Lane-level Vehicular Localization Utilizing Smartphones

Invited Paper

Siyu Zhu*, Xiong Wang*, Zhehui Zhang*, Xiaohua Tian*†, Xinbing Wang*
*School of Electronic, Info. & Electrical Engineering, Shanghai Jiao Tong University, China
†National Mobile Communications Research Laboratory, Southeast University, China
Email: {zhusy54, wangxiongsjtu, qiaomai, xtian, xwang8}@sjtu.edu.cn

***Abstract*—Lane-level vehicular localization has been regarded as a critical technology component for future vehicle navigation services. Current lane-level vehicular localization systems require dedicated devices, making the systems difficult to popularize. Moreover, most systems depend heavily on continuous accurate GPS data, which may be interfered under specific environments. Efficient lane-level map building is another problem to deal with. In this paper, we propose an integrated system with the capability of map building and lane-level localization using smartphones. This system employs a crowdsourcing-based approach to collect information from multiple sensors (including GPS, orientation sensor and acceleration sensor), such as lane changes and turns. Based on the information, a lane localization schemes is designed using the tool of machine learning. The experimental results show that the proposed system achieves high accuracy of map building and lane-level localization.**

Keywords: Localzation, Lane Detection, Machine Learning

I. Introduction

Digital maps and navigators are playing more and more important roles nowadays. Accurate vehicle navigation services are highly dependent on road information [1] and real-time vehicle position in the lane level. It is not only helpful to improve the quality of navigation services, but a critical technology for self-driving cars in the future. However, detailed lane information is limited to few regions, and it is difficult to update these information in time. As a result, an integrated system with the capability of map building and lane-level localization is vital for the popularity of lane-level localization.

An intuitive way to localize vehicles in lane-level is using cameras. Image processing techniques can be used for lane information retrieval [2], [3], [4], which is to identify the lane markings on the road from photos taken by the vehicle-mounted camera. Ieng *et al.* [5] even install two cameras to improve the system reliability. However, the performance of image processing is highly depended on light conditions, for example, the performance is poor at night.

Utilizing sensors to detect vehicle actions has been practiced in past years. Motion or rotation sensors mounted on cars or dedicated vehicle detection sensors in the road infrastructure are used to retrieve lane information [6]. Gruyer *et al.* propose a mechanism that fuses localization data, road marking detection and special digital map of the road in order to localize the car in the centimeter level [7], but it requires an accurate digital map for reference and facilitation of two lateral cameras to estimate the position of the car. These information obtained from sensors can help improving the accuracy of localization.

Clustering methods have been used for lane-level localization [8] by mining historical Differential GPS (DGPS) traces whose accuracy is much higher than smartphone GPS. Dao *et al.* in [9] utilize markov-based approach to infer lane-level localization of a car with only GPS data, and same grained vehicle localization is acquired based on mapping sensor data to Terrain [10]. However, both works need extra facilities such as wheel odometry sensors, and Dao's system have to construct an ad-hoc network. In addition, traditional clustering methods requires accurate GPS traces, but the accuracy of GPS is not stable especially when satellite signals is not satisfying, such as in tunnels and between high rises.

In this paper, we propose a machine learning based lane-level localization system utilizing smartphones. Sensor data of smartphones can help identify vehicle lane changes and turn via a Support Vector Machine (SVM) based algorithm. The information is then uploaded to a server together with GPS data. On the basis of uploaded data, the server can build lane-level map by a constrained K-means algorithm. It also sends vehicle locations to users by clustering GPS traces. The main contributions of the paper are as follows:

* An SVM based lane change detection algorithm to identify lane changes and turns.
* A constrained K-means based cluster algorithm with the capability of map building and lane-level localization.

The rest of the paper is organized as follows. Section II describes the architecture of the proposed system. The lane change detection algorithm is introduced in Section III , and Section IV designs the lane localization algorithm. System performance evaluation is conducted in Section V. Section VI concludes the paper.

