CS4758: Multiple Learning Methods for Autonomous Lane Navigation

Christopher Barrett Ames, College of Engineering/Independent Study, Sophomore,
cba33@cornell.edu

***Abstract*—Autonomous vehicles are becoming a thing of the present, it's not just science fiction anymore and one major task they must be able to do to become fully integrated in society is lane navigation. Lane navigation is often done with cameras, although it can be done with LIDAR's that provide an intensity reading. In this project, I present three different approaches to vision based lane navigation using supervised learning, SVM, and unsupervised learning.**

The three different learning algorithms all return the same information, that being the location of the lane. The edge information obtained from the image is then passed to a free space path planner. This planner will drive towards the largest open space in front of it that is also in the direction of its goal.

The experiments carried out are based on the Minesweeper Nero platform and were conducted at the same location on several different days.

I. Introduction

Autonomous driving vehicles have become more and more prevalent ever since the DARPA Grand Challenge. The first few vehicles were dependent on precise data provided by specialty sensors. However since those first few vehicles the interpretation of noisy data has become increasingly better and thus opened up a whole new array of sensors that can be used. The monocular camera has emerged as one of the top sensors for autonomous vehicles. From its images many helpful cues can be derived.

For example, Figure 1, a bird's eye view of an intersection gives a good idea of how many visual cues we rely upon while driving. There are crosswalks, stoplights, road dividers and lane lines to name a few. I will be focusing on the detection of lane lines in a less structured environment than what is typically found on roads. Instead of lanes on roads I will be looking at lanes painted on grass. This is similar to a road but the overall texture and color of grass can vary more than asphalt.

In this project, I tackled the lane detection problem through several learning methods to evaluate which is the best method to use. The methods used are a supervised learning, support vector machines, unsupervised learning.

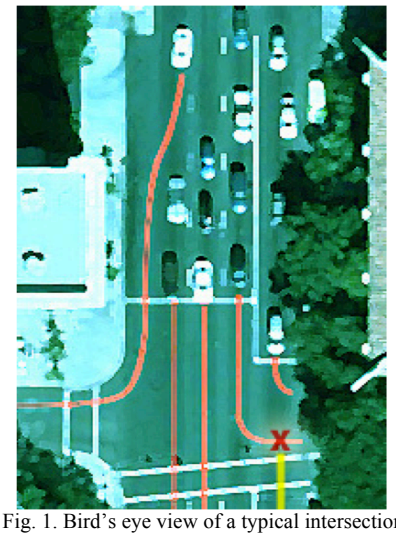


Fig. 1. Bird's eye view of a typical intersection

The paper is organized as follows. After discussing related work in Section II, I briefly describe the robot that was used for this paper in Section III. In Section IV I discuss the three learning methods that were applied. Section V presents the experimental results of the three learning algorithms. Lastly, I conclude in section VI and give a few proposed solutions to problems that arose.

II. Related Work

Lane navigation is one of the basic problems of robot navigation and there are many discussions about the issue in robotics literature. In [3] the lane is detected in real time by first taking a threshold of the image and then using an equation specific to their situation to pick out the lane based on its perspective geometry. This lacks the generality needed for use in other platforms and the algorithm would not be able to perform well if the camera's position was changed. In [4] the robot follows the curvature of a line on the ground to simulate the following of road curvature.

Our approach to supervised learning approach to lane detection is very similar to the algorithm used in [3]. The main difference being that we used supervised learning to determine the parameters of the algorithm instead of estimating them for our setup. This was done to increase the robustness of the algorithm and to fit the parameters of the algorithm to our specific robot.

In [6] the curvature of the road is obtained by using a Support Vector Machine (SVM) to classify what is part of the road and what is not. They then broke the image into horizontal sections. The centroid of the road sections in each section was found and then SVM regression was performed on the centroid locations to get the road curvature. This approach is very similar to the SVM approach that was used for this project.

III. The Robot

Nero V2.0 is a differential drive robot designed for competition in the Intelligent Ground Vehicle Competition (IGVC). The purpose of Nero V2.0 is to be a software test platform and thus is not mechanically complex.

However, the sensor suite that has been developed on Nero is extensive. Nero has a SICK LIDAR with a range of 30m, and a color camera with 45degree field of view, delivered at 15 frames per second. Also, each wheel has three hall-effect sensors that act as wheel encoders to provide odometry data. To complement the wheel encoders and to compensate for any situation in which the wheels might slip, there is a 6 DOF IMU paired with GPS that provides hardware Kalman Filtered position and velocity. Lastly, all of the sensors are tied together using the ROS architecture to allow the development of navigation algorithms that work on multiple platforms.

However, for this project the most important sensor is the camera. It runs at a very low resolution: 160x120. This is to keep the bandwidth down and to get rid of any fine edges. Placement of the camera can be seen in figure 2.

