Vehicle's Lane-changing Behavior Detection and Tracking

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 [12] 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.

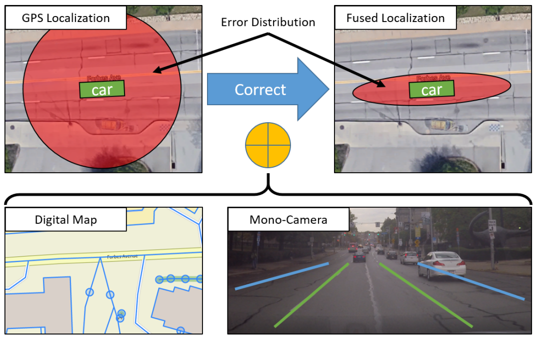


Figure 1. The illustration of our project's objective. We get a global localization result from GPS with a Gaussian error model. By integrating the digital map and the lane information from our lane-changing behavior detection and tracking algorithm, we can laterally correct the global localization error.

Therefore, the common and cheaper camera-based methods are preferred for affordable autonomous vehicles. The visual odometry method can achieve decimeter level relative localization [8]. By implementing filter-based, e.g. Kalman filter and its variants, or graph-based, i.e. non-linear 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.

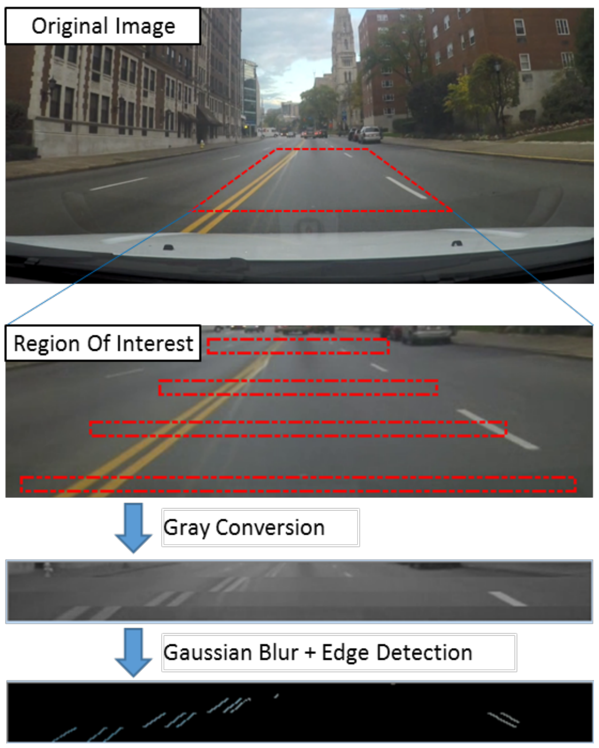


Figure 2. Top: the original image with a ROI in front of the vehicle. Middle: The ROI has four sample layers (red rectangles) and each layer represents a distance (1m, 10m, 20m, 30m) from the vehicle. Bottom: Stack the sampled layers of processed pixels (edge extraction on gray image) row by row to form the feature vector.

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.

Morevoer, 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 classfier w/ or w/o PCA dimension reduction.

[1] 18-794 Pattern Recognition Theory homework assignment 3

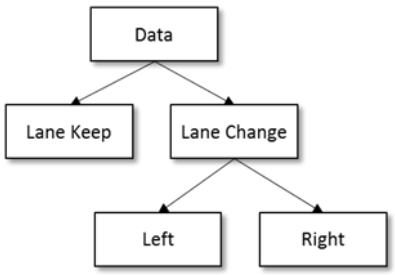


Figure 3. The two layer decision tree converts the three-class classification problem into a two-step binary classification problem.

2. Related Work

We have found 5 related papers [7, 10, 6, 9, 11], and we will finish this part after the mid-progress report.

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.

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 thie 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.

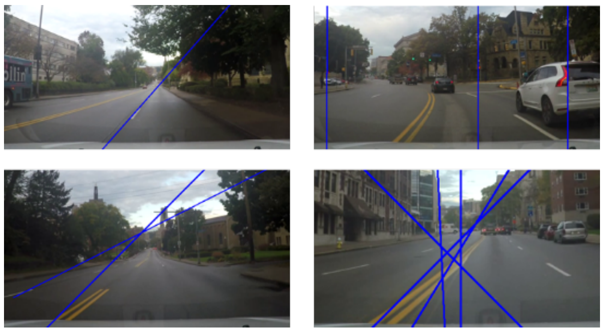


Figure 4. Top left/right [1]: Basic Hough transformation was conducted with canny edge detection result to find and fit lines representing the lane markers. It failed to detect the lanes in curved road scenes. Bottom left/right [2]: Color segmentation and background subtraction were applied to extract lane marker features. Then the features were fitted into lines using the Hough transformation and contour estimation.

