ECE 4881 / 588
Introduction to Robot Vision

Project Milestone #4
Line Following & Classification

Samuel Bellaire
Sam Khalaf
Taylor Tomblin
Ali Hassan

Table of Contents

1 Introduction

The primary deliverable of milestone #4 is to develop software for the robot to fulfill a line following function, as well as to design a classifier to identify objects in the robot's environment.

2 Line Follower Software

The first task of project milestone #4 is to develop a line following algorithm for the robot. For this project, the robot was designed to operate in the IAVS high bay; thus the image processing system will identify the edges of white line markings on the floor.

2.1 Image Processing Pipeline

The image processing system on the robot follows the pipeline detailed in Fig. 1. An image is first captured from the robot via ROS, and it is then cropped down to the region of interest (i.e. the ground in front of the robot), as any other objects above the horizon would introduce noise to the line extraction algorithm that are not necessary for performing the line following task.

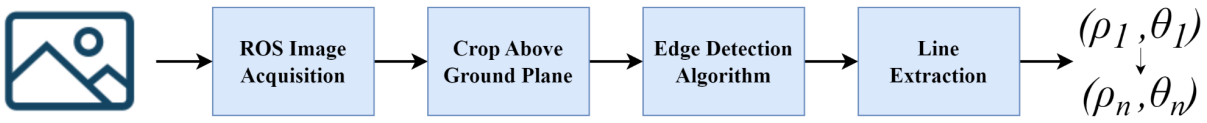

Figure 1 - Image Processing Pipeline

From milestone #2, it was determined that all rows above ~250 in the image can be discarded to obtain an image of the ground plane in front of the robot. Fig. 2 shows a sample cropped image captured from the robot's camera.

Figure 2 - Cropped Grayscale Image

After cropping the image, the Canny edge detection algorithm was used to identify all of the edges in the image. A threshold of 0.4 was used on the gradient image, with $\sigma = 0.3$ being used for the Gaussian smoothing kernel. Fig. 2 shows the resulting image after processing Fig. 1.

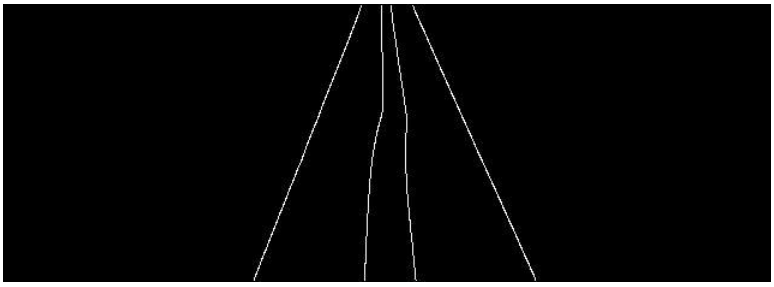

Figure 3 - Canny Edge Detector Output

Finally, the Hough Transform of the edge image was taken to identify lines in the image. The Matlab `houghlines` function was used to accomplish this, with a minimum line length of 20 being used. Gaps of up to 150 pixels between similar lines were also filled in by the algorithm. Fig. 4 shows the result of applying the Hough Transform to the edge image.

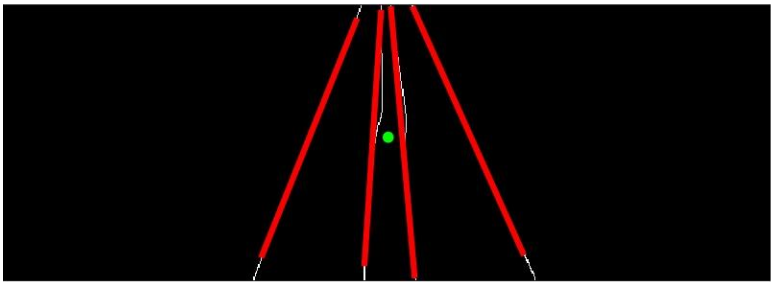

Figure 4 - Hough Line Extractor Output

Also take note of the green dot in the center of Fig. 4; this is a crucial aspect of the steering control system that will be elaborated later in Sec. 2.2.

