2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)
Windsor Oceanico Hotel, Rio de Janeiro, Brazil, November 1-4, 2016

Surround Vehicles Trajectory Analysis with Recurrent Neural Networks

Aida Khosroshahi, Eshed Ohn-Bar, and Mohan Manubhai Trivedi

***Abstract*— Behavior analysis of vehicles surrounding the ego-vehicle is an essential component in safe and pleasant autonomous driving. This study develops a framework for activity classification of observed on-road vehicles using 3D trajectory cues and a Long Short Term Memory (LSTM) model. As a case study, we aim to classify maneuvers of surrounding vehicles at four way intersections. LIDAR, GPS, and IMU measurements are used to extract ego-motion compensated surround trajectories from data clips in the KITTI benchmark. The impact of different prediction label space choices, feature space input, noisy/missing trajectory data, and LSTM model architectures are analyzed, presenting the strengths and limitations of the proposed approach.**

I. Introduction

Driving a vehicle requires interaction with other road occupants. An intelligent driving system must understand and predict the actions of its surrounding agents in order to maneuver in a safe manner [1]. This work aims to develop a framework for automatic activity classification of surrounding on-road agents using a Recurrent Neural Network (RNN) and 3D trajectory cues. Fig. 1 depicts an example of the type of data and activities studied in this work. As a case study, we focus on activity classification at intersections.

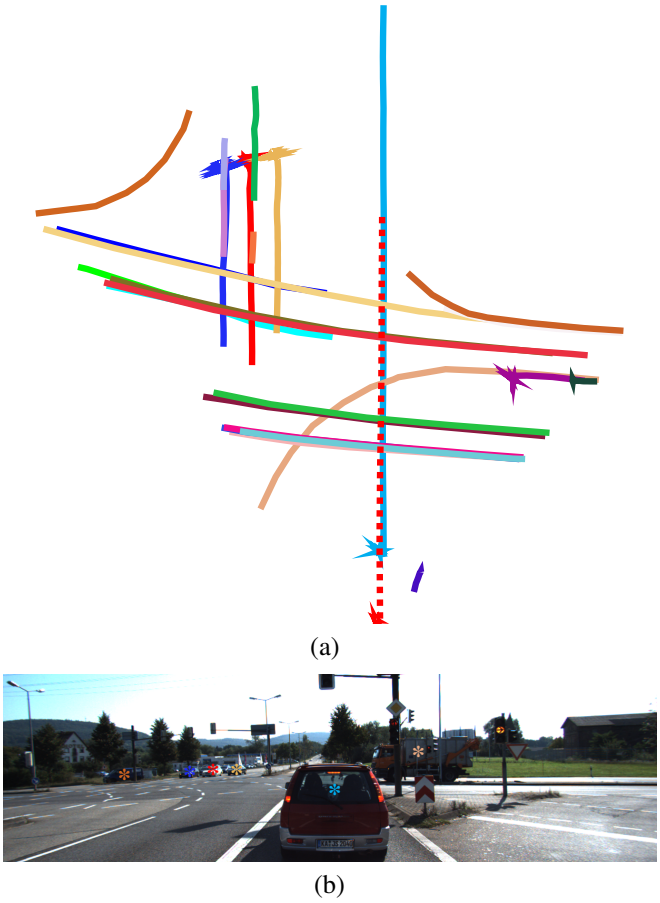
Studying the behavior of surrounding vehicles at intersections is an important issue for autonomous driving and driver assistance. For example when turning left in an intersection that does not have a separate left turn signal, the driver needs to know if the vehicles in the oncoming lane want to go straight through the intersection (driver waits) or they are also making a left turn (driver can make the left turn). Due to the crossing of multiple roads, crashes often occur at intersections [2]. In 2014, there were 4,441 crashes at four way intersections with at least 327 fatalities in U.S. [3]. An activity classification system can recognize dangerous situations and take precautionary measures to avoid an accident.

A. Contributions

Activity classification framework: Our goal is to build a robust system that can classify surround vehicle maneuvers. We propose to use a multi-layer (stacked) LSTM architecture, as behavior analysis and modeling involves reasoning over the evolution of temporal events in sequences. We find that increasing the abstraction capability of the model (up to 3 layers) works best for our task.

