2017 American Control Conference
Sheraton Seattle Hotel
May 24–26, 2017, Seattle, USA

Similarity-Based Vehicle-Motion Prediction

Kazuhide Okamoto[1] Karl Berntorp[2] Stefano Di Cairano[2]

Abstract— **Motion-prediction algorithms for vehicles often employ historical behavior of a vehicle, rely on the Markov property of the underlying system, and predict the future behavior of the vehicle. However, the Markov property alone may lead to conservative predictions and heavy computational burden. To overcome these drawbacks, this paper develops a method that uses the notion of similarity among vehicle trajectories. As traffic rules and driver intentions restrict the motions of a vehicle, the behavior of a road vehicle is typically similar to that of other vehicles. We hypothesize that if the motion of any two vehicles was similar in the past for a sufficiently long time span, then it is likely that it will be similar in the future. This paper introduces an algorithm that exploits this hypothesis to develop prediction methods, and from the results of numerical simulations, it verifies the effectiveness of the algorithm.**

I. Introduction

The safety of self-driving vehicles is to a great extent governed by the performance of the underlying motion-prediction and threat-assessment algorithms. Without motion predictions, path-planning systems, which are oftentimes the next building step in the design of an autonomous vehicle, have to assume a static, or perfectly known, environment, which is unrealistic because the environment is typically dynamic and highly uncertain. If precise prediction of the environment is available, path planners can compute safer, less conservative, and more robust trajectories. Furthermore, motion prediction plays a key role in the development of advanced driver assistance systems (ADAS) and semi-autonomous vehicles, and automobile control researchers have published several review papers that address the recent surge of automated driving [1]–[3].

Many preceding automobile motion-prediction methods assume the *Markov property* of the underlying system, which essentially hypothesizes that the future state of a system is uniquely determined by the current state. With this assumption, several studies, summarized in [3], developed methods with Markov chains and hidden Markov models for predicting the future motion of individual vehicles. However, as highlighted in [4], real-time prediction of the future behavior of other vehicles is difficult because the intentions of the other drivers are uncertain, the range and resolution of the sensors is limited, and the environment is uncertain and noisy. Therefore, motion-prediction methods typically acquire relevant information of each individual vehicle only over a short time span, which leads to imprecise predictions. Furthermore, because predicting each vehicle individually requires a large computational load, the methods have difficulty in simultaneous motion prediction of multiple vehicles in the region of interest.

To overcome the problem of inaccurate prediction and high computational cost, we propose a new method, which in addition to the Markov property relies on another assumption: the conservation of similarity (CoS). We define the CoS as follows: Given time-series data $U = [u_1, \dots, u_i, \dots, u_m]^\top \in \mathbb{R}^{m\times d}$ and $V_n = [v_1, \dots, v_j, \dots, v_n]^\top \in \mathbb{R}^{n\times d}$, where d is the dimension of a feature vector at each time step, if $U_{1:i}$ and $V_{1:j}$ are similar, then the CoS states that $U_{i+1:\hat{m}}$ and $V_{j+1:\hat{n}}$, where $\hat{m} \leq m$ and $\hat{n} \leq n$, are similar as well. The basic idea of the CoS is that if vehicles A and B behaved similarly for the previous several seconds, they will move similarly in the following several seconds, implying the similarity between the behavior of vehicles A and B is conserved. As an example, we consider race-car driving. While driving around a circuit, drivers typically follow similar paths, the racing line [5], because following the racing line results in the minimum time. This similarity of driving implies that we can predict the behavior of race-car vehicles by just observing the racing line of the track, and the resulting trajectories will be the racing line plus noise.

In this paper, we target our motion-prediction method to standard passenger road vehicles, the driving paths of which are the results of several constraints such as traffic rules (e.g., speed limits, traffic lights, and lane boundaries) and comfort and safety margins of drivers. These requirements result in similar automotive behaviors in similar environments, or at least a finite set of different behaviors. Thus, we conjecture that it is possible to assume that CoS holds to predict future behavior of passenger road vehicles.

