Video Processing on Vehicle Front-View Camera

EE368: Digital Image Processing Final Project

Yaqi Zhang
Department of Electrical Engineering
Stanford University
Stanford, California 94305
Email: yaqiz@stanford.edu

***Abstract*—In this project, image processing techniques are used to develop basic understanding on video recording taken by vehicle's front-view camera. By post processing on the video recording, the computer is able to extract road sign and traffic light information, and make prediction on how fast the vehicle is moving and steering. The road sign recognition is able to identify most of the signs in the video, and traffic light detection achieves decent accuracy on three videos from KITTI benchmark [1]. Forward and steering angular velocity have mean squared error around 0.94 m/s and 0.31 deg/s.**

I. Introduction

With increasing technology to improve driving security, surrounding camera is increasingly popular among recent models of family using vehicles. With abundant information collected by these cameras, there are few existing practices that automatically analyze and understand the content of the recording. By extracting information from the video, computer can better understand the driving condition and surrounding environment. The most common application is in the field of self-driving, where vehicle needs be able to detect objects and ground lane, and understanding traffic light as well as signs, to make decision about its direction and speed. For vehicle that are not self-driving but equipped with front-view cameras, it is still valuable to post process and analyze the recording by making inference on drivers decision based on detected object and other information learned from the recording, and potentially provide appropriate advice on drivers behavior.

This project focuses on three aspects of driving information:

- Detecting road signs
- Recognizing traffic light (exist or not and color)
- Predicting driving speed and steering speed, and making prediction on state of the vehicle (e.g. Forward, Still, Turn Left)

The major challenge for object detection is that front-view cameras usually do not have very high resolution. Additionally, due to distance to the detection target, the target usually occupies only a small region in each frame and contain limited number of pixels. This imposing challenge on making detection based on geometry and pattern of the target, as an edge no longer appears to be a line or a circle after discretizing into a few pixels. This project addresses the issue by blurring the comparison template and uses a bounding box to threshold thethe geometry of the shape. For speed prediction, the challenge comes from the fact that source of the video is a moving vehicle, which has non-zero vertical velocity and rotation along the forward axis. Moreover, moving objects such as pedestrian and other vehicles introduce additional interference on prediction of the speed of vehicle relative to ground. To overcome the issue, averaged flow inside a small window rather than original flow of each pixel is used in the learning model.

The dataset used in this project are three raw videos from the KITTI benchmark suite [1]. The labels on forward velocity and angular velocity along upward axis are used for training and testing purpose. The original dataset shoot in China mentioned in proposal was not used because the video does not contain velocity label.

II. Image Processing Pipeline

A. Roadsign Detection

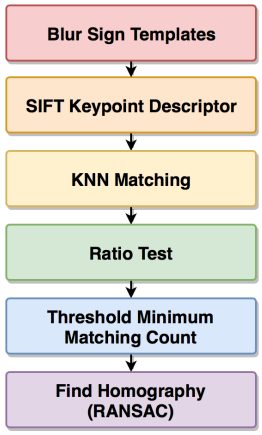

Fig. 1. Roadsign Detection Flow

Basic image processing flow for road sign detection is shown in Fig. 1. For the interest of this project, only 16 signs listed in Fig. 2 are used in template database. In order to detect signs, all sign templates are matched individually to each frame. Due to the shooting distance from the vehicle to road signs, the appeared signs have very low resolution in the original recording. This problem is illustrated in in Fig. 3, where each individual sign is only a few pixels wide and details inside the signs are very blurry. Additionally, the local illumination condition is affected by the direction the sign

Fig. 2. Road Sign Templete

is facing toward and whether it is shadowed by surrounding objects. To overcome the issue, each sign template is blurred by a 5x5 Gaussian filter before matching. Next, SIFT Keypoint descriptor is used for identify keypoints in both templates and frames. Then Hamming distances of the matched descriptors are sorted and the matches with lowest distance are selected. K Nearest Neighbor (KNN) matching with $k = 2$ is used in order to perform ratio test [2], with the assumption that best matching has much closer Hamming distance than the second best.

$$Dist(BestMatch) < 0.7 \times Dist(SecondBestMatch)$$

In order to speed up the matching process, Fast Library for Approximate Nearest Neighbors (FLANN) [3] is used to find the closest matching. As shown in Figure 4, the approximated matching almost produces identical result compared to the exact match in OpenCV implementation.

