Regression-Based Online Situation Recognition for Vehicular Traffic Scenarios

Daniel Meyer-Delius Jürgen Sturm Wolfram Burgard

***Abstract*— In this paper, we present an approach for learning generalized models for traffic situations. We formulate the problem using a dynamic Bayesian network (DBN) from which we learn the characteristic dynamics of a situation from labeled trajectories using kernel regression. For a new and unlabeled trajectory, we can then infer the corresponding situation by evaluating the data likelihood for the individual situation models. In experiments carried out on laser range data gathered on a car in real traffic and in simulation, we show that we can robustly recognize different traffic situations even from trajectories corresponding to partial situation instances.**

I. Introduction

To act intelligently, an agent must be capable of recognizing its temporal context or situation. Many real domains, like car driving, present complex and time-varying dynamics. Knowledge about the characteristic dynamics of a situation can be used by an agent to improve its performance by, for example, making informed decisions to avoid risks. A driving assistant, for instance, could use this knowledge to take actions to improve safety, for example turning on an acoustic alarm or executing an emergency brake. Such systems were introduced by several automobile manufacturers in the past few years. These systems, however, were designed to recognize acute situations of immediate danger, like wheel slippage or head-to-tail collisions, using specialized hardware. So only a very limited number of hand-crafted situation models have been implemented in today's cars. It is clear that a truly intelligent agent should be able to deal with more complex situations.

In this work we present a framework for the modeling and online-recognition of situations. As application and test-bed for our approach, we consider a driver assistant application in traffic scenarios and consider situations that typically occur in highway-like driving settings, like for example, a *passing* situation. Concretely, we deal with the problem of learning generalized models for our situations and recognizing instances of those situations online.

In our approach, the model for a given situation is learned from multiple trajectories of the corresponding situation. The insight behind this approach is that, although instances of a given situation are in general different from each other, there is an inherent similitude that characterizes the situation. Figure 1 plots multiple trajectories of a *passing* situation together with the learned model. In our concrete example, a trajectory is described by the bearing, distance, and speed of the passing car relative to the car being passed. We can clearly see that the different trajectories are similar; they all present a distinctive form that characterizes the *passing* situation. The learned model for the *passing* situation is visualized in the figure by the thick lines and the filled curves, corresponding respectively to the mean and variance of the features in time.

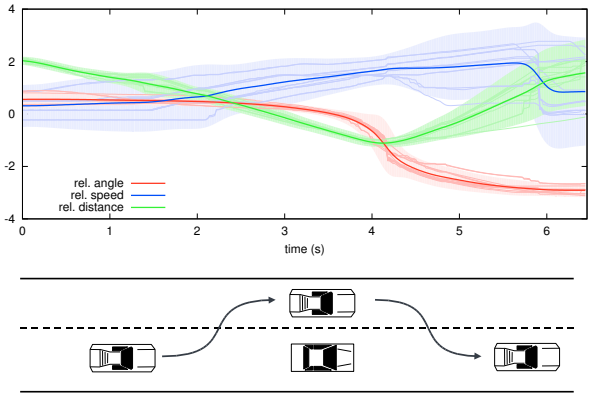

Fig. 1. Regression model learned from trajectories corresponding to a *passing* situation. The training trajectories are visualized as thin lines and each color represents a different dimension (bearing, distance, speed of the passing car relative to the car being passed) in state space. From these trajectories, a generalized model for the *passing* situation was learned using kernel regression, visualized by the thick line and the filled curve, corresponding to the mean and variance respectively.

Our approach is formulated as a dynamic Bayesian network (DBN) that represents, in a factorized way, the relevant aspects of the state of the system using random variables and conditional probability distributions between these variables. A high-level state variable that represents the current situation determines the dynamics of the lower-level state variables. The dynamics are described by a function that approximates the state of the system in time. This function is learned from a set of labelled training trajectories using kernel regression. Thus, each situation is modeled by an individual regression function that describes the characteristic dynamics of the situation. Trajectories can then be classified by evaluating their likelihood for the different models. This can be done even for trajectories that correspond only to a partial instance of a situation, allowing us to recognize situations online.

