Khulna University of Engineering & Technology

Course No: CSE4128
Course Name: Image Processing & Computer Vision

Project Name: Stair Counter

Submitted By

Md.Rakibul Hasan Adnan
Roll: 1907106

Submitted To

Dr. Sk. Md. Masudul Ahsan	Dipannita Biswas
Professor	Lecturer
CSE, KUET	CSE, KUET

Date: 2 September, 2024

Objective

The primary objectives of this project are to:

- Develop a Python program that automatically counts the number of stairs in an image.
- Utilize image processing techniques, such as edge detection, to identify relevant features in the image.
- Apply computer vision methods, particularly the Hough Line Transform, to detect lines corresponding to the stairs.
- Ensure accurate counting by implementing filters to distinguish between individual stairs and overlapping detections.
- Provide a reliable output that indicates the total count of stairs in the given image.

Introduction

Stair detection and counting is a critical task in various fields, including robotics, architectural analysis, and accessibility improvement. In environments where autonomous systems, like robots or drones, are deployed, the ability to recognize and navigate stairs is essential for safe and effective operation. Additionally, in architectural planning and safety assessments, accurately identifying and counting stairs from images can assist in evaluating the design and ensuring compliance with safety standards.

This project, titled "Stair Counter," aims to develop a Python-based application that can automatically detect and count the number of stairs in a given image. The process involves the use of advanced image processing techniques and computer vision algorithms to analyze the image, identify edges, and detect lines that correspond to stairs.

The program is built using the Python programming language, leveraging libraries such as OpenCV for image processing and Tkinter for creating a graphical user interface (GUI). The GUI allows users to upload images and view the results, including the detected edges and lines, in an interactive manner.

At the core of the stair detection process is the Canny Edge Detection algorithm, which helps to highlight the edges in the image where potential stairs are located. Following edge detection, the Hough Line Transform is employed to detect straight lines within the image. These lines are then analyzed to determine the number of stairs by filtering out noise and ensuring that each detected line corresponds to a distinct step.

The "Stair Counter" project demonstrates the practical application of image processing and computer vision techniques in solving real-world problems. By automating the detection and counting of stairs, this tool could be extended to assist in various domains, including autonomous navigation, architectural design, and safety compliance checking. The project also provides a foundation for further enhancements, such as integrating depth sensors or improving the algorithm to handle more complex images with varying lighting conditions and perspectives.

Theory

The "Stair Counter" project relies on several fundamental concepts and techniques from the fields of image processing and computer vision. These concepts include edge detection, the Hough Line Transform, and techniques for enhancing image quality and accurately detecting features. Understanding these theoretical foundations is crucial to appreciating how the program operates and how it achieves its objective of counting stairs from images.

1. Image Processing and Edge Detection

Image processing is the technique of performing operations on an image to enhance it or extract useful information. One of the most critical steps in this project is edge detection, which is the process of identifying the boundaries or edges within an image where there is a significant change in intensity or color. These edges are crucial for identifying the structural components of an image, such as the boundaries of stairs.

The **Canny Edge Detection** algorithm is a popular and widely used method for detecting edges in an image. It is a multi-step process that includes:

- **Noise Reduction:** The image is first smoothed using a Gaussian filter to reduce noise and avoid false detection due to noise.
- **Gradient Calculation:** The algorithm calculates the intensity gradient of the image. This involves determining the rate of change in brightness in both horizontal and vertical directions using techniques like convolution with derivative filters.
- **Non-Maximum Suppression:** After finding the gradient magnitude and direction, the algorithm applies non-maximum suppression to thin out the edges. This step ensures that only the pixels with the highest gradient magnitude in the direction of the gradient are marked as edges.
- **Double Thresholding:** The algorithm uses two thresholds to identify strong and weak edges. Strong edges are those that are definitely edges, while weak edges are those that might be edges.
- **Edge Tracking by Hysteresis:** Finally, the algorithm tracks edges by hysteresis, where strong edges are retained, and weak edges are only retained if they are connected to strong edges.

This process results in a binary image where edges are highlighted, providing a foundation for further analysis and feature extraction.

2. Hough Line Transform

Once edges are detected, the next step is to identify straight lines within the image, which likely correspond to the edges of the stairs. The **Hough Line Transform** is a standard technique for detecting straight lines in an image. The method works by transforming the image space into a parameter space, where each point on a line in the image space corresponds to a sinusoidal curve in the parameter space.