II. System Architecture

Our system consists of two parts: *smartphone client* and *lane localization server*, as shown in Fig. 1. Smartphone clients calculate lane changes and collect GPS data. Such

978-1-5090-1701-0/16/$31.00 ©2016 IEEE

information is then uploaded to the server via the Internet, and localization results are sent back to smartphone clients.

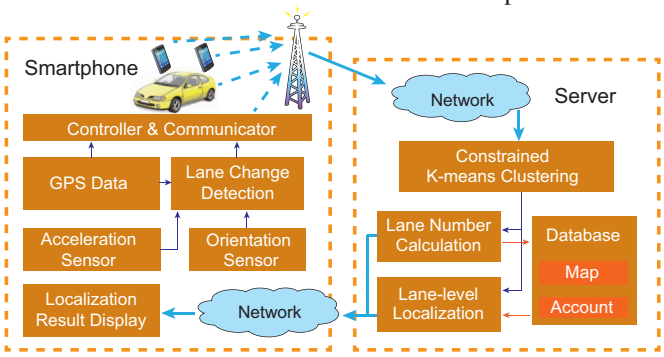


Fig. 1: System architecture.

Smartphone client: Drivers' smartphones first collect GPS, acceleration sensor and orientation sensor data continuously. The module of *lane change detection* obtains the acceleration and orientation data to identify lane changes and turns, and records the GPS data where the action happens. The processed data will be uploaded to the server for positioning.

Once receiving requested lane location at the beginning of a road, the client can change its lane location by detecting lane change actions without the server until the next turn to reduce the amount of uploaded data. This is especially effective when suddenly driving into a tunnel. Lane change information can also help to correct the lane position.

Lane localization server: On the basis of the uploaded GPS data, the server can calculate the number of lanes and their position by *constrained K-means algorithm* to build a lane-level map database. Once receiving the GPS data of a certain user, his/her lane position can be located according to the existing lane-level map. Then the result are sent back to the user's smartphone client.

III. Lane Change Detection

In this section, we introduce an SVM (Support Vector Machine)-based algorithm combining the orientation sensor and acceleration sensor to identify lane changes and turns, which is efficient enough for smartphones. Single sensor systems cannot identify lane changes and turns simultaneously because the orientation sensor is not sensitive enough to identify lane changes while the acceleration sensor can not identify change changes and turns.

Dogan *et al.* [11] compared some machine learning methods in classifying left and right lane changes, including Feed-Forward Neural Network (FFNN), Recurrent Neural Network (RNN) and Support Vector Machine (SVM), and proved that SVM had the best performance in classifying lane changes. Mandalia et al. [12] evaluate SVM to detect lane changes via real sensor data without other actions. However, they all didn't consider the impact of other actions such as turns. In our system, turns can be identified according to the change of orientation data while left and right lane changes can be calculated via the SVM.

A. Process of Lane Change Detection

A turn or a lane change can result in the change of acceleration and orientation. The change in acceleration can be easily detected as long as a car begins to turn or change lanes. But lane changes and turns can not be identified until the action is finished by comparing the driving direction before and after. Moreover, we put the recent acceleration sensor data to the SVM classifier to distinguish left and right lane changes, the model of which is trained in advance.

The lane change information can help locate the lane position real-timely and accurately. Once the initial lane position is determined by the K-means algorithm, the lane position can then be measured according to the lane change information. Moreover, lane change detection can help correct the lane position because a car never change to a non-existent lane. For example, when a car is at the rightmost lane, the lane position keeps unchanged if a right lane change is detected.

B. Lane Change Detection Algorithm

The process of lane change detection is shown in Algorithm 1. We first implement a Kalman Filter to deal with the noisy acceleration data. There is no need to filter orientation data since the raw data is good enough for further analysis. However, the orientation data ranges from 0 to 360 degrees and it must minus or plus 360 degrees when it jumps between 0 and 360 degrees to ensure the continuity.

