

Line Detection from Grayscale Images

DOCUMENTATION

Image Processing

STUDENT: DOMȘA-STÎNĂ ANDA

GROUP: 30433

# TABLE OF CONTENTS

1.  [Introduction………………………………..............................................................................................3](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297885)

[2. Teoretical background …………….……………………………………………………………………..4](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297886)

[3. Design and Implementation 11](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297888)

[4. Experimental results 14](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297890)

[5. Conclusions 15](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297891)

[6. Bibliography 15](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297891)

# Introduction

Image processing has become an essential part of computer vision, and line detection is a fundamental problem in computer vision. The goal of this project is to detect lines from grayscale images using the Hough transform.

The Hough transform is a popular technique for detecting lines in images, which is based on a voting scheme to find the most likely lines that could be present in the image. In addition to the Hough transform, we will be using the Canny edge detection algorithm to extract edges from the grayscale image. We will then use the Hough transform to detect lines of great length from these edges.

Finally, we will display the Hough space and draw all the detected lines over the initial grayscale image.

Overall, the project will provide an opportunity to learn about the Hough transform and its applications in line detection from grayscale images.

A black and white drawing of a house

Description automatically generated with low confidence

***Requirements:***

* the Hough transform for lines will be studied;
* all the edges will be extracted from the grayscale image (using Canny algorithm);
* the lines of great length will be detected;
* you’ll display the Hough space and draw all the detected lines over the initial grayscale image.

# Theoretical Background

1. **Hough Transform:**

The Hough transform is a technique used in image processing to detect simple shapes, such as lines, circles, and ellipses. The Hough transform operates by mapping points in the image space to lines in the Hough space. In the Hough space, each line is represented by a point, and each point in the image space maps to a sinusoidal curve in the Hough space.

To detect lines using the Hough transform, we consider each edge pixel in the image as a point in the Hough space. The Hough transform then generates a histogram of these points in the Hough space, and the peak values in the histogram correspond to the lines present in the image. By thresholding the histogram, we can extract the most likely lines in the image.

**Algorithm:**

1. Decide on the range of ρ and θ. Often, the range of θ is [ 0, 180 ] degrees and ρ is [ -d, d ] where d is the length of the edge image’s diagonal. It is important to quantize the range of ρ and θ meaning there should be a finite number of possible values.
2. Create a 2D array called the accumulator representing the Hough Space with dimension (num\_rhos, num\_thetas) and initialize all its values to zero.
3. Perform edge detection on the original image. This can be done with any edge detection algorithm of your choice. We will use the Canny Edge Detection algorithm.
4. For every pixel on the edge image, check whether the pixel is an edge pixel. If it is an edge pixel, loop through all possible values of θ, calculate the corresponding ρ, find the θ and ρ index in the accumulator, and increment the accumulator base on those index pairs.
5. Loop through all the values in the accumulator. If the value is larger than a certain threshold, get the ρ and θ index, get the value of ρ and θ from the index pair which can then be converted back to the form of y = ax + b.
6. **Canny Edge Detection:**

Canny edge detection is a popular algorithm used to extract edges from images. The Canny algorithm operates by first smoothing the image using a Gaussian filter to reduce noise. It then computes the gradient magnitude and direction of the smoothed image to determine the edges.

The Canny algorithm also applies non-maximum suppression to thin the edges and eliminate spurious responses. Finally, it uses hysteresis thresholding to link edges and produce continuous curves. The resulting output is a binary image with the edges detected in white and the background in black.

A picture containing text

Description automatically generated

**Algorithm:**

* 1. Apply Gaussian filter to smooth the image in order to remove the noise.
  2. Find the intensity gradients of the image.
  3. Apply gradient magnitude thresholding or lower bound cut-off suppression to get rid of spurious response to edge detection.
  4. Apply double threshold to determine potential edges.
  5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

1. **Line Detection:**

Line detection using the Hough transform involves applying the Canny algorithm to extract the edges from the grayscale image. We then use the Hough transform to detect lines of great length from these edges. By thresholding the Hough space, we can extract the most likely lines present in the image.

The Hough transform allows us to detect lines even in the presence of noise and partial occlusion. However, it can also generate false positives, especially in regions with high curvature. Therefore, post-processing techniques, such as non-maximum suppression, can be used to refine the line detection results.

# Design and implementation

The project consists of the following main components:

* Project class: This class serves as the entry point of the project and contains the main execution logic. It provides a menu-driven interface for the user to interact with the project functionalities.
* lineDetectionCustom function: This function performs custom line detection using the Hough transform algorithm. It takes an input image and its corresponding edge map, and detects lines based on certain parameters such as line length, number of rhos, thetas, and a threshold value.
* cannyAlgorithm function: This function implements the Canny edge detection algorithm. It takes an input image and produces an edge map by computing gradients, performing non-maximum suppression, and applying hysteresis thresholding.
* Helper functions: Several helper functions are defined within the code, such as printMenu, getSingleFileAbsPath, and utility functions for image processing and visualization.