The contribution of this paper is a practical approach for modeling and recognizing situations. We show how our framework can be used for learning models of typical situations in a vehicular traffic scenario. We also describe

All authors are with University of Freiburg, Dept. of Computer Science, D-79110 Freiburg, Germany.
{meyerdel,sturm,burgard}@informatik.uni-freiburg.de

how situation instances can be recognized while they are developing. Experimental results using real and simulated data show that our system can robustly recognize different traffic situations even for partially observed instances.

The remainder of this paper is organized as follows. In the next section we review related approaches. In Section III we introduce our framework for learning generalized models for situations from a set of training trajectories. We then describe how the situation models are learned in Section IV and in Section V we explain how to infer the current situation using the data likelihood as measure. Finally, we present our experimental results in Section VI and conclude in Section VII.

II. Related Work

The approach presented in this paper is similar, in essence, to the problem of programming a robot by human demonstration [1], where a human performs an action or task multiple times, and the robot must infer a generalized representation of the task. We build upon the work of Eppner *et al.* [2], where a framework for learning and reproducing tasks with a robotic manipulator is presented. The fundamental difference between our work and these approaches is that they focus on the learning and reproduction of the task, whereas our approach concentrates on the classification and online recognition of tasks (situations in our case) using the learned models.

Our dynamic Bayesian network formulation of the system where a high-level variable determines the dynamics of the lower-level states is similar to the one made by the Switching Linear Dynamic Systems (SLDS) models. By switching between multiple linear dynamic models, an approximate description of the continuous non-linear dynamics of the system is obtained. Pavlović and Rehg [3] apply SLDS models to the problem of classifying human motion. Oh *et al.* [4] present an extension of the SLDS framework that allows a parameterization of the duration model of standard SLDS models. They apply their approach for decoding the honeybee dance. In contrast to these SLDS-based techniques, our approach learns the non-linear dynamics of the system using individual regression models.

The hidden Markov model (HMM) is one of the most popular probabilistic models for representing sequences of states that have structure in time. Rabiner [5] gives an excellent introduction to HMMs and its application in speech recognition. HMMs have been used successfully in many other different applications. Brand *et al.* [6], for example, represent and classify sequences corresponding to T'ai Chi Ch'uan gestures using a variation of the hidden Markov model. Bennewitz *et al.* [7] present a complete framework based on HMMs for recognizing gestures for human-robot interaction. In a previous work [8] we used HMMs to model and recognize vehicular traffic situations. The main disadvantage of HMM-based approaches is that representing trajectories using a finite (and usually small) number of states imposes a sometimes unnatural discretization on the trajectories.

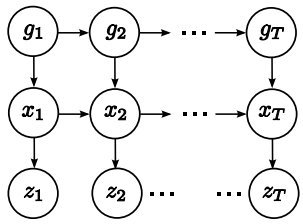

Fig. 2. Structure of the dynamic Bayesian network (DBN) used in our framework for learning the dynamics of the system in a given situation. For time t, x_t denotes the state of the system, z_t the observation of the state, and g_t is a high-level state variable that represents the current situation and determines the dynamics of the system.

III. Framework Overview

Each type of situation is associated with a particular system dynamics. The goal of our approach consists in learning the system dynamics corresponding to a situation from multiple training trajectories. Then, using the learned system dynamics, we can infer the most likely situation type for new trajectories.

In our framework, we model the system using a dynamic Bayesian network (DBN) (see Figure 2). At each point in time t, the random variable x_t represents the state of the system, z_t represents the observation of the state, and g_t is a high-level state variable that represents the current situation. The observation z_t depends on the current state of the system x_t, which in turn depends on the previous state x_{t-1} and the current situation g_t. Our DBN encodes two conditional probability distributions: the observation model $p(z_t|x_t)$ that represents the probability of observing z_t from state x_t, and the state transition model $p(x_t|x_{t-1},g_t)$ that corresponds to the state transition probability of the system at time t, and describes the dynamics of the system at that particular point in time.

For a given situation S of length T, we learn a model $p(x_{1:T}|g=S)$ that describes the dynamics of the state of the system during the complete development of the situation. Using a set of training trajectories $x^1_{1:T^1},\ldots,x^M_{1:T^M}$, where each trajectory $x_{1:T^n}=\{x_1,\ldots,x_{T^n}\}$ consists of a sequence of states corresponding to the same situation, we apply kernel smoothing to obtain the model that best approximates all trajectories. As the training trajectories are in general of different lengths, we apply Dynamic Time Warping to standardize them before learning the situation model. Finally, we can also learn the prior over situations $p(g)$ by counting how many training trajectories belong to each different situation type.