Here we set moving windows, whose lengths are N, for both sensors. We denote the mean value of orientation data in the moving window at time k as $E_{o,k}$, and its variance as $V_{o,k}$. Suppose the latest sample of orientation data at time k is o_k, then the moving window includes o_{k-N+1} to o_k. Similarly, the variance of acceleration sensor is $V_{a,k}$, and the moving window includes a_{k-N+1} to a_k.

The status of the car becomes unstable when $V_{o,k}$ exceeds the threshold D_o. The mean of orientation at this time is recorded as L_o. Similarly, the car becomes unstable when $V_{a,k}$ exceeds the threshold D_a. If both $V_{o,k}$ and $V_{a,k}$ do not exceed their thresholds, the status of the car is stable.

When an action is finished, $E_{o,k'}$ at this moment is compared with L_o to identify left and right turns, as shown in Algorithm 1. If the direction change does not exceed the threshold, the data stored in the moving window will be sent to SVM and the result can identify whether the action is a left or right lane change.

Note that one action may be detected twice as the change of both types of data exceed their threshold. If one of the sensors has distinguished the action, the system should be notified that the action has been detected, thus the other sensor will not detect the action again.

C. Parameter Selection

Considering the window length N and threshold D_o, D_a have significant impact on the algorithm performance, they should be carefully estimated. The window length should be longer than the number of data samples produced during an action. This should be determined according to the sample

Algorithm 1: Lane change detection

Input: New data at time k of orientation sensor o_k and acceleration sensor a_k, length of the moving window N, threshold D_a, D_o, $\Delta_{dir,k}$.

Output: The action of the car.

```
Calculate the variances V_{o,k}, V_{a,k}, and mean E_{o,k}
if V_{a,k}>D_a ∧ V_{a,k-1}<D_a ‖ V_{o,k}>D_o ∧ V_{o,k-1}<D_o then
    if No on-going actions then
        An action is detected, L_o=E_{o,k}
    else
        Duplicate detection
if V_{a,k}<D_a ∧ V_{a,k-1}>D_a then
    Run SVM classifier
    if direction detector finished ∧ | o_k − o_{k-1} | < Δ_dir then
        The action is the SVM result
    if Duplicate detection then
        Record the SVM result
if V_{o,k}<D_o ∧ V_{o,k-1}>D_o then
    if |L_o-E_{o,k}|>Δ_dir then
        if L_o-E_{o,k}>0 then
            The action is a right turn
        else
            The action is a left turn
    else
        if acceleration detector is finished then
            The action is the SVM result
        else
            Notify that orientation detector is finished
```

rate of sensors and the maximum time of an action. We set $N = 300$ in our test.

A too small acceleration threshold D_a will increase the impact of acceleration noise interference, but a high threshold may miss many actions, and so as D_o. We first collect sensor data from real tests, and calculate the variance $V_{o,k}$, $V_{a,k}$ at each time k. For the acceleration sensor, define the difference between two classes separated by D_a as:

$$H = \mu_0 * (E - E_0)^2 + \mu_1 * (E - E_1)^2,$$

where μ_0, μ_1 are the proportion of variance values larger and smaller than D_a respectively. E_0, E_1 is the mean of the two variance classes separated by D_a, and E is calculated as:

$$E = \mu_0 * E_0 + \mu_1 * E_1,$$

The value of D_a is determined by maximizing the value of H, and so as D_o.

D. Spatial Coordinates Transformation

We need to collect the lateral acceleration data of a vehicle because the acceleration sensor may be interrupted by speeding up and brake. Fig. 2 shows how the device coordinates are set. The orientation data provided by smartphones gives the orientation information of itself. By using this orientation data, we can transform the sensor data into world coordinates. The transformation can be expressed as following:

$$\vec{A'} = \vec{R} \times \vec{A},$$

where $\vec{A}$ denotes the vector of acceleration data in device coordinates and $\vec{A'}$ is that in world coordinates. $\vec{R}$ is the identity matrix when the device is aligned with world's coordinate system, which can be obtained by orientation sensor.

Fig. 2: Device coordinate system.

