

PROBABILISTIC ANTICIPATION FOR AUTONOMOUS URBAN ROBOTS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Frank Havlak

May 2015

© 2015 Frank Havlak
ALL RIGHTS RESERVED

PROBABILISTIC ANTICIPATION FOR AUTONOMOUS URBAN ROBOTS

Frank Havlak, Ph.D.

Cornell University 2015

The ability to anticipate the behavior of other vehicles on the road is a key part of how humans drive safely in complex environments. This thesis presents work enabling robotic systems to also anticipate the behavior of vehicles in the environment. The Hybrid Gaussian Mixture Model anticipation algorithm is presented, and enables the state of a dynamic system, such as a tracked vehicle, to be accurately predicted over useful time horizons, by using Gaussian Mixture Models to represent the state uncertainty, and adapting the Gaussian Mixture Models on the fly to any nonlinearities in the model of the dynamic system. Results show high accuracy predictions of a tracked vehicle state can be made in real time. The model used to anticipate the behavior of a vehicle in the environment must include both the vehicle dynamics and the driver behavior, so the Gaussian Process adaptive Gaussian Mixture Model (GP-aGMM) algorithm is presented, using Gaussian Processes to model human drivers and anticipate their behavior. Presented results show that the GP-aGMM can effectively anticipate the behavior of drivers even in complex situations. Finally, the lane-feature Gaussian Process Anticipation (LFGPG) algorithm is presented. The LFGPA algorithm is similar to the GP-aGMM, but abstracts the training data into a feature space that captures the relationship between the driver and the road, locally. This allows training data from one set of roads to be relevant to any roads. The power of the LFGPA algorithm to reduce the requirements for training data is demonstrated in presented results.

BIOGRAPHICAL SKETCH

Frank Havlak received his M.S. in Mechanical Engineering from Cornell University in 2012, and his B.S. in Mechanical Engineering from Rice University in 2008. He received an Honorable Mention from the NSF Graduate Fellowship Research Program in 2011. His research interests include probabilistic anticipation of human-driven road vehicles to enable safer autonomous driving, modeling human drivers, and Bayesian estimation.

ACKNOWLEDGEMENTS

- Professor Mark Campbell, for his support and guidance through the years.
- Jason Hardy, for his collaboration on parts of this research
- Benjamin Johnson, for his collaboration on other parts of this research
- Professor Campbell's research group, especially Jon Schoenberg, Mark McClelland, Nisar Ahmed, and Kevin Wyffles for their input and help throughout the years.
- Ryan Dougherty, Ryan Mitch, Mark McClelland, Micheal Bono, Matt Leineweber, Alex Moore, Juan Gomez, and everyone else who was willing to subject themselves to our experiments

TABLE OF CONTENTS

Biographical Sketch	iii
Acknowledgements	iv
Table of Contents	v
List of Tables	vii
List of Figures	viii
Introduction	1
1 Probabilistic Anticipation of Road Vehicles	4
1.1 Introduction	4
1.2 Hybrid Mixture Anticipation Algorithm	7
1.2.1 Obstacle State Model	7
1.2.2 Hybrid Mixture Probability Distribution Representation	8
1.2.3 Hybrid Mixture Propagation	9
1.2.4 Algorithm Overview	10
1.3 Mixand Continuous Propagation	12
1.3.1 No Splitting Case	13
1.3.2 Linearity Criteria	14
1.3.3 Gaussian Mixand Splitting	18
1.3.4 Benchmark Splitting Examples	26
1.4 Experimental Results	29
1.4.1 Simulation	29
1.4.2 Experimental Data	35
1.4.3 MIT-Cornell Collision Example	40
1.5 Conclusion	44
2 Probabilistic Anticipation Using Gaussian Process Models	46
2.1 Introduction	46
2.2 Gaussian Process Overview	49
2.3 Analytical Evaluation of GPs at Uncertain Inputs	50
2.3.1 GP Mean and Covariance at Uncertain Input	51
2.3.2 GP Input-Output Covariance	51
2.4 On-the-Fly Data Selection	53
2.5 Adaptive Gaussian Mixture Formulation	54
2.5.1 Analytical Kurtosis Evaluation for Non-Gaussianity Detection	55
2.5.2 Numerically Evaluated Local Linearity	56
2.5.3 GP-aGMM	57
2.6 GP-aGMM Probabilistic Prediction Examples	59
2.6.1 Univariate Nonstationary Growth Model	59
2.6.2 Two Stage Driver-Vehicle Model	62
2.6.3 Experimental Evaluation of Two Stage Driver-Vehicle Model	65
2.7 Conclusion	79

3	Learning Gaussian Process Models in Lane Feature Spaces	81
3.1	Introduction	81
3.2	Two-Stage Driver Vehicle Modeling	83
3.2.1	Parametric Vehicle Model	85
3.2.2	Gaussian Process Driver Model	86
3.2.3	Fast Approximate GPs	87
3.3	hciLab Driving Data Set	88
3.4	Human Driver Models in Lane Feature Space	90
3.4.1	Lane Feature Selection for Anticipation	92
3.5	Lane Feature GP Anticipation (LFGPA) Algorithm	96
3.5.1	Map and GPS Free Anticipation	98
3.6	Experimental Validation	100
3.6.1	Study of the Size of the Feature Space and Size of the SPGP	100
3.6.2	Generalization of LFGPA Algorithm Across Data Sets	103
3.7	Conclusion	115
	Conclusion	118
	A Kurtosis Derivation	121
	Bibliography	136