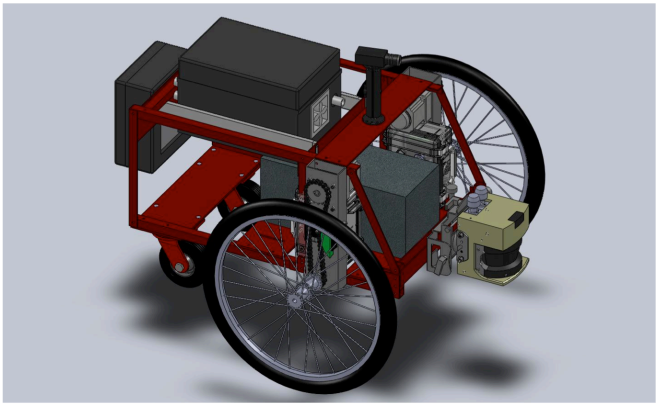


Fig. 2. Nero V2.0

IV. Lane Detection Methods

The goal of this work is to learn the pattern or shape that best describes the lane lines in an image. The following methods all work on the same basic algorithm. First, classify pixels as part of the lane. Next, fit a line or a curve to the pixels that were classified as lane lines. Lastly, the line is passed to the free space follower that is implemented on the robot.

A. Supervised Learning

The lanes are detected using the same approach as in [3]. First, the image is filtered using a median filter (figure 3(b)) to remove any outlier noise in the image. Then a threshold is applied to the image to make sure that only things that are bright and white are included. Taking the threshold of the image leaves a speckled image as in figure 3(c). Then each line of the speckled image is scanned to determine the width of contiguous segments of white in the binary image (figure 3(d)). If the width of the segment is approximately the same as an expected width of the lane then the contiguous white segment is kept in the image, otherwise it is removed. The expected width is calculated using parameters learned from more than 100 training images for equation 1.

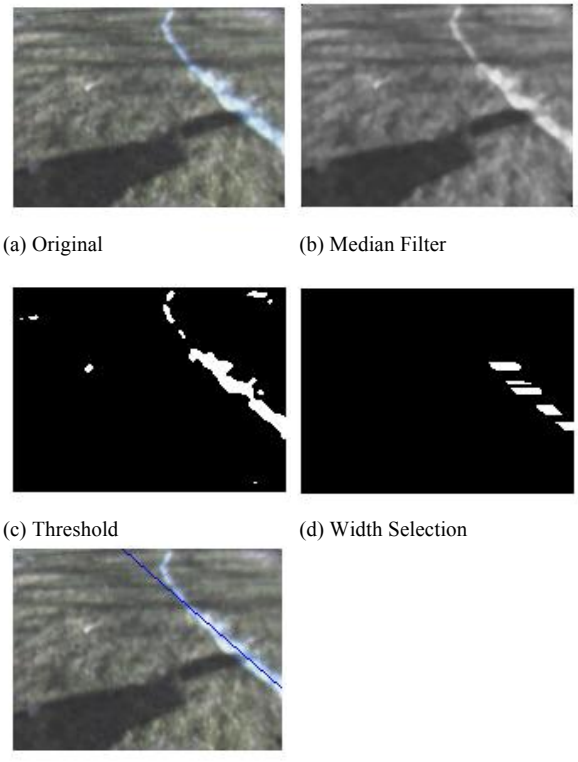
$$width = initial - \left(\frac{row}{h1} + \frac{row}{h2} \right) \quad (1)$$

Once all of the horizontal lines have been scanned there is a group of pixels left that are considered to be part of the lane line. A linear regression is performed on the lines to provide the x-intercept and slope of the line (figure 3(e)).

The supervised learning process consisted of labeling all of the lanes in the image, detecting the lines using the stated algorithm, and then determining the error of the estimated line. The lines were identified by their x-intercept, slope and their angle. The angle and slope were both recorded because as the line nears 90 degrees the slope goes to infinity. Several error metrics were tested: intercept error, angle error, and the distance between 8 x positions for the two lines.

The parameters that had to be learned were *initial, h1* and *h2* of (1). They were learned using a brute force algorithm followed by a gradient descent. The brute force algorithm was very rough and was implemented to make sure that the gradient descent did not get trapped in a local minimum. The gradient descent was run afterwards to refine the parameters.

