Discrete and Continuous, Probabilistic Anticipation for Autonomous Robots in Urban Environments

Frank Havlak and Mark Campbell

***Abstract*—This paper develops a probabilistic anticipation algorithm for dynamic objects observed by an autonomous robot in an urban environment. Predictive Gaussian mixture models are used due to their ability to probabilistically capture continuous and discrete obstacle decisions and behaviors; the predictive system uses the probabilistic output (state estimate and covariance) of a tracking system and map of the environment to compute the probability distribution over future obstacle states for a specified anticipation horizon. A Gaussian splitting method is proposed based on the sigma-point transform and the nonlinear dynamics function, which enables increased accuracy as the number of mixands grows. An approach to caching elements of this optimal splitting method is proposed, in order to enable real-time implementation. Simulation results and evaluations on data from the research community demonstrate that the proposed algorithm can accurately anticipate the probability distributions over future states of nonlinear systems.**

***Index Terms*—Field robots, Gaussian mixture models, nonlinear filtering.**

I. Introduction

AUTONOMOUS urban driving is an important and maturing field in mobile robotics. Intelligent vehicles promise to improve both road safety, vehicle efficiency, and convenience [1]–[3]. The finals of the 2007 DARPA Urban Challenge (DUC) was an empirical evaluation of the state of the art at the time, integrating 11 autonomous vehicles together with other robots and human drivers in an urban environment for long-duration operations (>4 h) [4]–[6]. Continued development in the field has led to autonomous cars beginning to drive in real urban environments alongside civilian traffic [7]–[10].

Many autonomous cars use primarily reactionary planners that rely on rapid replanning in order to respond to the dynamic environments in which they operate [11]. A collision between the Massachusetts Institute of Technology (MIT) and Cornell entries was one of several examples in the 2007 DUC that raised safety concerns about reactionary planning for autonomous driving [12]. One approach proposed to more intelligently handle autonomous driving in dynamic environments is to incorporate "anticipation" into path planning, or predicting the future motion of dynamic obstacles for use in planning. This area has been an active topic in mobile robotics in recent years [13], [14]. This paper presents a formal method for probabilistically anticipating the future behavior of tracked obstacles.

The problem of anticipation is inherently probabilistic, as it is impossible to know the true future behavior of dynamic obstacles that make their own independent decisions. In addition, the behavioral models used to anticipate obstacle behavior are often highly nonlinear. In the literature, several algorithms have been proposed to simplify the problem, such as assuming no uncertainty in future obstacle behavior [15]–[17]. These algorithms are well suited for cooperative situations, where the obstacles can communicate their intentions to the robot, or for short anticipation horizons. However, they do not provide sufficient information for reasoning about an obstacle with unknown intentions over a significant anticipation horizon. Similarly, several proposed methods consider only a subset of obstacle uncertainty, such as along track error [18]. These approaches reduce the complexity of the problem to a manageable level, while still considering the probabilistic aspects of obstacle anticipation, but are typically very simple and narrow in their application.

Another class of algorithms applies standard estimation filters (Kalman Filter, Extended Kalman Filter, etc.) to the problem of anticipation [19], [20]. Such approaches assume a model for the behavior of the obstacle, and provide mathematically rigorous, probabilistic estimates of that obstacle's state over the anticipation horizon. These approaches are well suited to obstacles that are accurately described by linear models because they maintain a single Gaussian to represent the uncertainty. For obstacles with more complex behaviors, such as those based on nonlinear dynamics (e.g., cars, bicycles, etc.) and those that make discrete decisions (e.g., intersections, passing, etc.), the uncertainty of the anticipated obstacle state becomes inaccurate very quickly, thus severely limiting the prediction horizon. In [21], [20] has been extended by the authors to use Gaussian mixture models (GMMs) to capture multiple obstacle hypothesis but is still focused primarily on linear obstacle models.

More complex uncertainties can be addressed, while avoiding the linearization problems of standard filters; for example, Monte-Carlo (MC) methods [22]. These approaches are attractive because they can consider complex, non-Gaussian uncertainties and allow for the use of nonlinear obstacle models to capture complex obstacle behavior. However, the accuracy of prediction scales with the number of particles, and there are no guarantees that the particle set effectively covers the true distribution of possible future obstacle states. Because the assumed dynamics model for the obstacle has to be evaluated at every particle used in anticipation, increasing confidence in the estimate is strongly traded with computational resources.

Manuscript received April 16, 2013; revised October 28, 2013; accepted November 11, 2013. Date of publication December 11, 2013; date of current version April 1, 2014. This paper was recommended for publication by Associate Editor R. Eustice and Editor G. Oriolo upon evaluation of the reviewers' comments.

The authors are with the Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853 USA (e-mail: fh92@cornell.edu; mc288@cornell.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2013.2291620

An GMM-based predictor is proposed in this paper to anticipate obstacle behaviors [23]. GMMs provide a well-suited representation to probabilistically anticipate non-Gaussian obstacle states. Here, the GMM is used to uniquely include discrete state elements that capture complex, high-level obstacle behaviors. Accurate anticipation of a wide variety of dynamic obstacles is ensured using a novel method for detecting linearization errors using sigma-point methods, and adjusting the mixture accordingly by optimally splitting inaccurately propagated mixands. This approach reduces the individual covariances of inaccurately propagated mixands, bringing them into a nearly linear regime. The presented algorithm provides accurate, probabilistic future obstacle state estimates and is shown to perform well even with highly nonlinear obstacle motion models.

This paper is outlined as follows. Section II-A defines the representation of the obstacle state, and Section II-B defines the mixture model. Sections II-C describes the discrete and continuous anticipation of the obstacle state. Section II-D provides an overview of the proposed algorithm. The details of temporal propagation are described in Section III, including the nonlinearity detection and mixand splitting. Section IV-A provides an example implementation of the anticipation algorithm in simulation, and Section IV-B demonstrates the efficacy of the algorithm on a real dataset. Section IV-C demonstrates the potential safety improvements by applying the proposed algorithm to the 2007 MIT-Cornell autonomous collision.

II. Hybrid Mixture Anticipation Algorithm

The hybrid mixture anticipation algorithm described in this paper is designed to predict the probability distribution over the state of a tracked obstacle forward in time. Here, hybrid is used to denote jointly discrete and continuous components. The intended application of the presented algorithm is to make accurate, probabilistic information about future obstacle behaviors available for use in path planning. In order to provide the most general algorithm, obstacle models can include nonlinear behaviors, as well as discrete variables to capture higher level decisions. To meet these requirements, the distribution over the obstacle state is described using a hybrid Gaussian/discrete mixture model (hGMM).

A. Obstacle State Model

The obstacle state at time k ($\mathbf{x}_k$) is assumed to have continuous elements ($\mathbf{x}_k^{\mathrm{C}}$, representing position, velocity, etc.) and discrete elements ($\mathbf{x}_k^{\mathrm{D}}$, representing behavioral modes). The state vector is partitioned accordingly:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^{\mathrm{C}} \\ \mathbf{x}_k^{\mathrm{D}} \end{bmatrix}. \tag{1}$$

A model for the evolution of the obstacle state is assumed to be available and partitioned into discrete (f^{D}) and continuous (f^{C}) components:

$$\begin{aligned} \mathbf{x}_{k+1}^{\mathrm{D}} &= f^{\mathrm{D}}\left(\mathbf{x}_k^{\mathrm{D}}, \mathbf{x}_k^{\mathrm{C}}, \mathbf{v}_k\right) \\ \mathbf{x}_{k+1}^{\mathrm{C}} &= f^{\mathrm{C}}\left(\mathbf{x}_{k+1}^{\mathrm{D}}, \mathbf{x}_k^{\mathrm{C}}, \mathbf{v}_k\right) \end{aligned} \tag{2}$$

where $\mathbf{v}_k$ is the process noise vector at time k. f^{D} and f^{C} are both functions that take as inputs *point values* for the vehicle state ($\mathbf{x}_k^{\mathrm{D}}, \mathbf{x}_k^{\mathrm{C}}$) and process noise ($\mathbf{v}_k$), as this is typically how models are defined. The following sections generalize these to h^{D} and h^{C}, which operate on distributions over, rather than samples from, the vehicle state and process noise.

B. Hybrid Mixture Probability Distribution Representation

The probability distribution $p(\mathbf{x}_k)$ is approximated using an hGMM. The hGMM extends the GMM presented in [23] by including discrete variables. Discrete variables allow the hGMM to capture both continuous behavior of a system (position, velocity, etc.) as well as high-level, abstract behaviors (turning left, going straight, etc.). The hGMM inherits the capability of GMMs to represent many general probability distributions, with increasing accuracy in the limit of a large number of mixands, while still maintaining the convenient computational properties of Gaussian distributions. The hGMM is defined as

$$p(\mathbf{x}_k) = \sum_{i=1}^{M_k} w_k^i \cdot p^i(\mathbf{x}_k)$$

where M_k is the number of mixands in the hGMM at time k, and w_k^i are the weights on each mixand at time k, such that

$$\sum_{i=1}^{M_k} w_k^i = 1 \quad \text{and} \quad w_k^i > 0 \quad \forall i. \tag{3}$$

The mixand $p^i(\mathbf{x}_k)$ is defined as a Gaussian distribution over the continuous state elements and a hypothesis (using a delta function) over the discrete state elements:

$$p^i(\mathbf{x}_k) = \delta(\mathbf{x}_k^{\mathrm{D}} - \alpha_k^i) \cdot \mathcal{N}(\mathbf{x}_k^{\mathrm{C}} | \mu_k^i, \Sigma_k^i) \tag{4}$$

where α_k^i is the mixand hypothesis of the discrete state, and μ_k^i and Σ_k^i are the mean and covariance of the Gaussian distribution over the continuous state.

C. Hybrid Mixture Propagation

The hGMM formulation enables each mixand to be propagated forward in time independently using the dynamics model (2), thereby reducing the complexity of the problem. The propagation of the mixands is complicated in two ways, however. First, more than one discrete state can be transitioned into (for example, a tracked obstacle approaching an intersection may turn left, turn right, or continue straight). Second, the variance on the continuous state elements may grow to the point where the mixand can no longer be accurately propagated through the dynamics model. Each of these are addressed in the proposed probabilistic anticipation algorithm.

Consider the discrete mixand propagation through the dynamics function from (2):

$$\begin{bmatrix} \left(\alpha_k^1, \mu_k^1, \Sigma_k^1, w_k^1\right) \\ \vdots \\ \left(\alpha_k^{M_k}, \mu_k^{M_k}, \Sigma_k^{M_k}, w_k^{M_k}\right) \end{bmatrix} \overset{h^{\mathrm{D}}}{\rightarrow} \begin{bmatrix} \left(\alpha_{k+1}^1, \mu_{k-}^1, \Sigma_{k-}^1, w_{k-}^1\right) \\ \vdots \\ \left(\alpha_{k+1}^{M_{k-}}, \mu_{k-}^{M_{k-}}, \Sigma_{k-}^{M_{k-}}, w_{k-}^{M_{k-}}\right) \end{bmatrix} \tag{5}$$

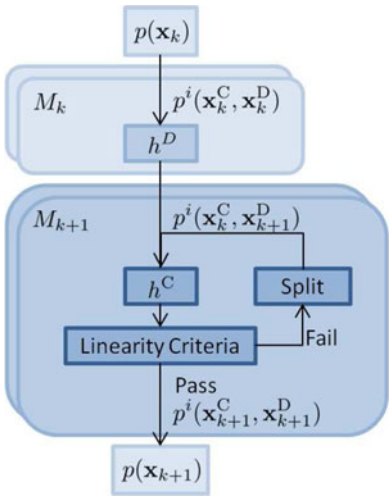

Fig. 1. Block diagram illustrating propagation of hybrid Gaussian mixture through a given dynamics model.

where α_{k+1}^i is the discrete state for the ith mixand at the next time step $(k+1)$. In cases where a mixand has multiple possible next discrete states (such as a choice of roads at an intersection), the mixand is split so that one copy of the mixand can transition to each possible discrete state. Although the continuous aspects of the mixand (μ_k^i, Σ_k^i) are not affected by the discrete propagation, the time index $k-$ is used to account for growth in the mixture size due to mixands transitioning to multiple discrete states.