LIST OF TABLES

2.1	Computation time statistics (sec)	70
3.1	First six optimal feature sets for lane feature steering model	94
3.2	First six optimal feature sets for lane feature throttle model	94
3.3	Anticipation Accuracy and Precision	112

LIST OF FIGURES

1.1	Block diagram illustrating propagation of hybrid Gaussian mixture through a given dynamics model	11
1.2	The transformation from an arbitrary Gaussian/splitting axis to the offline problem	22
1.3	J_{ISD} as a function of N and σ , collected for 5000 randomly generated two-dimensional Gaussian distributions and splitting axes. The one-sigma error bars are shown	26
1.4	Accuracy of propagation for the UNGM model 1.4(a) and cubic model 1.4(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)	28
1.5	NLL between “truth” particle set and computed probability distributions plotted against the lookahead time (up to a 3.5 second anticipation horizon) for an obstacle vehicle on a straight road 1.5(a), approaching a turn 1.5(b), and approaching an intersection 1.5(c).	31
1.6	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.	33
1.7	Performance of monte-carlo propagation of four-state obstacle vehicle state distribution	34
1.8	Log-likelihood and standard deviation of observed vehicles given by hGMM anticipation algorithm as a function of $e_{\text{res,max}}$	36
1.9	t-test results comparing Log-Likelihood data at different linearity thresholds $e_{\text{res,max}}$ to the data for $e_{\text{res,max}} = 1$	37
1.10	Expected off-track error and standard deviation of observed vehicles given by hGMM anticipation algorithm as a function of the linearity threshold $e_{\text{res,max}}$	38
1.11	t-test results comparing EOTE data at different linearity thresholds $e_{\text{res,max}}$ to the data for $e_{\text{res,max}} = 1$	39
1.12	Anticipation of a real tracked vehicle using the hGMM (1.12(a),1.12(b), and 1.12(c)) and a single Gaussian predictor (1.12(d),1.12(e), and 1.12(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.	39
1.13	MIT-Cornell collision in the 2007 DUC	40
1.14	hGMM anticipation algorithm applied to the MIT-Cornell Collision	41
1.15	Anticipated probability of collision	42
1.16	Computation time required to anticipate an obstacle to a 4.5 second horizon as a function of linearity threshold and maximum number of allowed mixands (enforced by Runnall’s reduction algorithm) [1]	43

1.17	hGMM prediction accuracy as a function of linearity threshold and maximum number of allowed mixands	44
2.1	Analytical prediction of a GP model trained from a simulated noisy UNGM model. The true output distribution is shown as a red line. The predicted distribution for the different methods is shown as a black dashed line. Adaptive splitting in (c) and (d) is based on the analytical kurtosis (k) metric and the numerical local linearity (NL) metric. A kurtosis threshold of $k_{\max} = 0.2$ and a local linearity threshold of $e_{\max} = 3$ are used.	60
2.2	Average KLD from true predicted distribution for $n_{\text{sim}} = 100$ random input distributions propagated through the UNGM function. .	61
2.3	Block diagram of the two stage driver-vehicle model consisting of a GP driver behavior model combined with a parametric vehicle dynamics model.	63
2.4	Turn left prediction results for each algorithm over 20 timesteps. Ellipses represent the prediction error uncertainty at 50% confidence intervals.	67
2.5	Average KLD from the Monte Carlo solution for the turn left maneuver.	68
2.6	Average KLD from the Monte Carlo solution for the drive straight maneuver. Note: the axes for this figure have been rescaled compared to the left turn maneuver in Figure 2.5 for clarity.	69
2.7	Predicted distribution of the GP-aGMM-K predictor at step 20 using $k_{\max} = 2$, $k_{\max} = 0.75$ and $k_{\max} = 0.1$	72
2.8	Average KLD from the Monte Carlo result at prediction step 20 for combined stop-at-line and turn left maneuvers using the GP-aGMM-K algorithm with the kurtosis splitting metric.	73
2.9	Average KLD from the Monte Carlo result at prediction step 20 for combined stop-at-line and turn left maneuvers using the GP-aGMM-NL algorithm with the nonlinearity splitting metric.	73
2.10	Average number of mixture elements before reduction for combined stop-at-line and turn left maneuvers using the GP-aGMM-K algorithm with the kurtosis splitting metric.	74
2.11	Average number of mixture elements before reduction for combined stop-at-line and turn left maneuvers using the GP-aGMM-NL algorithm with the nonlinearity splitting metric.	74
2.12	Average KLD from Monte Carlo result at prediction step 20 as a function of the average number of mixture elements before reduction for combined stop-at-line and turn left maneuvers.	75
2.13	KLD between the GP-aGMM solution and the Monte Carlo solution as a function of the on-the-fly data selection size n for the left turn maneuver, without using splitting.	76