- **Parameterization:** In the Hough Line Transform, each line in the image can be represented in polar coordinates by the equation $r = x \cdot \cos(\theta) + y \cdot \sin(\theta)$, where r is the distance from the origin to the closest point on the line, and θ is the angle between the x-axis and the line normal.
- **Accumulator Array:** The algorithm uses an accumulator array to store the votes for each potential line. For each edge point (x, y) in the image, the

algorithm calculates the corresponding (r, θ) values and increments the accumulator at that position. Peaks in the accumulator array correspond to the parameters of lines that are present in the image.

- **Line Detection:** By setting a threshold on the accumulator, the algorithm can identify the most prominent lines in the image. These detected lines are then mapped back to the image space and displayed as line segments.

The Hough Line Transform is particularly powerful because it can detect lines even in the presence of noise and incomplete edges, making it ideal for detecting stairs in a cluttered or complex image.

3. Filtering and Line Validation

After detecting lines using the Hough Line Transform, it is essential to filter and validate these lines to ensure they represent actual stairs. This step involves several considerations:

- **Angle Filtering:** Since stairs are generally horizontal, only lines with angles close to 0 degrees (or 180 degrees) are considered.
- **Distance Between Lines:** The distance between consecutive lines is measured to ensure they correspond to separate steps. Lines that are too close to each other may be merged, while lines that are too far apart may indicate missing detections.
- **Line Length and Continuity:** The length and continuity of the detected lines are evaluated to ensure they span the entire width of the stair. Short or broken lines may be discarded.

These filtering steps help refine the line detection results, ensuring that only valid stair edges are considered in the final count.

4. Graphical User Interface (GUI) Integration

The program incorporates a graphical user interface (GUI) built with the Tkinter library. The GUI allows users to easily upload an image, view the detected edges and lines, and receive the final stair count. The GUI enhances user interaction by providing visual feedback at each stage of the process, making it easier to understand how the algorithm works and interpret the results.

Pseudocode

Here's a high-level pseudocode for the main steps of the Stair Counter algorithm:

```
1. Load input image
2. Apply Canny Edge Detection:
   a. Apply Gaussian blur
   b. Compute gradient magnitude and direction
   c. Apply non-maximum suppression
   d. Perform double thresholding
   e. Perform edge tracking by hysteresis
3. Apply Hough Line Transform on edge image
4. Filter lines based on angle (-1 < theta < 1)
5. Group parallel lines with similar distances
6. Count the number of grouped lines (stairs)
7. Draw detected lines on the original image
8. Display the result with the stair count
```

