Vehicle’s Lane-changing Behavior Detection

Mengwen He
CyLab-ECE
4720 Forbes Ave. CIC 2223G
mengwenh@andrew.cmu.edu

Iljoo Baek
CyLab-ECE
4720 Forbes Ave. CIC 2224G
ibaek@andrew.cmu.edu

Abstract

The lane-level localization accuracy is very important for autonomous vehicles. The Global Navigation Satellite System (GNSS), e.g. GPS, is a generic localization method for vehicles, but is vulnerable to the multi-path interference in the urban environment. Integrating the vision-based relative localization result and a digital map with the GNSS is a common and cheap way to increase the global localization accuracy and thus to realize the lane-level localization. This project is to develop a mono-camera based lane-changing behavior detection and tracking algorithm module for the correction of lateral GPS localization. We implemented a Support Vector Machine (SVM) based framework to directly classify the driving behavior, including the lane keeping, left and right lane changing, from a sampled data of the raw image captured by the mono-camera installed behind the window shield. The training data was collected from the driving around Carnegie Mellon University, and we compared the trained SVM models w/ and w/o the Principle Component Analysis (PCA) dimension reduction technique. Next, we intend to compare the SVM based classification method with the CNN method.

1. Introduction

The autonomous vehicle highly relies on the accurate localization technique because it enables reliable planning and control operations for the safe autonomous driving. The Global Navigation Satellite System (GNSS), e.g. GPS, GLONASS, Beidou (Compass), and Galileo, provides commercial localization devices with affordable price but low accuracy to vehicles. It works well while driving on the highway; however, it is vulnerable to the multi-path interference caused by trees, buildings, or overhead bridges in the urban area.

The prevalent method to enhance the global localization accuracy is to use 3D Light Detection and Ranging (LiDAR) sensors, e.g. Velodyne, to conduct registration with 3D point-cloud map. This can guarantee centimeter level localization accuracy [13] so that this method can be found on various commercial or experimental autonomous vehicles like Google, Baidu, Uber, and Toyota. However, this method is very expensive because of the 3D LiDAR and dense point-cloud map.

Therefore, the common and cheaper camera-based methods are preferred for affordable autonomous vehicles. The visual odometry method can achieve decimeter level relative localization [9]. By implementing filter-based, e.g. Kalman filter and its variants, or graph-based, i.e. nonlinear least squares, methods, we can fuse the GNSS global localization with the visual odometry relative localization to get an enhanced global result.

In this project, we want to realize a vehicle’s lane-changing behavior detection and tracking algorithm based on a mono-camera. Following the integration of GNSS and camera, we want to use the road lane information from a mono-camera to enhance the global localization result; therefore, the detection and tracking of the lane-changing behavior is required to tell which lane the vehicle is on. Coupled with a digital map with road lane information, we can laterally reduce the global localization error from the GNSS as shown in Fig.1.
We employed a Support Vector Machine (SVM) based framework to detect the lane-changing behavior, and the training and test data/feature is directly sampled from the raw image’s region-of-interest (ROI) as shown in Fig.2. Therefore, we performed a Principle Component Analysis (PCA) dimension reduction technique to compress the feature’s dimensionality as well as to keep more than 98% of its original energy.

Moreover, because of the limitation of data size, we choose to use SVM instead of the Convolution Neural Network (CNN) \(^1\) to classify the lane-changing behavior as lane-keeping, right lane-changing, or left lane-changing. But, we plan to compare the training results between the SVM and CNN using the same training data.

We conducted the experiment around Carnegie Mellon University with a Go-Pro HD Camera (Hero 4, 1920 × 1080) mounted after the window shield of Iljoo’s vehicle. We used the off-the-shelf LibSVM [5] to train the lane-changing behavior classifier w/ or w/o PCA dimension reduction.