Given an observation sequence $z_{1:t}$, the trajectory $x_{1:t}$ can be estimated using the Bayesian recursive state estimation scheme [9]. We can then select the model that best fits the trajectory by computing the likelihood of each model S given the data

$$p(g=S|x_{1:t}) \quad \propto \quad p(x_{1:t}|g=S)p(g=S)\,, \qquad (1)$$

and selecting the model with the largest likelihood.

IV. Regression-Based Modeling

We treat the problem of learning a model for a situation as a non-linear regression problem. The goal is to estimate the function f that approximates the state of the system x as a function of time. Consequently, f will represent the characteristic dynamics of the state for that situation. Given a set of training trajectories we learn the function that best approximates all of them.

Assuming that at every time step t the state x_t of the system is normally distributed over all training instances, we formulate $f(t)$ as

$$f(t) \quad = \quad \mathcal{N}(\mu(t), \Sigma(t)), \tag{2}$$

where $\mu(t)$ is the mean vector with the same dimensionality k as x_t and $\Sigma(t)$ is the corresponding covariance matrix. Accordingly, the problem of estimating f can be stated as the problem of learning the mean $\mu(t)$ and covariance $\Sigma(t)$ of the normal distributions at each point in time t.

To estimate these parameters, we use kernel smoothing [10] which is a non-parametric technique for approximating the density function of a random variable from a set of sample instances. The idea is to approximate the value of the parameters as the weighted average of the neighboring sample points. These weights are given by a kernel function parameterized by a distance measure in the domain of the function. In our case, the weight of the samples depends on the temporal distance of the samples. Given a set of D trajectories corresponding to the same situation, each having a length of T, and assuming that the state dimensions are independent from each other, kernel smoothing estimates the mean $\mu_i(t')$ for dimension i at time t' as

$$\mu_i(t') \quad = \quad \frac{\sum_{d=1}^{D}\sum_{t=1}^{T} K(\frac{t'-t}{h}) x_{d,t}^{i}}{\sum_{d=1}^{D}\sum_{t=1}^{T} K(\frac{t'-t}{h})}, \tag{3}$$

where $x_{d,t}^{i}$ is the value for the i-th dimension of the state of the system in trajectory d at time index t, and $K(u)$ is a Gaussian kernel with bandwidth h. This bandwidth determines how the influence of the neighboring samples decreases with the distance in time. The variance $\sigma_i^2(t')$ for dimension i at time t' is estimated as

$$\sigma_i^2(t') \quad = \quad \frac{\sum_{d=1}^{D}\sum_{t=1}^{T} K(\frac{t'-t}{h})(x_{d,t}^{i} - \mu_i(t'))^2}{\sum_{d=1}^{D}\sum_{t=1}^{T} K(\frac{t'-t}{h})}. \tag{4}$$

The result of applying kernel smoothing to the training data is a function $f(t)$ that describes, for each point in time t, the characteristic state of the system by the mean $\mu(t)$ and covariance $\Sigma(t)$.

A. Aligning Trajectories

To be able to use the kernel density estimator method, the training trajectories must be of the same length. To handle temporal variations we apply Multi-Dimensional Dynamic Time Warping (MD-DTW) [11], a variation of Dynamic Time Warping (DTW) [12], to make all trajectories equally long. We select a reference trajectory from the training set, and all other trajectories are aligned to it. Kernel density estimation can then be directly applied on the aligned trajectories.

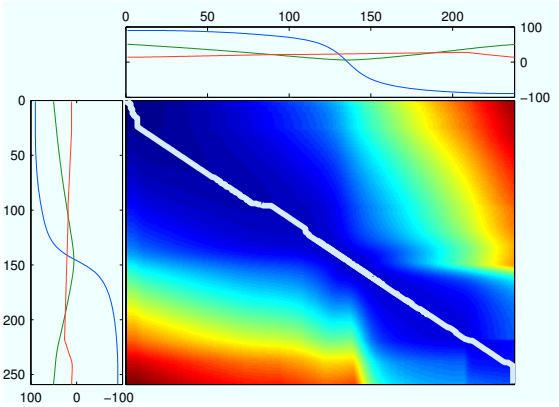

Fig. 3. Dynamic Time Warping between two different trajectories. The figure visualizes the cost matrix induced by the recursive computation of the cumulative warping cost. The bold white line in the figure corresponds to the minimum-cost warping path.