Fig. 3 Supervised Learning



(a) Original (b) Median Filter

(c) Threshold (d) Width Selection

(e) Least Squares Line

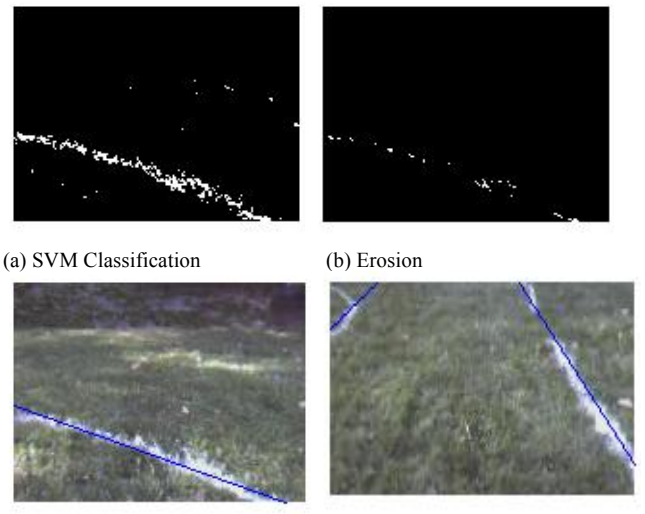
B. Support Vector Machine

For this approach the Support Vector Machine (SVM) was used to perform pattern recognition. The method outlined here is very similar in nature to [6], however this implementation focuses on the lane lines instead of the overall road shape.

The SVMLIN of Shogun Machine Learning [5] was used to create the SVM. The linear SVM was used because it is a two class SVM and it is fast at classifying new images. Many of the other SVMs that were tried had classification times of 30 seconds for an image. This is not acceptable for real time navigation. The SVM used a Gaussian kernel and the image features used were the color channels, the luminance, and the chrominance of a randomly selected group of pixels from more than 100 training images. After being trained the classifier was then run on a set of test images to check the generality of the classifier.

Once the SVM had classified all of the pixels in an image as in figure 4(a), then the image is cleared of unconnected white pixels and eroded using a cross structure (figure 4(b)). Next, the image was broken up into a left half and a right half. These halves then had linear regression run on them, which provide the blue lines seen in figure 4(c) and 4(d).

Fig. 4 Support Vector Machine



(a) SVM Classification (b) Erosion

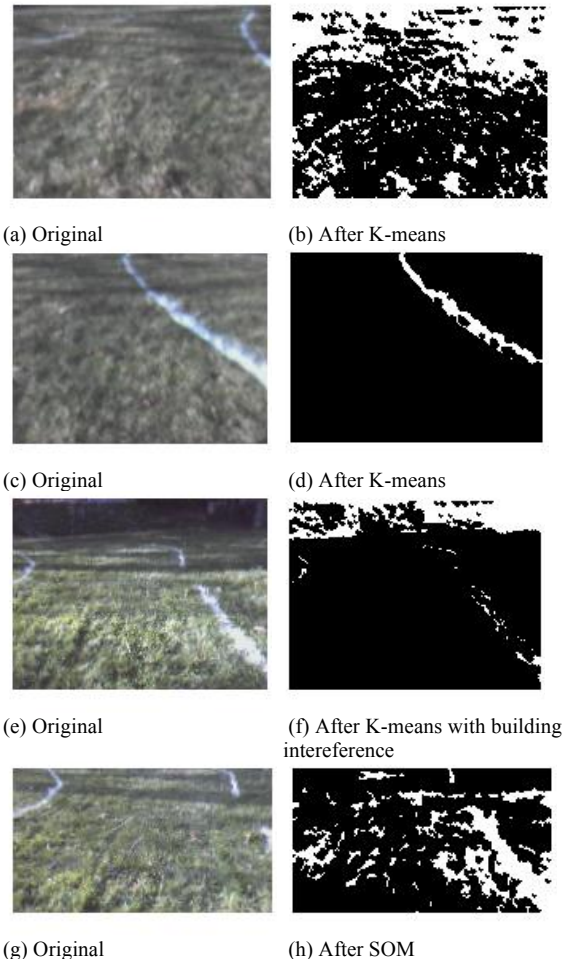
(c) Linear Regression on (b) (d) Linear Regression on both halves

C. Unsupervised Learning

The first attempt at unsupervised learning was using K-means clustering. K-means was passed

the same features that were passed to the SVM: the color channels, the luminance, and the chrominance. The K-means algorithm used the cosine distance function and only has two clusters. The cosine distance function looks at the angle between two points; this produces a centroid point that is the mean of all the points in the cluster after normalizing to unit length. This produced images that were not usable at all (figure 5(b)). Better results were reached when intensity and edges were included in the feature set passed to the algorithm. The images produced from the new set of features were better, but could not distinguish between the white building and the white lines (figure 5(f)).

Fig. 5 Unsupervised Learning



The second attempt at unsupervised learning was using Self-Organizing Maps (SOM). SOM is a type of neural network that clusters data based on the weighting o f neurons. The set up used had 2 neurons, the neuron positions were based upon a random configuration and the weighting was based upon the link distance between the neurons. The neurons were trained for 200 iterations over the training set of images, each iteration having 100 steps to reach convergence. These images did not contain any marking data, but the SOM learns to weight its neuron connections by training over an initial data set.