II. Notation and Preliminaries

We refer to a vehicle that uses the proposed method as an "ego vehicle" (EV). Suppose that an EV wants to compute the threat of another vehicle in its neighborhood, the region of interest (ROI). The ROI corresponds to the range of sensors such as radars, cameras, and lidars. All the other vehicles in the ROI are denoted by "other vehicles" (OVs) (see Fig. 1). Note that OVs can be autonomous, semi-autonomous, or completely human-driven. We assume that the EV can measure the current state of the OVs longitudinal and lateral position and speed. This information can, for example, be extracted from onboard sensors, such as cameras, radars, and positioning systems (GPS). In addition,

[1] K. Okamoto is with School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. This work was done while working at MERL. kazuhide@gatech.edu

[2] K. Berntorp and S. Di Cairano are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA {karl.o.berntorp,dicairano}@ieee.org

978-1-5090-5992-8/$31.00 ©2017 AACC

we assume the existence of a map of the road; for example, given by a car-navigation system.

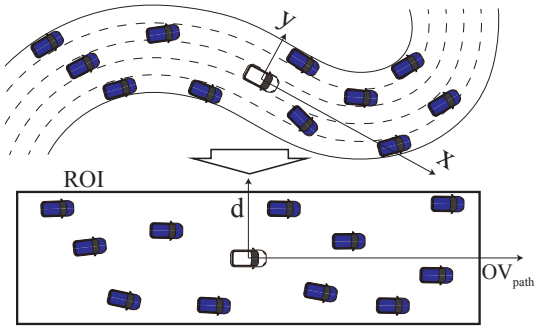

Fig. 1. The EV is the white vehicle in the center, and the OVs are the blue vehicles. Each state of OV is mapped to the road-aligned coordinate, and the shape of the ROI is not affected by the number of lanes or curvature of the road.

Our proposed method is based on the following line of thought. First, we assume a database of stored vehicle behavior. This database is either created iteratively online as vehicles enter the ROI, or is given a priori. Second, when a new OV enters the ROI, the algorithm checks the database for vehicles that share the history with such new OV. Third, after finding the similar vehicles in its database, the EV, using the history of the vehicles, predicts the future motion of the OV. If there is no similar vehicle in the database, the observed behavior with associated prediction information is added to the database.

A. Methods to Compute Similar Vehicles from Traffic Flow

The similarity computation algorithm entails the following two requirements. First, the algorithm can determine that each pair of points is similar. Second, the algorithm can identify that no similar vehicles are present in the database. To compute similarities among driving behaviors, we have explored three ways: clustering [6], [7], classification [8], and anomaly detection [9]. The first requirement rejects clustering algorithms, since belonging to the same cluster does not necessarily mean one behavior is similar to every other behavior in the same cluster. The second requirement rejects the classification algorithms since we do not have datasets of anomalous trajectories a priori or we cannot assume that the majority of a dataset is unfamiliar in the case of online data collection, especially in the early phase.

Because of these two requirements on the similarity computation algorithms, we employ an anomaly-detection method for our similarity-computation algorithm. The proposed method is inspired from an anomaly-detection algorithm in [9], which is a grid-based method. We, however, are not able to employ grid-based methods to evaluate similarities between vehicle trajectories, since vehicle-trajectory-similarity computation requires much finer grids than the problem setting in [9]. Thus, we need modifications as explained in Section III.

III. Dynamic Time Warping for Similarity Computation

This section explains the algorithm we employ to find similar behaviors in database. The algorithm for similarity computation is based on dynamic time warping (DTW) [10], a classical approach for speech recognition. DTW normalizes time and computes the similarity between two time-series data.