Dynamic Time Warping is a technique for aligning the time axis of two time-indexed sequences. The algorithm computes the minimum-cost alignment or warp between two series $x_{1:l_x}$ and $y_{1:l_y}$. Usually, the Euclidean distance is used as the distance measure $d(x_i, y_j)$ between the points in the sequences. The minimum-cost warp is then efficiently found using dynamic programming to compute the cumulative cost $\gamma(i,j)$ corresponding to the minimum-cost warp of the partial sequences $x_{1:i}$ and $y_{1:j}$. The value of $\gamma(i,j)$ is recursively computed as

$$\begin{aligned}\gamma(i,j) = d(x_i, y_j) + \\ min\{\gamma(i-1,j), \gamma(i,j-1), \gamma(i-1,j-1)\}.\end{aligned} \tag{5}$$

The cost C_W of the minimum-cost warp is then given by the value of $\gamma(l_x, l_y)$ and the warp is constructed by tracing back from $\gamma(l_x, l_y)$ to $\gamma(1,1)$. Figure 3 illustrates the cost matrix induced by the recursive computation of (5). Each entry i, j in the matrix corresponds to the value of the cumulative cost $\gamma(i,j)$ and the bold white line corresponds to the minimum-cost warp.

Multi-Dimensional Dynamic Time Warping is a generalization of DTW for multi-dimensional sequences where the distance between the points in the sequences corresponds to the n-dimensional Euclidean distance. To meaningfully compare different dimensions, each point x_i in the sequences is standardized as $x_i' = (x_i - \mu)\sigma^{-1}$, where μ and σ are the sample mean and standard deviation, before computing the distances. To make the alignments more robust, we compute, for each point in the sequence, an approximation of the derivative in each dimension as in [13]. This derivatives are added to the dimensions of the state space and included in the computation of the distance between the points, effectively incorporating information about the shape of the trajectories being aligned.

V. Recognition

Having trained a set of situation models, we want to select the one that best describes any given trajectory $x_{1:t}$. In other words, we want to select the model S^* such that

$$S^* = \operatorname*{argmax}_S p(g = S|x_{1:t}). \quad (6)$$

According to (1) this involves the computation of the likelihood $p(x_{1:t}|g = S)$ of the trajectory $x_{1:t}$ given the model S, for each model.

Given a regression model f corresponding to a situation S and a trajectory $x_{1:t}$, the likelihood of the trajectory given the model is computed as

$$p(x_{1:t}|g = S) = \prod_{t=1}^{t} p(x_t|f(t)), \quad (7)$$

where

$$p(x_t|f(t)) = \frac{1}{(2\pi)^{k/2}|\Sigma(t)|^{1/2}} e^{-\frac{1}{2}\left((x_t-\mu(t))^T\Sigma(t)(x_t-\mu(t))\right)}. \quad (8)$$

Equation (7) assumes that the states of the system are independent from each other. This is in general not the case, however, this approximation works well for our purposes and is easily computed.

Using the likelihood as a measure of the quality of a model, we can now select from a set of competing models, the one that produces the highest likelihood for a given trajectory. In this way, we can use our trained models for classifying trajectories. Note, however, that before computing the likelihood, the trajectory must have the same length as the model. This is achieved as explained in Section IV-A by aligning the trajectory against the reference trajectory for the corresponding model using MD-DTW.

A. Recognizing Partial Instances

The previous discussion about recognizing situations implicitly assumes that the start and the end of the trajectories that are being evaluated correspond, respectively, to the start and the end of the situation. For evaluating a trajectory that corresponds only to an incomplete instance of a situation, where the end of the instance doesn't correspond to the end of the situation, the MD-DTW approach as described in Section IV-A can not be directly applied.

The MD-DTW algorithm finds the best global alignment between two sequences. However, for aligning an incomplete trajectory we need to find the best local alignment beginning at the common starting point. Given an incomplete trajectory $x_{1:t}$ and a reference trajectory $y_{1:T}$, the minimum-cost local warp is constructed by tracing back from $\gamma(t, j^*)$ to $\gamma(1,1)$ where $j^* = \operatorname{argmin} \gamma(t, j)$ for $j = 1 \ldots T$.

