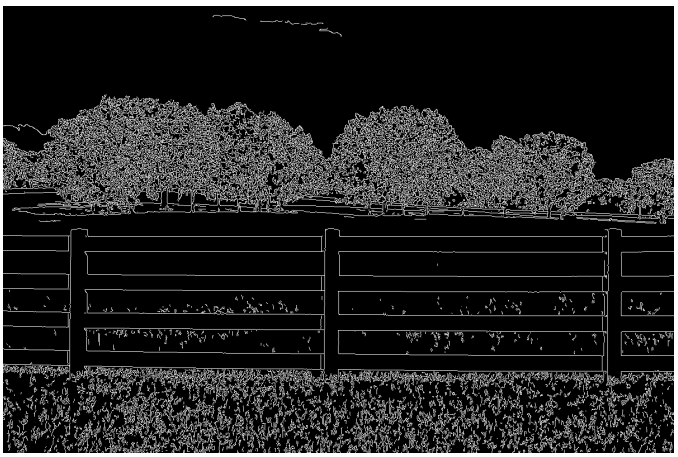

Fig. 13. fence.jpg fence_canny.jpg opencv 1 0 0 1200 800 cannycv

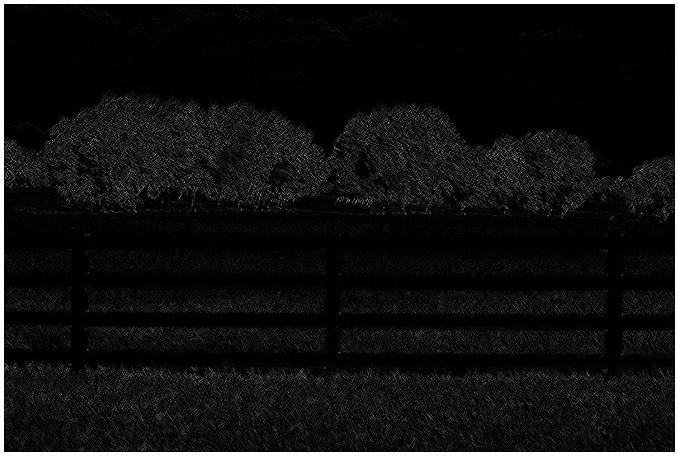

Fig. 14. fence.jpg fence_HEsobel.jpg opencv 1 0 0 1200 800 hesobel

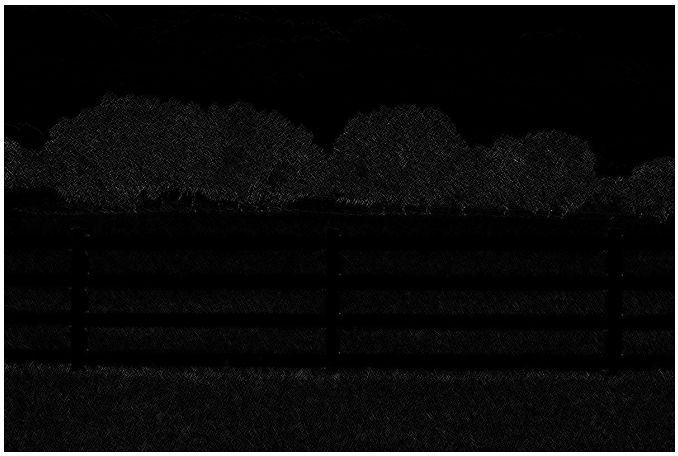

Fig. 15. fence.jpg fence_sobel.jpg opencv 1 0 0 1200 800 sobelcv