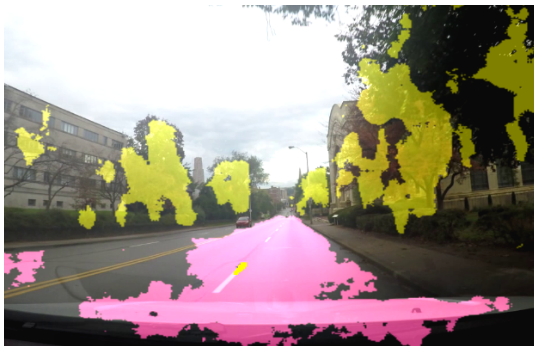


Figure 5. Each pixel in the image is classified through a sequence of non-linear processing layers (encoders) and a corresponding set of decoders followed.

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

Lastly, 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 robust and stable enough to provide continuous information for the SVM classifier.

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:

$$\begin{aligned} \Sigma\vec{v} &= \lambda\vec{v} \\ XX^T\vec{v} &= \lambda'\vec{v} \\ X^TXX^T\vec{v} &= \lambda' X^T\vec{v} \\ X^TX\vec{v}' &= \lambda''\vec{v}' \quad (\vec{v}' = \eta X^T\vec{v}) \end{aligned} \tag{1}$$

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

$$\left\{ \begin{aligned} XX^T\vec{v} &= \lambda'\vec{v} \\ \vec{v}' &= \eta X^T\vec{v} \end{aligned} \right. \Rightarrow \vec{v} = \eta' X\vec{v}' \tag{2}$$

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:

$$\Lambda = \sum_{i=1}^{d} \lambda_i \tag{3}$$

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

$$\begin{aligned} \textstyle\sum_{i=1}^{M} \lambda_i &\geq r\Lambda \\ \textstyle\sum_{i=1}^{M-1} \lambda_i &< r\Lambda \end{aligned} \tag{4}$$

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

$$P = [\vec{v}_1, \ldots, \vec{v}_M] \tag{5}$$

Therefore, the dimension reduced new centralized trainidng data matrix is $X' = P^TX$. 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 (μ_X) of the training data.
2. Reduce its dimension to get $Y' = P^T(Y - \mu_X)$

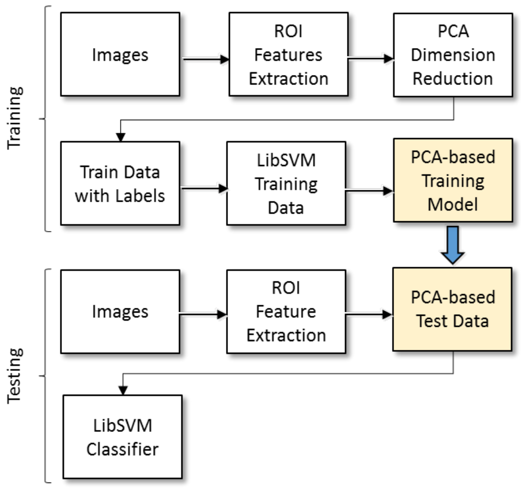


Figure 6. The pipeline of this SVM based lane-changing behavior detection project.