A. Dynamic Time Warping

Assume that we want to compare two time-series data U and V. DTW aligns U and V_n by minimizing the warping function $W = \{w_1, \ldots, w_k, \ldots, w_l\}$, where $w_k = (i_k, j_k)$ Each pair $(i, j) \in W$ indicates that u_i and v_j are aligned, and minimizing W corresponds to minimizing the total distance between U and V_n defined as $D_{U,V_n} \triangleq \sum_{k=1}^{l} d_{U,V_n}(i_k, j_k)$, where $d_{U,V_n}(i_k, j_k)$ is a distance measure between u_{i_k} and v_{j_k}. We employ the Euclidean distance, but other metrics can be used. Since computing all possible combinations of alignments is expensive, it is common to compute the minimum path with dynamic programming:

$$D(i, j) = d(i, j) + \min D, \tag{1}$$

where $D(i, j)$ is the cost of the minimum cost path from $(1, 1)$ to (i, j) with $D(1, 1) = d(1, 1)$, and

$$\min D = \min\left(D(i, j-1), D(i-1, j), D(i-1, j-1)\right). \tag{2}$$

Note that for simplicity we omitted the subscript here and $D(i, j) = D_{U,V_n}(i, j)$. The path can be calculated by computing backwards from $(i, j) = (m, n)$.

To make the results consistent, the computation of DTW has several constraints on the warping function: monotonicity, continuity, boundary conditions, adjustment window, and slope constraints. The monotonicity and continuity constraints are defined as

$$i_{k+1} \geq i_k,\ j_{\ell+1} \geq j_\ell. \tag{3}$$

$$i_{k+1} \leq i_k + 1,\ j_{\ell+1} \leq j_\ell + 1. \tag{4}$$

$\forall k \in [1, m-1], \forall \ell \in [1, n-1]$. Boundary conditions are introduced as

$$(i_1, j_1) = (1, 1), \tag{5}$$

$$(i_l, j_l) = (m, n). \tag{6}$$

The adjustment window condition is defined as

$$|i_s - j_s| \leq r \tag{7}$$

with window length $r > 0$. Finally, the slope constraints are defined as

$$\frac{j_{s_p} - j_{s_0}}{i_{s_p} - i_{s_0}} \leq p, \quad \frac{j_{s_q} - j_{s_0}}{i_{s_q} - i_{s_0}} \leq q, \tag{8}$$

where $q \geq 0$ and $p \geq 0$ are the maximum allowed number of steps in the x- and y- direction, respectively.

B. End-Free/Iterative Dynamic Time Warping

The standard DTW explained above compares two time-series data assuming that the entire sequence is available. However, we wish to find similar trajectories in the database *before* observing the entire trajectory of the vehicle of interest. Therefore, we cannot directly employ the standard DTW in our vehicle trajectory similarity computation. Several researchers have proposed ways to improve standard DTW [11], [12]. In this subsection, we introduce two modified DTW approaches that meet our specific requirements.

To have more closely aligned paths for partial time-series data comparison, we disregard some of the constraints of the standard DTW: the terminal constraint in the boundary conditions (6), the adjustment window conditions (7), and the slope constraint conditions (8). We refer to our two approaches as "end-free DTW" and "iterative DTW". The end-free DTW computes similarity between two trajectories without fixing the terminal point at (m, n). The iterative DTW is a computationally more efficient method than end-free DTW if the similarity of trajectories at the previous time step is given.

1) End-Free DTW Algorithm: In end-free DTW, we change the starting point of the back propagation of dynamic programming, that is, (m, n) for standard DTW. Here, we assume that U with length m is one of the data that the EV already has, and V_n with length $n \leq m$ is the query data. We calculate the minimum of $d_{U,V_n}(:, n)$, which corresponds to finding the closest data point in U to $V_n(n)$.

$$i_{\text{endfree}} = \underset{1 \leq i \leq m}{\operatorname{argmin}} \, d_{U,V_n}(i, n). \tag{9}$$

Then, we regard this point as the new terminal constraint and the starting point of dynamic programming: $w_l = (i_{\text{endfree}}, n)$. The rest of the process is as in (1) and (2).

2) Iterative DTW Algorithm: If we already have the minimum cost path $W(U, V_n)$ and receive a new data point v_{n+1}, then the computation of the cost path $W(U, V_{n+1})$, where $V_{n+1} = [V_n^\top, v_{n+1}]^\top$, from the beginning is redundant. Instead, we propose an iterative method to compute $W(U, V_{n+1})$. One of our preliminary results shows that the computational speed of iterative DTW is up to 60 times faster than that of the end-free DTW.