The warping algorithm can also be modified to include a scaling cost that penalizes the stretching or contraction of a sequence [13]. In this way, we can alleviate the problem of over- or under-scaling a sequence. Once the incomplete trajectory is aligned, its likelihood can be computed directly using (7). In this way, we are able to recognize situations as they are developing.

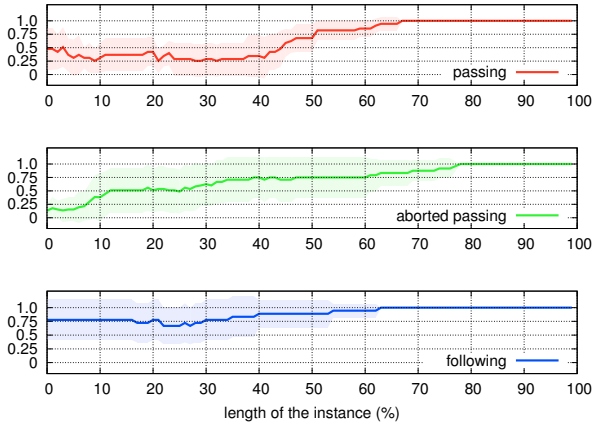

Fig. 4. Classification accuracy of the models for partial trajectories. The figure shows the average accuracy and standard deviation for each model as a function of the length of the partial trajectories (measured as the percentage of the length of the complete trajectory).

VI. Experimental Evaluation

Our framework was tested in a vehicular traffic scenario using real data as well as a simulated driving environment. As situations we considered three typical maneuvers: *passing*, *aborted passing* and *following*. The state of the system x_t was described by the bearing ψ, distance d, and speed v of the neighboring cars relative to the reference car. These features were sufficient to characterize the maneuvers, being also robust against variations in the different instances.

We first trained a model for each situation type using 30 trajectories generated in the simulation environment. Each trajectory started as soon as the neighboring car was closer than 50 meters and ended when the car was more than 50 meters away. As reference trajectory we selected one with average length. We then used MD-DTW to align the training trajectories and generated the corresponding model applying kernel smoothing. Figure 1 depicts some of the training trajectories and learned model for the *passing* situation. The likelihood for a set of 15 validation trajectories not used for training was computed to evaluate the classification performance of our models. For a better interpretation of the data, the average Mahalanobis distance over the length of the trajectory is used as fit error of a model. Table I presents the average fit error of the validation set for the learned models.

As can be seen in the table, the smallest error is obtained when the trajectories and the model correspond to the same situation type, which is the expected result. This experiment shows that our framework allows us to construct models that represent the characteristic dynamics of the maneuvers and could be used for recognition.

A. Recognizing Partial Instances

To evaluate how our approach performs at recognizing trajectories that correspond to incomplete instances of a situation, we repeated the previous experiment using only the initial segment of the trajectories. Figure 4 plots the the average accuracy and standard deviation for each model

TABLE I

AVERAGE AND STANDARD DEVIATION OF THE FIT ERROR FOR THE MODELS LEARNED ON ARTIFICIAL DATA.

	$\Theta_{\text{aborted passing}}$			$\Theta_{\text{following}}$			Θ_{passing}		
	ψ	d	v	ψ	d	v	ψ	d	v
aborted passing	**0.94** ± 0.42	**0.87** ± 0.37	**0.95** ± 0.34	6.49 ± 2.17	2.58 ± 1.31	4.22 ± 0.74	6.74 ± 0.51	2.53 ± 0.34	2.64 ± 0.43
following	2.09 ± 0.20	3.74 ± 3.17	3.04 ± 0.76	**0.80** ± 0.30	**2.43** ± 1.44	**2.48** ± 1.32	8.28 ± 0.22	9.08 ± 5.91	4.26 ± 0.91
passing	17.27 ± 0.83	5.66 ± 0.82	18.08 ± 6.45	38.55 ± 3.28	11.46 ± 0.82	20.12 ± 7.94	**0.79** ± 0.43	**1.02** ± 0.15	**1.24** ± 0.35

as a function of the length of the trajectory segments. The classification results were evaluated using k-fold cross-validation with $k = 6$.