Similarly, the continuous mixand propagation function h^{C} is defined as

$$\begin{bmatrix} (\alpha_{k+1}^1, \mu_{k-}^1, \Sigma_{k-}^1, w_{k-}^1) \\ \vdots \\ \left(\alpha_{k+1}^{M_{k-}}, \mu_{k-}^{M_{k-}}, \Sigma_{k-}^{M_{k-}}, w_{k-}^{M_{k-}}\right) \end{bmatrix} \xrightarrow{h^{\mathrm{C}}} \begin{bmatrix} (\alpha_{k+1}^1, \mu_{k+1}^1, \Sigma_{k+1}^1, w_{k+1}^1) \\ \vdots \\ \left(\alpha_{k+1}^{M_{k+1}}, \mu_{k+1}^{M_{k+1}}, \Sigma_{k+1}^{M_{k+1}}, w_{k+1}^{M_{k+1}}\right) \end{bmatrix} \tag{6}$$

where the right-hand side characterizes the propagated hGMM after one full step.

D. Algorithm Overview

The propagation steps of the hGMM are summarized in Fig. 1. First, each mixand $p^i(\mathbf{x}_k)$ in the hGMM at time k is propagated through the discrete dynamics. This step has the potential to increase the number of mixands in the hGMM due to the possibility of multiple discrete decisions being available to mixands. Next, each mixand in the hGMM after discrete propagation is propagated through the continuous dynamics. This step includes a linearity criteria (see Section III-B) that ensures the accuracy of propagation by splitting mixands that propagate inaccurately (see Section III-C). After the continuous propagation is completed, the probability distribution over the state at time $k+1$ can be written as

$$p(\mathbf{x}_{k+1}) = \sum_{i=1}^{M_{k+1}} w_{k+1}^i \cdot \delta(\mathbf{x}_{k+1}^{\mathrm{D}} - \alpha_{k+1}^i) \cdot \mathcal{N}(\mathbf{x}_{k+1}^{\mathrm{C}} | \mu_{k+1}^i, \Sigma_{k+1}^i). \tag{7}$$

Note that the number of mixands (M_{k+1}) in the hGMM after the continuous propagation can be larger than the number of mixands after the discrete propagation (M_{k-}) due to the possibility that some mixands were split to ensure accurate propagation through nonlinear dynamics.

III. Mixand Continuous Propagation

The first step in the mixture continuous propagation is to predict the continuous state distribution in each of the mixands forward one time step. The continuous dynamics function h^{C} (6) uses the sigma-points (SP) transform (sometimes called the unscented transform) [24]–[27] to propagate the mixand through the continuous dynamics function f^{C} (2). Sigma-points are well explored for use in estimation problems involving nonlinear dynamics and measurement models [24], [26]. The Gaussian distributions $\mathcal{N}(\mathbf{x}_k^{\mathrm{C}} | \mu_k^i, \Sigma_k^i)$ and $\mathcal{N}(\mathbf{v}_k | \mathbf{0}, \Sigma_{\mathbf{v}})$ are approximated by deterministically choosing sets of points $\boldsymbol{\chi}_k^i$ and $\boldsymbol{\Upsilon}_k$, which are called sigma points

$$\begin{aligned} \boldsymbol{\chi}_k^i &= [\, \chi_k^{i,1}, \ldots, \chi_k^{i,1+2n_{\mathbf{x}}+2n_{\mathbf{v}}} \,] \\ \boldsymbol{\Upsilon} &= [\, \Upsilon^1, \ldots, \Upsilon^{1+2n_{\mathbf{x}}+2n_{\mathbf{v}}} \,] \end{aligned} \tag{8}$$

where $n_{\mathbf{x}}$ is the dimension of $\mathbf{x}_k^{\mathrm{C}}$ and $n_{\mathbf{v}}$ is the dimension of $\mathbf{v}_k$; the process noise distribution is assumed to be time-invariant and Gaussian without loss of generality—time varying GMM process noise distributions can be used with minor modifications. To find the individual sigma points in (8), the matrix square-roots of the covariances Σ_k^i and $\Sigma_{\mathbf{v}}$ are first solved:

$$\begin{aligned} \mathbf{S}_{\mathbf{x},k}^i \cdot \left(\mathbf{S}_{\mathbf{x},k}^i\right)^T &= \Sigma_k^i \\ \mathbf{S}_{\mathbf{v}} \cdot \left(\mathbf{S}_{\mathbf{v}}\right)^T &= \Sigma_{\mathbf{v}} \end{aligned}$$

such that

$$\begin{aligned} \mathbf{S}_{\mathbf{x},k}^i &= [\, S_{\mathbf{x},k}^{i,1}, \ldots, S_{\mathbf{x},k}^{i,n_{\mathbf{x}}} \,] \\ \mathbf{S}_{\mathbf{v}} &= [\, S_{\mathbf{v}}^1, \ldots, S_{\mathbf{v}}^{n_{\mathbf{v}}} \,]. \end{aligned} \tag{9}$$

The individual sigma-points are then defined using these matrix square-roots and a parameter λ

$$\chi_k^{i,j} = \begin{cases} \mu_k^i, & j \in \{0, [2n_{\mathbf{x}}+1, 2n_{\mathbf{x}}+2n_{\mathbf{v}}]\} \\ \mu_k^i + \gamma \cdot S_{\mathbf{x},k}^{i,j}, & j \in [1, n_{\mathbf{x}}] \\ \mu_k^i - \gamma \cdot S_{\mathbf{x},k}^{i,j-n_{\mathbf{x}}}, & j \in [n_{\mathbf{x}}+1, 2n_{\mathbf{x}}] \end{cases}$$

$$\Upsilon^j = \begin{cases} \mathbf{0}, & j \in [0, 2n_{\mathbf{x}}] \\ \gamma \cdot \mathbf{S}_{\mathbf{v},k}^{j-2n_{\mathbf{x}}}, & j \in [2n_{\mathbf{x}}+1, 2n_{\mathbf{x}}+n_{\mathbf{v}}] \\ -\gamma \cdot \mathbf{S}_{\mathbf{v},k}^{j-2n_{\mathbf{x}}-n_{\mathbf{v}}}, & j \in [2n_{\mathbf{x}}+n_{\mathbf{v}}+1, 2n_{\mathbf{x}}+2n_{\mathbf{v}}] \end{cases} \tag{10}$$

where $\gamma = \sqrt{n_{\mathbf{x}} + n_{\mathbf{v}} + \lambda}$.

Each pair of points $(\chi_k^{i,j}, \Upsilon^j)$ is then individually propagated through $\chi_{k+1}^{i,j} = f^{\mathrm{C}}\left(\alpha_{k+1}^i, \chi_k^{i,j}, \Upsilon^j\right)$, and the resulting

set $\boldsymbol{\chi}^i_{k+1} = \left[\chi^{i,1}_{k+1}, \ldots, \chi^{i,1+2n_{\mathbf{x}}+2n_{\mathbf{v}}}_{k+1}\right]$ is used to evaluate the impact of nonlinearities (see Section III-B) on the accuracy of prediction and to find the predicted distribution at time $k+1, \mathcal{N}(\mathbf{x}^{\mathrm{C}}_{k+1}|\mu^i_{k+1}, \Sigma^i_{k+1})$ (see Section III-A).

A. No Splitting Case

If the mixand passes the linearity criteria described in Section III-B, the propagated sigma points are used to find the predicted mean and covariance (μ^i_{k+1} and Σ^i_{k+1}) for the mixand:

$$\begin{aligned}
\mu^i_{k+1} &= \sum_{j=1}^{1+2n_{\mathbf{x}}+2n_{\mathbf{v}}} \gamma^j \cdot \chi^{i,j}_{k+1} \\
\Sigma^i_{k+1} &= \sum_{j=1}^{1+2n_{\mathbf{x}}+2n_{\mathbf{v}}} \beta^j \cdot \left(\chi^{i,j}_{k+1} - \mu^i_{k+1}\right)\left(\chi^{i,j}_{k+1} - \mu^i_{k+1}\right)^T \\
\gamma^j &= \begin{cases} \dfrac{\lambda}{\lambda + n_{\mathbf{x}} + n_{\mathbf{v}}}, & \text{for } j = 1 \\ \dfrac{1}{2\left(\lambda + n_{\mathbf{x}} + n_{\mathbf{v}}\right)}, & \text{for } j \neq 1 \end{cases} \\
\beta^j &= \begin{cases} \dfrac{\lambda}{\lambda + n_{\mathbf{x}} + n_{\mathbf{v}}} + 2, & \text{for } j = 1 \\ \dfrac{1}{2\left(\lambda + n_{\mathbf{x}} + n_{\mathbf{v}}\right)}, & \text{for } j \neq 1. \end{cases}
\end{aligned} \tag{11}$$

Once the mean and covariance are computed, the propagation of the mixand is complete. If the mixand fails the linearity criteria, it is split into several mixands with reduced covariance, as described in Section III-C.

B. Linearity Criteria

The strength of Gaussian mixture models is that they can accurately approximate non-Gaussian probability density functions that arise either through non-Gaussian process or measurement noise or through nonlinearities in the system model [23], [28]. Traditional Gaussian mixture model propagation algorithms [29], [30] use standard nonlinear filtering techniques, such as Extended Kalman Filtering or Unscented Kalman Filtering, to propagate the individual mixands without considering the impact of local nonlinearities in the system model on the accuracy of propagation. Motivated by a careful examination of the sigma-points, this paper develops a natural measure for how accurately a given mixand is propagated through the system model, along with a splitting method to reduce covariances, which, in turn, improves propagation accuracy.