2.14	Computation time per prediction step as a function of the on-the-fly data selection size n for the left turn maneuver, without using splitting	77
2.15	Computation time of the excess kurtosis calculation as a function of the on-the-fly data selection size n	78
3.1	The two stage driver/vehicle model.	84
3.2	The route driven by subjects in the hciLab data set.	88
3.3	Illustration of lane features at the current vehicle state and the projected vehicle state t seconds in the future.	92
3.4	Residual error of LARS solution as a function of the number of features in the active set for the steering model.	95
3.5	Block diagram showing the LFGPA algorithm.	97
3.6	Expected value of the performance metrics $e_1(\mathbf{x}_k)$, $e_2(\mathbf{x}_k)$, and computation time for a vehicle driving on a straight road.	101
3.7	Expected value of the performance metrics $e_1(\mathbf{x}_k)$ and $e_2(\mathbf{x}_k)$ for a vehicle driving on a curved road.	102
3.8	Region partitioning of the hciLab data set. Much of the route after the last region defined here ($s5$) consists of a tunnel, so GPS data is not available.	103
3.9	Anticipation log-likelihood of LFGPA algorithms learned using different highway regions ($h1, h2, h3, hAll$) and the entire data set ($dAll$) anticipating the behavior of vehicles driving in the three highway regions ($h1, h2, h3$).	105
3.10	Anticipation log-likelihood of LFGPA algorithms learned using different highway ramp regions ($r1, r2, r3, rAll$) and the entire data set ($dAll$) anticipating the behavior of vehicles driving in the three highway ramp environments ($r1, r2, r3$).	106
3.11	Anticipation log-likelihood of LFGPA algorithms learned using different surface street regions ($s1, s2, s3, s4, s5, sAll$) and the entire data set ($dAll$) anticipating the behavior of vehicles driving in the five surface street environments ($s1, s2, s3, s4, s5$).	106
3.12	Prediction performance using a lane feature GP driver model trained with the entire data set. Points in green are initial conditions from which the anticipations were accurate. Points in red are initial conditions from which the anticipation was inaccurate. Here, 93.4% of the predictions were accurate.	110
3.13	Prediction performance using a lane feature GP driver model trained with the data from regions $\{h1, r2, s2, s5\}$. Points in green are initial conditions from which the anticipations were accurate. Points in red are initial conditions from which the anticipation was inaccurate. Here, 92.3% of the predictions were accurate.	111

3.14	Prediction performance using a lane feature GP driver model trained with the data from regions $\{h1, r2, s2, s5\}$. Points in green are initial conditions from which the anticipations were accurate. Points in red are initial conditions from which the anticipation was inaccurate. Here, 73.9% of the predictions were accurate. . . .	111
3.15	Prediction performance using a lane feature GP driver model trained with only surface street data ($\{s1, s2, s3, s4, s5\}$). Points in green are initial conditions from which the anticipations were accurate. Points in red are initial conditions from which the anticipation was inaccurate. Here, only 69.2% of the predictions were accurate, and accuracy on the highway regions is especially bad. . .	112
3.16	Comparison of predictions for left turn maneuver between a GP-aGMM driver model trained on the same data set (3.16(a)) and an LFGPA algorithm trained on the hciLab data set(3.16(b))	114
3.17	Log-likelihood comparison (smaller is better) of an LFGPA algorithm trained on data from the hciLab data set, and a GP-aGMM trained on the local data set.	114

INTRODUCTION

This thesis presents research into the probabilistic anticipation of tracked dynamic objects, specifically human-driven cars, to enable safer autonomy of road vehicles. Chapter 1 presents an adaptive propagation algorithm, capable of efficiently, and accurately, propagating the belief of an object's state through an assumed nonlinear dynamics model many steps into the future. Chapters 2 and 3 apply Gaussian Processes, a non parametric data driven modeling technique, to the problem of modeling human drivers for the purpose of anticipation. Chapter 2 develops tools that enable the efficient use of Gaussian Process dynamics models in an adaptive Gaussian Mixture Model estimator. Chapter 3 presents a generalization of the state space of tracked dynamic objects that enable drastic reductions in the training data requirements of Gaussian Process driver models, enabling the feasible use of such models in realistically large environments.

Chapter 1, published in IEEE Transactions on Robotics, 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 to a specified anticipation horizon. A Gaussian splitting method is proposed based on the sigma-point transform and the nonlinear dynamics function, which enables accurate predictions even with nonlinear dynamics models. 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.

Chapter 2, submitted to the International Journal of Robotics Research, presents an adaptive Gaussian Mixture Model formulation for performing multiple-step probabilistic state predictions using a nonparametric Gaussian Process regression model. The presented prediction algorithm is applicable to any dynamic system that is challenging to model parametrically, but where data is available. Gaussian mixture elements are propagated through the Gaussian Process by analytically evaluating expectation integrals for the moments of the output distribution. Two metrics are presented and compared for adaptively splitting the initial state distribution into a sum of Gaussians to reduce the effect of nonlinearities on prediction accuracy: 1) an analytical evaluation of the excess kurtosis which measures the non-Gaussianity of the output distribution, and 2) a weighted least squares regression model which evaluates the local nonlinearity of the Gaussian Process model with respect to the input distribution. In addition, an on-the-fly data selection method is presented to reduce the computational complexity associated with analytically evaluating the higher order moments of the Gaussian Process output distribution. The proposed adaptive Gaussian Process - adaptive Gaussian Mixture Model (GP-aGMM) formulation is applied to the case of anticipating driver behavior at road intersections using a Gaussian Process driver behavior model in combination with a parametric vehicle model. Prediction performance for this scenario is evaluated using driving data collected from three human subjects navigating a standard four-way intersection. Results demonstrate that the presented prediction algorithm is capable of accurately capturing multimodal behavior in the Gaussian Process training data.

