Save Luna !

Lalit Karthikeyan M A

Abstract— **1.1 This project implements an edge detection algorithm using Canny edge detection and Hough transform to detect edges of a table in an image. The Sobel operator is utilized for gradient computation, followed by non-maximum suppression and double thresholding for edge refinement. Subsequently, the Hough transform is applied to identify lines representing the table's edges. Challenges include parameter optimization and line detection accuracy. This finds applications in computer vision, robotics, and object detection systems. Scalability extends to automated inspection, image-based navigation, and augmented reality. In robotics, it aids in environment perception and navigation. Automated inspection benefits from edge detection in quality control processes. Image-based navigation systems utilize edge information for path planning and obstacle avoidance.**

1.2 The algorithm aims to estimate depth from stereo images using the Sobel operator and disparity mapping. By computing gradients and performing window-based matching, the algorithm generates a depth map revealing the scene's spatial layout. Challenges involve balancing accuracy and computational efficiency, requiring careful parameter adjustment and iterative refinement. Lessons learned encompass the importance of parameter optimization and algorithmic efficiency in stereo vision tasks.This has applications in autonomous vehicles, augmented reality, and medical imaging. Scalability extends to precision agriculture, construction, and retail. In autonomous vehicles, depth perception is vital for object detection and navigation.

I. INTRODUCTION

1.1 To ensure the safety of a robot named Luna on a table, a process using computer vision techniques was implemented. Canny Edge Detection was first applied to identify all edges on the table, including its boundaries, and this data was saved as "edge.png." Subsequently, Hough Line Transformation was utilized on the edge-detected image to locate lines that correspond to the table edges. By selecting two optimal points from these lines, a precise line was drawn using the slope equation along the table's edge. This line effectively marks the boundary that Luna must not cross, preventing the robot from falling off the table.

1.2 Luna is a robot that uses two cameras to see objects and avoid crashing into them. These cameras, placed side by side, capture two images called left.png and right.png. By using a technique called stereo vision, similar to how human eyes work, Luna can measure how far objects are. First, a small area from the left image is chosen. Then, the similar area in the right image is found using template matching. The difference in position between these matching areas in the two images helps determine the distance of objects from Luna. This process is repeated for many points to create a depth map, which helps Luna understand the space around her and navigate safely.

II. PROBLEM STATEMENT

1.1 Luna, a robot, is placed on a table and needs to detect the edges of the table to prevent falling off. To achieve this, a specific process using computer vision techniques was utilized.

The initial step involves applying Canny Edge Detection on an image called table.png. This method detects the edges of the table through several steps:

Noise Reduction: The image is first smoothed using a Gaussian filter to minimize noise. This smoothing helps in highlighting genuine edges during the detection process.

Gradient Calculation: Next, the image's gradient is calculated at each pixel to identify areas with significant changes in intensity, which are potential edges.

Non-maximum Suppression: Non-maximum Suppression: This technique thins the edges detected in the gradient step to ensure the edges in the final image are sharp and clear.

Double Thresholding: This involves two thresholds to differentiate between strong and weak edges. Strong edges are definite edges, while weak edges are only considered edges if they are connected to strong edges. The result of this detection is saved as edge.png, which clearly shows the table edges.

Following edge detection, the next step is to use this edge.png for line detection using the Hough Transform. The Hough Transform is a technique to detect lines in an image:

Binarization: Before applying the Hough Transform, edge.png is converted into a binary image, where edges are marked as white on a black background.

Hough Transform Implementation: This method maps points in the image to a space where lines can be detected as accumulations of intersecting curves. By setting appropriate thresholds in this space, the most prominent lines corresponding to the table edges are identified.

The final image is then adjusted to the best thresholds found during the process, ensuring that the line representing the table's edge is clearly visible. This line is crucial for Luna as it indicates where the table ends, helping the robot avoid falling off the table.

1.2 To prevent Luna, the robot, from colliding with objects on the floor, a depth map needs to be generated using images from two cameras mounted on her body, named left.png and right.png. The depth map will display closer objects in red and farther objects in blue, guiding Luna's movements.

Below is a simplified explanation of the process and a Python code to create such a depth map.

Capture Images: Luna uses her two cameras to take images of the scene in front of her. The left and right images capture slightly different perspectives due to the horizontal distance between the cameras.

