SAVE LUNA FROM FALLING

Shresth M Jha

Abstract— **In this task we have saved Luna from falling by detecting where the edge of the table is and avoiding the potential fall. It can be scaled in the real world for robots navigating themselves,it can be used in ARK for detecting where the walls or other obstacles are coming in the path of the drone.**

I. INTRODUCTION

the problem statement was to detect the edges present in the image using Gaussian, Sobel kernels,(not mentioned in the task but I used it)non-max suppression and once the edges are detected convert them to a binary occurrence image. after which we will apply the Hough transformation to detect the lines.

II. Problem Statement

This should cover one full column of page. In the first part of the task, we applied the Gaussian filter to reduce the noise in the image so that it would be easier for us to detect the edges, also the sigma value had to be set so that the noise reduction does not result in intensity reduction that it makes edges difficult to detect. Once the filtering is done the Sobel filter is used with two kernels as shown in Figure **1**, after which the gradient intensity is calculated and the edge direction is calculated near the edges there would be significant gradient intensity and hence they would be visible in the image after applying the filter. gradient intensity and edge direction finding formula is shown in Figure **2**. to make the edges more clear I have performed an extra step known as the non-max suppression in which we thin out the edges. The principle is simple: the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions.

Now explaining both of the methods in detail:

SOBEL FILTER CALCULATIONS: The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators.

Edges correspond to a change of pixels' intensity. To detect it, the easiest way is to apply filters that highlight this intensity change in both directions: horizontal (x) and vertical (y)

When the image is smoothed, the derivatives ***Ix*** and ***Iy*** w.r.t. ***x*** and ***y*** are calculated. It can be implemented by convolving ***I*** with Sobel kernels ***Kx*** and ***Ky***, respectively

Each pixel has 2 main criteria (edge direction in radians, and pixel intensity (between 0–255)). Based on these inputs the non-max-suppression steps are:

NON MAX SUPPRESSION METHOD:

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

Fig. 1. kernels used in Sobel filters

$$|G| = \sqrt{I_x^2 + I_y^2},$$
$$\theta(x, y) = arctan\left(\frac{I_y}{I_x}\right)$$

Gradient intensity and Edge direction

Fig. 2. gradient and the edge direction in the Sobel filter

- Create a matrix initialized to 0 of the same size of the original gradient intensity matrix;
- Identify the edge direction based on the angle value from the angle matrix;
- Check if the pixel in the same direction has a higher intensity than the pixel that is currently processed;
- Return the image processed with the non-max suppression algorithm.

III. Related Work

One other thing that could have been done is to do complete the canny edge detection but we did only a few steps involved in it.

IV. INITIAL ATTEMPTS

Write all the equations and results with pictures if applicable. As shown in Figure 3 the red line is the edge detected

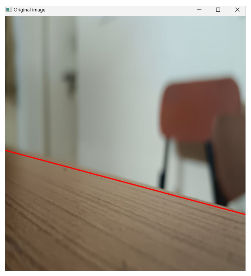

Fig. 3. Figure 3

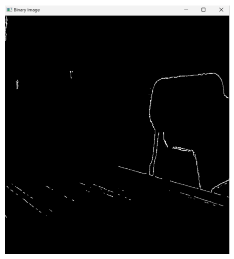

Fig. 4. Figure 4

from the image and image 4 is the binary occupancy image that we get after the filtering and the non max suppression and converting it to a binary image. in the initial attempts, the problem was that sometimes due to confusion between the x, and y coordinates and the row, and column convention, the code was giving wrong results which on closer look solved the problem.

V. Final Approach

My final approach to this problem was as follows: first converted the image to a grayscale image. then I applied the Gaussian filter to reduce the noise the value of sigma had to be set such that the noise was reduced but the intensity was also not affected much. after this, I applied the Sobel filter which basically takes the intensity change values and takes the hypotenuse of it for the gradient intensity values the value of the gradient intensity values is high near the edges hence only they are visible in the image prominently and edge direction value tells us in which direction is the edge oriented near that point. after that to make the edges more clear we take the edges and apply the non-max suppression method on them which proceeds according to the following steps.

Create a matrix initialized to 0 of the same size of the original gradient intensity matrix;

Identify the edge direction based on the angle value from the angle matrix;

Check if the pixel in the same direction has a higher intensity than the pixel that is currently processed;

Return the image processed with the non-max suppression algorithm. once we get the image after this we convert it to a binary image with some specified threshold decided according to the visibility of the required lines in the image. now the binary image that we get is put to hough line detection method in this method we convert space to a polar space and then in an accumulator matrix we store the number of times a particular line is recurring the lines with more than a certain threshold is recorded and given as the output. the final output that we show is the Figure 3.

VI. Future Work

Write about the problems in your algorithm / approach and limitations in testing (if any due to hardware or otherwise) and how to tackle them and any future work which can be done to improve the results further. One of the limitation of my code is that it will work for this particular image but for other images the value of the sigma and the threshold values will have to be set accordingly otherwise some more edges than required can come after the hough transformation. also one more limitation is that I have written the code only for linear edge detection so for some curved edges the code would have to be changed.

Conclusion

Overall this task taught me a lot about how the images have to handled in a computer and different algorithms that are used in computer vision.and as i am applying for the perception division in the ARK it will definetely help.