Chapter 3 presents a novel method for performing road vehicle anticipation using Gaussian Process driver models. The presented algorithm relaxes the need

for a specific training set for each environmental space, which is a key limitation of using Gaussian Process driver models. Tracked vehicle states are generalized using a transformation into a lane feature space that describes the relationship between vehicle state and the local environment. The lane feature space allows a Gaussian Process driver model trained using data from one environment to be applied to the problem of anticipating the behavior of a vehicle driving in a different environment, thereby eliminating the need for full environmental coverage by training data, and enabling practical application. The challenge of defining relevant lane features is addressed here by specifying a large number of lane features, and automatically identifying relevant subsets of the lane feature space during training. The presented algorithm is trained using the hciLab human driver data set, and evaluated both in simulation and on examples from the hciLab data set.

PROBABILISTIC ANTICIPATION OF ROAD VEHICLES

1.1 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 [2, 3, 4]. 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 hours) [5, 6, 7]. Continued development in the field has led to autonomous cars beginning to drive in real urban environments alongside civilian traffic [8, 9, 10, 11].

Many autonomous cars use primarily reactionary planners that rely on rapid re-planning in order to respond to the dynamic environments in which they operate [5]. A collision between the 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 non-linear. In the literature, several algorithms have

been proposed to simplify the problem, such as assuming no uncertainty in future obstacle behavior [15, 16, 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 non-linear 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. Du Toit and Burdick [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 [21].

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 un-

certainties and allow for the use of non-linear 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.

A GMM based predictor is proposed in the 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 non-linear obstacle motion models.

This paper is outlined as follows: Section 1.2.1 defines the representation of the obstacle state and Section 1.2.2 defines the mixture model. Sections 1.2.3 describes the discrete and continuous anticipation of the obstacle state. Section 1.2.4 provides an overview of the proposed algorithm. The details of temporal propagation are described in Section 1.3, including the non-linearity detection and mixand splitting. Section 1.4.1 provides an example implementation of the anticipation algorithm in simulation, and Section 1.4.2 demonstrates the efficacy of the algorithm on a real data set. Section 1.4.3 demonstrates the potential safety improvements

by applying the proposed algorithm to the 2007 MIT-Cornell autonomous collision.

1.2 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. Hybrid, here, 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 non-linear 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).

1.2.1 Obstacle State Model

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

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^C \\ \mathbf{x}_k^D \end{bmatrix}. \quad (1.1)$$

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:

$$\begin{aligned} \mathbf{x}_{k+1}^D &= f^D(\mathbf{x}_k^D, \mathbf{x}_k^C, \mathbf{v}_k) \\ \mathbf{x}_{k+1}^C &= f^C(\mathbf{x}_{k+1}^D, \mathbf{x}_k^C, \mathbf{v}_k) \end{aligned} \quad (1.2)$$

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 ($\mathbf{x}_k^D, \mathbf{x}_k^C$) 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.

1.2.2 Hybrid Mixture Probability Distribution Representation

The probability distribution $p(\mathbf{x}_k)$ is approximated using an hGMM. The hGMM extends the GMM presented in Alspach and Sorenson [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:

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

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

$$\sum_{i=1}^{M_k} w_k^i = 1, \text{ and } w_k^i > 0 \quad \forall i. \quad (1.3)$$

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:

$$p^i(\mathbf{x}_k) = \delta(\mathbf{x}_k^D - \alpha_k^i) \cdot \mathcal{N}(\mathbf{x}_k^C | \mu_k^i, \Sigma_k^i) \quad (1.4)$$

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.

1.2.3 Hybrid Mixture Propagation

The hGMM formulation enables each mixand to be propagated forward in time independently using the dynamics model (Equation 1.2), 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 Equation 1.2:

$$\begin{bmatrix} (\alpha_k^1, \mu_k^1, \Sigma_k^1, w_k^1) \\ \vdots \\ (\alpha_k^{M_k}, \mu_k^{M_k}, \Sigma_k^{M_k}, w_k^{M_k}) \end{bmatrix} \xrightarrow{h^D} \begin{bmatrix} (\alpha_{k+1}^1, \mu_{k-}^1, \Sigma_{k-}^1, w_{k-}^1) \\ \vdots \\ (\alpha_{k+1}^{M_{k-}}, \mu_{k-}^{M_{k-}}, \Sigma_{k-}^{M_{k-}}, w_{k-}^{M_{k-}}) \end{bmatrix} \quad (1.5)$$

where α_{k+1}^i is the discrete state for the i^{th} 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:

$$\left[\begin{array}{c} \left(\alpha_{k+1}^1, \mu_{k-}^1, \Sigma_{k-}^1, w_{k-}^1 \right) \\ \vdots \\ \left(\alpha_{k+1}^{M_{k-}}, \mu_{k-}^{M_{k-}}, \Sigma_{k-}^{M_{k-}}, w_{k-}^{M_{k-}} \right) \end{array} \right] \xrightarrow{h^C} \left[\begin{array}{c} \left(\alpha_{k+1}^1, \mu_{k+1}^1, \Sigma_{k+1}^1, w_{k+1}^1 \right) \\ \vdots \\ \left(\alpha_{k+1}^{M_{k+1}}, \mu_{k+1}^{M_{k+1}}, \Sigma_{k+1}^{M_{k+1}}, w_{k+1}^{M_{k+1}} \right) \end{array} \right] \quad (1.6)$$