In early attempt of the project, Hough circle transform was considered to identify round signs, which assists feature matching by localizing the signs in the image. However, Hough transform is subject to false positive when detecting really small circles. In Fig. 5 (a), although the round sign is correctly picked out by Hough transform, many false positive circles are introduced (the thresholds were adjusted to just allow the correct sign to be identified and minimize falsely identified circles). When applying the same set of threshold parameters to a second frame, Hough transform fails to pick out the red “no entry” sign, which is very small and blurry. Overall, it is really hard to find a set of thresholds that performs consistently well for all frames with a large variation in round sign radius. Therefore, direct matching between sign templates and the frames is used, which later we can see that global matching can cause inaccuracy under certain scenarios.

Fig. 3. Low Resolution Road Sign

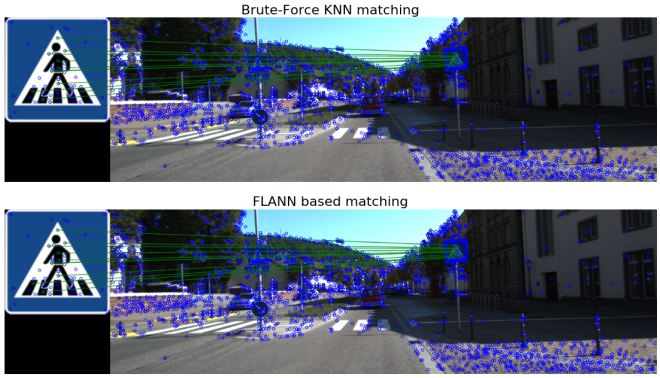

Fig. 4. Nearest Neighbor Matching Comparison

B. Traffic Light Recognition

The original idea of recognizing traffic light is also using Hough circle transform to identify three circles that are roughly of the same radius, and whose centers are roughly aligned, which can be detected using Hough line transform over centers of all detected circles. This approach facing the same challenge that low resolution traffic light does not appears to be a circle, and three lights does not light up at the same time. The alternative approach is to use color to localize the potential candidates of traffic light and using additional conditions to filter out false positives. The assumption is that since traffic lights are color sources, the RGB value of the lights themselves are relevantly invariant across different scenes and illumination condition. To identify the color, four color masks for red, yellow, green (actually sky blue for green light in Germany), and black are created using following equations [4]. Colors that do not belong to any of these masks will be set to white. To reduce noise, image is blurred before thresholding. An example of a purified scene is shown in Fig. 6 (The color in the image is for visualization purpose

Fig. 5. Detecting round sign with Hough Circle Transform

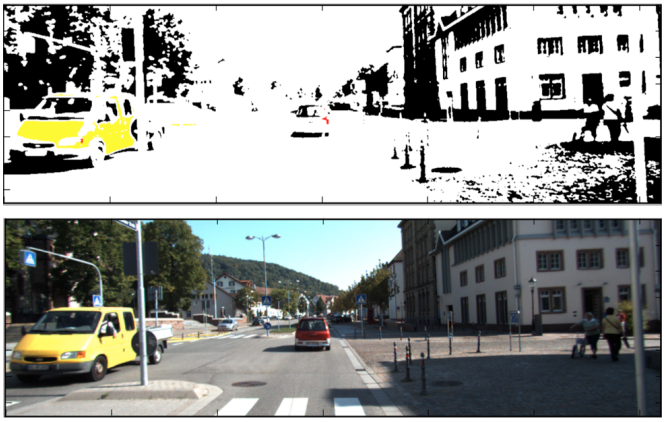

Fig. 6. Purified scene after color thresholding

only. The actual masks are 5 separated binary images). Using red light as an example, contour of each connected component in red mask is computed. Then convex hull is calculated for each contour boundary. If the difference between the maximum diameter (D_{max}) and minimum diameter (D_{min}) is within a threshold, the contour is considered as a "circle". For actual implementation, the process was simplified by fitting a minimum area rectangle (MAR) to the contour, and check whether width (W) and height (H) of the rectangle is within a threshold.

$$|D_{max}[hull[contour]] - D_{min}[hull[contour]]| < threshold$$

$$|W[MAR[contour]] - H[MAR[contour]]| < threshold$$

Any contours that roughly fits in a square would pass the checking criteria above. Therefore, additional checking on surrounding colors of the contour is conducted. Specifically for red light, area below the red region with rough the same width and twice the hight as the qualified red region should be in black for a vertical traffic light (Fig. 7). If let the red region to be at coordinate (0,0), the surrounding color is collected by masking the red region over (logical and with) all color