Suppose that we employed end-free DTW and already have $W(U, V_n)$, which is mapped as a trajectory on an $m \times n$ grid as shown in Fig. 2a. If we obtain new data v_{n+1}, then we need to find the path that leads from $(1, 1)$ to the minimum of the $(n+1)$th row as described in Fig. 2b. The algorithm is as follows. First, we compute the minimum of the $(n+1)$th row of the grid.

$$i_{\text{Iterative}} = \underset{1 \leq i \leq m}{\operatorname{argmin}} \, d_{U,V_{n+1}}(i, n+1). \tag{10}$$

Then, we perform dynamic programming (1) and (2) starting from $(i_{\text{Iterative}}, n+1)$. If the path collides with $W(U, V_n)$, then we stop calculation, and add the rest of $W(U, V_n)$ to the path. Since we already have the minimum-cost trajectory from $(1, 1)$ to the point where $W(U, V_n)$ and $W(U, V_{n+1})$ collides, we can reduce computational cost.

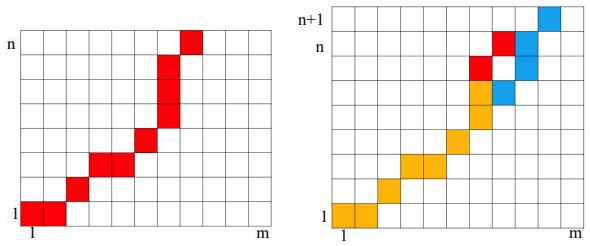

Fig. 2. Iterative dynamic time warping. The red area is the minimum path between U and V_n only. The blue area is the minimum path between U and V_{n+1} only. The yellow area is the shared portion of the two minimum paths.

IV. Similarity-based Vehicle Motion Prediction

Having modified the DTW methods to our purposes, we introduce our similarity-based vehicle motion prediction algorithm, whose structure is shown in Algorithm 1. The framework is similar to the algorithm in [9], but it differs mainly in two ways: the `map` function (Line 10) and the `hasPath` function (Line 16).

Mapping of the trajectory: To directly compute behavioral similarity, we need the shape of the ROI to be unchanged depending on the geometry of the road. The mapping function in our algorithm maps trajectories from the EV-fixed Cartesian coordinate frame from to a road-aligned coordinate frame (see Fig. 1). The sensors of the EV can obtain position (x, y) of the vehicle of interest. Having a map of the road, the EV can measure the distance to OV along the path OV_{path}, and deviation from the center line d.

Existence of similar paths: The `hasPath` function returns a set of similar trajectories to the incoming trajectory. We evaluate this similarity with the methods introduced in Section III as follows:

$$\texttt{hasPath}(U, \omega) = \left\{ \bar{U} \subseteq U | \texttt{DTW}(\omega, \bar{U}) < \theta_{\text{sim}} \right\}, \tag{11}$$

where θ_{sim} denotes the threshold for trajectory similarity computation. If the output of the `DTW` function is less than this threshold, we regard the trajectories as similar. The `DTW` function employs either end-free or iterative DTW depending on where the data window ω starts (see Algorithm 2). If the StartPoint, where the distance between $\bar{U}$ and $\omega(1)$ is the minimum, is the same as before, then the warping function $W(\bar{U}, \omega)$ at the previous time step is available to compute the warping function at the current step. Because of such availability, iterative DTW is employed. Otherwise, end-free DTW is used.

Evaluating the Similarity: The `support` function on Line 16 in Algorithm 1 evaluates the similarity of the query trajectory to the trajectories in the database. It calculates how much of the database the incoming trajectory is similar to; for example, one possibility is to define similarity as $\texttt{support}(U_i, U_{i-1}) = |U_{i-1}|/|U_i|$, where $|\cdot|$ is the cardinality. If the output of this `support` function is less than the specified threshold θ, the algorithm resets the database

Algorithm 1 Similarity-based Motion Prediction.
U: database of mapped trajectories; θ: anomaly threshold; ws: window size.