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

1.2.4 Algorithm Overview

The propagation steps of the hGMM are summarized in Figure 1.1. 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 (Section 1.3.2) that ensures the accuracy of propagation by splitting mixands that propagate

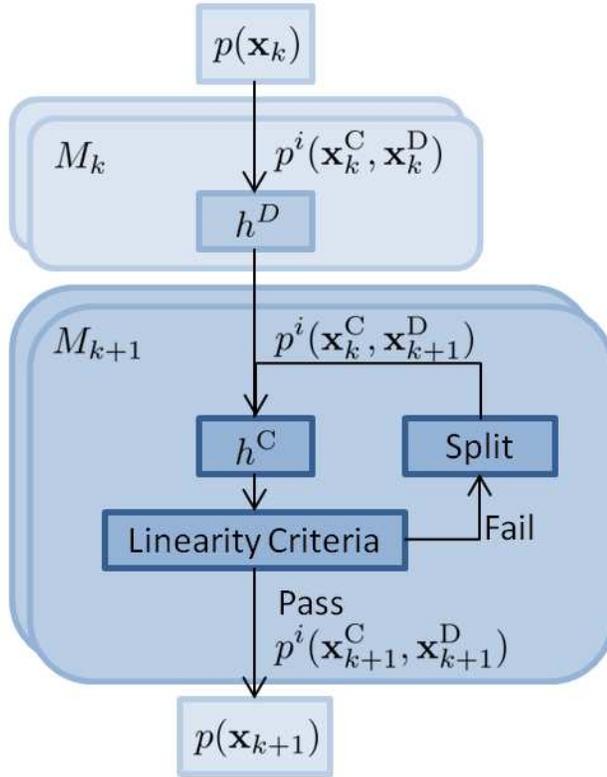


Figure 1.1: Block diagram illustrating propagation of hybrid Gaussian mixture through a given dynamics model

inaccurately (Section 1.3.3). After the continuous propagation is completed, the probability distribution over the state at time $k + 1$ can be written as:

$$p(\mathbf{x}_{k+1}) = \sum_{i=1}^{M_{k+1}} w_{k+1}^i \cdot \delta(\mathbf{x}_{k+1}^D - \alpha_{k+1}^i) \cdot \mathcal{N}(\mathbf{x}_{k+1}^C | \mu_{k+1}^i, \Sigma_{k+1}^i). \quad (1.7)$$

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.

1.3 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 (Equation 1.6) uses the sigma-points (SP) transform (sometimes called the unscented transform) [24, 25, 26, 27] to propagate the mixand through the continuous dynamics function f^C (Equation 1.2). Sigma-points are well-explored for use in estimation problems involving nonlinear dynamics and measurement models [24, 26]. The Gaussian distributions $\mathcal{N}(\mathbf{x}_k^C | \mu_k^i, \Sigma_k^i)$ and $\mathcal{N}(\mathbf{v}_k | \mathbf{0}, \Sigma_v)$ are approximated by deterministically choosing sets of points $\boldsymbol{\chi}_k^i$ and $\boldsymbol{\Upsilon}_k$, called sigma points:

$$\begin{aligned}\boldsymbol{\chi}_k^i &= \left[\chi_k^{i,1}, \dots, \chi_k^{i,1+2n_x+2n_v} \right] \\ \boldsymbol{\Upsilon}_k &= \left[\Upsilon^1, \dots, \Upsilon^{1+2n_x+2n_v} \right]\end{aligned}\tag{1.8}$$

where n_x is the dimension of \mathbf{x}_k^C and n_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 Equation 1.8, the matrix square-roots of the covariances Σ_k^i and Σ_v are first solved:

$$\begin{aligned}\mathbf{S}_{\mathbf{x},k}^i \cdot (\mathbf{S}_{\mathbf{x},k}^i)^T &= \Sigma_k^i \\ \mathbf{S}_v \cdot (\mathbf{S}_v)^T &= \Sigma_v\end{aligned}$$

such that

$$\begin{aligned}
\mathbf{S}_{\mathbf{x},k}^i &= \left[S_{\mathbf{x},k}^{i,1}, \dots, S_{\mathbf{x},k}^{i,n_{\mathbf{x}}} \right] \\
\mathbf{S}_{\mathbf{v}} &= \left[S_{\mathbf{v}}^1, \dots, S_{\mathbf{v}}^{n_{\mathbf{v}}} \right].
\end{aligned} \tag{1.9}$$

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

$$\begin{aligned}
\chi_k^{i,j} &= \begin{cases} \mu_k^i, & j \in \{0, [2n_{\mathbf{x}} + 1, 2n_{\mathbf{x}} + 2n_{\mathbf{v}}]\} \\ \mu_k^i + \gamma \cdot S_{\mathbf{x},k}^{i,j}, & j \in [1, n_{\mathbf{x}}] \\ \mu_k^i - \gamma \cdot S_{\mathbf{x},k}^{i,j-n_{\mathbf{x}}}, & j \in [n_{\mathbf{x}} + 1, 2n_{\mathbf{x}}] \end{cases} \\
\Upsilon^j &= \begin{cases} \mathbf{0}, & j \in [0, 2n_{\mathbf{x}}] \\ \gamma \cdot \mathbf{S}_{\mathbf{v},k}^{j-2n_{\mathbf{x}}}, & j \in [2n_{\mathbf{x}} + 1, 2n_{\mathbf{x}} + n_{\mathbf{v}}] \\ -\gamma \cdot \mathbf{S}_{\mathbf{v},k}^{j-2n_{\mathbf{x}}-n_{\mathbf{v}}}, & j \in [2n_{\mathbf{x}} + n_{\mathbf{v}} + 1, 2n_{\mathbf{x}} + 2n_{\mathbf{v}}] \end{cases}
\end{aligned} \tag{1.10}$$