2.2 Control System Design

The approach taken for steering control of the robot in this milestone was unorthodox, but nonetheless proved to be more effective than the bang-bang controller implemented in milestone #2.

Each line i detected in the image has a midpoint M_i. The average M_μ is then given by eq. 1.

$$M_\mu = \sum_{i=1}^{N} \frac{M_i}{N} \tag{1}$$

This "average midpoint" is the green dot shown roughly in the center of Fig. 4, and is used as the target for the robot to follow. Its longitudinal deviation from the center of the image is given by the error $e = 0.5W - M_\mu$ (where W is the image width in pixels). The steering controller was

then implemented as a simple proportional controller, with $K_p = 2/W$ to prevent oversteering in the event that the average midpoint appears near the edges of the image. The controller is given by eq. 2.

$$\omega = K_p e \tag{2}$$

2.3 Results & Analysis

Despite being simple, the control law given by eq. 2 showed fairly robust performance and disturbance rejection for this application. Normally, a more complex control algorithm would be developed, but due to the low speed of the robot and the controlled environment in which testing was performed, a proportional controller performed acceptably. The attached demo video(s) demonstrate the performance of the line following software.

3 Object Classifier

The second task of project milestone #4 is to develop a simple classifier to distinguish objects in the robot's environment. For this project, a classifier was designed to pick out an orange cone in the IAVS high bay environment.

3.1 Classifier Design and Training

Before designing the classifier, a training image was captured that contains objects representative of what the robot will see in its environment. This image is shown in Fig. 5.

Figure 5 - Classifier Training Image

Two regions were extracted from the image to represent the cone and the environment. A roughly 100 x 100 pixel patch from the center of the cone was chosen to represent the object of interest, and the entire right half of the training image was taken to represent the environment.

The color covariance matrix (given by eq. 3) was then calculated for the three color channels (red, green, and blue), where C_{ij} denotes the covariance between i and j.

$$C = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \tag{3}$$

The mean of each color channel was also calculated and arranged in a vector, given by eq. 4.

$$\boldsymbol{\mu} = [\mu_r, \mu_g, \mu_b] \tag{4}$$

These two quantities can then be used to compute the Fisher Discriminant, which is given by eq. 5. This discriminant is used to determine if a pixel in an image belongs to the object of interest, or the environment.

$$\boldsymbol{U} = (\boldsymbol{\mu}_{obj} - \boldsymbol{\mu}_{env})(C_{obj} - C_{env})^{-1} \tag{5}$$

For any pixel p, a Fisher Index can then be calculated as in eq. 6. Thresholding the normalized index after calculating it for all pixels p then yields a binarized image where all white pixels are considered as objects, and all black pixels are classified as the environment.

$$I_f = \frac{|(p - \boldsymbol{\mu}_{obj})\boldsymbol{U}^T|}{\det(C_{obj})} - \frac{|(p - \boldsymbol{\mu}_{env})\boldsymbol{U}^T|}{\det(C_{env})} \tag{6}$$

3.2 Results & Analysis

For milestone #4, the Fisher Index given by eq. 6 was calculated for every pixel in each image, and the normalized Fisher Index was then thresholded at a value of 0.2. The result of this operation is shown in Fig. 6; it can be seen that the classifier finds a clear distinction between the orange cones and the rest of the environment. The real-time performance of the classification algorithm can be seen in the attached demo video(s).

Figure 6a - RGB Image

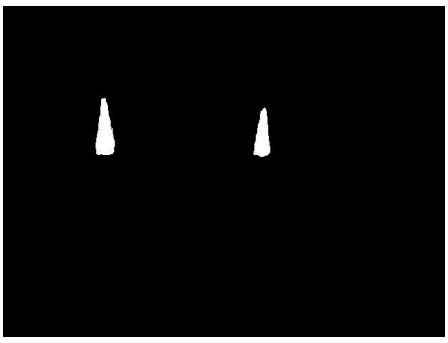

Figure 6b - Classified Image