After either of the unsupervised algorithms ran the image they produced was then cleaned of single pixels and then eroded using the same cross structure as in the SVM case. Figure 5 shows the results of these two methods.

V. Experiments

A. Data

The dataset we are working from is based on what the robot will be seeing in the IGVC competition. This means that dataset is made up of flat grassy areas with various obstacles and white lines no less than six feet apart. The obstacles are trashcans of different colors and various construction barricades. The current dataset includes pictures of white lines from multiple angles, poorly maintained and multicolored grass, scattered leaves, and shadows cast from nearby trees. The training data set is 119 images. The test data set is 120 images.

B. Results

Supervised Learning

	Train	Test
RMS Distance Error (pixels)	33	6308

SVM- Pattern Recognition

	Train	Test
Observations	59619	59619
Correct Rate	0.9823	0.9818
Sensitivity	0.9975	0.9972
Specificity	0.5182	0.475
Prevalence	0.9825	0.9825
Correctly Marked Lanes	84.40%	82%

Self Organized Maps

	Train	Test
Observations	119119	119119
Correct Rate	0.5977	0.5966
Sensitivity	0.6024	0.601
Specificity	0.4286	0.4338
Prevalence	0.9732	0.9733

K-means

	Test
Observations	960000
Correct Rate	0.4379
Sensitivity	0.4252
Specificity	0.9878
Prevalence	0.9774

Fig. 6. The percent of correctly marked lines is only noted for the SVM because the other two methods had such a low correct rate that it did not warrant calculating the percentage. The RMS distance error was calculated by taking the distance between a point and a line for eight different points between the marked line and the estimated line. Sensitivity is the number of correctly marked positive samples divided by the total number of positive samples. Specificity is the number of correctly marked negative samples divided by the total number of negative samples. Prevalence is the number of true positives divided by the total number of samples.

Overall, the data shows that the SVM was the most correct, partly because it's amazing ability to generalize. It worked well across the variation of the color of the grass, leaves and flowers. In addition, it worked well across mild changes in lighting conditions, failing when it became too bright, but not when it became dim. The down side to the SVM being that if too many training cases are used the time to classify becomes very high, and thus the lane identification becomes too slow for use on a robot in action. However by using a linear SVM solves this run time issue.

The supervised learning approach works well for test cases that were very similar to what it had been trained on. However if there was a variation in lighting or grass color then this method suffered. This show that supervised learning can suffer from a lack of generality and thus shouldn't be expected to perform well unless the images passed to it are very similar to images that it trained on.

Lastly, the unsupervised learning showed promise and was quite good at classifying some of the images, but there were a considerable number that did poorly. It worked well on images where the line was the brightest thing in the image, but it didn't do very well on images that were closer in brightness. The major advantage of the unsupervised learning is that no time needs to be spent marking data as correct or not. Unsupervised learning could be viable, if more image features were used.

VI. Conclusion

I presented three methods for lane detection all based on learning. The best method was the SVM due to its ability to generalize and classify in many different situations. However both supervised and unsupervised learning could do better if given more features.

In the future I would like to increase the number of features used by all three of the algorithms and see if this would increase their accuracy. Lastly, I would like to include SVM regression for function finding so that a better model of the lane can be derived from the image.

Acknowledgment

The author would like to thank Cornell Minesweeper for allowing me to use their platform and for their technical support. In addition, I'd like to thank Professor Saxena for his guidance on the direction and form this project should take.

References

[1] Jeff Michels, Ashutosh Saxena, Andrew Y. Ng. High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning, Proceedings of the Twenty-first International Conference on Machine Learning (ICML), 2005

[2] Eric Royer, et al. Outdoor autonomous navigation using monocular vision, International Conference on Intelligent Robots and Systems, 2005

[3] Chung-Yen Su and Gen-Hau Fan. An Effective and Fast Lane Detection Algorithm, Advances in Visual Computing, 2008

[4] Yi Ma, Jana Kovseck, and Shankar Sastry. Vision guided navigation for a nonholonomic mobile robot, The Confluence of Vision and Control, 1998

[5] S.Sonnenburg, G.Raetsch, C.Schaefer and B.Schoelkopf, Large Scale Multiple Kernel Learning.Journal of Machine Learning Research,7:1531-1565, July 2006, K.Bennett and E.P.-Hernandez Editors.

[6] Hao Zhang, Dibo Huo, Zeki Zhou. Novel Lane Detection Algorithm Based on Support Vector Machine, Progress in Electromagnetics Research Symposium, August 2005, pp.390-394