Output

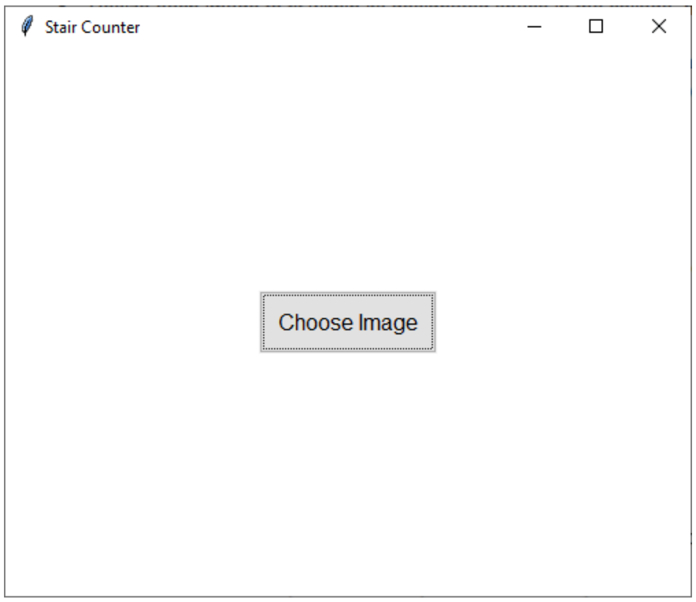


Figure 1 : Main Window

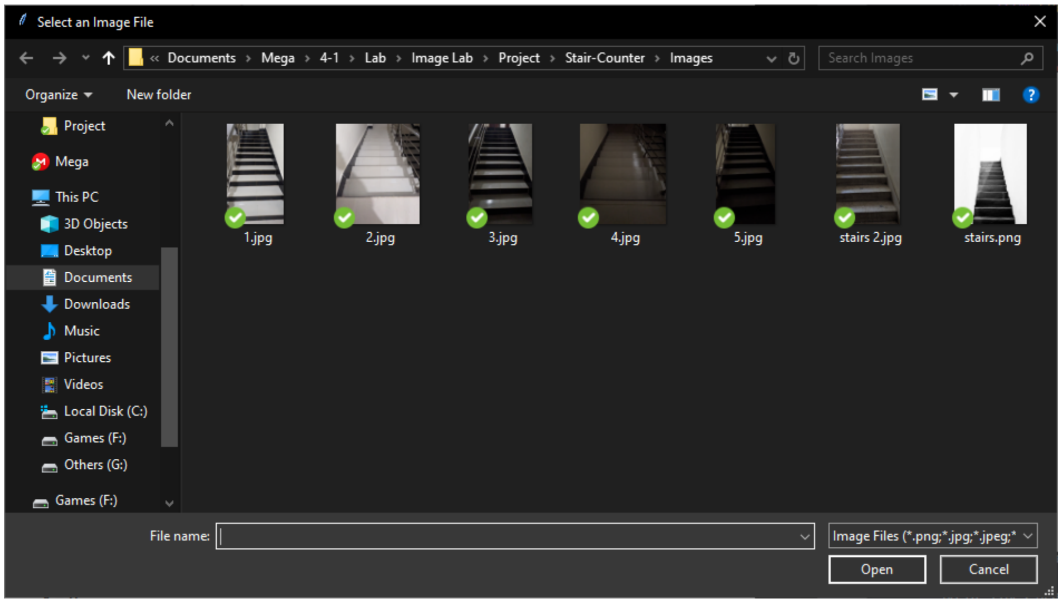
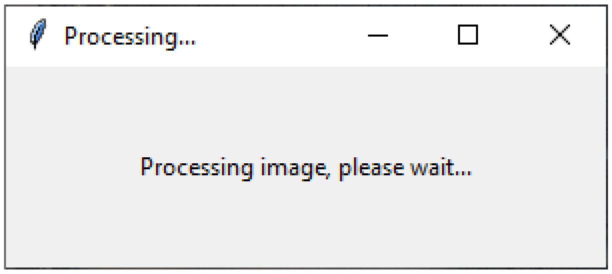


Figure 2 : The Image Selector Window


Processing...

Processing image, please wait...