Moreover, GPS data can provide the angle between driving direction and the North which can be denoted as θ. Then we can transform the sensor data from world coordinates into car coordinates as following:

$$\vec{A''} = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \times \vec{A'},$$

where $\vec{A''}$ is the acceleration data in vehicle coordinates. In this way, we can obtain the lateral acceleration data of a driving vehicle.

IV. Lane Localization

An intuitive way for lane localization is to collect GPS traces and then classify which lane they belong to. However, the accuracy of GPS data is limited, so it can not be used to locate the vehicle without further processing. In this section, we propose a constrained K-means algorithm to localize vehicles in lane level. Since we have known the lane change and turn points, the GPS traces can be divided into segments before clustering. The accuracy of clustering can then be improved. Both the lane level map generation and localization are finished by utilizing this algorithm to cluster GPS traces.

A. Lane Level Map Generator

In the real world, the GPS data tends to aggregate in the corresponding centers of lanes when a vehicle does not change its lane. Traditional K-means algorithm can cluster the GPS traces, but some information, such as lane change points and turns, are not utilized. However, such information which can help improving the system accuracy, such as lane change points and turns. Based on the lane change detection algorithm, we can know the clustering relationships of GPS data, for example, the GPS data of every lane change point. To make full use of the known information, we propose the constrained K-means clustering algorithm in this section. Moreover, the usage in mining GPS traces are also described in this subsection.

1) Constrained K-means Algorithm: The algorithm adds two sample level constraints to the traditional K-means algorithm. The added constraints are based on the lane change infomation obtained from the previous section. The algorithm calculates the result of the clusters that each sample belongs to without breaking all the specified constraints.

Must-link: Specify that some samples must be in the same cluster. In Section III, GPS data is separated into segments based on lane change points. We can intuitively know that the samples between two lane change points must be in one cluster. These samples formulate the must-link constraint M.

Cannot-link: Specify that two samples cannot be clustered in the same cluster. Suppose there is a lane change point between two data samples in data set D, they cannot be in a same cluster and it is the cannot-link constraint N.

The constrained K-means clustering algorithm is shown in Algorithm 2. Algorithm 3 describes the function Violate-Constraints (d_i, C_j, M, N) in Algorithm 2.

Algorithm 2: Constrained K-means algorithm

Input: Data set D, Must-link constraints M, Cannot-link constraints N.
Output: The lane clusters $\{C_1, C_2, ...C_k\}$.

```
Randomly select the initial cluster centers C_1, C_2, ...C_k.
while If don't converge do
    for each sample d_i, d_i ∈ D do
        Find its closest cluster C_j, j ∈ {1, 2, ...k}
    if Violate-Constraints ( d_i, C_j, M, N ) = false then
        Assign it to C_j
    else
        Do not assign
    for each cluster center C_i, i ∈ {1, 2, ...k} do
        Update its value by calculating the average
          euclidean distance of all of the points d_j that
          have been assigned to it.
Return {C_1, C_2, ...C_k} and average distance d_ave.
```

Algorithm 3: Violate-Constraints (d_i, C_j, M, N)

```
if There exits a d_x, {d_x, d_i} ∈ M, d_x ∉ C_j then
    Return true
if There exits a d_x, {d_x, d_i} ∈ N, d_x ∈ C_j then
    Return true
else
    Return false
```

2) Mining GPS Traces: In traditional K-means algorithm, the center of a cluster is a point. However, the fused GPS data is two dimensional and can not be used directly. To dealwith this problem, we use a line parallel to the centerline of one road to represent the lane cluster. The distance between one point and the lane cluster line can represent the euclidean distance in constrained K-means algorithm. We adopt the Road Centerline Generation approach in [8]. In the following discussions, the directions of roads are normalized, which makes it easier to illustrate our experiments.