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
```

3.5. Lane Changing Behavior Tracking

Not done yet, will be finished after the mid-progress report.

4. Experiments and Progress

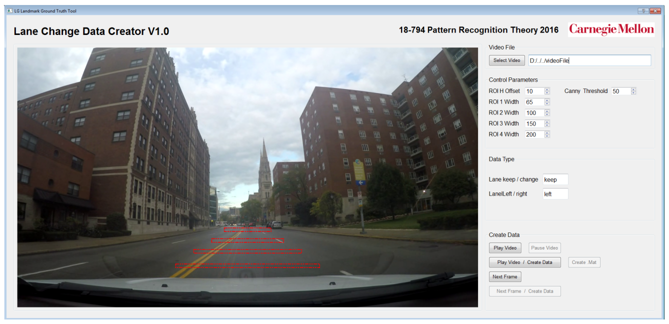
The implementation pipeline of this project is shown as Fig.6.

4.1. Data Collection and Labeling

A tool shown in Fig.7 was developed to create the training and ground truth data. We defined three classes which are 'Lane Keep', 'Lane Change - Left', and 'Lane Change-Right'. The class type is manually determined, and the feature data is automatically labeled using the user interface provided by this tool. The binary edge feature presents the distinguishing pattern between two class types as shown in Fig.8.

4.2. SVM Training Result

The training and test data are collected from the roads around Carnegie Mellon University as shown in Fig.9.


Lane Change Data Creator V1.0
18-794 Pattern Recognition Theory 2016
Carnegie Mellon

Video File
Select Video: D:/../../videoFile

Control Parameters
ROI H Offset: 10
Canny Threshold: 50
ROI 1 Width: 65
ROI 2 Width: 100
ROI 3 Width: 150
ROI 4 Width: 200

Data Type
Lane keep / change: keep
LaneLeft / right: left

Figure 7. The tool to collect data and extract feature.

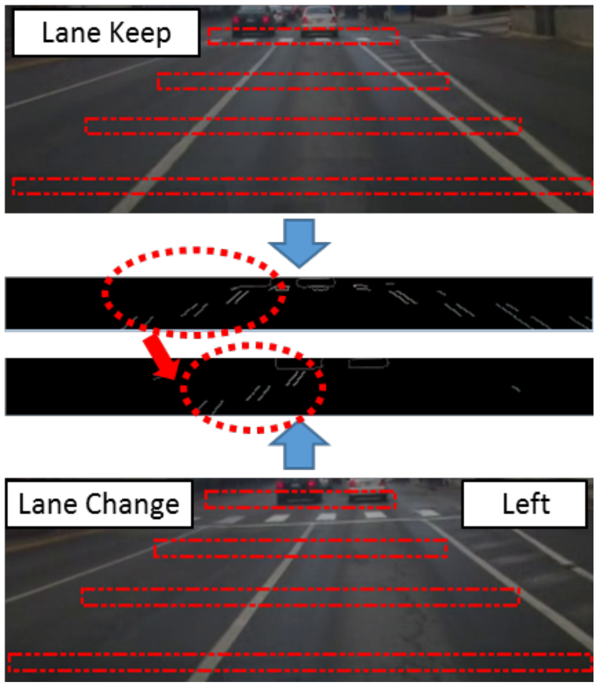


Figure 8. Compare the features extracted from the 'Lane Keep' behavior and the 'Lane Change - Left' behavior.

From these two roads, we extracted and labeled 321 training data and 285 test data. Then we present the training results as two groups: w/o PCA dimension reduction v.s. w/ PCA dimension reduction, and for each group, we also present the training results as two subgroups: w/o RBF kernel v.s. w/ RBF kernel.

4.2.1 Without PCA Dimension Reduction

Without RBF kernel

- Train data: $16,000 \times 321$ (dimension $\times$ data size)
- Test data: $16,000 \times 285$
- Accuracy: 88.0702% (251/285)
- Confusion matrix: Tab.1

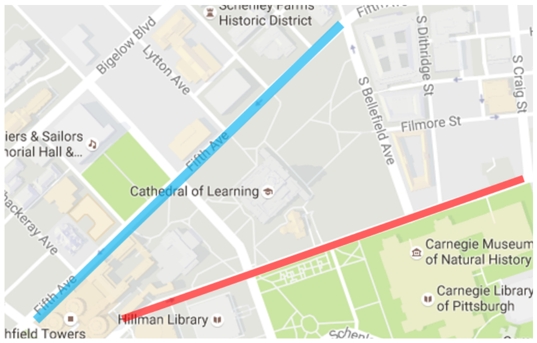


Figure 9. The locations where we collect the training data (blue) and test data (red).

Table 1. Confusion Matrix w/o PCA, w/o RBF

	Lane Keeping	Lane Changing
Lane Keeping	159	19
Lane Changing	15	92

Table 2. Confusion Matrix w/ PCA, w/o RBF

	Lane Keeping	Lane Changing
Lane Keeping	178	0
Lane Changing	107	0

Because the feature is already in high-dimensional space, the linear SVM's accurate rate, 88%, is acceptable with a small size of training data.

With RBF kernel Currently, we did not use the kernel in our experiment, and we will implement it after the mid-progress report.

4.2.2 With PCA Dimension Reduction

Without RBF kernel

- PCA energy: $> 98\%$
- PCA Train data: 154×321
- PCA Test data: 154×285
- Accuracy: 62.4561% $(178/285)$
- Confusion matrix: Tab.2

After PCA dimension reduction, the feature is in a low-dimensional space; therefore, the linear SVM's accurate rate, 62%, significantly reduced compared to the result without PCA and RBF kernel. Actually, from the confusion matrix, we find this SVM classifier does not work, because the output is always 'Lane Keeping'. Therefore, we assert that the dimension reduced feature is not linear-separable, and the kernel is required to increase its performance.

With RBF kernel Currently, we did not use the kernel in our experiment, and we will implement it after the mid-progress report.

4.3. CNN Training Result

Not done yet, will be finished after the mid-progress report.

5. Conclusion

Not done yet, will be finished after the mid-progress report.

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. 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.

[7] 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.

[8] 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.

[9] E. C. Olsen. *Modeling slow lead vehicle lane changing*. PhD thesis, Virginia Polytechnic Institute and State University, 2003.

[10] A. Pentland and A. Liu. Modeling and prediction of human behavior. *Neural computation*, 11(1):229–242, 1999.

[11] 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.

[12] 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.