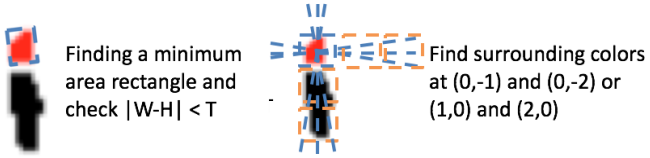

Fig. 7. Red Light Detection

masks at (0,-1), (0, -2) (Resulting color should be one of the 5 masks). The region over these coordinates are considered to be a specific color if the corresponding area contain more than a threshold % non-zero pixel in corresponding color masks, to take into account the fact that the vertical axis of the traffic light can be tilted due to movement of the vehicle. Similar approach can be performed to check a horizontal traffic light using neighbor coordinate at (+1, 0), (+2, 0) instead (though do not seem to exist in KITTI benchmark from Germany). Siimilar approach is used to check color at (0, +1) and (0, -1) for yellow light and (0, +1) and (0, +2) for green light. The corner case where red light and yellow light can light up at the same time can be easily handled by checking if the color at (0, -1) for red mask is either yellow or black and (0,+1) for yellow mask is red or black.

C. Speed Prediction

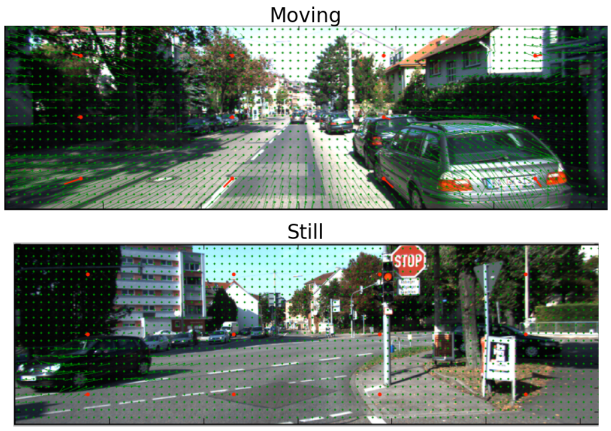

Fig. 8. Dense Optical Flow and Averaged Flow

In order to make prediction on how fast the vehicle is moving and steering, a simple linear regression model is trained to make prediction based on optical flow of the video. Optical flow assume matching pixels in consecutive frames have relationship

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

where d_x and d_y are small changes in space and d_y is small change in time. The equation can be simplified to optical equation [5]:

$$\frac{df}{dx}u + \frac{df}{dy}v + \frac{df}{dt} = 0$$

where $\frac{df}{dx}u$, and $\frac{df}{dy}$ are gradient of the image along space and $\frac{df}{dt}$ is the gradient along time. In this project, dense optical flow based on Gunner Farneback's algorithm [6] is used. To compute dense optical flow, gray scale images of two consecutive frames are input to the algorithm, which outputs a flow matrix with a flow vector for each individual pixel. The dense optical flow is very noisy because the vehicle has non-zero vertical and left-right velocity and rotation. Additionally, there are other moving objects in the scene such as pedestrian and other vehicles. To reduce the effect of relevant moving object in prediction of vehicle's absolute speed. Each frame is segmented into n x m grid windows, where flow vectors inside each window is averaged by taking average of u and v components for all (u, v) in that window. Fig. 8 shows dense flow vectors (shown in green) and averaged flow vectors (shown in red) with 3x4 grids for moving and still scenes. As shown in the moving scene, flow vector close to the gray vehicle on the right has large magnitudes and angles along body of the vehicle due to its vicinity to the camera. In the still scene, there are non-zero vectors around the black car moving from right to left. Averaged flow, on the other side, is much less prone to noise introduced by local movement or inaccuracy in flow computation (mostly around region with similar colors such as ground or trees). There are two linear regression models that both take in magnitudes and angles of all averaged flow vectors as input (concatenated as an array) and are trained to produce forward velocity and angular velocity along upward axis separately. To make an inference on state of the vehicle (Still, Forward, Turning Left, or Turinig Right) the model simply thresholding on the predicted velocity and angular velocity.