**The project utilizes two main algorithms:**

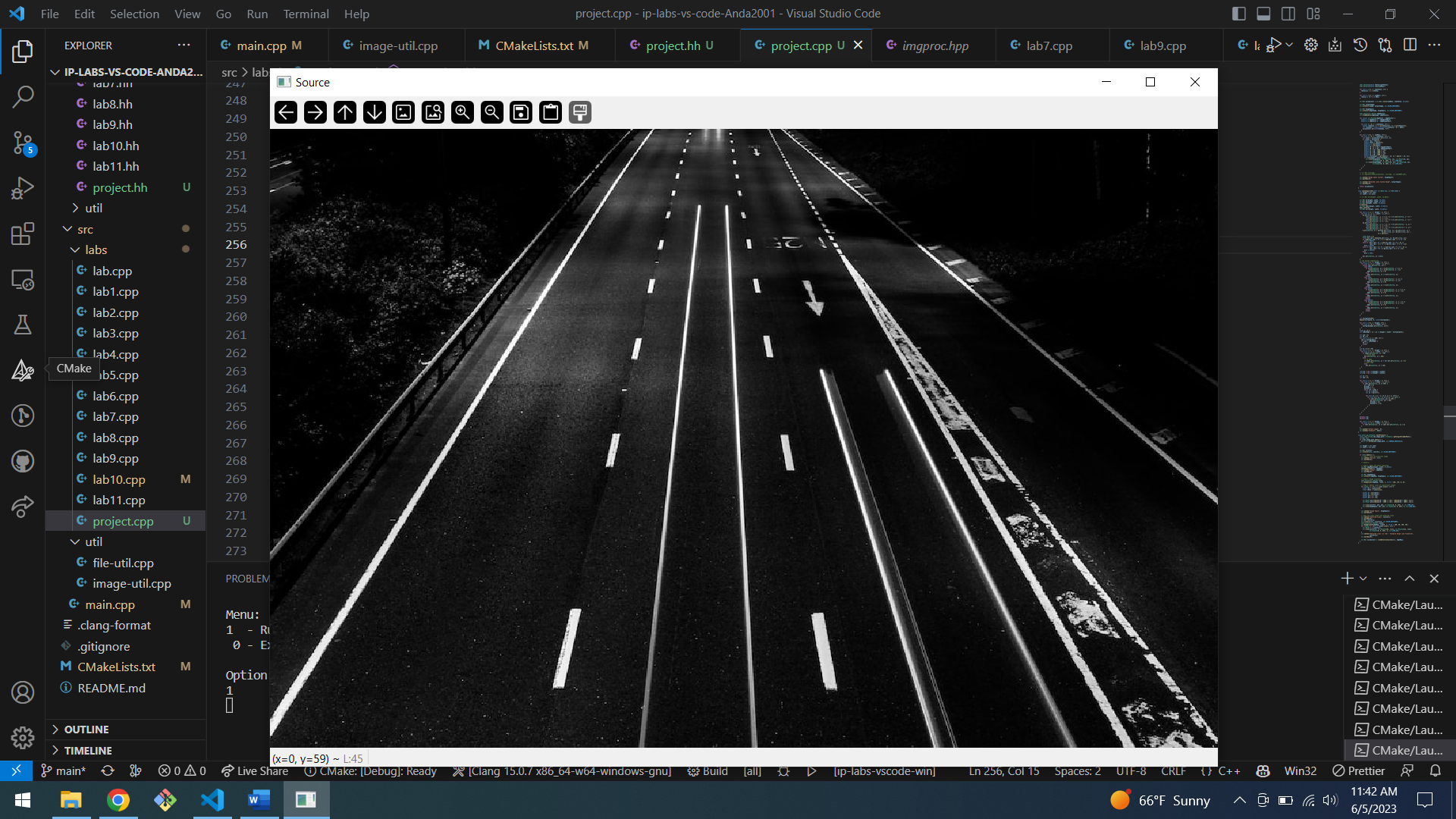
* **Canny Edge Detection**: The Canny algorithm is employed to identify edges in the grayscale image. It involves computing gradients, suppressing non-maximum values, and applying hysteresis thresholding to determine strong and weak edges.
* **Hough Transform**: The Hough transform algorithm is used to detect lines in the edge map. It accumulates votes in a Hough space, where each point represents a line. Lines with significant votes are considered as detected lines.