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 $\chi_{k+1}^{i,j} = f^C(\alpha_{k+1}^i, \chi_k^{i,j}, \Upsilon^j)$, and the resulting set $\boldsymbol{\chi}_{k+1}^i = [\chi_{k+1}^{i,1}, \dots, \chi_{k+1}^{i,1+2n_{\mathbf{x}}+2n_{\mathbf{v}}}]$ is used to evaluate the impact of nonlinearities (Section 1.3.2) on the accuracy of prediction, and to find the predicted distribution at time $k + 1$, $\mathcal{N}(\mathbf{x}_{k+1}^C | \mu_{k+1}^i, \Sigma_{k+1}^i)$ (Section 1.3.1).

1.3.1 No Splitting Case

If the mixand passes the linearity criteria described in Section 1.3.2, the propagated sigma points are used to find the predicted mean and covariance (μ_{k+1}^i and Σ_{k+1}^i) for the mixand:

$$\begin{aligned}
\mu_{k+1}^i &= \sum_{j=1}^{1+2n_{\mathbf{x}}+2n_{\mathbf{v}}} \gamma^j \cdot \chi_{k+1}^{i,j} \\
\Sigma_{k+1}^i &= \sum_{j=1}^{1+2n_{\mathbf{x}}+2n_{\mathbf{v}}} \beta^j \cdot (\chi_{k+1}^{i,j} - \mu_{k+1}^i) (\chi_{k+1}^{i,j} - \mu_{k+1}^i)^T \\
\gamma^j &= \begin{cases} \frac{\lambda}{\lambda+n_{\mathbf{x}}+n_{\mathbf{v}}} & \text{for } j = 1 \\ \frac{1}{2(\lambda+n_{\mathbf{x}}+n_{\mathbf{v}})} & \text{for } j \neq 1 \end{cases} \\
\beta^j &= \begin{cases} \frac{\lambda}{\lambda+n_{\mathbf{x}}+n_{\mathbf{v}}} + 2 & \text{for } j = 1 \\ \frac{1}{2(\lambda+n_{\mathbf{x}}+n_{\mathbf{v}})} & \text{for } j \neq 1 \end{cases}.
\end{aligned} \tag{1.11}$$

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, described in Section 1.3.3.

1.3.2 Linearity Criteria

The strength of Gaussian mixture models is that they can accurately approximate non-Gaussian pdfs that arise either through non-Gaussian process or measurement noise or through non-linearities in the system model [23, 28]. Traditional Gaussian mixture model propagation algorithms [29, 30] use standard non-linear filtering techniques, such as Extended Kalman Filtering or Unscented Kalman Filtering, to propagate the individual mixands without considering the impact of local non-linearities 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:

$$\begin{aligned}\bar{\boldsymbol{\chi}}_{k+1}^i &= \mathbf{A}\bar{\boldsymbol{\chi}}_k^i + \mathbf{b} \cdot \mathbf{1}_n \\ &= [\mathbf{A}, \mathbf{b}] \begin{bmatrix} \bar{\boldsymbol{\chi}}_k^i \\ \mathbf{1}_n \end{bmatrix}\end{aligned}\tag{1.12}$$

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

$$\begin{aligned}\bar{\boldsymbol{\chi}}_k^i &= [\chi_k^{i,0}, \dots, \chi_k^{i,2n_x}] \\ \bar{\boldsymbol{\chi}}_{k+1}^i &= [\chi_{k+1}^{i,0}, \dots, \chi_{k+1}^{i,2n_x}].\end{aligned}\tag{1.13}$$

For a linear system, Equation 1.12 can be solved exactly for \mathbf{A} and \mathbf{b} . For a non-linear system, the residuals can be calculate by casting the linearization as a least-squares problem.

$$[\mathbf{A}^*, \mathbf{b}^*] = \operatorname{argmin} \left(\left\| \bar{\boldsymbol{\chi}}_{k+1}^i - [\mathbf{A}, \mathbf{b}] \begin{bmatrix} \bar{\boldsymbol{\chi}}_k^i \\ \mathbf{1}_n \end{bmatrix} \right\| \right)\tag{1.14}$$

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

$$\begin{aligned}
e_{\text{res}} &= \min \left(\left\| \bar{\boldsymbol{\chi}}_{k+1}^i - [\mathbf{A}, \mathbf{b}] \begin{bmatrix} \bar{\boldsymbol{\chi}}_k^i \\ \mathbf{1}_n \end{bmatrix} \right\| \right) \\
&= \left\| \bar{\boldsymbol{\chi}}_{k+1}^i - [\mathbf{A}^*, \mathbf{b}^*] \begin{bmatrix} \bar{\boldsymbol{\chi}}_k^i \\ \mathbf{1}_n \end{bmatrix} \right\|.
\end{aligned} \tag{1.15}$$

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 non-linear the underlying model is in the neighborhood of the sigma-points.