Calculate Disparity: By comparing similar features or blocks of pixels between the two images, the horizontal shift (disparity) for each pixel is determined. Larger shifts indicate closer objects, and smaller shifts suggest objects are farther away.

Generate Depth Map: Using the disparity data, a map is created where each pixel's intensity is inversely proportional to its disparity. Therefore, pixels representing closer objects will have higher intensity (shown in red), and those representing farther objects will have lower intensity (shown in blue).

Visualize and Save: The depth map is then saved as depth.png, providing a visual guide for Luna to navigate safely.

III. Related Work

1.1 Canny edge detection is a widely used technique in computer vision to identify edges within images. It operates through a multi-step process to accurately detect edges while minimizing noise and false positives. Firstly, the image undergoes Gaussian smoothing, which reduces noise and highlights important features by convolving it with a Gaussian kernel. Next, gradients of intensity are calculated at each pixel, revealing areas of significant change in intensity, this is done using Sobel Kernel. Non-maximum suppression is then applied to thin out edges, retaining only the strongest responses and eliminating weaker ones. Finally, double thresholding categorizes edge pixels as strong, weak, or non-edges based on their gradient intensity, resulting in a binary edge map that highlights prominent features in the image.

1.2 Constructing a depth map involves comparing images from Luna's two cameras to determine the disparity between corresponding pixels. Greater pixel shifts signify closer objects, while smaller shifts indicate farther ones. The disparity data is converted into depth values, with larger disparities representing shorter distances. A color-coded map is generated, where warm hues depict closer objects, and cool tones signify distant ones.

IV. INITIAL ATTEMPTS

At first, I considered solely using the Sobel Kernel for edge detection. However, upon understanding the concept of Canny edge detection and its double thresholding feature, I realized its potential for achieving more accurate results. By utilizing Canny edge detection and adjusting the thresholds, I became confident that I could accurately detect the edges of the table. This method provided greater flexibility and control, allowing me to fine-tune the parameters to suit the specific characteristics of the table's edges, ultimately leading to more precise edge detection.

V. Final Approach

1.1 In my endeavor to perceive and navigate my environment safely, I utilize various image processing techniques to detect the edges of the table. Initially, I capture an image and convert it to grayscale to simplify processing. Then, I apply the Sobel kernel to highlight areas of significant intensity changes, indicative of edges. This step serves as the foundation for subsequent edge detection processes.

Building upon the Sobel-filtered image, I utilize non-maximum suppression to refine the detected edges. This technique ensures that only the local maxima along the edges are retained, enhancing the clarity and sharpness of the detected features. By iteratively comparing each pixel's intensity with its neighbors along the gradient direction, I identify and preserve the most prominent edge pixels while suppressing weaker responses. This process helps maintain a clear delineation of the table's boundaries, critical for accurate navigation.

With non-maximum suppression applied, I further optimize the edge detection process through double thresholding and hysteresis. Double thresholding enables me to distinguish between strong, weak, and non-edge pixels by setting high and low threshold values. Pixels with gradient magnitudes above the high threshold are classified as strong edge pixels, while those between the high and low thresholds are considered weak edge pixels. This step effectively separates true edges from noise in the image, improving the overall edge detection accuracy.

Hysteresis complements double thresholding by connecting weak edge pixels to strong ones, ensuring the continuity of detected edges. I iterate over the image, promoting weak edge pixels to strong edge pixels if they are adjacent to existing strong edge pixels. This iterative process bridges gaps between weak edge segments and reinforces the detected edges, resulting in a more robust edge detection outcome. Together, double thresholding and hysteresis provide me with a comprehensive approach to optimizing edge detection and enhancing the quality of the detected edges.

Once I have obtained refined edge detection results, I employ the Hough transformation to identify points along the edges corresponding to lines in the image. The Hough transformation allows me to detect lines, even if they are broken or have gaps, by representing each point in the image space as a line in the parameter space. I utilize this information to select the best two points along the detected lines and construct a line along the edge of the table using the slope equation.

By leveraging image processing techniques such as Sobel filtering, non-maximum suppression, double thresholding, hysteresis, and the Hough transformation, I successfully detect the edges of the table in the environment.

1.2 In my Python script, I've implemented the construction of a depth map from stereo images using various image processing techniques. It starts by importing the left and right stereo images and converting them to grayscale to simplify further processing. Then, I apply Gaussian blur to smooth out noise and the Sobel operator to compute the gradient magnitude, highlighting areas of significant intensity changes, which often indicate edges.