$U_0 \leftarrow U$ % Initialize similar-trajectory database
$\omega \leftarrow \emptyset$ % Data window in V to compute similarity
for $t \leftarrow 1$ **to** T **do**
  if A new incoming trajectory V appears **then**
    $i \leftarrow 0$
  end if
  if V is not completed **then**
    $i \leftarrow i + 1$
    $\tilde{v}_i \leftarrow \texttt{map}(v_i)$
    if $i \geq ws \wedge$ fixed window size **then**
      $\omega \leftarrow \{\omega(2:\texttt{end}), \tilde{v}_i\}$
    else
      $\omega \leftarrow \{\omega, \tilde{v}_i\}$
    end if
    $U_t \leftarrow \texttt{hasPath}(U_{t-1}, \omega)$
    if $\texttt{support}(U_{t-1}, U_t) < \theta$ **then**
      $U_t \leftarrow U$
      if fixed window **then**
        $\omega \leftarrow \tilde{v}_i$
      end if
      Predict future trajectory using method in Section IV-A
    else
      Predict future trajectory with U_t
    end if
  end if
  $t \leftarrow t + 1$
end for

Algorithm 2 DTW function used in `hasPath`

Input: ω: query trajectory; $\bar{U}$: one trajectory in the database; oldStartPoint, oldWarpingFunction
Output: Similarity measure $\tilde{\theta}$, newWarpingFunction, newStartPoint

StartPoint $\leftarrow \text{argmin}_i \|\bar{U}(i) - \omega(1)\|^2$,
if StartPoint **equals** oldStartPoint **then**
  [$\tilde{\theta}$,newWarpingFunction]$\leftarrow$ IterativeDTW(ω,$\bar{U}$,oldWarpingFunction)
else
  [$\tilde{\theta}$, newWarpingFunction] $\leftarrow$ EndFreeDTW(ω,$\bar{U}$)
end if

and the starting point of the window in the adaptive-window-size case. Note that when the window size is fixed, since the StartPoint always changes, we only use the end-free DTW.

A. Intention Recognition and Motion Prediction for Unfamiliar Vehicles

When the database is not rich enough, which can happen in the early stage of executing the algorithm, there might not be any similar trajectories in the database. Hence, CoS cannot be used for predicting the future motion of a new vehicle. We solve this problem by building a hybrid system with an intention recognizer and a subsequent sampling-based motion-prediction method.

Hence, when there are no trajectories in the database that are similar to the one we are currently interested in, we activate an intention-recognition method, based on random forests [13], a supervised-learning algorithm. The available intentions are driving straight, turning left, or turning right; and brake, accelerate, or maintain velocity. We label the data for training the algorithm, and then let the algorithm output the two intentions.

As a motion predictor, we employ sequential Monte-Carlo [14]. In previous work we have developed a sampling-based motion planner based on particle filtering [15], in which task specifications y were used to guide the motion planner to the relevant parts of the state space. Particle filters numerically estimate probability distributions $p(x_k|y_{0:k})$ by generating N random states $\{x_k^i\}_{i=1}^N$ at each time step k and assigning a probability weight w_k^i, which reflects how well the state explains the observations y_k. In this work, the random states are generated by sampling the conditional distribution $q(x_{k+1}|x_k^i, y_{k+1}) = p(x_{k+1}|x_k^i, y_{k+1})$. This choice leads to the weight update $w_{k+1}^i \propto p(y_{k+1}|x_k^i)w_k^i$. For a linear, Gaussian measurement relation in the form $y_k = Hx_k + e_k$, where e_k is the Gaussian measurement noise, the expression is analytic,

$$p(x_{k+1}|x_k^i, y_{k+1}) = \mathcal{N}\left(x_{k+1}|\hat{x}_{k+1}^i, (\Sigma_{k+1}^i)^{-1}\right) \quad (12)$$

where $\mathcal{N}(x|\mu, \Sigma)$ is the Gaussian density given mean μ and covariance Σ,

$$\hat{x}_{k+1}^i = f(x_k^i) + L_k^i(y_{k+1} - \hat{y}_{k+1}^i), \quad (13)$$

$$\Sigma_{k+1}^i = \left((H_k^i)^\top R_{k+1}^{-1} H_k^i + Q_k^{-1}\right)^{-1}, \quad (14)$$

