Lane Markings From Satellite

Danjie Wenren,* Tianxin Zhao,† and Kunming Qu‡

In this project we attempt to apply techniques from digital image processing and machine learning to extract lane marking information from the satellite images obtained from Google Maps. Edge detectors are used to reduce the amount of pixels that need to be examined. Histogram of Oriented Gradients are used as our feature descriptors and Hough transform is used to obtain the final marking of lanes.

I. INTRODUCTION

Autonomous driving technology has advanced remarkably during the past few years. It has not, however, become practical due to the high cost of its major components. One approach that could significantly lower the cost is to purely use computer vision technology to detect the real-time driving environment for autonomous vehicles. An important issue such kind of technology needs to solve is to detect lane markings, with which the vehicle can adjust its direction.

Most work in the literature at the moment is on the real-time detection ([1] and references therein), where lane markings are identified using the images taken by the vehicle. The techniques developed in such ways already have a very high detecting accuracy. However, they might not provide the detecting results fast enough in real-time due to their computational complexity in examining one frame of image. This problem is even more severe when the vehicle is moving at a high speed.

The goal of this project is to build a machine learning system that could generate lane markings from satellite images. The images we will use are taken from Google Maps. With such a system, we can then build a database of various properties of lanes, such as latitude-longitude, or the types of lanes, etc. Tools that could map the latitude-longitude information of lane points into points in a vehicle's real-time vision have been made, and the complexity of such a mapping is expected to be relatively small. Therefore given such a database of the lane markings, a vehicle can identify its nearby lane markings rather quickly.

At the current stage of the project, we implemented a Support Vector Machine (SVM) to identify the white lane markings, which separate lanes along the same direction. For this purpose, a two-class off-the-shelf SVM package is sufficient. Specifically, we used the *LibLinear* package ([2]).

Many features can be used to classify whether a point is on a lane or not. [1] uses both the local features, such as the color and intensity of the point, and the spatial context features of the point, such as those obtained by applying a Haar-like filters. In this work, we use the Histogram of Oriented Gradients (HOG, [3]) as our features. Adding local features of a point can also increase the accuracy of the SVM, as the SVM trained using purely the HOG may mistakenly identify some of the non-lane-marking points on the color changing edges as lane-marking points. However, at this stage of the work, we have not added such features. As we will show in the follows, even purely using the HOG feature, our technique already has a remarkable accuarcy.

The support vector machine trained as above can examine the image pixel by pixel. A layer of Hough transform is added afterward, which further extracts lane patterns from a bit-map generated by the SVM, where 1 means the SVM thinks the point is on lane and 0 otherwise. At the moment, we only extract the straight lane patterns. To reduce the amount of pixels the SVM will need to examine, a layer of edge detection is added before the SVM.

In section II, we will give a brief overview of the digital image processing techniques we will use. Section III will explain our supervised learning process and how we selected models. In section IV we will show some detail of how a lane marking are generated for a satellite image using our technique. Some results are also provided. We conclude our current stage of work in section V.

II. OVERVIEW OF TECHNIQUES

In this section we briefly introduce the techniques from digital image processing that we will use. We will mainly focus on the role they play in our technique of extracting lane markings.

A. Edge detection

In the context of this work, *edge* means the set of points in a digital image where the color vector (r, g, b) changes significantly. This could be used as a primitive feature of an image.

Due to the high color contrast between the lane markings and the pavement around them, it is almost always the case that there is a color edge surrounding the lane marking. In specific, we use the *Canny* detector ([4]) to locate the edges of the satellite images.

*SUNet ID: djwenren
†SUNet ID: tianxin
‡SUNet ID: kunming

B. Histogram of oriented gradients

Histogram of oriented gradients ([3]) are strong feature descriptors for detecting objects in images. Originally developed for the purpose of pedestrian detection, it actually performs very well for other purposes, such as detecting lane marking points.

This technique divides the image into small regions, termed cells, and compute the histograms of binned gradients for the cells. The idea behind it is that local objects can be described by the distribution of color gradient and edge directions. The technique further organizes the cells (and their histograms) into a larger group, called block, to improve detection accuracy. It contrast-normalizes the histograms of the cells inside the same block. The normalization can reduce the effect of different illumination and shadowing. It maintains an invariance against geometric and photometric transformations.