Afterwards, I add padding to the images to accommodate the chosen kernel size for local window matching. This ensures that no information is lost at the image boundaries during subsequent processing steps. Then, the script iterates through each pixel in the left image, conducting a local window search in the corresponding region of the right image to find the best matching disparity value.

Once the best matching disparity value is determined for each pixel, I construct a disparity map. This map represents the disparity value (depth information) for each pixel in the left image. In this script, I normalize the disparity values to a range between 0 and 255 and convert them to a depth map visualization.

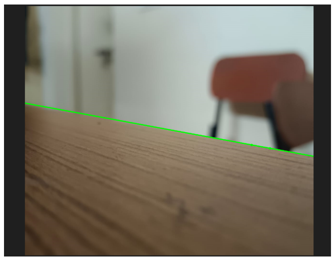
For the depth map construction, I map the normalized disparity values to color intensities to create a visually interpretable depth map. In this mapping scheme, the disparity values are mapped to the blue channel of the depth map, while the inverse of the disparity values (representing depth) is mapped to the red channel.

Finally, I display the depth map using OpenCV, allowing for visualization of the spatial layout and depth information captured by the stereo images. This depth map provides valuable insights into the scene's structure, facilitating tasks such as 3D reconstruction, object detection, and navigation.

Overall, my script demonstrates a comprehensive approach to constructing a depth map from stereo images using image processing techniques such as edge detection, local window matching, and disparity mapping. By leveraging these techniques, I extract depth information from stereo image pairs, enabling various applications in computer vision and robotics.

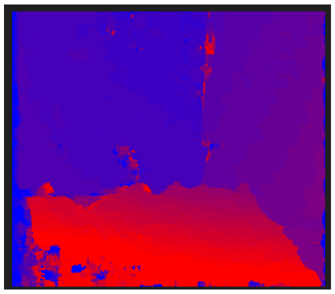
VI. Results and Observation

The result for 1.1 is as follow:



As it can be seen the edge is detected accurately and line is drawn.

The result for 1.2 is as follow:



As it can be seen there are noises and if we use large kernels and complex images the computational cost will be high.

VII. Future Work

Noise Reduction: Implementing noise reduction techniques, such as bilateral filtering or median filtering, can help mitigate the effects of noise on depth estimation. By smoothing the disparity map while preserving edge information, these techniques can improve the algorithm's robustness to noise.

Disparity Optimization: Optimizing the disparity estimation process can lead to more accurate depth maps. Techniques such as dynamic programming or graph cuts can be employed to perform global optimization and refine the disparity values, particularly in areas with occlusions or discontinuities.

Conclusion

1.1 Conclusively, the implemented algorithm successfully detects edges on the table using Canny edge detection and Hough transform. Challenges included parameter fine-tuning, ensuring line detection accuracy, and optimizing the algorithm for efficiency. Lessons learned encompassed the significance of parameter optimization for effective edge detection, ensuring robustness in edge detection methods, and the importance of algorithmic efficiency for real-time applications. Through overcoming these challenges and applying learned lessons, the algorithm achieved its objective of accurately identifying table edges, highlighting the importance of meticulous parameter adjustment and algorithmic optimization in computer vision tasks.

1.2 In conclusion, the depth estimation algorithm demonstrates potential for extracting depth from stereo images. Balancing accuracy and efficiency posed challenges, particularly in optimizing parameters for diverse scenes. Lessons learned include the significance of fine-tuning parameters for optimal performance and the iterative nature of disparity estimation. Despite difficulties, the process underscored the importance of understanding stereo vision fundamentals and the potential for improvement through advanced techniques like feature matching and noise reduction. With further refinement, the algorithm holds promise for applications in robotics, autonomous driving, and 3D reconstruction, offering valuable insights into scene perception and spatial understanding.

REFERENCES

[1] ”https://en.wikipedia.org/wiki/Cannyedgedetector”

[2] ”https://www.analyticsvidhya.com/blog/2022/06/a-complete-guide-on-hough-transform/”

[3] ”https://medium.com/analytics-vidhya/distance-estimation-cf2f2fd709d8”

[4] ”https://medium.com/@rohit-krishna/coding-canny-edge-detection-algorithm-from-scratch-in-python-232e1fdceac7”