Evaluation: The capability of the proposed approach to classify activities and capture different levels of temporal context is analyzed using a real-world case study at four way intersections. The annotated intersection activity dataset is employed to gain insights into the optimal LSTM architecture (specifically, the number of layers and cells in each layer), as well as the impact of changing the label space (e.g. turning or not turning vs. specific types of turns) or the feature space (e.g. velocity, orientation, etc.). We also analyze the impact of noisy or missing trajectory data on classification accuracy of the proposed framework. Such an experiment studies robustness and generalization capabilities, necessary for on-road systems.

Authors are with the Laboratory for Intelligent and Safe Automobiles (LISA), University of California, San Diego, {aikhosro, eohnbar, mtrivedi}@ucsd.edu



(a)

(b)

Fig. 1. This study develops a framework for classification of surround vehicles' trajectories. (a) Vehicle trajectories from a drive in the KITTI [15] dataset, mapped to the road surface. The ego vehicle path is shown with a dotted line. (b) A corresponding image of the studied scene, where colored tags refer to the paths shown in (a)

II. Related Research Studies

As stated in [5], a vehicle's velocity can indicate the intention of the driver at an intersection. For example a

driver who does not slow down as she is approaching the intersection, will probably go straight compared to a driver who slows down and waits at the side of the intersection. Efficient modeling of such subtle cues from trajectory data in on-road settings is the main aim of our study. In [4], pose and motion parameters, such as the vehicle's yaw rate, are used with Kalman filtering for a trajectory prediction task. This work focuses only on oncoming traffic and detects if the oncoming vehicle is going straight or turning. On the other hand, we experiment with a variety of label space choices for the activity vocabulary.

Some related research studies employ unsupervised extraction of activities. In [16], a vocabulary of recurrent motion patterns is generated in a data-driven manner. Consequently, activities are analyzed using a Hidden Markov Models (HMM). Although both our study and the study in [16] employ trajectories for activity classification, [16] focuses on surveillance-type settings. In [17], traffic activities as well as scene topology and geometry are inferred using a probabilistic generative model. While we employ LIDAR data, [17] employs visual cues from a stereo camera.

Previous approaches for behavior classification and prediction include explicit models of the velocity profile of vehicles [5], Bayesian networks [6], Monte Carlo Simulation [7], Hidden Markov Models (HMM) [8], [9], [10], Support Vector Machines [11] prototype based methods [12], and Conditional Random Fields [13].

Here we propose a method based on a Long Short Term Memory [18] network. RNNs, and in particular LSTMs have recently emerged as powerful temporal data models [14], and this work discusses employing them for a trajectory classification task. In particular, we propose a stacked architecture and a set of temporal features, and evaluate their performance on a naturalistic trajectory dataset of KITTI videos [15].

III. Activity Classification Framework

A. Dataset

The KITTI benchmark [15] was used for training and testing. KITTI dataset has camera images, Velodyne LIDAR data, and IMU/GPS recordings, as well as annotations of 3D bounding boxes. From the IMU/GPS data and 3D bounding boxes, the trajectories for visible vehicles were ego-motion compensated and mapped onto a 2D plane (Fig. 1). Consequently, the trajectories that involved going through intersections were manually selected and labeled for training and evaluation. There are 51 samples from 8 videos covering 2247 frames.

B. Label Space

The annotated labels are explained in Fig. 2. For classification, several categorizations were defined. The most specific categorization had 12 labels (as shown in Fig. 2). Fig. 3) shows the number of instances in each class. The labels can be used for more general activity categorizations, produced by merging together labels in Fig. 2. Throughout the experiments, we refer to the resulting number of enumerated classes as M. For example in $M = 3$, the starting point

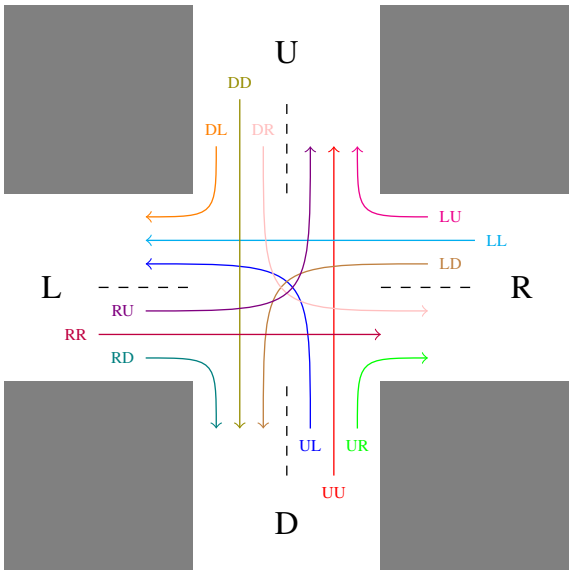


Fig. 2. We annotate trajectories using the depicted labels and the paths they represent. U-turns were excluded because there were very few instances of them in the dataset. The labels shown are used to study different activity vocabulary choices when training classification models.