III. Experiment Setup

This project uses three videos from the raw dataset in the KITTI benchmark suite [1]. In order to figure out the optimal segmentation size for the linear model used in speed prediction described in section II-C, a space exploration experiment was conducted by sweeping the horizontal and vertical segmentations for velocity and angular velocity. Error of the model is measured by mean squared error of the prediction (Percentage error is not a valid metric here because a very small speed would naturally correspond to a very large percentage error and speed at zero corresponds to error of infinity).

$$MSE = mean\big((\hat{v} - v)^2\big)$$

Among the three videos, 80% of the frames are randomly sampled for training and remaining 20% are used for testing. Road sign matching and speed sweeping experiments are really time consuming because feature matching and optical flow require significant computation. These two experiments are pre-computed on a server machine with 1 TB of RAM and 140 processors, and results are loaded from files when running the demos. Road sign matching was parallelized with over 600 threads for about half an hour and sweeping experiments was parallelized around 250 threads runing for 1 hour and 20 min.

TABLE I
Road Sign Detection Accuracy

signs	Identified	Misidentified
children_crossing	True	True
give_way	False	False
keep_right	True	False
no_entry	False	True
no_parking_end	True	False
no_parking_start	True	False
no_stopping	True	True
no_stopping_end	True	False
no_stopping_start	True	False
parking	False	True
parking_area_end	True	False
parking_area_start	True	False
pedestrian_crossing_left	True	True
pedestrian_crossing_right	True	False
speed_limit_30	True	False
stop_sign	True	True

Traffic light detection is very simple operation performed on binary masks and hence is computed on the fly.

IV. Experimental Result

Due to time limit and missing road sign labels for the data, there is not quantitative evaluation on precision of the detection other than speed prediction. For road sign detection and traffic light recognition, a rough statistics of correctly and incorrectly recognitions are provided. A few scenarios under which recognition failed are also analyzed.

A. Roadsign Detection

Fig. 9. Correct Sign Detection

Fig. 9 shows a set of scenes where road signs are correctly identified while Fig. 10 shows a set of scenes where road signs

Fig. 10. Incorrect Detection or Unrecognized Signs

Fig. 11. Feature matching affected by illumination condition

are either unrecognized or misidentified (incomprehensive). Most misclassification are due to the fact that original sign templates do not have a lot of keypoints, and hence is really easy to falsely matched to random keypoints in the scene. The major cause of unrecognized sign are usually the low resolution of the signs in the scenes.

Apart from resolution, Fig. 11 shows a situation where illumination affects the quality of the feature matching. Although, in theory SIFT descriptors should be invariant to lightening condition of the picture, we can see that there are less keypoints identified from the sign in video in Fig. 11 (a) than Fig. 11 (b), where Fig. 11 (b) is obtained by performing histogram equalization on a small window of the frame. One hypothesis about this outcome is that histogram equalization changes the gradient of the edges around the keypoint, which allows them to standout when picking the top most keypoints in SIFT descriptor. As a result, the sign is matched after appropriate illumination adjustment and localization. One approach to identify this sign is to perform a sliding window on the frame, where each window is first histogram equalized and then matched to the template. However, sliding window is not used in the entire video processing, as it significantly slow down the matching process and only works for a few occasions after tuning, and also introduces false positive recognition.

The other challenge comes with road sign detection is that some signs are really simple in its pattern and geometry, such that there are really few keypoints associated with the sign template itself. An example of such road sign is the "no entry" sign shown in Fig. 12. The sign only has a few keypoints around two ends of the weight bar, which corresponds to a 1 pixel wide horizontal bar in the video frame. The other hypothesis is that for the symmetric signs, there are actually

Fig. 12. Road sign with very simple pattern

Fig. 13. Identical signs after rotation

two almost equally good matching by rotating the signs 180 degree since the matching is rotation invariant. This will unfortunately fails the ratio test, as the test assumes the best matching is significantly better than the second best. The other interesting signs are "keep right", "keep left", and "ahead only", which are identical after rotation. (Fig. 13). As a result, rotation angle in homography has to be used to identified the which is the actual matched sign (not a problem in this dataset). The need of localization is also reflected when a sign or similar signs appear multiple times in a scene, which will result in more than one good matching in global matching.

B. *Traffic Light Recognition*

Fig. 14. Color masks of traffic light

Although the set of hand crafted rules developed for traffic light detection is very specific to traffic light's layout and the color threshold might be sensitive to different regions and countries, it is able to identify traffic light that are only 2-3 pixel wide in the video when they are quite far away. The color threshold is also relevantly stable for the set of video tested. Fig. 14 shows a list of detected traffic lights and their color masks. Fig 15 shows set of scenes where traffic light are correctly identified. For the video tested, there is no traffic light undetected or mis-colored. However, there is false positive where back light of the black vehicle is identified as red light, purely because the hand crafted rules fail under such occasion Fig. 16.