The histograms of the blocks are then concatenated to form a feature vector for the entire image, which can effectively describe the content of the image. The length of the feature vector can be determined as the follows. Let the size of the image be $N_x \times N_y$ and the side length of a cell be c, then there would be roughly $(N_x N_y/c^2)$ cells in the image. In each block there are $B \times B$ cells and in each cell, there are b bins in the histogram. Due to the overlapping of the blocks, each cell is roughly counted for B^2 times. Therefore there are approximately

$$\mathcal{O}\left(B^2 \frac{bN_x N_y}{c^2}\right) \tag{1}$$

features for the image.

In the context of our work, the image that we use to generate a HOG feature vector is the patch around some pixel we examine, whose size is roughly 48×48. The side length of a cell is set to be 4 and there are 2×2 cells in a block. Assume the number of bins for each cell is 40, then the length of the feature vector for a point is around

$$4 \times 40 \times \left(\frac{48}{4}\right)^2 = 23040. \tag{2}$$

The exact parameters, such as patch size or cell size etc., will be selected using ten-fold cross validation later.

C. Hough transform

Hough transform is a standard technique used in computer vision and digital image processing to extract line pattern from an image. In the current status of our project, we only used Hough transform to extract straight-line patterns from the bit-map image generated by the SVM. This works as follows.

A straight line in an (x, y) space can expressed by

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right) x + \frac{r}{\sin\theta}, \tag{3}$$

FIG. 1: The training process.

where the (θ, r) fixes the line. For a particular point (x_0, y_0) on a line parametrized by (θ, r), (3) can also be written as

$$r = x_0 \cos\theta + y_0 \sin\theta = \sqrt{x_0^2 + y_0^2}\sin(\theta + \phi_0), \tag{4}$$

where $\phi_0 = \arctan(x_0/y_0)$. We can treat r in (4) as a function of θ for fixed (x_0, y_0). This gives a sinusoidal curve in the (r, θ) space. Each point on the curve represents a straight line through (x_0, y_0) in the original space.

Given a set of candidate points in the (x, y) space, we can compute the sinusoidal curve $r(\theta)$ for each point (x, y). The weight of a point in the (r, θ) space is the number of such sinusoidal curves through that point. Therefore if there are n candidate points in the (x, y) space on the straight line parametrized by (r, θ), then the weight of point (r, θ) is n. Thus the most weighted points in the (r, θ) represent the most likely straight line patterns in the original (x, y) space.

III. TRAINING

The supervised learning process can be summarized by Fig 1. The training set is generated by manually selecting points from satellite images obtained from Google Maps. In the current stage of work, we have only generated a training set of white lane marking points without touching other type of lane points, such as the yellow ones. In order to better train the support vector machine, we selected many more points on the color changing edges than those not on edges. The ones on the edge are more important because they are more likely to be on the decision boundary, since those non-lane-marking points on the edge will look very alike with the lane-marking points. We need to provide the support vector machine with more such points in order for it to learn better. In other words, in doing so, we are providing more support vectors and the decision boundary can be better tuned.

For each training point, we pick a patch of size $P_s \times P_s$ and use it to generate a HOG feature vector for the point. The length of the feature vector can be estimated using (1), which is typically of order $\mathcal{O}(10^4)$. In order to have a uniform size of patch and thus length of feature vector, we impose a periodic boundary condition for patch generation. This might have a negative impact on labelling points on the boundary of the satellite image. However it is not likely that those edge cases would have an important effect on the entire labelling process, especially

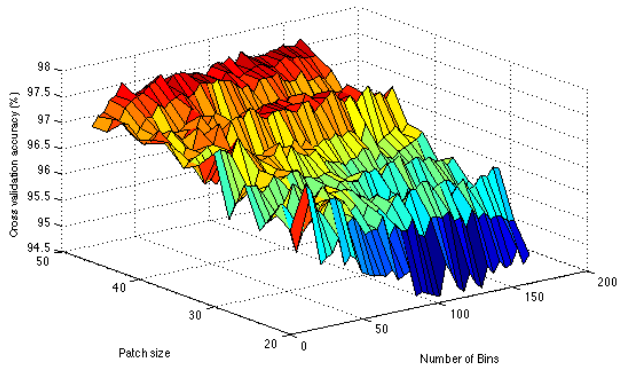

FIG. 2: The ten-fold cross validation rate as a function of patch size and number of bins used to generate HOG features. Purely from the perspective of cross validation accurarcy, it is indeed better to choose large numbers for patch size and number of bins. But large numbers also make the computation expensive.

when there is a Hough transform layer after the pixel-wise labelling.