The constrained K-means clustering algorithm expects the number of clusters k as input. However, for the number of lane calculation, we have to determine the most likely number of lanes automatically. This requires methods to evaluate the solutions that use different values of k and choose the most appropriate one. We adopt a measure that trades cluster cohesiveness under different number of clusters to select the most appropriate k. In this measure, it first calculates the average euclidian distance d_{ave} from each clustered instance to its assigned cluster center, then it penalizes for the complexity of the solution by multiplying the quadratic of k. The equation of cluster cohesiveness is:

$$G_k = d_{ave} * k^2.$$

Our problem can be transformed to find the minimum value of all the $G_k, k \in \{1, 2, ...10\}$, and the corresponding k is the number of lanes the road has.

B. Online Lane Localization

The localization is based on the online classification of GPS traces and the correction with lane change points. The initial prediction of a vehicle's location relies on the classification of GPS trace segments. After making a turn, former lane localization is no longer valid. The client will send new GPS data to the server. Since clustering data is stored with lane information, the server can locate the lane position. Data set collected by the vehicle is firstly converted to the offset to the road centerline. Then it will be classified according to its distance from lane centers. Once the online localization is done, the following lane localization can be done by lane change detection algorithm until the next turn.

The lane number L of x is calculated as:

$$L = \arg\min_n \sum_{i=1}^{N} (x_i - C_n)^2,$$

where n stands for the lane number and C_n represents lane offsets from the road centerline. The scale of data set N is determined by the number of uploaded samples. To guarantee the accuracy, the number should not be too small.

V. Evaluation

In this part, we evaluate the performance of different modules in this system. In consideration of the diversity, the evaluation is conducted using different cars with two kinds of smartphones, Google Nexus 4 and Nubia Z7 Max.

A. Lane Change Detection

TABLE I: Lane change detection test

Action	Left lane change	Right lane change	Turn
Number of tests	50	50	20
Results	50	49	21

In the lane change detection algorithm test, we compare the performance of single-sensor and dual-sensor systems. The

TABLE II: Experimental results of lane localization

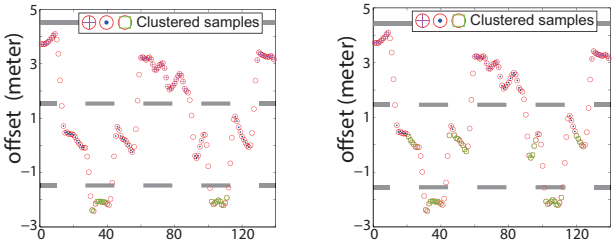
GPS Trace ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Mean Accuracy(%)	98.8	96.3	95.6	97.1	99.2	95.6	95.4	93.8	100	94.7	94.6	97.2	92.8	95.4	100

algorithm is implemented on smartphones with an open source SVM library called lib-svm [13]. Since we already know that using acceleration sensor alone is unable to detect turns, the single sensor system only uses the orientation sensor. In this test, nearly 20 lane changes and several turns are tested. We set $N = 300$, D_a=0.35, and $D_o = 225$ after the test in Section III-C. There are 18 errors using orientation sensor only, while there are no errors using both sensors. It is obvious that the dual-sensor system performs much better.

Then we conduct 100 lane changes to further test the accuracy of lane change detection algorithm, and the test result is shown in Table I. There was 1 error in total, which was identified as a turn. Some turns are identified incorrectly in Nubia Z7 Max while all turns are detected in Nexus 4. In spite of this, the accuracy of lane detection is still around 99%.

B. Lane Localization

GPS data is divided into several segments according to lane change points obtained from lane change detection algorithm. Denote N is a constant number determined in advance, then N data points are deleted from the start and the end of each segment. This procedure can improve the accuracy of the final results because these GPS points are the locations where a car is changing lanes. In the evaluation, we set $N = 4$.



(a) Constrained (b) Traditional

Fig. 3: Clustering with two kinds of K-means

Next, we implement the constrained K-means clustering algorithm. In this step, we set the number of clusters $k = 1, 2, ...10$ and get the corresponding cluster cohesiveness $G_k, k \in \{1, 2, ...10\}$. We then compare the constrained K-means with traditional K-means under the same selected k criteria to prove its effectiveness, as shown in Fig. 3. We conducted the experiment for 100 times. And the accuracy of constrained K-means algorithm is 97% on average, while the average accuracy of traditional K-means is less than 50%. There are three lanes separated by dotted gray lines in Fig. 3, and red circles are the deleted points during the process of lane changes. Red circles with purple plus signs, blue dots and green squares indicate the three different clusters.