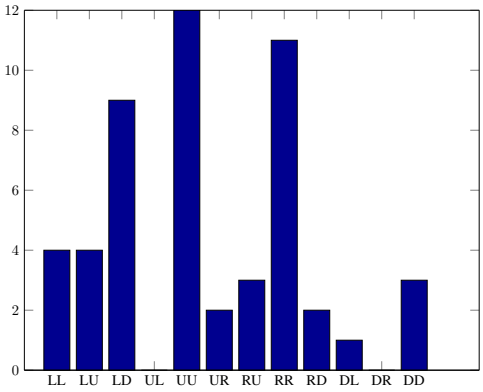


Fig. 3. The number of instances in each class.

in the intersection can be inferred from the location of the cars with respect to the road instead of having the classifier predict that. This leads to simpler classifiers that have the benefit of having more training data. We study the following activity label spaces,

- M = 12: all the classes are present.
- M = 8: for each side of intersection, only two classes for *going straight* and *turning* are considered (e.g. UL and UR are merged).
- M = 3: three global classes for *going straight*, *turning left* and *turning right* (left and right are respective to the vehicle's orientation).
- M = 2: two classes for *going straight* vs *turning*.

C. Features

For each time instance, we found it useful to transform the road-plane position information of the trajectory into the following types of features. The features are computed after resampling each trajectory to a fixed length of $L = 20$.

Algorithm 1 Extracting Histogram Features

Require: loc ▷ vehicle location in 2D per frame
Require: edges ▷ Edges for the histogram buckets
Require: L ▷ length of the output
Ensure: hist ▷ Histogram features per frame

```
loc ← resample loc to have length L + 2
for i = 1 to L do
    cur ← loc[i+1]-loc[i]
    next ← loc[i+2]-loc[i+1]
    ang[i] ← clockwise angle between cur and next
    euc[i] ← 2-norm of cur
    k ← the bucket for ang[i]
    h[k] ← h[k] + euc[i]
    hist[i] ← h
```

1) Linear Changes: The original trajectories were resampled to have length $L+1$. For each location after the first one (i), the difference between current location and the previous one was calculated,

$$linear_i = location_{i-1} - location_i$$

2) Angular Changes: For each location after the second one (i), calculate the vector A that connects $location_{i-2}$ to $location_{i-1}$ and vector B that connects $location_{i-1}$ to $location_i$ and find the angle between A and B. The angles are between $-\pi$ and π and sensitive to the order of A and B: $get\_angle(A, B) = -get\_angle(B, A)$.

$$A = location_{i-2} - location_{i-1}$$
$$B = location_{i-1} - location_i$$
$$\theta_i = get\_angle(A, B)$$

3) Angular Changes Histogram: The same angular changes from the previous part were used to make histograms per step. Algorithm 1 shows how the histogram features were extracted. For each angle, instead of adding 1, the corresponding euclidean distance was added because stationary vehicles have noisy readings. Two sets of buckets were tested:

1) edges = $[-\infty, -\frac{\pi}{40}, \frac{\pi}{40}, \infty]$
2) edges = $[-\infty, -\frac{\pi}{20}, -\frac{\pi}{40}, \frac{\pi}{40}, \frac{\pi}{20}, \infty]$

4) Vehicle Orientation: As KITTI provides bird's eye view orientation, the value was also studied as a potential useful feature in intersection scenarios.