Then the HOG feature vectors and training point labels are fed into a SVM to generate a model. In our work, the training set consists of 1800 points taken from the El Camino Real near Redwood City and CA 280 near the Page Mill Road, of which 600 are positive samples and the other 1200 points are negative. The setting of the SVM is to use L2-regularized L2-loss support vector classification.

To select a better model, we use ten-fold cross validation. The model space we scan through is where P_s goes from 20 to 50 and b (number of bins in the histogram of a cell) goes from 20 to 160. Other factors such as block size and cell size are not expected to be important. In our work, we set cell size to be 4 and each block consists of 2×2 cells. The result is shown in Fig 2. The plot shows that it is indeed true that the larger the patch size and number of bins, the better the cross validation accuracy. However, the improvement is marginal while the cost of computation time is high when both numbers are large. As shown by eq (1), the feature vector length is $\mathcal{O}(bP_s^2)$. Classification with large values of b and P_s can be computationaly prohibitive.

Therefore we choose to work with the model with patch size 50 and number of bins 40, whose cross validation accuracy is 97.50%. There are 561 support vectors for this model out of the 1800 training points. For such a model, the length of the HOG feature vector is 23040, which is the same with the one given in eq (2) because 50 is not a multiplier of 4 while 48 is (50 is rounded to 48).

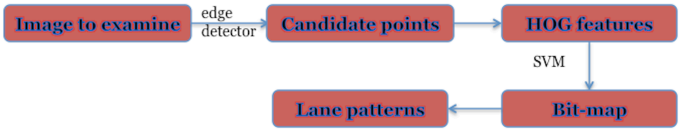

FIG. 3: Examining process for a satellite image. The bit-map is a matrix where 1 means the SVM classifies the corresponding point as a lane-marking point and 0 otherwise.

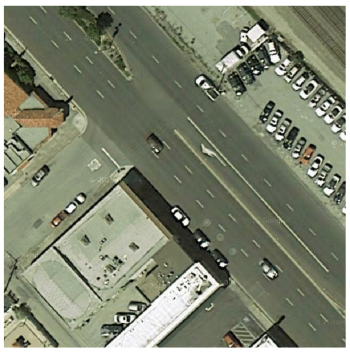

FIG. 4: The original satellite of the example. This image showing a region along the El Camino Real near Red Woodcity is obtained from Google Maps.

IV. MARKING LANES

Having trained a support vector machine as described in the previous section, we can then use it to extract lane patterns from satellite images. This process can be summarized as in Fig 3. An example is shown in Fig 4, 5, 6 and 7.

The step of edge detection can be regarded as a preprocessing step. The total number of pixels in an satellite image can be large. A typical image from Google Maps is $1280 \times 1280 = 1638400$. Examining every pixel on the image can take a long time. However, as argued in section

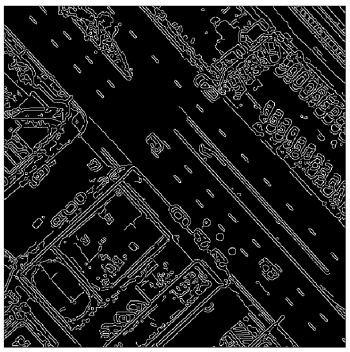

FIG. 5: The result after the preprocessing step to the satellite image shown in Fig 4. The white points are the candidate points that will be further examined by the support vector machine.

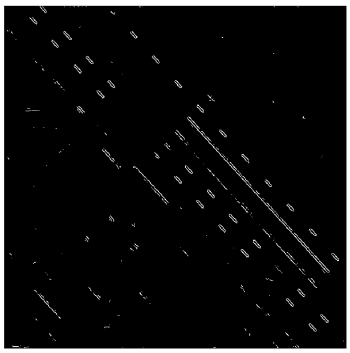

FIG. 6: The bit-map generated by the support vector machine after examining the candidate points as shown in Fig 5.

FIG. 7: The final result after applying a Hough transform to the bit-map as shown in Fig 7. The green lines are the lane markings identified by our technique.

II, the number of points we need to examine is actually relatively smaller, since it is almost always the case that a lane marking is surrounded by a color edge. Therefore adding a layer of edge detector before examining the pixels using the support vector machine can significantly reduce the processing time. For the example we use, the original image to examine is of size $640 \times 640 = 409600$. After applying the edge detector, the amount of pixels we need to examine reduces to 44611.