Fig. 15. Correct Traffic Light Recognition

Fig. 16. Incorrect Traffic Light Recognition

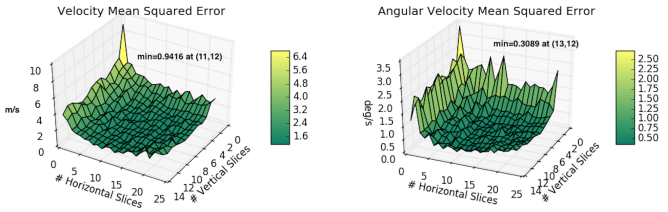

Fig. 17. Space Exploration on Degree of Segmentation on Testing Error

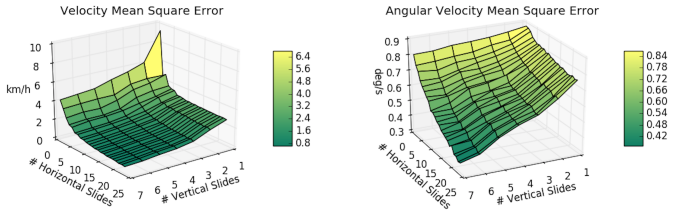

Fig. 18. Space Exploration on Degree of Segmentation on Training Error

C. Speed Prediction

In order to find the optimal segmentation size, we can explore the space by sweeping the number of horizontal and vertical slices in segmentation that minimizes the minimum squared error. One extreme of the segmentation is to directly use the dense flow matrix as inputs (with 1 pixel wide and height window size), while the other extreme is treating the entire frame as one window (with original image's width and height as window size). Intuitively, both extrema will not perform well because the first extreme suffers from noise in fine-grain flow matrix while the other is too coarse-grain to capture any variation in different scenarios. The result of the space exploration is shown in Fig. 17. As expected, the minimum MSE occurs at middle of the parabola with 11x12 number of windows at 0.9416 m/s and 13x12 at 0.3089 deg/s for velocity and angular velocity. One interesting result shown in Fig. 18 is that when using all frames rather than 80% and 20% for both training and testing, the training error have a completely different surfaces that favors maximum segmentation. This is because of overfitted model that although gives minimum error at maximum segmentation, too much details caused by the noise are captured by the model. As a result, its output is very sensitive to variation in the input, and will perform poorly on any unseen data. A more rigorous testing error can be calculated with k-fold cross-validation. However, due to the amount of computation time it requires, it is not explored in this project.

V. Limitation and Future Work

The apparent drawback in approach for road sign detection in this project is the matching speed, which make it impossible to perform real time recognition. The thresholding for traffic light is also sensitive to color in different country and can not be generalized to a large variation of different lights. For speed recognition, one potential improvement is instead of using dense optical flow, Lucas-Kanade's optical flow [7] can be used to only keep track of flow of major keypoints, which might improve accuracy due to noise reduction. Furthermore, a better learning model other than linear regression with more training inputs can also improves accuracy and generalization to different videos.

VI. Demonstration

Three videos from KITTI dataset with labeled detected signs and traffic lights, and predicted speeds are available as demos. The project is on github repository: https://github.com/blackwings-01/DrivingPerception.git. Due to size of the repository (containing data), cloning the repository can potentially fail due to connection timeout. So please download the zip file instead at https://github.com/blackwings-01/DrivingPerception/archive/master.zip. More information about how to setup and run the demos are available in `README.md`.

VII. Conclusion

This project uses image processing techniques to develop a basic "video interpreter" of vehicle's front-view camera, which allows computer to understand some important information about the driving by "watching" what driver can see. In reality, there are much more advanced techniques that can perform the same tasks with much better accuracy. However, the road sign detection in this project is able to achieve an moderate accuracy. Comparably, traffic light detection is more robust, and velocity predictions are able to achieve a reasonable accuracy.

REFERENCES

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pp. 331–340, INSTICC Press, 2009.

[4] C. Yu and Y. Bai, "A traffic light detection method," in *Advanced Technology in Teaching*, pp. 745–751, Springer, 2012.

[5] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 433–466, 1995.

[6] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*, pp. 363–370, Springer, 2003.

[7] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision.," in *IJCAI*, vol. 81, pp. 674–679, 1981.

APPENDIX

This project is related to a research project in computer vision, in which the same dataset is used and tasks are performed but detection and prediction are using convolutional neural network. The result of this project will become a baseline as a comparison to the research project. However, the implementation of this project was performed independently.