D. Long Short-Term Memory for Intersection Activity Classification

Recurrent Neural Networks can use contextual temporal information for mapping input sequences to output sequences, but a problem arises when the influence of the input decays or blows up exponentially during the network's recurrent connections [19]. This problem makes it difficult to discover patterns in long input sequences. LSTM is a form of RNNs that addresses this problem (also called the problem of *vanishing gradient*) by replacing each node of the network by a memory cell (Fig. 4). Each memory block has

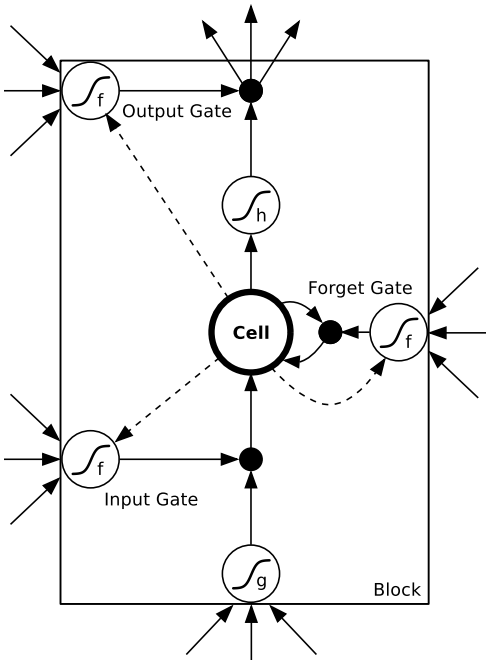


Fig. 4. An LSTM memory cell [19], suitable for modeling activities of surrounding vehicles.

- memory cell: that remembers and accumulates what the cell is intended to remember
- forget gate: that decides what proportion of the cell memory should be kept for the next time step
- input gate: that decides whether the input should be allowed into the block
- output gate: that decides whether the output of the block should be sent out.

This architecture allows LSTM blocks to store and retrieve information for arbitrary length of time. Unlike general RNNs, in LSTM the back-propagated error doesn't vanish exponentially over time and they are easily trainable [20]. Hence, studying the usefulness of such an approach for the purpose of surround trajectory classification is well motivated.

The Keras implementation of Long Short Term Memory for Python using Theano as backend was used for classification [21]. To better learn the temporal representations, three layers of LSTM were stacked on top of each other. The last layer outputs a vector with the same size as the number of classes (Fig. 5).

IV. Experimental Analysis

In this section, the impact of varying the prediction label space, model parameters, and input features is analyzed on the annotated trajectory dataset. Due to the large variation in the number of samples in each maneuver type (from 0-12), a normalized accuracy metric was used. First the percent

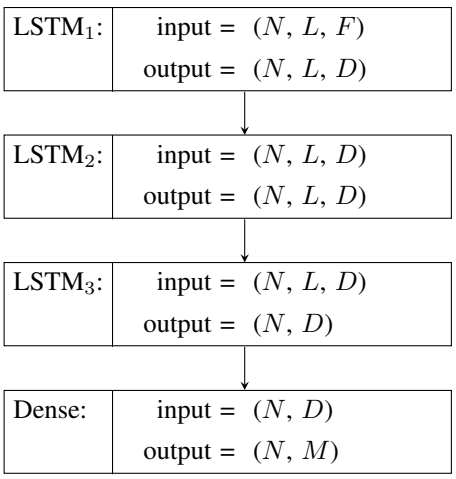


Fig. 5. Architecture of the layers in the classifier with $S = 3$. N = number of training samples, L = length of features, F = dimension of features (e.g. $F = 2$ for linear changes in , $F = 3$ for linear and angular changes), M = number of categories, and D = number of cells in each layer.

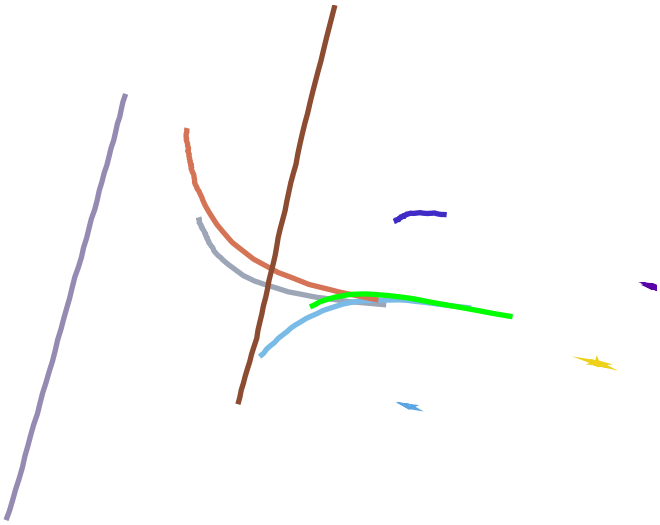


Fig. 6. The green path shows a right turn that was misclassified as going straight when using histogram features.

accuracy within each class was calculated and then averaged over all the classes. If one of the classes did not have any samples, it was not considered for accuracy calculation (classes UL and DR).

$$\text{accuracy}_i = \frac{\text{correct predictions for class } i}{\text{instances from class } i}$$

$$\text{accuracy} = \frac{1}{M}\sum_{i=1}^{M} \text{accuracy}_i$$

To make results consistent, the random number generator's seed was set to a fixed value before each training.