2. Related Work

We first referred to [8] which uses the SVM for lane-change detection. The features they used are all from the vehicle’s CANBUS containing the steering angle, speed, etc. However, because this method does not use the camera or other external sensors to observe the environment, especially the strongly related lanes, it cannot perform perfectly for the real application, e.g. on the curved road or under the driving situation with large speed variance. [11, 7] both implemented Dynamic/Hidden Markov Models to characterize and detect driving maneuvers. They used a simulation platform to collect drivers’ behavior as well as the simulated vehicle’s steering angle. However, this method only provides the simulated experiment results, and it is necessary to develop more general models and assure robustness corresponding to actual driving situations. [10] conducted many real experiments with cameras which observe the front as well as the driver’s appearance and behavior. This dissertation’s objectives are 1) to characterize normal lane changes in which a slow lead vehicle was present, 2) develop predictive models distinguishing baseline (straight-ahead) driving from lane changes, and 3) provide design guidelines relevant to the issuing of lane change collision warnings. After statistically analyzing the collected huge dataset, the author implemented a Logistic Regression to build the predictive model for lane-change behavior. The result is good, however, in this project, we don’t have too much time or such resources to follow his work. In [12], the author try to map a driver’s observable actions to their unobservable intentions, say lane-change, and this method is named as mind-tracking. To build this map, they implemented many cognitive models of driving behaviors to inference the driver’s behavior. However, similar to [8, 11, 7], we think this method still does not touch the direct clue for lane-change detection, the relative position between ego-vehicle and lanes captured by camera.

3. Approach and Algorithm

3.1. Assumptions

For the time limitation, we make some assumptions to simplify this project; however, in the future research, we will gradually get rid of these assumptions.

- This project will only work on the road with clear lane markers, and the intersection is not in its scope.
- The occlusion caused by other vehicles will not be considered in our training and test data.
- An accurate global digit map already exists and contains the lane information.
- The initially occupied lane is known from an upstream algorithm module.

---

\(^1\) 18-794 Pattern Recognition Theory homework assignment 3
3.2. Training Data Collection and Labeling

While training the SVM, it is very important to choose the most effective set of features that can represent significant differences during a lane change against lane keeping. We have tried three types of methods to collect the desired features in this project. For the labeling, the classifier has two layers of decision as shown in Fig.3. It firstly figures out whether the vehicle changes the lane or not, and then it finds the direction of lane change.

3.2.1 Hough Transformation Based Method

We firstly tried to extract the lane position information by detecting lanes using open source lane detectors [1, 2], and both of them implement the Hough transformation in the OpenCV to extract the lane edge. However, the detectors could not provide robust feature information because the ratio of true negative was too high. Additionally, the frequent false positive result (e.g., Fig.4) from the detectors has a negative impact on the SVM’s performance.

3.2.2 Deep Neural Network Based Method

Then we used the deep learning based SegNet tool [3, 4] to distinguish the lane markers from a road scene. However, the lane markers were often misclassified as the road in the segmentation result as shown in Fig.5. This is also not suitable for the SVM based classifier.

3.2.3 ROI Sampled Edge Features

We tried to extract the binary edge information from a pre-defined ROI as shown in Fig.2. The ROI has four layers and each layer represents a distance (1m, 10m, 20m, 30m) from the vehicle. The ROI pixels are converted to gray, blurred by Gaussian, and filtered by Canny Edge detector. According to our test, these binary edge features are not stable enough to provide continuous information for the SVM classifier.

3.2.4 ROI Sampled HOG descriptor

Lastly, the Histograms of Oriented Gradients (HoG) [6] was applied to a rectangular area which includes the four layers and intermediate areas between the layers as shown in Fig.6. We converted the sub image to gray, and computed gradients in 16x16 cells with 8 orientations. We did not conduct block overlap normalization because our test result showed insignificant performance improvement with the block normalization.