The SP transform propagates Gaussian mixands through a temporal model using sigma-points; intuitively, this process performs a statistical linearization of the model in the neighborhood of the Gaussian mixand. However, the SP transform does not provide a measure of error introduced by the statistical linearization. In order to evaluate the error introduced by the SP propagation, the linearization residuals are proposed here as a metric. The process is as follows. First, a linear model is defined:

$$\begin{aligned}
\bar{\boldsymbol{\chi}}^i_{k+1} &= \mathbf{A}\bar{\boldsymbol{\chi}}^i_k + \mathbf{b} \cdot \mathbf{1}_n \\
&= [\mathbf{A}, \mathbf{b}] \begin{bmatrix} \bar{\boldsymbol{\chi}}^i_k \\ \mathbf{1}_n \end{bmatrix}
\end{aligned} \tag{12}$$

where the notation $\bar{\boldsymbol{\chi}}^i_k, \bar{\boldsymbol{\chi}}^i_{k+1}$ describes the subset of sigma-points related to the state uncertainty and not to process noise:

$$\begin{aligned}
\bar{\boldsymbol{\chi}}^i_k &= \left[\chi^{i,0}_k, \ldots, \chi^{i,2n_{\mathbf{x}}}_k\right] \\
\bar{\boldsymbol{\chi}}^i_{k+1} &= \left[\chi^{i,0}_{k+1}, \ldots, \chi^{i,2n_{\mathbf{x}}}_{k+1}\right].
\end{aligned} \tag{13}$$

For a linear system, (12) can be solved exactly for $\mathbf{A}$ and $\mathbf{b}$. For a nonlinear system, the residuals can be calculated by casting the linearization as a least-squares problem.

$$[\mathbf{A}^*, \mathbf{b}^*] = \operatorname{argmin}\left(\left\|\bar{\boldsymbol{\chi}}^i_{k+1} - [\mathbf{A}, \mathbf{b}]\begin{bmatrix} \bar{\boldsymbol{\chi}}^i_k \\ \mathbf{1}_n \end{bmatrix}\right\|\right). \tag{14}$$

The residual linearization error (e_{res}) is defined as

$$\begin{aligned}
e_{\mathrm{res}} &= \min\left(\left\|\bar{\boldsymbol{\chi}}^i_{k+1} - [\mathbf{A}, \mathbf{b}]\begin{bmatrix} \bar{\boldsymbol{\chi}}^i_k \\ \mathbf{1}_n \end{bmatrix}\right\|\right) \\
&= \left\|\bar{\boldsymbol{\chi}}^i_{k+1} - [\mathbf{A}^*, \mathbf{b}^*]\begin{bmatrix} \bar{\boldsymbol{\chi}}^i_k \\ \mathbf{1}_n \end{bmatrix}\right\|.
\end{aligned} \tag{15}$$

The residual error is a direct metric of how well the optimal linear model explains the propagation of sigma points through the nonlinear dynamics. If the underlying model is linear, the residual error is zero. For nonlinear systems, the residual linearization error indicates how locally linear or nonlinear the underlying model is in the neighborhood of the sigma-points.

An L–Q factorization can be used to compute e_{res}:

$$\mathbf{LQ} = \begin{bmatrix} \bar{\boldsymbol{\chi}}^i_k \\ \mathbf{1}_n \end{bmatrix}, \mathbf{L} \in \mathbb{R}^{n_{\mathbf{x}}+1, 2\cdot n_{\mathbf{x}}+1} \quad \text{and} \quad \mathbf{Q} \in \mathbb{R}^{2\cdot n_{\mathbf{x}}+1, 2\cdot n_{\mathbf{x}}+1}$$

where

$$\mathbf{L} = [\mathbf{L}_0, \mathbf{0}], \quad \mathbf{L}_0 \in \mathbb{R}^{n_{\mathbf{x}}+1, n_{\mathbf{x}}+1}, \quad \text{and} \quad \mathbf{0} \in \mathbb{R}^{n_{\mathbf{x}}+1, n_{\mathbf{x}}}$$

such that $\mathbf{L}_0$ is lower triangular, and therefore cheaply invertible, and $\mathbf{Q}$ is orthonormal:

$$\mathbf{QQ}^T = \mathbf{I}. \tag{16}$$

Substituting the factorization from (16) into (14) yields

$$\begin{aligned}
[\mathbf{A}^*, \mathbf{b}^*] &= \operatorname{argmin}\left(\|\bar{\boldsymbol{\chi}}^i_{k+1} - [\mathbf{A}, \mathbf{b}]\,\mathbf{LQ}\|\right) \\
&= \operatorname{argmin}\left(\|\bar{\boldsymbol{\chi}}^i_{k+1}\mathbf{Q}^T - [\mathbf{A}, \mathbf{b}]\,\mathbf{L}\|\right).
\end{aligned} \tag{17}$$

Because of the structure of $\mathbf{L}$, the arguments $[\mathbf{A}, \mathbf{b}]$ only effect the first $n_x + 1$ columns of the norm in (17). The matrix $\bar{\boldsymbol{\chi}}^i_{k+1}\mathbf{Q}^T$ is partitioned according to what is explained by the linear model ($\hat{\boldsymbol{\chi}}^i_{k+1}$) and what is not explained by a linear model ($\hat{\boldsymbol{\chi}}^i_{k+1,\mathrm{res}}$). In a block form, this is written as

$$[\mathbf{A}^*, \mathbf{b}^*] = \operatorname{argmin}\left(\left\|\left[\hat{\boldsymbol{\chi}}^i_{k+1}, \hat{\boldsymbol{\chi}}^i_{k+1,\mathrm{res}}\right] - [\mathbf{A}, \mathbf{b}]\,[\mathbf{L}_0, \mathbf{0}]\right\|\right). \tag{18}$$

The optimal arguments $[\mathbf{A}^*, \mathbf{b}^*]$ can be extracted from the partitioned matrix in (18) and $\mathbf{L}_0$:

$$[\mathbf{A}^*, \mathbf{b}^*] = \hat{\boldsymbol{\chi}}^i_{k+1}\mathbf{L}_0^{-1}. \tag{19}$$

The desired linearization error e_{res}, which is defined in (15), simplifies to

$$e_{\text{res}} = \left\| \left[\mathbf{0}, \hat{\boldsymbol{\chi}}^i_{k+1,\text{residual}} \right] \right\|. \tag{20}$$

Note that the linearization error e_{res} can be calculated without requiring the computed optimal arguments $\mathbf{A}^*$ and $\mathbf{b}^*$ in (19). The scalar residual error in (20) quantifies how well the temporal propagation of the sigma-points can be explained by a linear model and, therefore, how well the local linearity assumptions made by the SP transform hold for the mixand being propagated. This metric is a refinement of the metric proposed in [31] and similar to the metric proposed in [32]. Other accuracy metrics for the sigma-point transform have been developed, for example [33] proposes a metric based on the Taylor series expansion of the dynamics function, which requires a differentiable dynamics model. If the error metric is above some defined threshold ($e_{\text{res,max}}$), the mixand can be targeted for covariance reduction. However, this metric is scalar, and in order to effectively reduce the covariance of the mixand to a specified level of the linearization error, it is desirable to identify which sigma-points are explained the least by the optimal linear model.

Define $\boldsymbol{\mathcal{E}}^i_{k+1}$ as the difference between the sigma-points propagated through the system model and the optimal linear model:

$$\boldsymbol{\mathcal{E}}^i_{k+1} = \bar{\boldsymbol{\chi}}^i_{k+1} - \left(\mathbf{A}^* \bar{\boldsymbol{\chi}}^i_k + \mathbf{b}^* \right) = \left[\mathcal{E}^{i,1}_{k+1}, \ldots, \mathcal{E}^{i,2\cdot n_{\text{x}}+1}_{k+1} \right]. \tag{21}$$

The jth column of $\boldsymbol{\mathcal{E}}^i_{k+1}, \mathcal{E}^{i,j}_{k+1}$ is the residual linearization error associated with the jth sigma-point. Similar to the linearization error $e_{\text{res}}, \boldsymbol{\mathcal{E}}^i_{k+1}$ can be computed without computing the optimal linearization, which is given by

$$\boldsymbol{\mathcal{E}}^i_{k+1} = \left[\mathbf{0}, \hat{\boldsymbol{\chi}}^i_{k+1,\text{residual}} \right] \mathbf{Q}. \tag{22}$$

To effectively reduce the covariance of the mixand in order to reduce the linearization error, the direction along which the linearization error is greatest is identified as the optimal splitting axis (e_{split}). To find e_{split}, the sigma points before propagation ($\chi^{i,j}_k$) are weighted by the norm of their associated residual linearization errors ($\|\mathcal{E}^{i,j}_{k+1}\|$). The first eigenvector of the second moment of this set of weighted sigma points is the axis along which the residual linearization error is the greatest and is used as the optimal splitting axis.

C. Gaussian Mixand Splitting

Given the sigma point residual error of the propagated mixand in (22), the next step is to adapt the mixture to decrease this error (and increase the accuracy of the temporal propagation). This is accomplished by identifying mixands that are poorly propagated by the SP transform [using (22)] and replacing them with several mixands with smaller covariances.

The approximation of a GMM by a GMM with fewer mixands has been explored in the literature [34]–[38]) as classical GMM filters experience geometric growth in the number of mixands and require a mixture reduction step in order to remain computationally tractable. The problem here, however, is the reverse—splitting a mixand in order to reduce the variance of mixands. This problem is only beginning to receive attention in the literature [31], [32], [39], [40], although there has been interest in the topic in the machine learning community [41]–[43]. Other adaptive GMM filters include Caputi and Moose [44], which relies on non-Gaussian process noise to increase the number of mixands, and [45], which adapts the GMM to nonlinearities by developing an update rule for the mixand weights rather than by increasing the number of mixands.

There are two competing goals in approximating a single Gaussian by a GMM. First, the GMM must closely approximate the original Gaussian. Second, the covariance of the mixands in the GMM must be significantly smaller than the covariance of the original Gaussian. A common metric used to evaluate how closely two distributions approximate each other (e.g., comparing the GMM to the original Gaussian) is the Kullback Leibler divergence (KLD), which measures the relative entropy between two distributions [46]. However, the KLD between a Gaussian and a GMM cannot be evaluated analytically, and effective approximations can be resource intensive [47], [48]. An alternative metric is the integral-squared difference (ISD), which has a closed-form analytic solution when comparing a single Gaussian to a Gaussian mixture [49]. Because the ISD can be exactly and quickly evaluated, it is used here as the statistical error metric when comparing a Gaussian and its GMM [49].

$$J_{\text{ISD}} = \int_x \left(\mathcal{N}(x|\mu, \Sigma) - \hat{p}(x) \right)^2 dx. \tag{23}$$

Formally, the Gaussian mixand splitting problem is defined as the approximation of a single Gaussian mixand $\mathcal{N}(x|\mu, \Sigma)$ by a GMM $\hat{p}(x) = \sum_{i=1}^N w_i \cdot \mathcal{N}(x|\mu_i, \Sigma_i)$ such that the covariances of the mixands (Σ_i) are smaller than the original (Σ), and the ISD between the original Gaussian and its GMM (23) is minimized. The formal problem is to solve for $w_i, \mu_i,$ and Σ_i for all i such that

$$\begin{aligned} &[w_i, \mu_i, \Sigma_i]_{i=1\ldots N} \\ &\quad = \text{argmin} \left(\int_x \left(\mathcal{N}(x|\mu, \Sigma) - \sum_{i=1}^N w_i \cdot \mathcal{N}(x|\mu_i, \Sigma_i) \right)^2 dx \right). \end{aligned} \tag{24}$$

Because this optimization is computationally expensive, an offline optimal solution is computed and stored for use as needed by the anticipation algorithm in real time. Precomputing an optimal split for real time use requires that the precomputed split can be applied to an arbitrary single Gaussian $\mathcal{N}(x|\mu, \Sigma)$ and splitting axis $\mathbf{e}_{\text{split}}$ (i.e., the general problem). Any single Gaussian and splitting axis combination can be transformed to the problem of splitting a unit, zero-mean Gaussian along the x-axis ($\mathbf{e}_1$) using an affine transformation.