A. Results

The evaluation results for various categorization methods, architectures, and features are shown in the tables II, III, IV, and V. The classifiers with a three layer architecture ($S = 3$) had higher accuracy compared to $S = 2$. However adding another layer in $S = 4$ does not seem to improve the accuracy significantly. Higher values of D (cells per layer) also give better accuracy but they add computational cost. $D = 128$ did not give results that were significantly better than $D = 64$, and even $D = 32$ is shown to produce comparable results. As the highly accurate 3D bounding boxes and Velodyne measurements in KITTI may not always be available (i.e. other sensors) and for studying sensitivity to noise, a Gaussian noise with standard deviation of $0.1m$ was added to the vehicle locations before any processing for each experiment.

In general, smaller values for M resulted in higher accuracy, which is expected as it results in fewer classes, more data per class, and consequently an easier inference problem. Some distinctions can not be easily made from the selected features. For example a UU looks exactly like a DD unless some information about the ego vehicle, such as its relative location and orientation are given. However these distinctions may not be necessary, as only the intentions of the vehicle to turn left/right or to go straight are needed for a system that knows the initial location and orientation of the vehicles.

The linear changes have the best results for $M = 2, 8$ and 12. Adding angular changes or orientations does not seem to increase the accuracies. For $M = 3$, the histograms seem to be more accurate, though they have difficulty detecting right turns because they are often short compared to left turns and their trajectories are not fully captured. Fig 6 shows an example of a misclassification. In this case the observed path started late into the turn, so the extracted features cannot capture the changes in orientation well enough for the classifier.

Table VI shows the results for a classifier based on linear and angular changes with $S = 3$ and $D = 64$. As it was mentioned earlier, since the features do not hold any information about the ego vehicle location, it is difficult to classify for the complete 12 classes. Also instances for left turns get detected almost perfectly, but the right turns are more difficult to classify using these features. However, the $M = 2$ or $M = 3$ classifiers achieve better performance, of 0.75 and 0.6, respectively.

The best performing settings in terms of accuracy and performance were:

- $M = 2$: $L = 3, D = 32$, feature type = histogram 3.
- $M = 3$: $L = 3, D = 32$, feature type = histogram 5.
- $M = 8$: $L = 3, D = 32$, feature type = linear changes.
- $M = 12$: $L = 3, D = 64$, feature type = histogram 3.

V. Concluding Remarks

In this work we proposed LSTM for detecting the surrounding vehicles' trajectory types. We demonstrated that the selection of features, architecture, and object localization quality all play key roles in classifying activities. Trajectory classification showed more promise when considering a coarse label (e.g. turning vs forward), but finer activity label

	without noise				with noise			
feature type	$M = 2$	$M = 3$	$M = 8$	$M = 12$	$M = 2$	$M = 3$	$M = 8$	$M = 12$
linear	0.64	0.61	**0.46**	**0.30**	0.64	**0.61**	**0.46**	**0.30**
linear & angular	0.71	0.56	0.38	0.20	0.71	0.56	0.38	0.20
linear & orientation	0.71	0.36	0.38	0.20	0.71	0.36	0.38	0.20
histogram 3	**0.77**	**0.67**	0.17	**0.30**	**0.77**	**0.61**	0.17	0.13
histogram 5	0.48	**0.67**	0.27	0.20	0.55	**0.61**	0.04	0.23

TABLE I: Classification Accuracy for $S = 2$, $D = 32$.

	without noise				with noise			
feature type	$M = 2$	$M = 3$	$M = 8$	$M = 12$	$M = 2$	$M = 3$	$M = 8$	$M = 12$
linear	0.70	0.50	**0.65**	**0.30**	0.70	0.50	**0.65**	**0.30**
linear & angular	0.70	0.58	0.33	**0.30**	0.70	0.58	0.33	**0.30**
linear & orientation	0.62	0.53	0.33	**0.30**	0.69	0.53	0.33	**0.30**
histogram 3	**0.85**	0.56	0.21	0.17	**0.71**	**0.67**	0.21	0.17
histogram 5	0.77	**0.75**	0.33	0.25	0.69	0.36	0.21	0.22