3.3. PCA Dimension Reduction

The training data/feature’s dimensionality is 16,000; therefore, we want to use the PCA to shrink its dimensionality and focus on the main components. Moreover, because the dimensionality is greatly larger than the data size, we will use the Gram Matrix Trick to accelerate the eigen decomposition operation.
3.3.1 Gram Matrix Trick

Given a centralized training data matrix $X$ with dimension $d \times N$, we derive its covariance matrix as $\Sigma = E(XX^T)$. The PCA needs to solve the problem as below:

$$
\begin{align*}
\Sigma \vec{v} &= \lambda \vec{v} \\
XX^T \vec{v} &= \lambda \vec{v} \\
X^TX \vec{v} &= \lambda X^T \vec{v} \\
X^TX \vec{v} &= \lambda X^T \vec{v} \\
X^T \vec{v} &= \eta X^T \vec{v} \\
\end{align*}
$$

Therefore, we only need to solve the eigen decomposition of the Gram matrix $X^T X$ with dimension $N \times N$. To get the final eigenvectors $\{\vec{v}_i\}$, we need to solve:

$$
\begin{align*}
XX^T \vec{v} &= \lambda \vec{v} \\
\vec{v} &= \eta X^T \vec{v} \\
\end{align*}
$$

Then we can use $\Sigma \vec{v} = \lambda \vec{v}$ to derive all the corresponding eigenvalues $\lambda_i$.

3.3.2 Dimension Reduction

Firstly, We sort the derived eigenvalues (associated with the corresponding eigenvectors) in a descending order, and we calculate the total energy as below:

$$
\Lambda = \sum_{i=1}^{d} \lambda_i
$$

Then, we choose the first $M$ biggest eigenvalues whose summation is just larger than a threshold ratio $r$ of the total energy $\Lambda$.

$$
\sum_{i=1}^{M} \lambda_i \geq r \Lambda \\
\sum_{i=1}^{M-1} \lambda_i < r \Lambda
$$

Finally, we use the first $M$ biggest eigenvalues’ corresponding eigenvectors to form a PCA dimension reduction matrix $P$ as below:

$$
P = [\vec{v}_1, \ldots, \vec{v}_M]
$$

Therefore, the dimension reduced new centralized training data matrix is $X' = P^T X$. If we use the PCA dimension reduction on the training data as well as the SVM, we also need to apply the dimension reduction on the test data $Y$ following these two steps:

1. Centralize the test data with the mean ($\mu_X$) of the training data.

2. Reduce its dimension to get $Y' = P^T (Y - \mu_X)$

3.4 LibSVM Training and Test

The LibSVM already provides some off-the-shelf applications to conduct and evaluate the binary classification, and what we need to provide is the training and test data following the format required by the LibSVM. The data is stored in ASCII format, and each row in it presents a labeled feature. Because the LibSVM supports sparse matrix operations, the labeled feature is stored as a label followed with a sequence of indexed non-zero values as below:

-1 1:1 11:1 18:1 20:1 37:1 42:1 59:1 +1 5:1 18:1 19:1 39:1 40:1 63:1

4. Experiments and Progress

The implementation pipeline of this project is shown as Fig.7.

4.1 Data Collection and Labeling

A tool shown in Fig.8 was developed to create the training and ground truth data. We defined four classes which are ‘Lane Keep’, ‘Lane Change - Left’, ‘Lane Change-Right’, and ‘Unknown’ (when the vehicle is at intersections where no lane information is available). The class type is manually determined, and the feature data is automatically labeled using the user interface provided by this tool. The...
Figure 8. The tool to collect data and extract feature.

Figure 9. Compare the Canny features extracted from the ‘Lane Keep’ behavior and the ‘Lane Change - Left’ behavior.

Figure 10. Compare the HoG features extracted from the ‘Lane Keep’ behavior and the ‘Lane Change - Right’ behavior.

Figure 11. The locations where we collect the training data (blue) and test data (red).

4.2. SVM Training Result

The training and test data are collected from the roads around Carnegie Mellon University as shown in Fig.11. From these two roads, we extracted and labeled 4059 training data and 1611 test data.