$$L_k^i = Q_k(H_k^i)^\top (H_k^i Q_k (H_k^i)^\top + R_{k+1})^{-1}, \quad (15)$$

$$\hat{y}_{k+1}^i = H_k^i f(x_k^i), \quad (16)$$

$$H_k^i = \left.\frac{\partial h}{\partial x}\right|_{f(x_k^i)}, \quad (17)$$

and Q_k is the process-noise covariance. The likelihood in the weight update is given as

$$p(y_{k+1}|x_k^i) = \mathcal{N}\left(y_{k+1}|\hat{y}_{k+1}^i, H_k^i Q_k (H_k^i)^\top + R_{k+1}\right), \quad (18)$$

where R is the measurement-noise covariance.

Now, our intentions define desired locations and velocity profiles in the road-aligned coordinate frame we employ. For instance, the intention of turning left implies an intended lateral position equal to the middle lane of the lane left to our current lane, and similar for the other options. Also, intention of braking implies a desired deceleration profile. Hence, transforming intentions to actual expressions on the road gives us a measurement relation $y_k = Hx_k + e_k$, which is linear because we model everything in the road-aligned frame (Fig. 1). The Q and R matrices are tuning parameters in the current setup, but future work is to learn these online. We will show the effectiveness of this hybrid method in Section V-B, and refer to an upcoming paper for details about the intention recognition and motion prediction.

Remark 1: When there is no database a priori, the intention recognition and sampling-based motion prediction must be activated for at least one trajectory. Thus, a probability distribution is associated with each trajectory that is outputted from Algorithm 1. Hence, our algorithm does not only provide a predicted path of the OV, but also a measure of the threat level of the OV relative to the EV. This implies that when motion prediction has been activated and the resulting observed trajectory added to the database, the motion prediction does not need to be invoked for new trajectories, as long as they are similar to one in the database. Note that since we eventually will have observed a whole trajectory, we can leverage smoothing techniques to adjust the predicted probability distribution, which can be applied to later observed vehicles.

V. Simulation

This section evaluates the performance of the proposed algorithm in three numerical simulations. The first scenario clarifies the concept of our method. In the second example, we build a hybrid system with a sampling-based method to overcome a weakness of similarity-based motion prediction methods The last example demonstrates the lower computational cost of our algorithm than conventional motion prediction methods.

A. Motion Prediction only with Similarity-based Method

Suppose that the EV has observed ten vehicle trajectories (see Fig. 3a), and the EV has a new vehicle on the right lower edge of its ROI (Fig. 3c). The new vehicle will follow a trajectory depicted in Fig. 3b. As five of the trajectories in the database start from the bottom right of the ROI, these five trajectories can be possible future behaviors for the new vehicle, as depicted in Fig. 3b. The other five trajectories, starting from the bottom left of the ROI, are not similar. These trajectories are denoted by dashed lines. As time passes, the number of candidate paths decreases since the observed data will deviate from some of the trajectories in the database (Fig. 3d). Eventually, only one trajectory in the database remains similar to the path of the new vehicle (see Fig. 3e). As the new vehicle remains similar to one of the trajectories in the database, the EV does not have to activate classical motion prediction algorithms, which predict with less data and offer a shorter prediction time horizon.

B. Overtaking by Similar Vehicles

In the first scenario, we showed the concept of the proposed method. The second example clarifies the benefit of our method with a more realistic driving scenario. The dynamics of each vehicle in this scenario is a single-track vehicle model [5]. Suppose the EV and one OV (OV1) drive on a straight road at 60km/h, and another OV (OV2) has just entered the ROI of the EV from behind at 70km/h. Since OV2 is faster than the EV and OV1, OV2 will overtake the EV and OV1 in the near future. Suppose that the EV has never observed a vehicle behavior similar to that of OV2, implying the EV cannot predict the motion of OV2