As can be seen in the figure, 80% of the length of the trajectory was enough to correctly classify all the segments. Even 50% of the length was enough to obtain reasonable classification results for all the models. Models corresponding to complex situations require more evidence to correctly classify a trajectory. This can be observed in the figure by the poor performance for proportionally short trajectories of the *passing* and *aborted passing* models. The *following* model, on the other hand, describes a relatively simple situation. As can be seen in the figure, the *following* model had a good classification accuracy even for very short trajectories.

B. Real-time Recognition

To evaluate the complete situation recognition approach in a real-time setting, we integrated our framework into a driving simulator (TORCS [14]) which features a simple 3D physics model and provides us with the absolute position and velocity of the vehicles at 50Hz. At every time step, that is, every 0.02 seconds, the state of the neighboring vehicles is computed. This means that, for each vehicle, the relative bearing, distance, and speed is estimated, and added to the corresponding trajectory. Then, after standardizing the state's values and computing the derivatives, the trajectories are aligned and the likelihood computed for every situation model.

Figure 5 shows a series of screenshots of the simulation environment in a two vehicle scenario together with the results of our situation recognition framework. The images show a car (orange) passing another one (yellow). The bars in the plots correspond to the normalized likelihood of the trajectory of the passing car for the *passing* (red), *aborted passing* (green), and *following* model (blue). As can be seen, at first, in the leftmost image, the *following* situation is the most likely, but as the maneuver develops (from left to right in the images), the *passing* situations becomes more likely. The framework was also tested in scenarios with up to 9 vehicles (plus the reference one). The experiments showed that our approach can be used in such scenarios, where multiple vehicles are being simultaneously tracked, over extended periods of time.

The most time demanding step in the whole process is aligning the trajectories to the situation models. If the length of the reference trajectory of a model is N and the length of the trajectory being aligned is M, the time complexity for the alignment is $O(NM)$, that is, $O(N^2)$. However, since the cumulative cost $\gamma(i, j)$ can be incrementally computed as new states are added to the trajectories, the important factors in the performance of the approach are the number of situation models, the length of their reference trajectories, and number of neighboring vehicles.

TABLE II

AVERAGE AND STANDARD DEVIATION OF THE FIT ERROR FOR THE MODELS LEARNED ON REAL DATA.

	$\Theta_{\text{following}}$			Θ_{passing}		
	ψ	d	v	ψ	d	v
following	**0.84** ± 0.33	**0.87** ± 0.39	**0.84** ± 0.45	2.61 ± 0.49	7.03 ± 3.12	1.47 ± 0.51
passing	8.23 ± 2.53	1.4 ± 0.19	2.59 ± 1.71	**0.73** ± 0.57	**0.84** ± 0.31	**0.81** ± 0.6

C. Real Data

The framework was also evaluated using real data. Two SICK laser range scanners were mounted on a convertible as illustrated in Fig. 6. Data was gathered by driving over more than 50 kilometers on highways and state roads at velocities of up to 110 km/h. Note that in this work, we do not deal with the recognition and tracking of vehicles, and the trajectories were manually extracted from the data.

Due to the technical limitations of the sensors together with their arrangement, many situation instances could not be captured, or were captured only partially. From the gathered data, only 19 (10 *passing* and 9 *following*) useful trajectories could be extracted. Table II presents the fit error of the training set for the two trained models. As can be seen, the smallest error lies on the diagonal of the table where the sequence and the model correspond to the same maneuver type.

We also evaluated the classification accuracy of the models trained on real data against the trajectories generated in the simulation environment and vice versa. It must be noted that the trajectories from the real data where sampled at 75Hz while the ones obtained from the simulator were sampled at 50Hz. Also, because of the configuration of the lasers, the trajectories corresponding to passing maneuvers where truncated when the passing vehicle was left of the