We compared the accuracy rate between using Canny features and HoG features as shown in Tab.1. For the canny feature, we also compared the classification results w/ and w/o PCA dimension reduction pre-processing. The performance gap between linear SVM and Kernel SVM was narrow in our test because most of the features in the ROI are regularly distributed around left and right lane markers. But the performance of HoG feature is better than that of Canny feature. The canny feature can extract the edge of lanes (also work on other object like vehicle, curb, etc.), but it only represents the edge in pixel level. However, the HoG feature can not only response to the lane’s edge but also represent its orientation. Therefore, the HoG feature is more suitable than the canny feature for the lane-change behavior detection.

| Feature | Pre-Process | L-SVM | K-SVM |
|---------|-------------|-------|-------|
| Canny   | w/o PCA (16,000) | 53.4% | 55.9% |
|         | w/ PCA (362) | 53.1% | 55.9% |
| HoG     | N/A (1200) | 66.9% | 68.5% |

Additionally, we use the confusion matrices to show the details of each SVM based classifier’s accuracy rate as shown in Tab.2,3,4. The SVM classifier using Canny edge feature completely failed to detect lane-change behavior due to the insufficiency of edges in the test scenes and so many noisy features from the front obstacles in both the training and testing data sets. Because of the orientation representation in HoG feature, the HoG based SVM classifier can efficiently detect the lane-change behavior. However, it generated so many miss classifications between ‘Keep’ and ‘Unknown’. But in this test scene, the vehicle really kept its direction (a kind of ‘Keep’) during the ‘Un-
known' data segments, and at the most time of approaching intersections, the ROI contains the lane information of the road after the intersection. Therefore, if we omit these miss classifications, the accuracy rate can reach 80%. To further investigate the performance of the classifiers, we present the classification results compared to the ground truth as shown in Fig. 12.

Table 2. Confusion Matrix using Canny Features

| GT    | Res. | Keep | Left | Right | Unknown |
|-------|------|------|------|-------|---------|
| Keep  | 900  | 0    | 0    | 0     | 0       |
| Left  | 305  | 0    | 0    | 0     | 0       |
| Right | 124  | 0    | 0    | 0     | 0       |
| Unknown | 281  | 0    | 0    | 0     | 0       |

Table 3. Confusion Matrix using Hog Features and L-SVM

| GT    | Res. | Keep | Left | Right | Unknown |
|-------|------|------|------|-------|---------|
| Keep  | 808  | 22   | 21   | 49    |         |
| Left  | 135  | 133  | 0    | 37    |         |
| Right | 23   | 0    | 94   | 7     |         |
| Unknown | 214* | 5    | 19   | 43    |         |

* Classification accuracy can be increased to 80% if this unknown misclassification is considered as true positive.

Table 4. Confusion Matrix using HoG Features and K-SVM

| GT    | Res. | Keep | Left | Right | Unknown |
|-------|------|------|------|-------|---------|
| Keep  | 826  | 19   | 24   | 31    |         |
| Left  | 129  | 135  | 0    | 41    |         |
| Right | 28   | 0    | 96   | 0     |         |
| Unknown | 215* | 5    | 15   | 46    |         |

* Classification accuracy can be increased to 80% if this unknown misclassification is considered as true positive.

Table 5. Confusion Matrix using CNNs

| GT     | Res. | Keep | Change (L&R) | Unknown |
|--------|------|------|--------------|---------|
| Keep   | 951  | 22   | 3            |         |
| Change (L&R) | 126  | 331  | 28           |         |
| Unknown | 134  | 14   | 2            |         |

4.3. CNN Training Result

To measure the efficiency of our SVM based lane-change behavior detector, we implemented a CNN based classifier as a baseline. The input of this network is the original image in the same ROI defined in Fig. 2, and the network has 10 layers as Tab. 6. We only trained this network to detect the lane-change behavior (the first layer in Fig. 3); therefore for CNN test, we only have three labels: ‘Keep’, ‘Change’, and ‘Unknown’. The training accuracy is 99.93%, and the test accuracy rate is 79.70%. The confusion matrix of using CNN is shown in Tab. 5, and the detailed performance is shown in Fig. 12.