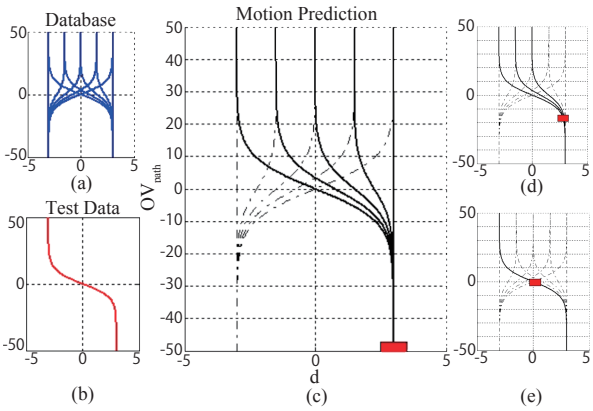

Fig. 3. Example of similarity-based motion prediction. Based on the database (a), the EV predicts the future motion of the vehicle in (c). The true path for the OV is in (b). Based on the observation, five trajectories become the future path candidates of the new vehicle as depicted in (c). As time passes, the number of path candidates decreases to three (d) and eventually one (e).

from the database. The EV can predict the future behavior of OV2 by activating a motion prediction algorithm with a relatively shorter time horizon, depicted in Figs. 4a-c. The dots in magenta are predicted OV2 positions in the near future, generated by a sampling-based prediction algorithm introduced in Section IV-A

Now suppose that another OV (OV3) appears at the bottom of the ROI, and the EV finds that the behavior of OV3 is similar to that of OV2. Then, the EV does not perform a motion prediction algorithm on OV3, but assumes that OV3 will behave similarly to OV2. Figure 4d shows this situation. The magenta dots are predictions of the future behavior of OV2, and the green line is the predicted path that OV3 follows and that corresponds to the past trajectory of OV2. Once the OV2 exits the ROI, the EV no longer has to activate motion prediction, as shown in Fig. 4e. The EV predicts that OV3 will follow the trajectory drawn by OV2 with the same probability distribution estimated by the motion predictor since they remain similar.

C. Road Test

The last scenario demonstrates that as time passes, the number of new vehicle behaviors decreases, implying that classical motion prediction is not necessary. In this scenario, we simulate 16 vehicles on a stretch of an actual road. This course simulates a sub-urban asphalt road, and the minimum radius of curve is 60 meters. Again, we employ a single-track vehicle model, and each vehicle is controlled by a proportional controller that takes the deviation of the direction angle relative to the road angle as a reference. All the vehicles drive at a constant speed, and the EV is the fastest among all of the vehicles.

We simulate one hour driving of the EV to determine if the algorithm can distinguish the vehicle as one in the database (a familiar vehicle) or not in the database (an unfamiliar vehicle). Since the dynamics and the controller of each OV do not change, it is expected that as time passes the number

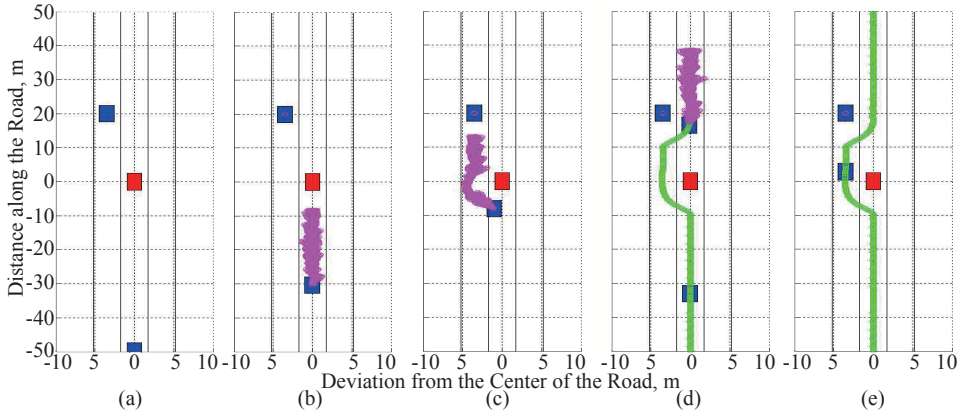

Fig. 4. Example of our motion prediction algorithm, a hybrid of similarity-based and sampling based. The EV (red) and OV1(blue on the left in (a)) are driving at relative speed zero. Another OV (OV2) (blue on the bottom in (a)) comes from behind at a faster speed and overtakes the EV and OV1. The EV conducts motion prediction of this motion of OV2 with a sampling-based method as depicted in (b) and (c). However, for the next OV3 (blue on the bottom of (d)), which behaves similarly to the OV2, the EV does not employ a sampling-based method but instead, a similarity-based method to predict the future motion ((d) and (e)). Please note that the scale on the x- and y-axes is different.