At last, we conduct the localization experiment. Fifteen GPS traces of a road are collected in different time periods. We test each GPS trace for 10 times with more than 100 sample points. The test results are listed in Table II. From the experimental results, the mean accuracy of our approach is 96.4% with standard deviation 4.98×10^{-4}, which prove the effectiveness of our system to localize vehicles in lane level.

VI. Conclusion

This paper introduces a lane localization system based on machine learning approaches. We leverage the smartphones to collect both sensor and location data, then the server can localize lane position and detect the number of lanes in each road. An SVM-based method is proposed to identify lane changes and turns. We also use a constrained K-means algorithm to derive the number of lanes and realize online localization with the collected information. The experiment results shows the accuracy of lane change detection algorithm is higher than 98%, and the accuracy of lane localization is higher than 95%.

VII. Acknowledgement

This work is supported by National Natural Science Foundation of China (No. 61572319, U1405251, 61532012, 61325012, 61271219, 61428205); National Mobile Communications Research Laboratory, Southeast University (No. 2014D07).

References

[1] S. Kinmlioglu, "Fusion of sensor signals for navigation on an unmanned land vehicle prototype," in *IEEE Signal Processing and Communications Applications Conference (SIU)*, pp. 369-372, 2014.

[2] F. Kou, W. Chen, J. Wang, and Z. Zhao, "A lane boundary detection method based on high dynamic range image," in *IEEE International Conference on Industrial Informatics (INDIN)*, pp. 21-25, 2012.

[3] K. Chiu and S. Lin, "Lane detection using color-based segmentation," in *Proc. of IEEE Intelligent Vehicles Symposium*, pp. 706-711, 2005.

[4] Y. Dong, J. Xiong, L. Li, and J. Yang, "Robust lane detection and tracking for lane departure warning," in *IEEE International Conference on Computational Problem-Solving (ICCP)*, pp. 461-464, 2012.

[5] S. Ieng, J. Vrignon, D. Gruyer and D. Aubert "A new multi-lanes detection using multi-camera for robust vehicle location," in *Proc. of IEEE Intelligent Vehicles Symposium*, pp. 700-705, 2005.

[6] J. Nilsson, J. Fredriksson, and A. Odblom, "Reliable vehicle pose estimation using vision and a single-track model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2630-2643, Dec. 2014.

[7] D. Gruyer, R. Belaroussi, and M. Revilloud, "Map-aided localization with lateral perception," in *Proc. of IEEE Intelligent Vehicles Symposium*, pp. 674-680, 2014.

[8] S. Schroedl, K. Wagstaff, S. Rogers, P. Langley, and C. Wilson, "Mining gps traces for map refinement," *Data mining and knowledge Discovery*, vol. 9, no. 1, pp. 59-87, 2004.

[9] T. Dao, K. Y. K. Leung, C. M. Clark and J. P. Huissoon, "Co-operative lane-level positioning using Markov localization," in *IEEE Conference on Intelligent Transportation Systems*, pp. 1006 - 1011, 2006.

[10] A. J. Dean and S. N. Brennan, "Terrain-based road vehicle localization on multi-lane highways," in *American Control Conference*, pp. 707-712, 2009.

[11] U. Dogan, H. Edelbrunner, I. Iossifidis, "Towards a driver model: Preliminary study of lane change behavior," in *Proc. IEEE ITSC*, pp. 931-937, 2008.

[12] H. M. Mandalia, M. D. D. Salvucci, "Using support vector machines for lane-change detection," in *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, vol. 49, no. 22, pp. 1965-1969, 2005.

[13] National Science Council of Taiwan. (2015, May 15). "LIBSVM – A Library for Support Vector Machines," [Online]. Avaliable: http://www.csie.ntu.edu.tw/ cjlin/libsvm/