The number of candidate points can actually be further reduced by applying another layer of simple classification according to the local property of a pixel. The lane marking points typically have a clear white or yellow color. As illustrated by the example in Fig 4 and 5, a large portion of the candidate points after the edge detection step will drop out according to such a classifier. In our current stage of work, however, we have not implemented such a step.

The support vector machine as trained in section III is then applied to examine the candidate points. The features are again the HOG features, generated using the parameters selected in section III. One example of outputs of such a step is shown in Fig 6. It shows that SVM

FIG. 8: Another result from examining a region along CA280. The green lines are the lane markings identified by our technique.

can identify the lane marking points pretty accurate, although there are still some outliers.

We then apply a Hough transform to the bit-map generated by the SVM and look for the peaks in the (r, θ) space. As there are misalignments between the short lane markings that are supposed to be along the same lane and lane markings have a width itself, we set the resolution of r to be 5, that is, lines with a difference of 5 pixels in r are regarded as the same line. We find all the lines whose intensity in the (r, θ) space is greater than or equal to 20% of the maximum intensity. The result is shown in Fig 7.

As one can see from Fig 7, our technique can identify the lanes quite accurately, although there is one short false positive in the middle of the image out of all the true positives. Another example is shown in Fig 8.

At this stage it appears difficult to us to run error analysis beyond the pixel-by-pixel level, since the training set is constituted by the labellings at the pixel level while our final result is given in the form of lines. One might try to run such a diagnostic for the bit-map result given by the SVM. Indeed there are misclassified points as in Fig 6. Those points, however, do not appear in the final result due to the Hough transform. Also the cross validation accuracy can already be viewed as a good proxy for the test error. A better idea would be to manually take all the lane lines for some maps and then test the results given by our algorithm against those lines. We leave designing such diagnostics to future work.

V. CONCLUSION AND FUTURE WORK

In this work we applied three standard techniques from digital image processing to develop a machine learning based technique of identifying lane markings. Parameters for the support vector machine model is selected using a ten-fold cross validation. Using the training set of 1800 points we generated by hand, the cross validation is 97.50%. The performance of the technique is shown in

section IV. With a relatively small training set (manually generated within half an hour by one person), it can already accurately identify almost all of the lane markings.

It is also worth noting that the lane marking we found also contains the latitude-longitude information. As mentioned earlier, the goal of project is to produce lane marking information for a vehicle to use in real-time. With the latitude-longitude information of the lane markings, the vehicle can use a GPS and other relatively simple programs to compute its location with respect to the its nearby lanes.

There are also some interesting future directions we can follow. The most straight forward extension is to parallelize our technique. At the moment, our implementation is single-threaded. As a result, it takes a relatively long time to finish examining one image. Since we basically do the same thing for every pixel and the work done for each pixel does not depend on the work done for the other pixels, it is very natual to parallelize the process. We expect a large boost in run time efficiency when applying GPU computation to our technique.

Another extension would be to generated more training data. We will need to collect training samples of other types of lane markings such as the singe or double solid yellow lines. It also came to our attention lately that an SVM trained using horizontal or diagnoal lanes does not perform very well for identifying vertical lanes. Therefore training set with different lane orientations is in need. A larger amount of training sample from images with diverse photometric conditions is also expected to increase the quality of our technique.

The images we currently use are those taken by satellites. However, there are cases where such images are not available, such as the roads inside tunnels. In those cases, we can still apply our technique by first taking images on the ground along with the relevant GPS information. We expect our technique to work for those images at least as well as the satellite images, since in a real-time image of a road, there are typically less color edges that "look very like" (loosely defined) lane markings but are actually not.

Acknowledgments

We thank Tao Wang for providing the project topic and helpful discussion, and Prof. Ng for giving such a nice course. We also appreciate the hard and helpful work done by the TAs.

[1] R. Gopalan, T. Hong, M. Shneier, R. Chellappa, *A learning approach towards detection and tracking of lane markings*, Intelligent Transportation Systems, IEEE Transactions on (Volume:13 , Issue: 3), 1088 - 1098

[2] LIBLINEAR – A Library for Large Linear Classification, `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

[3] Dalal, N, Triggs, B., *Histograms of oriented gradients for human detection*, Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference, 886 - 893 vol. 1

[4] Canny edge detector, `http://en.wikipedia.org/wiki/Canny_edge_detector`