Given a Gaussian mixand $\mathcal{N}(x|\mu, \Sigma)$ and a splitting axis $\mathbf{e}_{\text{split}}$, the following transformations are applied in order to arrive at the precomputed problem:

$$\hat{x} = \mathbf{R}_{\text{x}} \mathbf{T}^{-1} (x - \mu)$$

where $\mathbf{T}$ is the matrix square-root of Σ

$$\Sigma = \mathbf{T}\mathbf{T}^T$$

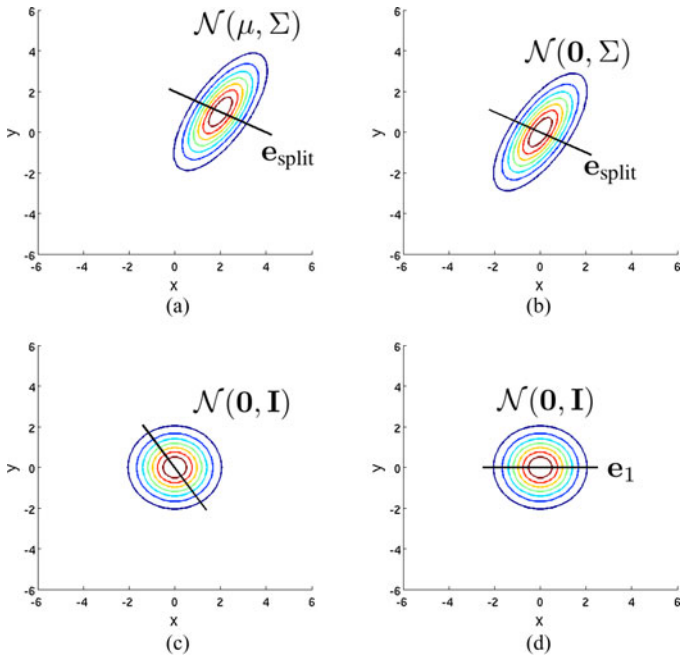

Fig. 2. Transformation from an arbitrary Gaussian/splitting axis to the offline problem. (a) Single Gaussian. (b) Translated to origin. (c) Transformed to unit variance. (d) Rotated splitting axis.

and $\mathbf{R}_{\text{x}}$ is a rotation matrix, computed such that the final splitting axis aligns with the x-axis:

$$\mathbf{e}_1 = \mathbf{R}_{\text{x}}\mathbf{T}^{-1}\mathbf{e}_{\text{split}}. \tag{25}$$

Fig. 2 illustrates the transformation in (25) applied to a general 2-D Gaussian. Fig. 2(a) shows the original single Gaussian to be split, and the axis along which the Gaussian is to be split e_{split}. Fig. 2(b) and (c) shows the translation of the Gaussian to the origin $(x - \mu)$ and the transformation that yields a unit covariance of the single Gaussian. Finally, Fig. 2(d) shows the rotation ($\mathbf{R}_{\text{x}}$) applied that aligns the splitting axis with the x-axis.

The splitting problem has now been posed as splitting a zero-mean, unit Gaussian along the x-axis into an N-component mixture of Gaussians:

$$\{(w_1^*, \mu_1^*, \Sigma_1^*), \ldots, (w_N^*, \mu_N^*, \Sigma_N^*)\} = \text{argmin}\left(\int_x \left(\mathcal{N}(x|\mathbf{0}, \mathbf{I}) - \hat{p}(x)\right)^2 dx\right). \tag{26}$$

The solution to this optimization problem is computed offline, and can be applied at runtime to the general problem (24) using the transformation from (25):

$$\begin{aligned} w_i &\approx w_i^* \\ \mu_i &\approx \mathbf{T}\mathbf{R}_{\text{x}}^T \cdot \mu_i^* + \mu \\ \Sigma_i &\approx \mathbf{T}\mathbf{R}_{\text{x}}^T \Sigma_i^* \mathbf{R}_{\text{x}}\mathbf{T}^T. \end{aligned} \tag{27}$$

The offline optimization (26) solves for a Gaussian mixture with an odd number (N) elements $\hat{p}(x) = \sum_{i=1}^{N} w_i \cdot \mathcal{N}(x|\mu_i, \Sigma_i)$ such that the ISD between the split Gaussian mixture and the Gaussian distribution $\mathcal{N}(x|\mathbf{0}, \mathbf{I})$ is minimized. For an n_x dimensional Gaussian, there are $N \cdot n_{\mathbf{x}}$ parameters associated with the means, $N \cdot \frac{1}{2}(n_{\mathbf{x}}^2 + n_{\mathbf{x}})$ parameters associated with the covariances, and N parameters associated with the weights. Due to the large parameter space, the optimization problem is ill-posed, and computationally intractable, even for offline optimization. To address these problems, several constraints are imposed to reduce the parameter space. First, the means μ_1 through μ_N are restricted to lie on the x-axis and be evenly spaced, reducing the optimization parameters associated with the means to a single spread parameter δ_μ:

$$\mu_i = \begin{bmatrix} \left(i - \dfrac{N-1}{2}\right)\delta_\mu \\ \mathbf{0} \end{bmatrix}. \tag{28}$$

This assumption is reasonable, as it spreads the means of split mixture along the splitting axis and only this dimension of the covariance is being targeted for reduction. Furthermore, the derivative of J_{ISD} with respect to the off-x-axis elements of the means evaluated at the proposed μ_i's is exactly zero.

The next parameter reduction constrains the covariance matrices Σ_1 through Σ_N to be diagonal, equal, and of the form

$$\Sigma_i = \begin{bmatrix} \sigma & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{29}$$

where

$$\sigma \in [0, 1]. \tag{30}$$

This targets only the covariance along the splitting axis; all other elements of the split covariance matrices σ are optimally 1 (the derivate of J_{ISD} with respect to these elements evaluated at the proposed value is zero); therefore, they are not considered. This step reduces the parameters associated with the variance matrices to a single parameter σ.

Finally, a further reduction to the parameter space is made by recognizing that the optimal weights (w_1 through w_N) for a given set of means (μ_1 through μ_N) and variances (Σ_1 through Σ_N) can be found using a quadratic program, removing the weights from the parameter search space entirely. Expanding the ISD, which is defined in (23), yields

$$\begin{aligned} J_{\text{ISD}} &= \int_x \mathcal{N}(x|\mathbf{0}, \mathbf{I})^2 - 2\mathcal{N}(x|\mathbf{0}, \mathbf{I})\hat{p}(x) + \hat{p}(x)^2 dx \\ &= \int_x \mathcal{N}(x|\mathbf{0}, \mathbf{I})^2 dx - 2\int_x \mathcal{N}(x|\mathbf{0}, \mathbf{I})\hat{p}(x) dx \\ &\quad + \int_x \hat{p}(x)^2 dx \\ &= J_{1,1} - 2 \cdot J_{1,2} + J_{2,2}. \end{aligned}$$

Williams provides a closed-form solution for each of the above terms [49]:

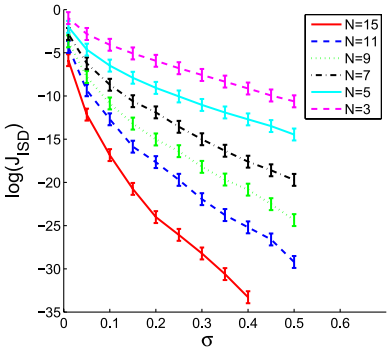

Fig. 3. J_{ISD} as a function of N and σ collected for 5000 randomly generated 2-D Gaussian distributions and splitting axes. The one-sigma error bars are shown.

$$J_{1,1} = \mathcal{N}(\mathbf{0}|\mathbf{0}, 2\mathbf{I})$$
$$J_{1,2} = \sum_j w_j \cdot \mathcal{N}(\mathbf{0}|\hat{\mu}_j, \mathbf{I} + \Sigma_j)$$
$$J_{2,2} = \sum_i \sum_j w_i w_j \mathcal{N}(\mu_i|\mu_j, \Sigma_i + \Sigma_j).$$

Rearranging the terms yields a quadratic program with the weights as arguments

$$J_{\mathrm{ISD}} = J_{1,1} - 2\mathbf{f}^T\mathbf{w} + \mathbf{w}^T\mathbf{H}\mathbf{w}$$

where

$$\mathbf{w} = [w_1, \ldots, w_N]^T$$
$$\mathbf{H}_{l,k} = \mathcal{N}(\mu_l|\mu_k, \Sigma_l + \Sigma_k)$$
$$f_l = \mathcal{N}(\mathbf{0}|\mu_l, \mathbf{I} + \Sigma_l)$$

such that

$$w_i \geq 0 \quad \forall i$$
$$\sum_i w_i = 1. \tag{31}$$

Equation (31) can be solved with a constrained quadratic program to find the optimal weights for the split w_i^*. This allows the weights to be removed from the parameter space as the optimal weights can be easily computed for each set of parameters considered.

The resulting parameter search space includes only the spacing of the means (δ_μ) and the reduction of the covariance (σ), making a high-resolution exhaustive parameter search possible. The optimization is an interesting trade between goals. First, J_{ISD} should be small as this represents the error introduced by approximating the original Gaussian by the split distribution. Second, σ should be small as this reduces the effects of nonlinearities in the propagation. Finally, N should be small in order to create manageable computation requirements.

Fig. 3 plots the mean and standard deviation of the ISD as a function of σ and N for 5000 randomly generated examples of 2-D Gaussians distributions and splitting axes. For each value of N, σ is varied between 0.01 and 0.5, and the optimal spread parameter and weights are computed. The optimized split is then applied to 5000 randomly generated 2-D Gaussians and splitting axes, and the ISD between each Gaussian and the resulting GMM after the split is computed. Note that the tight error bounds indicate that the optimized split consistently works well, even as Gaussian/splitting axis pairs vary, supporting the approximation in (27). Intuitively, $\sigma = 1$ minimizes the ISD, while $\sigma = 0$ mostly reduces the variance along the x-axis. It is proposed here to first choose a single value for σ and then compute the optimal values for the spread parameter δ_μ and the weights w_i for different values of N.

D. Benchmark Splitting Examples

To explore the impact of the parameter choices N and σ on the accuracy of propagation of a single Gaussian through a nonlinear temporal model, two nonlinear models are used as benchmark examples. The first is the univariate nonstationary growth model (UNGM) [50], [51]:

$$x_{k+1} = \alpha x_k + \beta \frac{x_k}{1 + x_k^2} + \gamma \cos 1.2k$$
$$\alpha = 0.3, \beta = 1, \gamma = 1 \tag{32}$$

and the second is a univariate cubic:

$$x_{k+1} = ax_k^3 + bx_k^2 + cx_k + d$$
$$a = 6, b = 1, c = 1, d = 1. \tag{33}$$

The UNGM is commonly used to evaluate the performance of nonlinear filters [50], [51]. In addition, for both the UNGM and the cubic model, the propagated distribution $p(x_{k+1})$ can be found analytically for comparison.