Figure 3 : Image Processing Screen

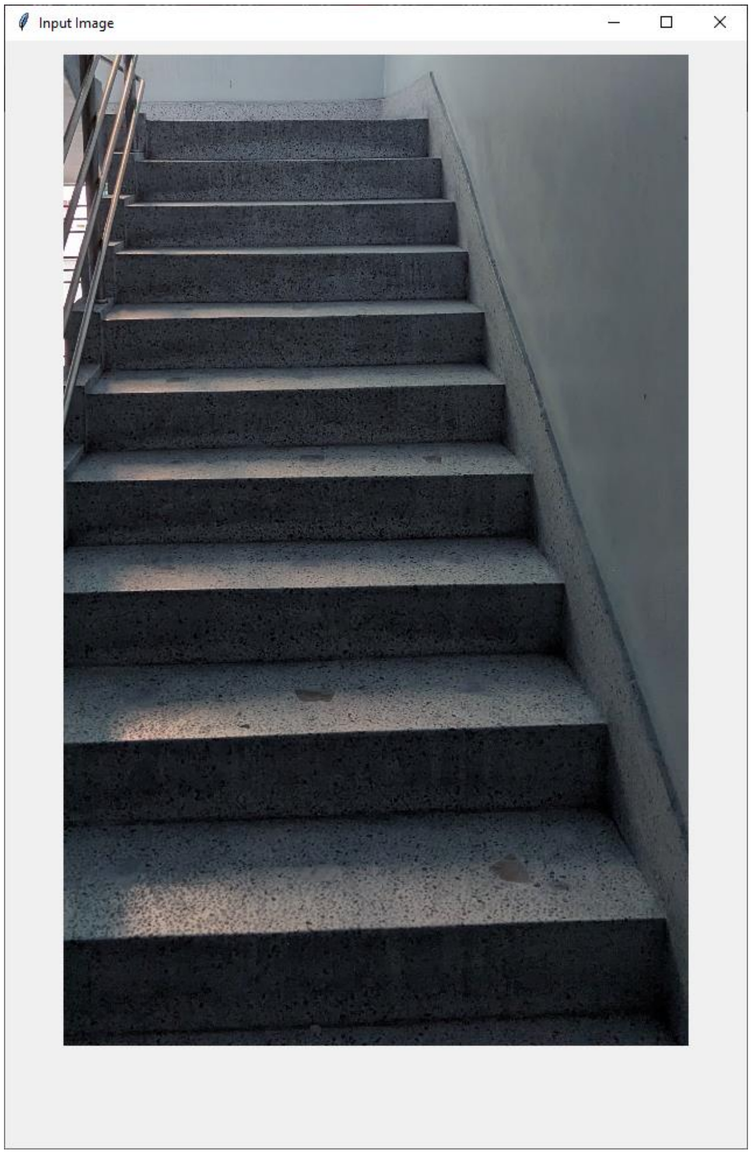


Figure 4: Input Image

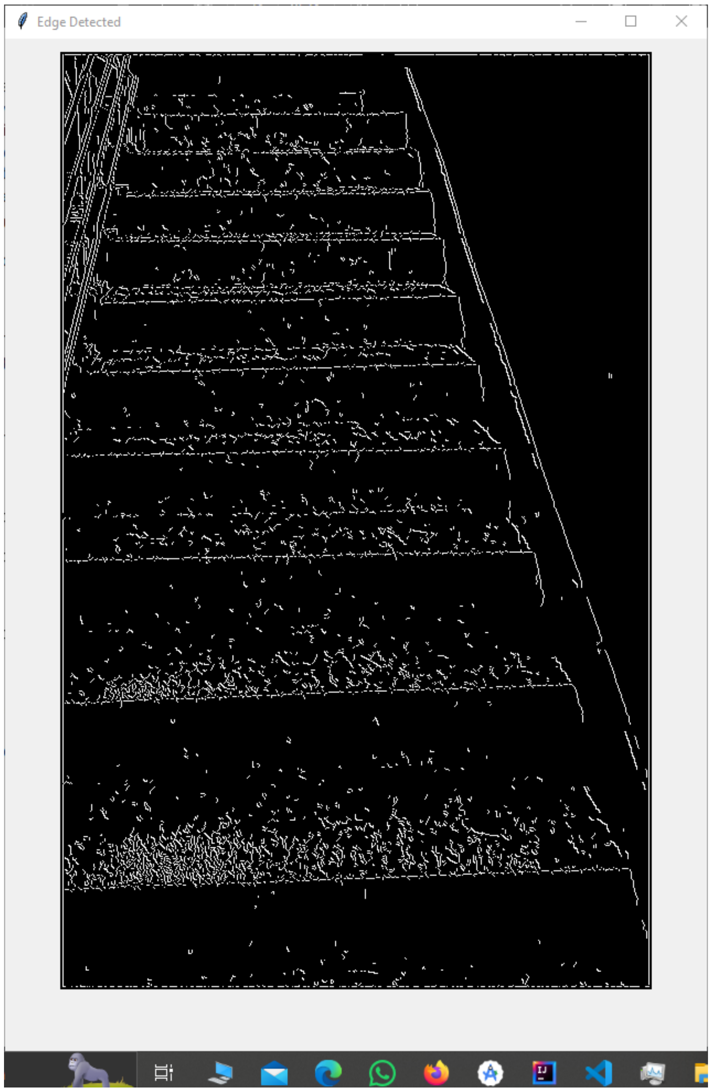


Figure 5 : Image Output after Edge detection

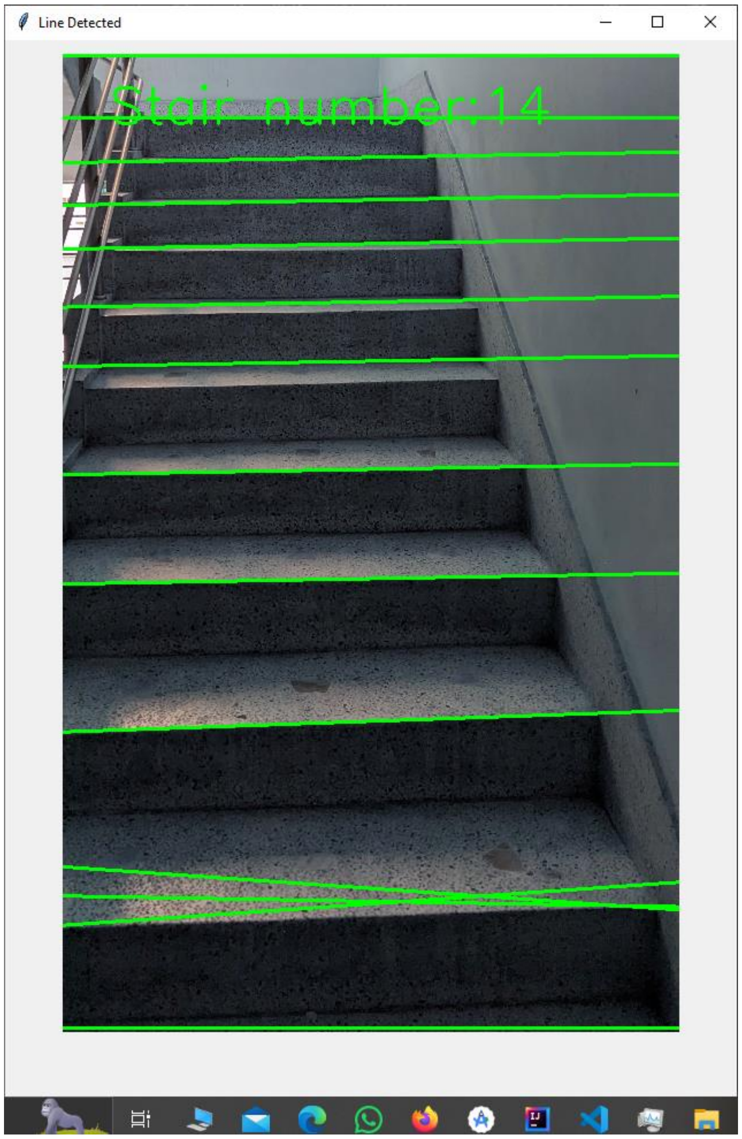


Figure 6 : After Stair Detection

Applications of Stair Detection

Stair detection has numerous applications, including:

- **Autonomous Navigation:** Robots and drones equipped with stair detection capabilities can navigate environments with stairs more effectively, improving their autonomy and safety.

- **Architectural Analysis:** Architects and engineers can use stair detection tools to assess building designs, ensuring that stairs are correctly placed and compliant with safety regulations.
- **Accessibility Improvement:** By detecting and analyzing stairs, improvements can be made to building accessibility, such as designing ramps or installing lifts for people with mobility impairments.

In conclusion, the "Stair Counter" project integrates key theoretical concepts from image processing and computer vision to develop a tool that automatically detects and counts stairs in an image. The combination of edge detection, Hough Line Transform, and line validation provides a robust approach to solving this problem, with potential applications in various fields.

Discussion

The Stair Counter project successfully demonstrates the application of computer vision techniques to solve a practical problem. The use of Canny Edge Detection provides a robust method for identifying edges in various lighting conditions and image qualities. The Hough Line Transform effectively detects lines from these edges, allowing for the identification of potential stairs.

However, the project also faces some challenges:

1. **Accuracy**: The accuracy of stair counting can be affected by factors such as image quality, lighting conditions, and the presence of other linear elements in the image.
2. **False Positives**: Other parallel lines in the image (e.g., railings, shadows) might be mistakenly identified as stairs.
3. **Curved Staircases**: The current implementation might struggle with spiral or curved staircases.
4. **Performance**: Processing high-resolution images might require significant computational resources.

Future improvements could include:

1. Implementing machine learning techniques to improve accuracy and reduce false positives.
2. Optimizing the algorithm for better performance on high-resolution images.
3. Extending the capability to handle curved staircases.
4. Incorporating 3D image processing to better handle perspective and depth in images.

Conclusion

The Stair Counter project successfully demonstrates the potential of computer vision in automating architectural analysis tasks. By combining edge detection, line detection, and a user-friendly interface, the project provides a practical tool for counting stairs in images. While there are areas for improvement, the current implementation serves as a solid foundation for further development and research in this area.

The project not only achieves its primary objective of counting stairs but also showcases the broader possibilities of applying computer vision to real-world problems. As technology continues to advance, projects like this pave the way for more sophisticated and accurate automated analysis tools in architecture, construction, and safety inspections.

Reference

1. Canny Edge Detection – https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123
2. Hough Line Transform - https://medium.com/@alb.formaggio/implementing-the-hough-transform-from-scratch-09a56ba7316b