TABLE II: Classification Accuracy for $S = 3$, $D = 32$.

	without noise				with noise			
feature type	$M = 2$	$M = 3$	$M = 8$	$M = 12$	$M = 2$	$M = 3$	$M = 8$	$M = 12$
linear	**0.85**	0.58	**0.56**	0.30	**0.85**	0.58	**0.56**	**0.30**
linear & angular	**0.85**	0.67	0.33	**0.40**	**0.85**	0.67	0.33	**0.30**
linear & orientation	0.76	0.61	0.38	0.30	0.76	0.61	0.38	**0.30**
histogram 3	0.77	**0.69**	0.40	**0.40**	0.63	0.61	0.21	0.25
histogram 5	0.77	0.64	0.29	0.30	0.71	**0.78**	0.25	0.27

TABLE III: Classification Accuracy for $S = 3$, $D = 64$

	without noise				with noise			
feature type	$M = 2$	$M = 3$	$M = 8$	$M = 12$	$M = 2$	$M = 3$	$M = 8$	$M = 12$
linear	0.70	0.58	**0.38**	**0.30**	0.70	**0.58**	**0.38**	**0.30**
linear & angular	0.79	0.56	0.33	**0.30**	**0.79**	0.56	0.33	**0.30**
linear & orientation	0.69	0.50	0.33	**0.30**	0.69	0.50	0.33	**0.30**
histogram 3	**0.85**	0.61	0.17	0.15	0.63	0.56	0.33	0.15
histogram 5	0.77	**0.67**	0.29	0.20	0.70	0.39	0.17	0.27

TABLE IV: Classification Accuracy for $S = 4$, $D = 32$.

	without noise				with noise			
feature type	$M = 2$	$M = 3$	$M = 8$	$M = 12$	$M = 2$	$M = 3$	$M = 8$	$M = 12$
linear	0.77	**0.69**	0.40	0.30	0.77	**0.69**	0.40	0.30
linear & angular	**0.85**	0.58	**0.46**	0.30	**0.85**	0.58	**0.50**	0.30
linear & orientation	0.69	0.53	0.44	0.30	0.69	0.53	**0.50**	0.30
histogram 3	0.77	0.67	0.27	**0.38**	0.71	0.44	0.21	**0.40**
histogram 5	0.77	0.67	0.29	0.13	0.63	0.58	0.13	0.07

TABLE V: Classification Accuracy for S = 4, D = 64.

		Target											
		LL	LU	LD	UL	UU	UR	RU	RR	RD	DL	DR	DD
Output	LL	0	0	0	0	0	0	0	0	0	0	0	0
	LU	0	0	0	0	0	0	0	0	0	0	0	0
	LD	0	0	2	0	0	0	0	1	1	1	0	0
	UL	0	0	0	0	0	0	0	0	0	0	0	0
	UU	1	0	0	0	1	1	0	1	0	0	0	0
	UR	0	0	0	0	0	0	0	0	0	0	0	0
	RU	0	0	0	0	0	0	1	0	0	0	0	0
	RR	0	1	0	0	0	0	0	0	0	0	0	0
	RD	0	0	0	0	0	0	0	0	0	0	0	0
	DL	0	0	0	0	0	0	0	0	0	0	0	0
	DR	0	0	0	0	0	0	0	0	0	0	0	0
	DD	0	0	0	0	0	0	0	0	0	0	0	1

TABLE VI: An example confusion matrix for features of linear and angular changes and $M = 12$. The green cells show the instances that were confused with similar classes starting on different sides (e.g. UU instead of LL). The orange cells show the instances that were confused for another type of turn (left turn instead of right turn).

		Target		
		left	straight	right
Output	left	3	0	0
	straight	0	6	3
	right	0	0	1

TABLE VII: Confusion matrix for feature type histogram and $M = 3$.

space was shown to be challenging ($M = 12$). In the future, additional trajectory samples and classes can be used to further study the proposed framework. Collecting more comprehensive data and resampling the current data may improve the performance of the framework further. Comparing the effectiveness of other classifiers in the ability to model temporal trajectory evolution may provide further insights. Training predictive models and improving robustness to 3D localization noise are important next steps. Furthermore, generalization of the proposed approach to other types of on-road maneuvers, such as lane changes [22], will be studied.

