



Problem Set 2 Report

İsmail Can Yağmur

April 2023

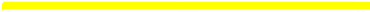

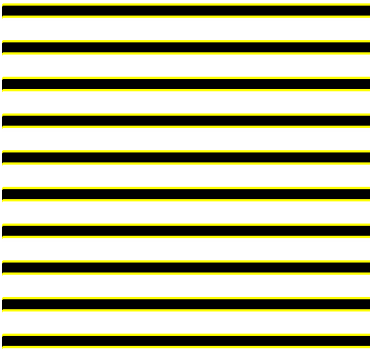
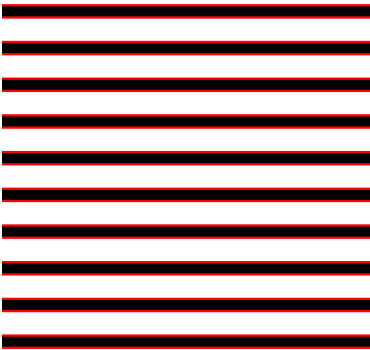
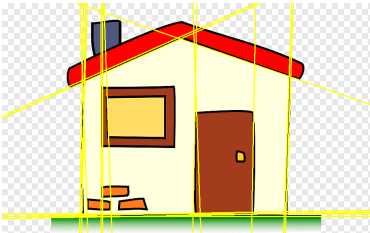
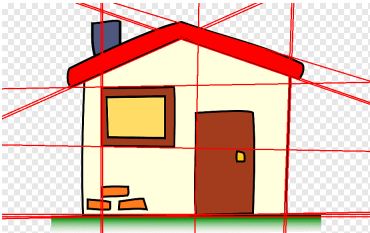
1 Problem 1

The first script is designed to perform Hough Transform on an input image to detect lines. The Hough Transform is a well-known computer vision technique used to detect geometric shapes in an image. The script loads an image using the OpenCV library, and then creates an instance of the HoughTransform class. The HoughTransform class provides two methods for computing the accumulator matrix, which is used to detect lines in the image. The script uses the vectorized version of the Hough Transform algorithm to compute the accumulator matrix, which is much faster than the traditional method.

After computing the accumulator matrix, the script uses the `hough_peaks_vectorized` method to find the peaks in the accumulator matrix. These peaks correspond to the lines detected in the image. The script then uses the `hough_lines_draw` method to draw the lines on the input image.

The script also displays the original input image, the edge image, and the image with detected lines. It uses the OpenCV `imshow` method to display these images. Finally, the script waits for a key press event and then closes all the windows. Overall, this script provides an efficient and accurate way of detecting lines in an image using the Hough Transform algorithm.

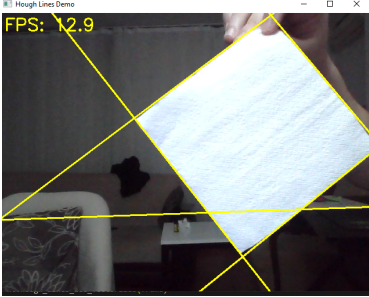
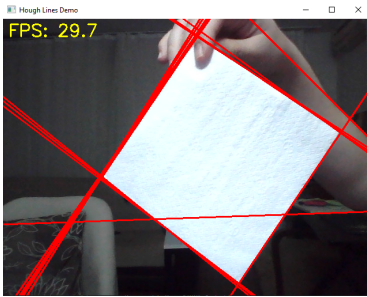
Table 1: Comparison of line detection implementations with threshold=150

My Implementation	OpenCV Implementation
	
(a) Author's Single Line	(b) OpenCV's Single Line
	
(c) Author's Horizontal Lines	(d) OpenCV's Horizontal Lines
	
(e) Author's House Detection	(f) OpenCV's House

2 Problem 2

The second script uses the Hough transform to detect and draw lines in real-time video streams. The program uses OpenCV library to capture frames from a video device and apply the Hough transform to identify lines in the image. The HoughTransform class is initialized with the use of dilation and erosion techniques to improve edge detection. The program then calculates the accumulator matrix and uses it to find the peaks in the matrix corresponding to the strongest lines in the image. The program then draws lines on the original image that correspond to the peaks. Additionally, the program tracks the frames per second (FPS) of the video stream and displays it as an overlay on the output image. The program runs in a continuous loop until the user presses the 'q' key to quit the program. This project can be used in various real-world applications, such as lane detection in self-driving cars and tracking the movement of objects in security footage.

Table 2: Comparison of line detection implementations with threshold=100

My Implementation	OpenCV Implementation
 <p>(a) Author's Video Frame</p>	 <p>(b) OpenCV's Video Frame</p>

As can be seen from Table 2, my implementation runs slower than OpenCV's implementation even though I vectorized all functions that I used to predict lines real-time. This could be related to better and more sophisticated optimization methods that are used by OpenCV. Notably, I also applied NMS to peaks that are found to overcome the issue of multiple local maxima. This problem can be observed from OpenCV implementation which doesn't support built-in NMS.