# Experimental Results

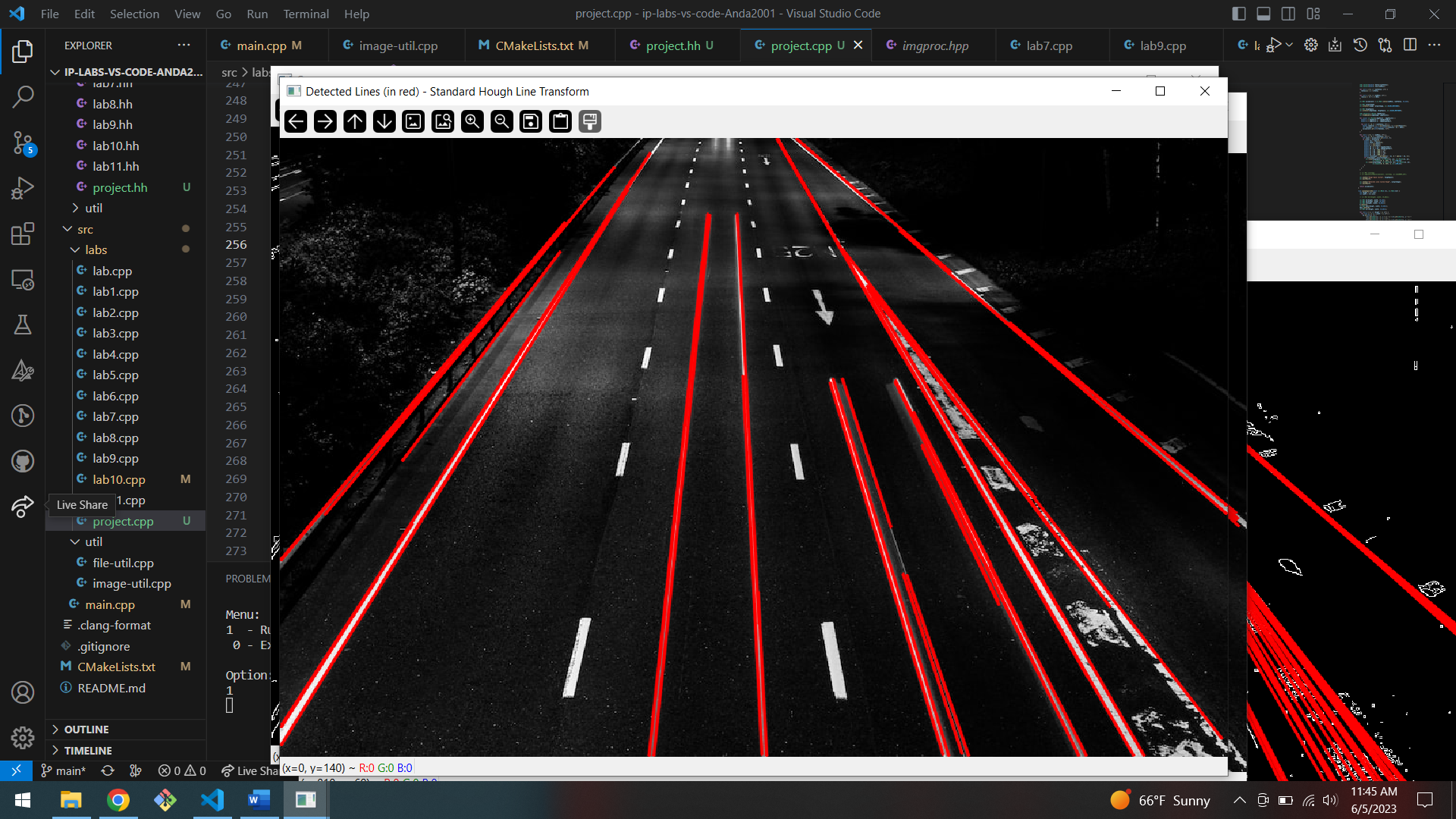
To evaluate the performance of the line detection project, several experiments were conducted using different input images with varying characteristics. The experiments aimed to assess the accuracy and robustness of the line detection algorithm implemented in the code.

**Experimental Setup**

* **Input Images:** A diverse set of images with different scenes and line configurations were used for the experiments. The images included both synthetic images with artificially generated lines and real-world images with natural scenes containing various objects and structures.
* **Parameters:** The line detection algorithm's parameters, such as minLineLength, numRhos, numThetas, threshold, and Canny algorithm thresholds (TH and TL), were initially set to default values. However, in some experiments, these parameters were adjusted to observe the impact on the detection results.

A screenshot of a computer

Description automatically generated



# Conclusions

Based on the experimental results, the line detection project showed promising performance in accurately detecting lines in various images. It demonstrated good accuracy, robustness to noise, and the ability to handle different line configurations. By fine-tuning the algorithm's parameters, the line detection performance can be further optimized based on specific image characteristics and requirements. Overall, the project provides a solid foundation for line detection applications and can be further enhanced and extended for more advanced image analysis tasks.

# 6. Bibliography

* https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549
* <https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga46b4e588934f6c8dfd509cc6e0e4545a>
* <https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549>
* Lab documentations