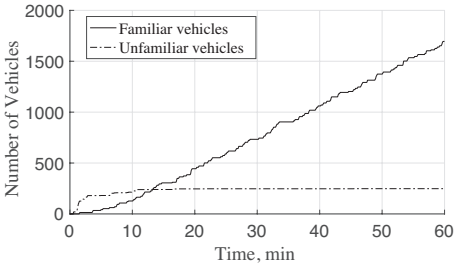

Fig. 5. Cumulative sum of the number of unfamiliar/familiar vehicles detected. As time passes, unfamiliar vehicles are not detected, but more familiar vehicles are detected.

of unfamiliar vehicle detections decreases, and the number of familiar vehicle detections increases.

Figure 5 shows the cumulative sum of the number of familiar/unfamiliar vehicles detected at each time step. As expected, unfamiliar vehicles are detected at first, but as time passes, their behavior is recorded in the database, and their detection becomes rare. By contrast, familiar vehicles are not detected at first since the number of vehicles recorded in the database is small. As the number of vehicles in the database increases, the detection of familiar vehicles occurs, indicating that motion prediction is conducted with a similarity-based method. All predictions are eventually performed by the similarity-based method.

VI. Conclusion

This paper addressed a new approach to predict future behavior of vehicles based on a similarity measure. The proposed method differs from preceding research because of the assumption that some vehicles behave similarly due to restrictive constraints such as traffic rules. We exhibited the effectiveness with three numerical simulations.

Future research direction may include compressing database size with trajectory clustering, investigating fast algorithms to find similar trajectories, and leveraging the proposed framework to design robust motion planners [15].

References

[1] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *arXiv preprint arXiv:1604.07446*, 2016.

[2] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty - a control perspective," *European J. Control*, vol. 24, pp. 14–32, 2015.

[3] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *Robomech Journal*, vol. 1, no. 1, p. 1, 2014.

[4] L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. How, J. Leonard, I. Miller, M. Campbell, D. Huttenlocher *et al.*, "The MIT–Cornell collision and why it happened," *J. Field Robotics*, vol. 25, no. 10, pp. 775–807, 2008.

[5] K. Berntorp, B. Olofsson, K. Lundahl, and L. Nielsen, "Models and methodology for optimal trajectory generation in safety-critical road–vehicle manoeuvres," *Veh. Syst. Dyn.*, vol. 52, no. 10, pp. 1304–1332, 2014.

[6] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[7] C. Sung, D. Feldman, and D. Rus, "Trajectory clustering for motion prediction," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vilamoura, Portugal, Oct. 2012.

[8] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1081–1094, 2008.

[9] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, S. Li, and Z. Wang, "iBOAT: Isolation-based online anomalous trajectory detection," vol. 14, no. 2, pp. 806–818, 2013.

[10] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," vol. 26, no. 1, pp. 43–49, 1978.

[11] X. Anguera, R. Macrae, and N. Oliver, "Partial sequence matching using an unbounded dynamic time warping algorithm," in *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Dallas, TX, USA, May 2010.

[12] S. Dixon, "Live tracking of musical performances using on-line time warping," in *Int. Conf. Digital Audio Effects*, Madrid, Spain, Sep. 2005.

[13] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[14] A. Doucet, N. De Freitas, and N. Gordon, "An introduction to sequential Monte Carlo methods," in *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 3–14.

[15] K. Berntorp and S. Di Cairano, "Particle filtering for online motion planning with task specifications," in *Amer. Control Conf.*, Boston, MA, Jul. 2016.