Table 6. The Construction of CNNs

| # | Type      | Dimension           | Stride |
|---|-----------|---------------------|--------|
| 0 | Input     | 110 × 400 × 3 × 1   | 1      |
| 1 | Conv      | 5 × 5 × 5 × 40      | 1      |
| 2 | Pool (Max)| 2 × 4               | 2 × 4  |
| 3 | Conv      | 5 × 5 × 40 × 80     | 1      |
| 4 | Pool (Max)| 2 × 4               | 2 × 4  |
| 5 | Conv      | 5 × 5 × 80 × 160    | 1      |
| 6 | Pool (Max)| 2 × 2               | 2 × 2  |
| 7 | Conv      | 5 × 5 × 160 × 1600  | 1      |
| 8 | ReLU      | N/A                 | N/A    |
| 9 | Conv      | 6 × 5 × 1600 × 3    | 1      |
| 10| SoftMaxLoss| N/A                | N/A    |

From this test, we found the HoG based SVM classifier performs better than Canny based SVM classifier and is also as efficient as the CNN to detect the lane change behavior. Sometimes, it performs more stable than the CNN, although its accuracy rate is lower than that of CNNs. At the end of the detailed performance comparison shown in Fig. 12, both HoG based SVM classifier and CNNs cannot detect the lane change behavior continuously, because a vehicle just
overtook our vehicle, which is not present in our training data set.

5. Conclusion

From this project, we went through all the steps of SVM based classifier training and testing framework, and thus have handled the SVM tool for other pattern recognition applications in our future research. Moreover, we also practiced the CNN in this project as a baseline. Although the CNN is powerful, we don’t know what exactly happens in the network, what the network refers to, and how to further improve it. However, after trying different features, we find a reasonable and controllable way to improve the SVM performance.

In the future, we want to integrate more information into the SVM classifier to improve the lane-change behavior detection performance, e.g. the visual odometry information from two consecutive images, the motion information from IMU sensor, and the steering information from the CAN-BUS. Furthermore, we can also combine many independent lane-change behavior detectors (mainly focused on the different visual features) together to implement an AdaBoost framework.

References

[1] Eric gonzalez’s lane-detection using the hough transformation in opencv. https://github.com/Eric-Gonzalez/lane-detection.
[2] Funningboy’s carcv using opencv. https://github.com/funningboy/carCV.
[3] Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. http://mi.eng.cam.ac.uk/projects/segnet/.
[4] V. Badrinarayanan, A. Handa, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. arXiv preprint arXiv:1505.07293, 2015.
[5] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3):27, 2011.
[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), volume 1, pages 886–893. IEEE, 2005.
[7] N. Kuge, T. Yamamura, O. Shimoyama, and A. Liu. A driver behavior recognition method based on a driver model framework. Technical report, SAE Technical Paper, 2000.
[8] H. M. Mandalia and M. D. D. Salvucci. Using support vector machines for lane-change detection. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, volume 49, pages 1965–1969. SAGE Publications, 2005.
[9] C. Merfels and C. Stachniss. Pose fusion with chain pose graphs for automated driving. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2016.
[10] E. C. Olsen. Modeling slow lead vehicle lane changing. PhD thesis, Virginia Polytechnic Institute and State University, 2003.
[11] A. Pentland and A. Liu. Modeling and prediction of human behavior. Neural computation, 11(1):229–242, 1999.
[12] D. D. Salvucci. Inferring driver intent: A case study in lane-change detection. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, volume 48, pages 2228–2231. SAGE Publications, 2004.
[13] E. Takeuchi and T. Tsubouchi. A 3-d scan matching using improved 3-d normal distributions transform for mobile robotic mapping. In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3068–3073. IEEE, 2006.