The accuracy of the GMM splitting routine is evaluated using 100 randomly generated samples; each is a normal distribution, with means uniformly distributed between -2 and 2, and variances uniformly distributed between 0 and 2 and propagated through the UNGM (32) and cubic (33) models using cached splits. In order to explore the sensitivities in the proposed anticipation algorithm, the cached splits are generated using different variance reductions (σ) and different numbers of mixands in the split (N). The GMM approximation to the propagated distribution $\hat{p}(x_{k+1})$ is compared with the exact propagated distribution $p(x_{k+1})$. The performance metric used to evaluate the accuracy of propagation is the numerically evaluated KLD between the propagated mixture and the true distribution, or

$$\mathrm{KLD} = \int_{x_{k+1}} \log\left(\frac{\hat{p}(x_{k+1})}{p(x_{k+1})}\right) \hat{p}(x_{k+1}) dx_{k+1}. \tag{34}$$

Fig. 4 plots the mean and standard deviation of the KLD between the true distribution $p(x_{k+1})$ and the GMM approximation $\hat{p}(x_{k+1})$ as a function of N and σ. The trends in each are similar. As σ decreases from 0.5, the accuracy of the split mixtures increases [smaller KLD compared with the true propagated distribution $p(x_{k+1})$]. As σ becomes very small, approaching zero, the split mixture becomes a poor approximation of the prior, introducing an error in the propagation (higher KLD values). As expected, larger values of N allow smaller values of σ and, therefore, more accurate mixture propagation. Even the

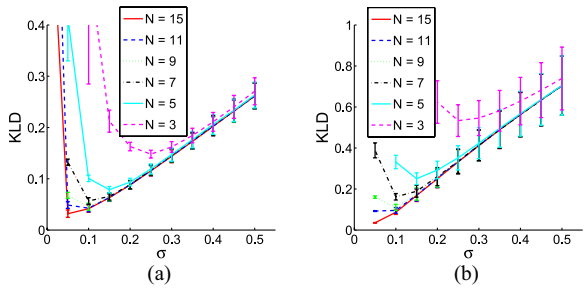

Fig. 4. Accuracy of propagation for the UNGM model (a) and cubic model (b). For comparison, the mean and variance of the computed KLDs between a propagation with no splitting and the truth are 0.5977 and 0.0630 (UNGM model) and 1.4918 and 0.4259 (cubic model).

least aggressive splits ($\sigma = 0.5$) result in KLDs of approximately one half that of propagation with no splitting, while more aggressive splits show KLDs of approximately one tenth that of the no-split solution.

These example problems can also be used to analyze how effective the linearity metric e_{res} (20) is at predicting propagation errors. e_{res} is stored for each single Gaussian propagated through the UNGM and the cubic function and the Pearson correlation between the residual linearization error, which is a metric that predicts the error in propagation, and the KLD between the no-split solution and truth, which is the actual measured propagation error, is computed. The Pearson correlation coefficient is 0.778 (UNGM) and 0.535 (cubic model), indicating strong correlation between the residual linearization error metric and the actual KLD from the true mixture after propagation.

These two example problems indicate that the linearity criteria and the mixand splitting algorithms greatly improve the accuracy of propagation of a single Gaussian through a nonlinear model. The choice of the linearity threshold ($e_{\mathrm{res,max}}$) and of the split parameters N and σ are problem dependent but can be explored offline prior to implementation.

IV. Experimental Results

Two systems, one simulated and one experimental, are used to analyze and evaluate the hGMM anticipation algorithm. In both systems, the hGMM is used to predict the future behavior of a tracked car driving on a known roadmap. Additionally, the case of the 2007 Cornell-MIT autonomous collision is studied anecdotally. In order to manage mixture complexity, the GMM reduction method proposed in [38] is used in the following examples.

A. Simulation

The simulation uses the hGMM to predict the future behavior of a simple simulated car driving in a Manhattan grid. Implementing in simulation allows the assumed dynamics model to exactly match the true dynamics model of the simulated obstacle vehicle. This allows the performance of the hGMM in predicting the distribution over future obstacle vehicle states to be studied independently of the assumed dynamics model.

The hGMM algorithm is implemented using a four-state obstacle vehicle model described in [31]. Obstacle vehicles are assumed to drive on a known road network and to obey specified traffic regulations. Obstacle vehicles are modeled as four-state bicycle robots, with the continuous state

$$\mathbf{x}_k^{\mathrm{C}} = \begin{bmatrix} x_k \\ y_k \\ v_k \\ \theta_k \end{bmatrix} \tag{35}$$

where x_k and y_k are the two-axis position of the center of the rear axle, v_k is the speed of the obstacle vehicle, and θ_k is the heading of the obstacle vehicle, all at time k. The continuous dynamics model for the obstacle vehicle is simple, though nonlinear.

$$\mathbf{x}_{k+1}^{\mathrm{C}} = f(\mathbf{x}_k^{\mathrm{C}}, \mathbf{u}_k) = \begin{bmatrix} x_k + \Delta t \cdot \cos(\theta_k) \cdot v_k \\ y_k + \Delta t \cdot \sin(\theta_k) \cdot v_k \\ v_k + \Delta t \cdot (u_{1,k} + \mathbf{v}_{1,k}) \\ \theta_k + \Delta t \cdot l \cdot v_k \cdot (u_{2,k} + \mathbf{v}_{2,k}) \end{bmatrix}. \tag{36}$$

Control inputs are throttle ($u_{1,k}$) and steering angle ($u_{2,k}$), and zero-mean, Gaussian process noise ($\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$) enters on the control inputs. The time step is $\Delta t = 0.1$ s in this example.

The hybrid aspect of the obstacle vehicle anticipation problem enters through an assumed route planning controller, where the discrete states are road segments in the map. The discrete dynamics function assumes that the route planning controller takes all possible routes with equal probability. A path following controller is assumed, which generates control inputs $u_{1,k}$ and $u_{2,k}$ that enables the obstacle vehicle to follow the current route and maintain a speed of 10 m/s:

$$\mathbf{u}_k = \begin{bmatrix} u_{1,k} \\ u_{2,k} \end{bmatrix} = h(\mathbf{x}_k^{\mathrm{C}}, \mathbf{x}_k^{\mathrm{D}}). \tag{37}$$

This path following controller (37) is composed with the bicycle dynamics (36) to arrive at the continuous dynamics function f^{C} (2) required by the hGMM:

$$\mathbf{x}_{k+1}^{\mathrm{C}} = f^{\mathrm{C}}(\mathbf{x}_{k+1}^{\mathrm{D}}, \mathbf{x}_k^{\mathrm{C}}, \mathbf{v}_k) = f(\mathbf{x}_k^{\mathrm{C}}, h(\mathbf{x}_k^{\mathrm{C}}, \mathbf{x}_k^{\mathrm{D}})). \tag{38}$$

The anticipation algorithm is used with a time horizon of 3.5 s (approximately the time required for an obstacle to fully traverse an intersection) to predict the behavior of an obstacle car in several different scenarios. For comparison to the hGMM, a large particle set is used to approximate the true distribution of the obstacle state at future times. Three example scenarios (straight road, turn, and intersection) are used to compare the output of the hGMM algorithm with the approximate true distribution. The negative log-likelihood (NLL) is used to evaluate the difference between the true distribution, as represented by a particle set, and the hGMM approximation. The performance of the hGMM is evaluated for different values of $e_{\mathrm{res,max}}$, and compared with the baseline of the hGMM algorithm with no splitting, which is equivalent to a single Gaussian UKF anticipation algorithm.

Fig. 5 shows the NLL metric for the three considered scenarios. The NLL is plotted as a function of lookahead time. Results are compared for four values of $e_{\mathrm{res,max}}$ for each scenario: the

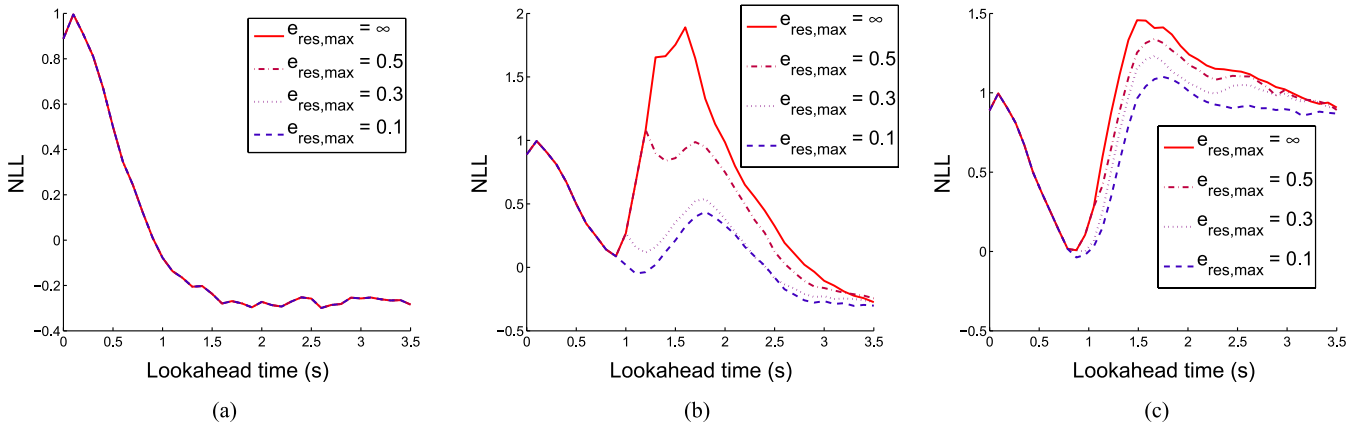

Fig. 5. NLL between "truth" particle set and computed probability distributions plotted against the lookahead time (up to a 3.5 s anticipation horizon) for an obstacle vehicle on (a) a straight road, (b) approaching a turn, and (c) approaching an intersection.

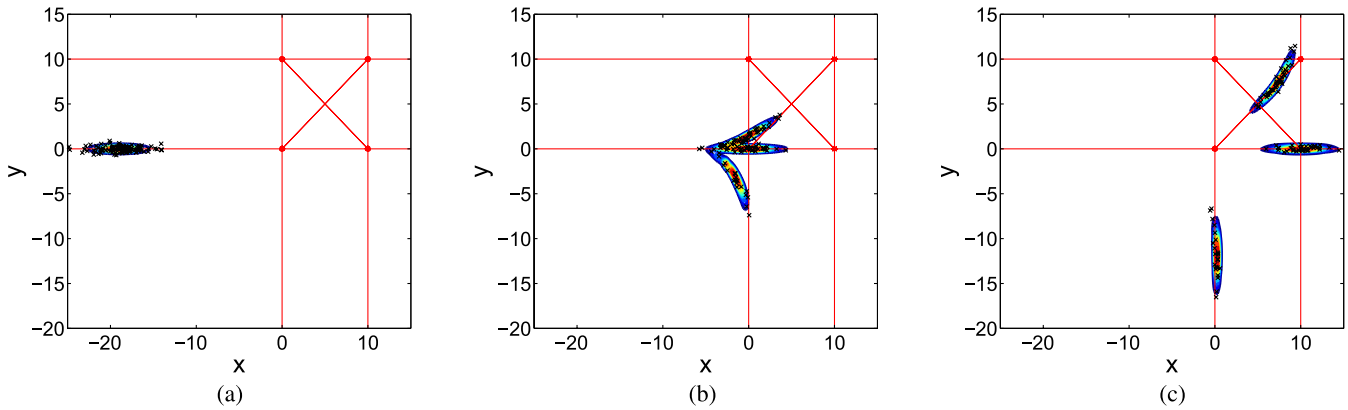

Fig. 6. Anticipated distribution $p(x, y)$ (position of the rear axle) as obstacle traverses an intersection. The time at which the vehicle is observed is t, and the time into future the obstacle is anticipated is t_{LA}. The particle set used as truth is overlaid as black dots. (a) $t_{\mathrm{LA}} = 0.1$ s. (b) $t_{\mathrm{LA}} = 2$ s. (c) $t_{\mathrm{LA}} = 3$ s.

hGMM with $e_{\mathrm{res,max}} = 0.1$, the hGMM with $e_{\mathrm{res,max}} = 0.3$, the hGMM with $e_{\mathrm{res,max}} = 0.5$, and, finally, the hGMM with $e_{\mathrm{res,max}} = \inf$; the latter is equivalent to a single Gaussian UKF predictor. When the obstacle vehicle is traveling on a straight road [see Fig. 5(a)], all of the anticipation predictors show similar performance, because the dynamics model is very close to linear in this case. In Fig. 5(b) and (c), the hGMM anticipation algorithm shows significant performance improvement over the no splitting predictor as nonlinearities have greater impact on the accuracy of propagation. The peaks in NLL that appear in the second and third scenario correspond to the future time when the vehicle is anticipated to be negotiating the bend on the road when the model is exhibiting a high degree of nonlinear behavior. Once the turn is negotiated and the vehicle is anticipated to be on the straight road after the turn, the nonlinearities fade, and the NLL decreases again.