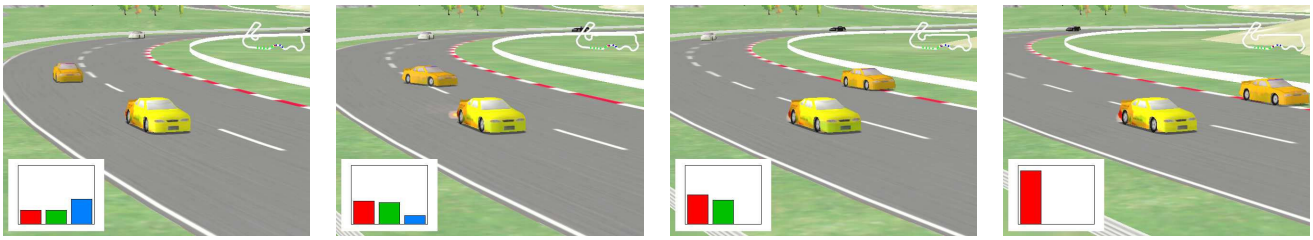

Fig. 5. Screen shots of the simulation environment showing a car (orange) passing another one (yellow). The bars in the plots correspond to the likelihood of the trajectory of the passing car for the *passing* (red), *aborted passing* (green) , and *following* (blue) model. At first, in the leftmost image, the *following* situation is the most likely, but as the maneuver develops (from left to right in the images), the *passing* situations becomes more likely.

Fig. 6. Left: arrangement of two SICK laser range finders on a convertible (middle) used for gathering real data. Each laser has a field of view of 180 degrees and can detect objects as far as 80 meters with an angular resolution of 1 degree at 75Hz. The arrangement of the two lasers provided a 340 degree field of view (right).

TABLE III

CLASSIFICATION ACCURACY OF MODELS TRAINED AND EVALUATED WITH REAL AND ARTIFICIAL DATA.

	real training	artificial training	
following	-	0.77	real testing
	1.0	-	artificial testing
passing	-	0.80	real testing
	0.57	-	artificial testing

reference one. Thus, the corresponding models only describe the maneuver until that point. When evaluating the trajectories generated in the simulation against the models trained with real data, we only considered the first half of the trajectories. This affected the classification accuracy, as shown in the previous experiment. Despite all these we obtained reasonable classification results as can be seen in Table III

VII. CONCLUSIONS

In this paper, we presented a general framework for modeling and recognizing situations. We take a model-based approach in which each situation type is described by an individual regression model that describes the characteristic dynamics of the system over time. We formalize the problem using a DBN and learn the characteristic dynamics of a situation from training instances. We then use the likelihood of the data as criterion for model selection and describe how to classify trajectories online. The approach was evaluated experimentally using real and simulated data in the context of a driver assistant application in traffic scenarios. The results show that our approach can robustly recognize different traffic situations even from partially observed instances.

REFERENCES

[1] S. Calinon and A. Billard, "Stochastic gesture production and recognition model for a humanoid robot," *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2769–2774 vol.3, Sept.-2 Oct. 2004.

[2] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard, "Imitation learning with generalized task descriptions," in *Proceedings of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

[3] V. Pavlovic and J. M. Rehg, "Impact of dynamic model learning on classification of human motion," in *CVPR*, 2000, pp. 1788–1795.

[4] S. M. Oh, J. M. Rehg, and F. Dellaert, "Parameterized duration mmodeling for switching linear dynamic systems," in *CVPR (2)*, 2006, pp. 1694–1700.

[5] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition." in *Proceedings of the IEEE*, vol. 77 (2), 1989, pp. 257–286.

[6] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *CVPR*, 1997, pp. 994–999.

[7] M. Bennewitz, T. Axenbeck, S. Behnke, and W. Burgard, "Robust recognition of complex gestures for natural human-robot interaction," in *Proc. of the Workshop on Interactive Robot Learning at Robotics: Science and Systems Conference (RSS)*, 2008.

[8] D. Meyer-Delius, C. Plagemann, and W. Burgard, "Probabilistic situation recognition for vehicular traffic scenarios," in *Proceedings of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.

[9] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ, 2003.

[10] E. Nadaraya, "On estimating regression," *Theory of Probability and Its Application*, vol. 9, pp. 141–142, 1964.

[11] G. A. ten Holt, M. J. T. Reinders, and E. A. Hendriks, "Multi-dimensional dynamic time warping for gesture recognition," in *In Proc. of the conference of the Advanced School for Computing and Imaging (ASCI 2007)*, 2007.

[12] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, Feb 1978.

[13] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *In Proceedings of First SIAM International Conference on Data Mining (SDM'2001)*, 2001.

[14] E. Espié and C. Guionneau, "Torcs - the open racing car simulator, http://torcs.sourceforge.net/."