VI. Acknowledgments

The authors would like to thank our associated industry partners and our LISA UCSD colleagues for assistance.

References

[1] E. Ohn-Bar and M. M. Trivedi, "Looking at Humans in the Age of Self-Driving and Highly Automated Vehicles," IEEE Trans. Intelligent Vehicles, 2016.

[2] NHTSA, "Crash Factors in Intersection-Related Crashes: An On-Scene Perspective Crash Factors in Intersection-Related," 2010.

[3] NHTSA, Traffic safety facts, 2014.

[4] A. Barth and U. Franke, "Tracking oncoming and turning vehicles at intersections," IEEE Conf. Intelligent Transportation Systems, 2010.

[5] M. Liebner, F. Klanner, M. Baumann, C. Ruhhammer and C. Stiller, "Velocity-Based Driver Intent Inference at Urban Intersections in the Presence of Preceding Vehicles," IEEE Intelligent Transportation Systems Magazine, vol. 5, no. 2, pp. 10-21, 2013.

[6] S. Lefèvre, C. Laugier and J. Ibañez-Guzmàn, "Exploiting map information for driver intention estimation at road intersections," IEEE Intelligent Vehicles Symposium, 2011.

[7] S. Danielsson, L. Petersson and A. Eidehall, "Monte Carlo based Threat Assessment: Analysis and Improvements," IEEE Intelligent Vehicles Symposium, 2007.

[8] H. Berndt and K. Dietmayer, "Driver intention inference with vehicle onboard sensors," IEEE Conf. Vehicular Electronics and Safety, 2009.

[9] C. Tay "Analysis of dynamic scenes: Application to driving assistance" L'Institut Polytechnique de Grenoble, FranceDèpt. Tèlècommun., 2009

[10] J. Firl, H. Stübing, S. Huss and C. Stiller "Predictive maneuver evaluation for enhancement of Car-to-X mobility data," IEEE Intelligent Vehicles Symposium, 2012.

[11] G. S. Aoude, V. R. Desaraju, L. H. Stephens and J. P. How, "Behavior classification algorithms at intersections and validation using naturalistic data," IEEE Intelligent Vehicles Symposium, 2011.

[12] C. Hermes, C. Wohler, K. Schenk and F. Kummert, "Long-term vehicle motion prediction," IEEE Intelligent Vehicles Symposium, 2009.

[13] E. Ohn-Bar, A. Tawari, S. Martin, and M. M. Trivedi, "On surveillance for safety critical events: In-vehicle video networks for predictive driver assistance systems," Computer Vision and Image Understanding, vol 134, pp. 130-140, 2015.

[14] A. Jain, A. Singh, H. S. Koppula, S. Soh, and A. Saxena, "Recurrent Neural Networks for Driver Activity Anticipation via Sensory-Fusion Architecture," IEEE Intl. Conf. Robotics and Automation, 2016.

[15] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," International Journal of Robotics Research, 2013.

[16] B. T. Morris and M. M. Trivedi, "Trajectory Learning for Activity Understanding: Unsupervised, Multilevel, and Long-Term Adaptive Approach," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 33, no. 11, pp. 2287-2301, 2011.

[17] A. Geiger, M. Lauer, C. Wojek, C. Stiller and R. Urtasun, "3D Traffic Scene Understanding From Movable Platforms," in IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 36, no. 5, pp. 1012-1025, 2014.

[18] S. Hochreiter, J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[19] A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks." [Online]. Available: `https://www.cs.toronto.edu/~graves/preprint.pdf`. [Accessed: 10-Jun-2016].

[20] A. Shahroudy, J. Liu, T. Ng and G. Wang, "NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis," IEEE Conf. Computer Vision and Pattern Recognition, 2016.

[21] F. Chollet, Keras, 2015, `https://github.com/fchollet/keras`

[22] M. S. Kristoffersen, J. V. Dueholm, R. K. Satzoda, M. M. Trivedi, A. Mgelmose, and T. B. Moeslund, "Towards Semantic Understanding of Surrounding Vehicular Maneuvers: A Panoramic Vision-Based Framework for Real-World Highway Studies," CVPRW, 2016.