Fig. 6 shows the hGMM anticipated distributions of the rear-axle position for the intersection scenario for $e_{\mathrm{res,max}} = 0.1$. The hGMM anticipation algorithm clearly captures the non-Gaussian nature of the probability masses that turn right and left. This example illustrates both the splitting due to the discrete dynamics (seen by separate probability masses taking each of the three options available at the intersection) and the splitting due to nonlinearities (seen by the non-Gaussian shape of the probability masses making the left and right turns). This figure also demonstrates why Fig. 5(c) has a peak just before 2 s lookahead—the non-Gaussian shape of the individual probability masses is at its highest as the obstacle vehicle is anticipated to be negotiating the intersection, as in Fig. 6(b).

The four-state obstacle vehicle simulation example is also used to compare the hGMM with Monte-Carlo (MC) propagation (particle filtering). MC methods are attractive because they inherently capture nonlinearities and multimodal behavior. However, the number of particles required to capture state propagation can be intractably large, especially for higher dimensional problems, and it can be difficult to work with the discrete distributions produced by such methods.

To compare MC propagation with the hGMM, the true distribution is computed by using a very large (6 million particles) particle set. The performance of an algorithm is defined by the ISD between the distribution produced by the algorithm and the true distribution, computed using a 4-D grid. MC methods using between 500 and 500 000 particles are used to propagate the distribution over the obstacle vehicle state forward in time, with multiple trials for each particle set size. Fig. 7 plots the result of the study. The trend is exactly as expected – increasing the number of particles results in more accurate capture of the true propagated distribution. The hGMM performance is shown

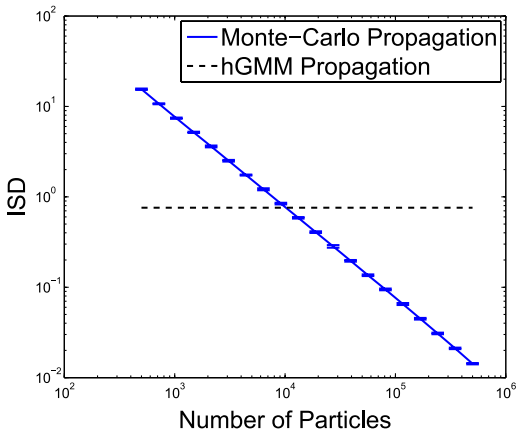

Fig. 7. Perfomance of Monte-Carlo propagation of four-state obstacle vehicle state distribution.

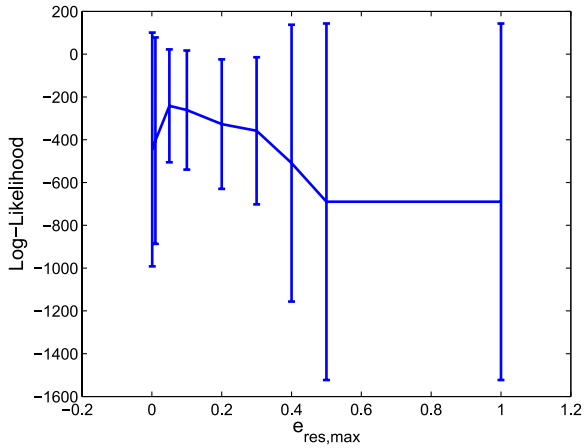

Fig. 8. LL and standard deviation of observed vehicles given by hGMM anticipation algorithm as a function of $e_{\mathrm{res,max}}$.

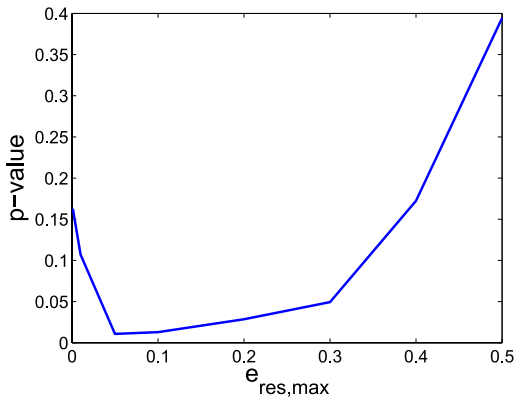

Fig. 9. *t*-test results comparing LL data at different linearity thresholds $e_{\mathrm{res,max}}$ to the data for $e_{\mathrm{res,max}} = 1$.

for comparison. To match the accuracy of the hGMM, approximately 10 000 particles are required. In MATLAB, the hGMM propagation takes 1.23 s, while MC propagation for a 10 000 particle set takes 356 s. The hGMM enjoys an enormous computational advantage over MC methods for this problem, and this advantage enables a real-time implementation of the hGMM, discussed in Section IV-C.

B. Experimental Data

In order to validate the probabilistic anticipation approach on more realistic problems, the hGMM algorithm was evaluated on a set of tracked vehicle data near intersections. The dataset used for this validation is the 2007 Columbus large image format (CLIF 2007) dataset made available by the United States Air Force [52]. The dataset consists of aerial imagery of an urban environment, at approximately two frames per second, collected by a large-format electronics observation platform. A large number of vehicles are observed driving in a variety of conditions. The validation focuses on vehicles observed traversing intersections, in relatively light traffic, that have the right-of-way. A total of 40 vehicle tracks at three different intersections are used.

In the proposed validation approach, an observed vehicle is tracked at time t, giving a state estimate. This state is then predicted forward in time using the hGMM with various values for $e_{\mathrm{res,max}}$, and an anticipation horizon equal to the time over which the vehicle is tracked. The estimate of the observed vehicle state at time t is used as the initial condition for the hGMM; subsequent measurements are used to evaluate how well the hGMM predicts the behavior of the vehicle. The same vehicle behavior model described in Section IV-A [see (36)] is used in this experimental validation to describe the dynamics of observed vehicles. Successive measurements of the tracked obstacle at times after t are compared with the anticipated distribution over the vehicle state to evaluate how well the hGMM algorithm anticipates the behavior of the tracked vehicle.

Fig. 8 plots the log-likelihood (LL) of the hGMM anticipation agreeing with CLIF observations as a function of $e_{\mathrm{res,max}}$. At high values of $e_{\mathrm{res,max}}$, the LL is low with larger uncertainty bounds, indicating poorer predictive performance. The figure shows that smaller values for $e_{\mathrm{res,max}}$ provide more accurate anticipation of observed cars, as well as much more consistent performance. At values for $e_{\mathrm{res,max}}$ above approximately 0.5, the standard deviation becomes quite large. This indicates that the covariance of mixands in the hGMM becomes large enough that the sigma-point propagation becomes a poor and unpredictable approximation. For very small values of $e_{\mathrm{res,max}}$, performance also degrades due to errors introduced by repeated mixand splitting in the hGMM.

Fig. 9 plots the p-values of *t*-test comparisons between the LL data for each linearity threshold $e_{\mathrm{res,max}}$ to the LL data for the maximum linearity threshold ($e_{\mathrm{res,max}} = 1$). This figure shows that the improved performance seen in Fig. 8 for linearity thresholds $e_{\mathrm{res,max}} = [0.05, 0.1, 0.2]$ is statistically significant as the corresponding p-values are below 0.05. For linearity thresholds below this range, the accumulated errors introduced by repeated mixand splitting degrade performance, and the LL performance is not statistically different from the data taken with a linearity threshold high enough that no splitting occurs ($e_{\mathrm{res,max}} = 1$).

The LL metric studied in Figs. 8 and 9 considers only the likelihood that the observations agree with the anticipated obstacle state and, therefore, only evaluates the hGMM at the observa-

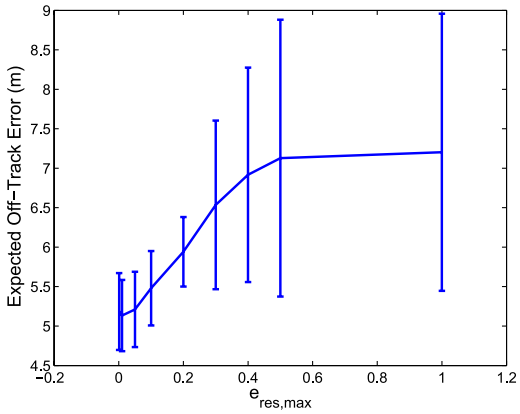

Fig. 10. EOTE and standard deviation of observed vehicles given by hGMM anticipation algorithm as a function of the linearity threshold $e_{\mathrm{res,max}}$.

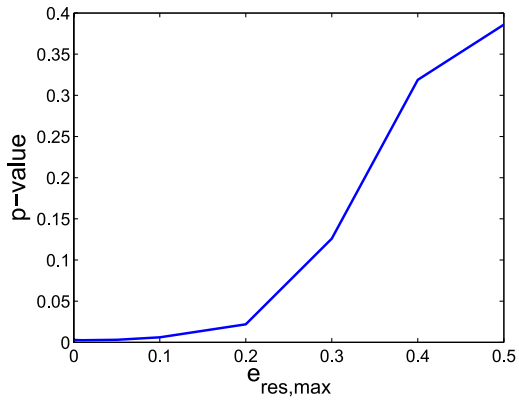

Fig. 11. *t*-test results comparing EOTE data at different linearity thresholds $e_{\mathrm{res,max}}$ to the data for $e_{\mathrm{res,max}} = 1$.

tions. Because of this, unreasonable predictions (i.e., the car being very far outside the driving lane) are not penalized by this metric. To compliment this metric, a second metric is proposed: the expected off-track error (EOTE). The EOTE is defined as

$$\mathrm{EOTE} = \sum_{k=1}^{K} \int_{x_k} \int_{y_k} d(x_k, y_k) p(x_k, y_k) dy_k dx_k \tag{39}$$

where K is the anticipation horizon, $d(x_k, y_k)$ is the distance of the rear axle position (x_k, y_k) from the lane center, and $\hat{p}(x_k, y_k)$ is the anticipated probability of the vehicle being at that position from the hGMM. This metric is equivalent to the expected value of the out-of-lane position of the observed vehicle. This metric directly measures how reasonable the output of the hGMM is, assuming the observed vehicle is not behaving anomalously.

Fig. 10 plots the EOTE as a function of the linearity threshold $e_{\mathrm{res,max}}$ used in the hGMM. At high values of $e_{\mathrm{res,max}}$, the EOTE is large and has a large standard deviation, indicating poor anticipation performance. For small values of $e_{\mathrm{res,max}}$, the EOTE and its standard deviation decrease, indicating that the hGMM predictions are more reasonable and more consistent. As with Fig. 8, for $e_{\mathrm{res,max}}$ above 0.5 the standard deviation of this metric becomes large as the mixand covariances become too large for the sigma-point approximation.

Fig. 11 plots the results of *t*-test comparisons between the EOTE data for linearity threshold to the EOTE data for the maximum linearity threshold. These results show that the decreased EOTE seen in Fig. 10 for small linearity thresholds ($e_{\mathrm{res,max}} \leq 0.2$) is statistically significant.

Fig. 12 shows the results of the hGMM and single Gaussian anticipation algorithms, respectively, for a vehicle making a turn at an intersection. The initial observation of the vehicle (green circle) is plotted along with the anticipated distribution of the rear-axle position at future anticipated times. The actual measurements are plotted as green crosses. Comparing the two approaches, the hGMM predicts the measurements about as well as the single Gaussian method, according to the LL metric in Fig. 8. However, Fig. 12(c) and (f) show that the hGMM distribution is much more reasonable than the single Gaussian method distribution in predicting locations that are actually on the road network, particularly as the look ahead time increases. This is demonstrated by the large amounts of probability mass outside of the actual expected driving corridor for the vehicle in Fig. 12(f) and is quantified by the EOTE metric, which clearly shows superior performance of the hGMM algorithm.

C. MIT-Cornell Collision Example

A motivating example for the importance of this study is the collision between the Cornell and MIT entries in the 2007 DARPA Urban Challenge (DUC). The collision occurred when the Cornell entry (Skynet) stopped due to a perceived blockage that was the result of a misplaced waypoint in the roadmap. MIT's entry (Talos), observing that Skynet had stopped, initiated a pass. While Talos was executing the pass, Skynet recovered and began moving again. Talos, not recognizing that Skynet was no longer a static obstacle, moved back into the driving lane while Skynet, not able to anticipate the behavior of Talos, continued forward. The result was the low-speed collision shown in Fig. 13. Further details on the collision are available in [12].

To demonstrate the efficacy of the hGMM anticipation algorithm at improving safety, the scenario is revisited using logged data from the 2007 collision. In this example, Skynet uses the hGMM to anticipate the behavior of Talos (using the simple vehicle model described in Section IV-A). At the same time, Skynet simulates its own trajectory for the proposed behavior of resuming motion. The probability of collision between Skynet's proposed motion and the hGMM over Talos' anticipated state is evaluated using the approximation developed in [13] over a horizon of 3 s at 10 Hz.

Fig. 14 shows the results of the anticipation as a function of the lookahead time. Fig. 14(a) shows the initial condition—Talos is shown as the black car; Cornell is shown as the green car. Fig. 14(b)–(d) shows the scenario as it evolves at three different lookahead times. The hGMM probability distribution prediction of Talos is plotted—in this case, the *xy* position of the rear axle. Skynet's proposed position at the given lookahead time is also shown, and the color corresponds to the probability of collision, with green being safe (probability of collision of zero) and red being dangerous (probability of collision of one). Fig. 15 plots

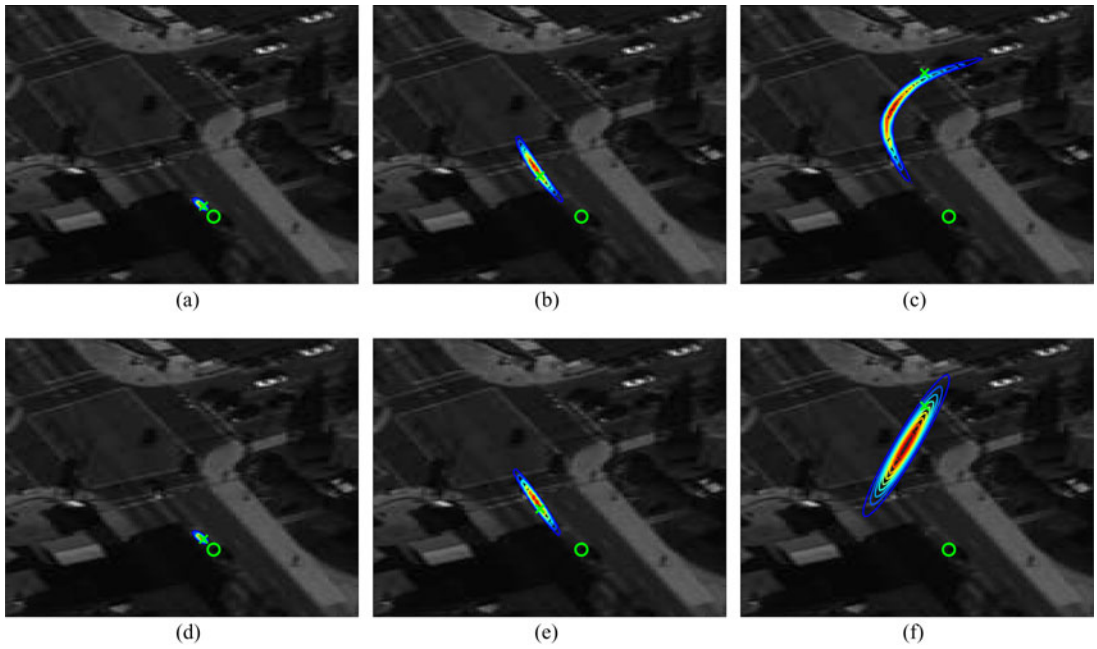

Fig. 12. Anticipation of a real tracked vehicle using the hGMM [(a)–(c)] and a single Gaussian predictor [(d)–(f)]. The vehicle state at time t is shown as a green circle. The observed vehicle state at the lookahead time is shown as a green cross. (a) $t_{\mathrm{LA}} = 1$ s. (b) $t_{\mathrm{LA}} = 3$ s. (c) $t_{\mathrm{LA}} = 7$ s. (d) $t_{\mathrm{LA}} = 1$ s. (e) $t_{\mathrm{LA}} = 3$ s. (f) $t_{\mathrm{LA}} = 7$ s.

Fig. 13. MIT-Cornell collision in the 2007 DUC.

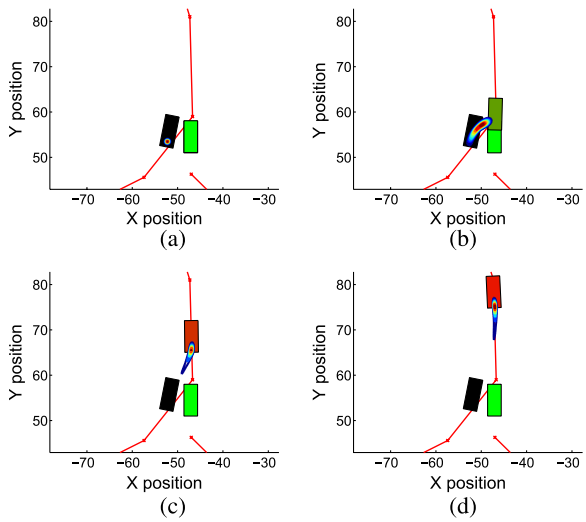

Fig. 14. hGMM anticipation algorithm applied to the MIT-Cornell Collision. (a) $t_{\mathrm{LA}} = 0.1$ s. (b) $t_{\mathrm{LA}} = 1.0$ s. (c) $t_{\mathrm{LA}} = 2.0$ s. (d) $t_{\mathrm{LA}} = 3.0$ s.

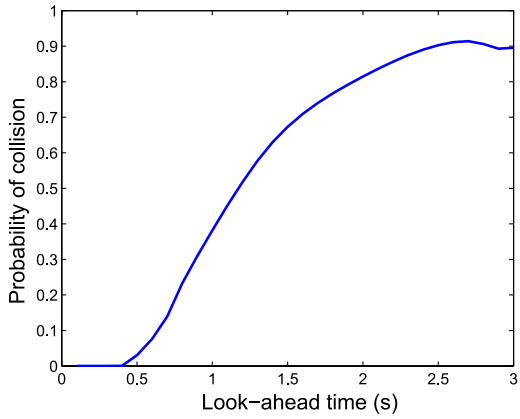

Fig. 15. Anticipated probability of collision.

the anticipated probability of collision as a function of lookahead time. The anticipated probability of collision increases as the anticipation algorithm looks ahead in time, and a collision is almost guaranteed by 2.5 s. These figures clearly show that even with a basic obstacle model, the hGMM could have predicted, and therefore prevented, the MIT-Cornell collision.

Fig. 16 plots the time required to predict an obstacle's state forward 4.5 s for a range of values for both the linearity threshold and the maximum number of mixands allowed at the end of each prediction step. The hGMM is implemented in C# in Skynet's planning algorithms on a modern, Intel Core i7 rack-mount computer. The computation time grows with the number of allowed mixands and the inverse of the linearity threshold, as expected.

Fig. 17 plots the prediction accuracy of the hGMM as a function of the linearity threshold and the maximum number of

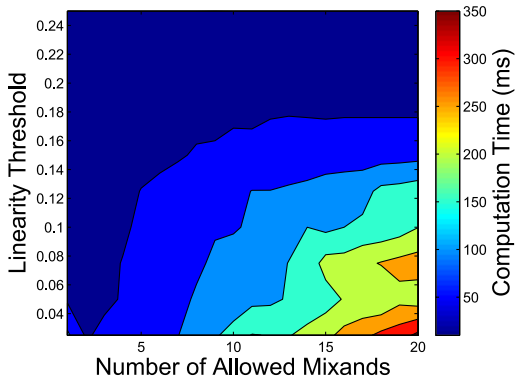

Fig. 16. Computation time required to anticipate an obstacle to a 4.5 s horizon as a function of linearity threshold and maximum number of allowed mixands (enforced by Runnall's reduction algorithm) [38].

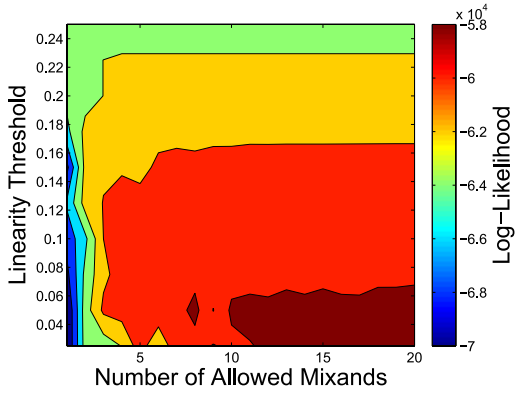

Fig. 17. hGMM prediction accuracy as a function of linearity threshold and maximum number of allowed mixands.

allowed mixands. Here, the prediction accuracy is measured by computing the LL between the hGMM result and a large particle set propagated using the same vehicle dynamics model as the hGMM. Fig. 17 shows that prediction accuracy improves as the linearity threshold becomes smaller and degrades as the number of allowed mixands becomes small. Note that the performance is independent of the number of allowed mixands until the number of mixands becomes very small. Comparing Figs. 16 and 17 provides insight on how to select the linearity threshold and the number of allowed mixands for the assumed dynamics model.

The hGMM is currently implemented in Skynet's planning algorithms with a linearity threshold of 0.1 and a maximum number of allowed mixands of 10. This implementation allows Skynet to anticipate the behavior of up to three obstacle vehicles to a horizon of 4.5 s at greater than 3 Hz. Because the anticipation result is never older than 1/3 s, but extends to 4.5 s in the future, this is considered real-time performance.

V. Conclusion

An anticipation algorithm is developed which uniquely recognizes and mitigates the impact of nonlinearities in the hybrid dynamics function on the accuracy of probability distribution propagation. A unique method for evaluating the accuracy of propagation of a Gaussian distribution through nonlinear dynamics is proposed, using the propagated sigma-points from the distribution. In addition, a new method for splitting propagated Gaussian mixands due to nonlinearities is developed. By detecting, and reacting to, propagation errors introduced by nonlinearities, the proposed algorithm is shown to have significant improvements in accuracy over standard propagation methods, such as the Unscented Transform.

The behavior of the nonlinearity detection and the mixand splitting algorithms are explored using several simulated example problems and compared with a single Gaussian sigma point method. In the case of benchmark nonlinear problems, the hGMM anticipation algorithm is shown to better approximate the true distributions that arise than the single Gaussian sigma point method. The second set of simulations use an obstacle vehicle driving on a known road network. The accuracy of the anticipated probability distributions is compared across choices for the linearity threshold parameter in the hGMM anticipation algorithm and compared with a single Gaussian sigma point predictor. Using a large particle set as truth, the hGMM is shown to more accurately capture the behavior of the distribution over future obstacle vehicle states, which is defined as the NLL between the particle set and the anticipated distribution.

The hGMM anticipation algorithm is validated on an experimental dataset. Specifically, the hGMM algorithm is used to predict the behavior of vehicles in the CLIF 2007 dataset provided by the USAF. The hGMM is compared with a single Gaussian sigma point method by comparing predictions of a tracked vehicle state to observations of the tracked vehicle. The hGMM is shown to provide increased accuracy over the single Gaussian sigma point method using the LL as a metric. The hGMM is also shown to provide predictions that are more reasonable (i.e., that do not include predictions of anomalous behavior for vehicles that are not behaving anomalously by the EOTE metric) than propagation with no splitting.

Additionally, the hGMM is applied to the scenario leading to the MIT-Cornell collision in the 2007 DUC, and it is shown that the hGMM could have anticipated the collision in time to prevent it.

References

[1] T. Folsom, "Social ramifications of autonomous urban land vehicles," presented at the IEEE Int. Symp.Technol. Society, Chicago, IL, USA, May 2011.
[2] R. Bishop, "Intelligent vehicle applications worldwide," *IEEE Intell. Syst. Appl.*, vol. 15, no. 1, pp. 78–81, Jan./Feb. 2000.
[3] Z. Juan, J. Wu, and M. McDonald, "Socio-economic impact assessment of intelligent transport systems," *Tsinghua Sci. Technol.*, vol. 11, no. 3, pp. 339–350, 2006.
[4] I. Miller *et al.*, "Team cornell's skynet: Robust perception and planning in an urban environment," *J. Field Robot.*, vol. 25, no. 8, pp. 493–527, 2008.
[5] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.
[6] A. Bacha *et al.*, "Odin: Team victortango's entry in the darpa urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 467–492, 2008.
[7] J. Markoff, "Collision in the making between self-driving cars and how the world works," Jan. 2012.
[8] J. Levinson *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Proc. IEEE Intell. Veh. Symp.*, 2011, pp. 163–168.
[9] A. Labs, "Autonomous car navigates the streets of Berlin."
[10] J. Markoff, "Google cars drive themselves, in traffic," *New York Times*, Oct. 2010.

[11] I. Huttenlocher, F. Sergei, L. Pete, M. Mike, and K. Fujishima, "Team Cornell's Skynet: Robust perception and planning in an urban environment," *J. Field Robot.*, vol. 25, no. 8, pp. 493–527, 2008.
[12] L. Fletcher *et al.*, "The MIT-Cornell collision and why it happened," *J. Field Robot.*, vol. 25, no. 10, pp. 775–807, 2008.
[13] J. Hardy and M. Campbell, "Contingency planning over probabilistic hybrid obstacle predictions for autonomous road vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 2237–2242.
[14] B. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. Bagnell, M. Hebert, A. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 3931–3936.
[15] S. Petti, T. Fraichard, I. Rocquencourt, and E. D. M. D. Paris, "Safe motion planning in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2005, pp. 3726–3731.
[16] J. Choi, G. Eoh, J. Kim, Y. Yoon, J. Park, and B. Lee, "Analytic collision anticipation technology considering agents' future behavior," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 1656–1661.
[17] T. Ohki, K. Nagatani, and K. Yoshida, "Collision avoidance method for mobile robot considering motion and personal spaces of evacuees," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Conf.*, 2010, pp. 1819–1824.
[18] J. Miura and Y. Shirai, "Probabilistic uncertainty modeling of obstacle motion for robot motion planning," *J. Robot. Mechatron.*, vol. 14, no. 4, pp. 349–356, 2002.
[19] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Probabilistic rapidly-exploring random trees for autonomous navigation among moving obstacles," in *Proc. IEEE Int. Conf. Robot. Autom. Workshop Safe Navigat.*, 2009.
[20] N. Du Toit and J. Burdick, "Robotic motion planning in dynamic, cluttered, uncertain environments," in *Proc. IEEE Conf. Robot. Autom.*, 2010, pp. 966–973.
[21] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 101–115, Feb. 2012.
[22] D. Ferguson, M. Darms, C. Urmson, and S. Kolski, "Detection, prediction, and avoidance of dynamic obstacles in urban environments," in *Proc. IEEE Intell. Veh. Symp.*, 2008, pp. 1149–1154.
[23] D. Alspach and H. Sorenson, "Nonlinear bayesian estimation using gaussian sum approximations," *IEEE Trans. Automat. Control*, vol. 17-AC, no. 4, pp. 439–448, Aug. 1972.
[24] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proc. Int. Symp. Aerospace/Defense Sensing, Simul. Controls*, 1997, vol. 3, p. 26.
[25] J. Uhlmann, "Dynamic map building and localization: New theoretical foundations" Ph.D. dissertation, Univ. Oxford, Oxford, U.K., 1995.
[26] S. Julier and J. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," Robotics Res. Group, Dep. Eng. Sci., Univ. Oxford, Oxford, U.K., Tech. Rep., 1996.
[27] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proc. IEEE Adapt. Syst. Signal Process., Commun., Contr. Symp.*, 2000, pp. 153–158.
[28] D. Musicki and R. J. Evans, "Measurement gaussian sum mixture target tracking," in *Proc. 9th Int. Conf. Inf. Fusion,*, 2006, pp. 1–8.
[29] M. Šimandl, J. Dunık, "Sigma point Gaussian sum filter design using square root unscented filters," in *Proc. 16th IFAC World Congress*, Preprints, Prague, Czech Republic, 2005.
[30] H. Sorenson and D. Alspach, "Recursive Bayesian estimation using Gaussian sums," *Automatica*, vol. 7, no. 4, pp. 465–479, 1971.
[31] F. Havlak and M. Campbell, "Discrete and continuous, probabilistic anticipation for autonomous robots in urban environments," *Proc. SPIE*, vol. 7833, p. 078330H, 2010.
[32] M. Huber, "Adaptive Gaussian mixture filter based on statistical linearization," 2011.
[33] R. Van Der Merwe, "Sigma-point kalman filters for probabilistic inference in dynamic state-space models" Ph.D. dissertation, Univ. Stellenbosch, Stellenbosch, South Africa, 2004.
[34] M. Huber and U. Hanebeck, "Progressive Gaussian mixture reduction," in *Proc. IEEE 11th Int. Conf. Inf. Fusion*, 2008, pp. 1–8.
[35] D. Salmond, "Mixture reduction algorithms for point and extended object tracking in clutter," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 45, no. 2, pp. 667–686, Apr. 2009.
[36] V. Garcia, F. Nielsen, and R. Nock, "Levels of details for gaussian mixture models," in *Proc. Asian Conf. Comput. Vis.*, 2010, pp. 514–525.
[37] C. Hennig, "Methods for merging gaussian mixture components," *Adv. Data Anal. Classif.*, vol. 4, no. 1, pp. 3–34, 2010.
[38] A. R. Runnalls, "Kullback-Leibler approach to gaussian mixture reduction," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 3, pp. 989–999, Jul. 2007.
[39] S. Ali-Löytty and N. Sirola, "Gaussian mixture filter in hybrid navigation," in *Proc. Eur. Navigat. Conf. Global Navigat. Satellite Syst.*, 2007, pp. 831–837.
[40] M. L. Psiaki, J. R. Schoenberg, and I. T. Miller, "Gaussian mixture approximation by another Gaussian mixture for "blob" filter re-sampling," in *Proc. AIAA Guidance, Navigation, Control Conf.*, 2010, pp. 2–5, AIAA Paper, nos. 2010–7747.
[41] N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton, "SMEM algorithm for mixture models," *Neural Comput.*, vol. 12, p. 2109, 1999.
[42] F. Faubel, J. McDonough, and D. Klakow, "The split and merge unscented gaussian mixture filter," *IEEE Signal Process. Lett.*, vol. 16, no. 9, pp. 786–789, Sep. 2009.
[43] Z. Zhang, C. Chen, J. Sun, and K. L. Chan, "Em algorithms for gaussian mixtures with split-and-merge operation," *Pattern Recog*, vol. 36, pp. 1973–1983, 2003.
[44] M. J. Caputi and R. L. Moose, "A modified gaussian sum approach to estimation of non-gaussian signals," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 29, no. 2, pp. 446–451, Apr. 1993.
[45] G. Terejanu, P. Singla, T. Singh, and P. D. Scott, "Uncertainty propagation for nonlinear dynamic systems using gaussian mixture models," *J. Guid., Control, Dyn.*, vol. 31, no. 6, p. 1623, 2008.
[46] S. Kullback, *Information Theory and Statistics*. New York, NY, USA: Dover, 1968.
[47] J. Goldberger and H. Aronowitz, "A distance measure between GMMS based on the unscented transform and its application to speaker recognition," in *Proc. 9th Eur. Conf. Speech Commun. Technol.*, 2005.
[48] J. Hershey and P. Olsen, "Approximating the Kullback Leibler divergence between Gaussian mixture models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2007, vol. 4.
[49] J. Williams and P. Maybeck, "Cost-function-based gaussian mixture reduction for target tracking," in *Proc. 6th Int. Conf. Inform. Fusion*, 2003, vol. 2, pp. 1047–1054.
[50] J. Kotecha and P. Djuric, "Gaussian sum particle filtering," *IEEE Trans. Signal Process.*, vol. 51, no. 10, pp. 2602–2612, Oct. 2003.
[51] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for Bayesian filtering," *Statist. Comput.*, vol. 10, no. 3, pp. 197–208, 2000.
[52] U. S. Air Force. (2007). "2007 clif data set," [Online]. Available: https://www.sdms.afrl.af.mil/index.php?collection=clif2007

Authors' photographs and biographies not available at the time of publication.