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

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

where

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

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

$$\mathbf{Q}\mathbf{Q}^T = \mathbf{I}. \tag{1.16}$$

Substituting the factorization from Equation 1.16 into Equation 1.14 yields:

$$\begin{aligned}
[\mathbf{A}^*, \mathbf{b}^*] &= \operatorname{argmin} \left(\left\| \bar{\boldsymbol{\chi}}_{k+1}^i - [\mathbf{A}, \mathbf{b}] \mathbf{LQ} \right\| \right) \\
&= \operatorname{argmin} \left(\left\| \bar{\boldsymbol{\chi}}_{k+1}^i \mathbf{Q}^T - [\mathbf{A}, \mathbf{b}] \mathbf{L} \right\| \right).
\end{aligned} \tag{1.17}$$

Because of the structure of \mathbf{L} , the arguments $[\mathbf{A}, \mathbf{b}]$ only effect the first $n_{\mathbf{x}} + 1$ columns of the norm in Equation 1.17. The matrix $\bar{\boldsymbol{\chi}}_{k+1}^i \mathbf{Q}^T$ is partitioned according to what is explained by the linear model ($\hat{\boldsymbol{\chi}}_{k+1}^i$) and what is not explained by a linear model ($\hat{\boldsymbol{\chi}}_{k+1,\text{res}}^i$). In block form, this is written:

$$[\mathbf{A}^*, \mathbf{b}^*] = \operatorname{argmin} \left(\left\| \begin{bmatrix} \hat{\boldsymbol{\chi}}_{k+1}^i & \hat{\boldsymbol{\chi}}_{k+1,\text{res}}^i \end{bmatrix} - [\mathbf{A}, \mathbf{b}] [\mathbf{L}_0, \mathbf{0}] \right\| \right). \quad (1.18)$$

The optimal arguments $[\mathbf{A}^*, \mathbf{b}^*]$ can be extracted from the partitioned matrix in Equation 1.18 and \mathbf{L}_0 :

$$[\mathbf{A}^*, \mathbf{b}^*] = \hat{\boldsymbol{\chi}}_{k+1}^i \mathbf{L}_0^{-1}. \quad (1.19)$$

The desired linearization error e_{res} , defined in Equation 1.15, simplifies to:

$$e_{\text{res}} = \left\| \begin{bmatrix} \mathbf{0} & \hat{\boldsymbol{\chi}}_{k+1,\text{residual}}^i \end{bmatrix} \right\|. \quad (1.20)$$

Note that the linearization error e_{res} can be calculated without requiring the computed optimal arguments \mathbf{A}^* and \mathbf{b}^* in Equation 1.19. The scalar residual error in Equation 1.20 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 by the authors in Havlak and Campbell [31], and similar to the metric proposed in Huber [32]. Other accuracy metrics for the sigma-point transform have been developed, for example Van Der Merwe [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 linearization error, it is desirable to identify which sigma-points are explained the least by the optimal linear model.

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

$$\boldsymbol{\mathcal{E}}_{k+1}^i = \bar{\boldsymbol{\chi}}_{k+1}^i - (\mathbf{A}^* \bar{\boldsymbol{\chi}}_k^i + \mathbf{b}^*) = [\boldsymbol{\mathcal{E}}_{k+1}^{i,1}, \dots, \boldsymbol{\mathcal{E}}_{k+1}^{i,2 \cdot n_{\mathbf{x}}+1}]. \quad (1.21)$$

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

$$\boldsymbol{\mathcal{E}}_{k+1}^i = [\mathbf{0}, \hat{\boldsymbol{\chi}}_{k+1,\text{residual}}^i] \mathbf{Q}. \quad (1.22)$$

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 ($\chi_k^{i,j}$) are weighted by the norm of their associated residual linearization errors ($\|\boldsymbol{\mathcal{E}}_{k+1}^{i,j}\|$). 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.

1.3.3 Gaussian Mixand Splitting

Given the sigma point residual error of the propagated mixand in Equation 1.22, 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 Equation 1.22) 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, 35, 36, 37, 1], 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 [32, 31, 38, 39], although there has been interest in the topic in the machine learning community [40, 41, 42]. Other adaptive GMM filters include Caputi and Moose [43], which relies on non-Gaussian process noise to increase the number of mixands, and Terejanu et al. [44], 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 [45]. However, the KLD between a Gaussian and a GMM can not be evaluated analytically, and effective approximations can be resource intensive [46, 47]. 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 [48]. 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 [48].

$$J_{\text{ISD}} = \int_x (\mathcal{N}(x|\mu, \Sigma) - \hat{p}(x))^2 dx \quad (1.23)$$

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 (Equation 1.23) is minimized. The formal problem is to solve for $w_i, \mu_i,$ and Σ_i for all i such that:

$$[w_i, \mu_i, \Sigma_i]_{i=1\dots N} = \operatorname{argmin} \left(\int_x \left(\mathcal{N}(x|\mu, \Sigma) - \sum_{i=1}^N w_i \cdot \mathcal{N}(x|\mu_i, \Sigma_i) \right)^2 dx \right). \quad (1.24)$$

Because this optimization is computationally expensive, an off-line optimal solution is computed and stored for use as needed by the anticipation algorithm in real time. Pre-computing an optimal split for real time use requires that the pre-computed 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 pre-computed problem:

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

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

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

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

$$\mathbf{e}_1 = \mathbf{R}_x \mathbf{T}^{-1} \mathbf{e}_{\text{split}}. \quad (1.25)$$

Figure 1.2 illustrates the transformation in Equation 1.25 applied to a general two dimensional Gaussian. Figure 1.2(a) shows the original single Gaussian to be split, and the axis along which the Gaussian is to be split, e_{split} . Figures 1.2(b) and 1.2(c) show the translation of the Gaussian to the origin ($x - \mu$) and the transformation that yields a unit covariance of the single Gaussian. Finally, Figure 1.2(d) shows the rotation (\mathbf{R}_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:

$$\{(w_1^*, \mu_1^*, \Sigma_1^*), \dots, (w_N^*, \mu_N^*, \Sigma_N^*)\} = \underset{\{w_i, \mu_i, \Sigma_i\}}{\text{argmin}} \left(\int_x (\mathcal{N}(x|\mathbf{0}, \mathbf{I}) - \hat{p}(x))^2 dx \right). \quad (1.26)$$

The solution to this optimization problem is computed off-line, and can be applied at runtime to the general problem (Equation 1.24) using the transformation from Equation 1.25:

$$\begin{aligned} w_i &\approx w_i^* \\ \mu_i &\approx \mathbf{T} \mathbf{R}_x^T \cdot \mu_i^* + \mu \\ \Sigma_i &\approx \mathbf{T} \mathbf{R}_x^T \Sigma_i^* \mathbf{R}_x \mathbf{T}^T. \end{aligned} \quad (1.27)$$

The offline optimization (Equation 1.26) 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.

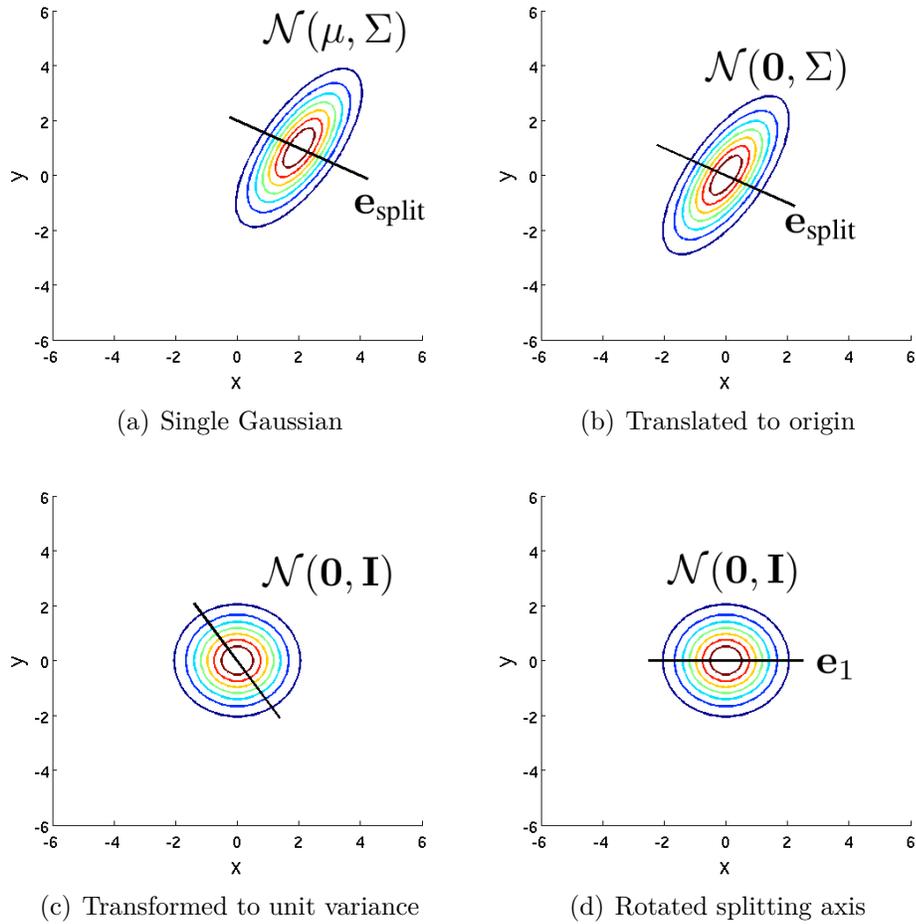


Figure 1.2: The transformation from an arbitrary Gaussian/splitting axis to the offline problem

For an $n_{\mathbf{x}}$ 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 off-line 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} :

$$\mu_i = \begin{bmatrix} \left(i - \frac{N-1}{2}\right) \delta_{\mu} \\ \mathbf{0} \end{bmatrix}. \quad (1.28)$$

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:

$$\Sigma_i = \begin{bmatrix} \sigma & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (1.29)$$

where

$$\sigma \in [0, 1]. \quad (1.30)$$

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), so 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, defined in Equation 1.23, yields:

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

Williams provides a closed-form solution for each of the above terms: [48]

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

Re-arranging the terms yields a quadratic program with the weights as arguments:

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

where

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

such that

$$\begin{aligned} w_i &\geq 0 \quad \forall i \\ \sum_i w_i &= 1. \end{aligned} \tag{1.31}$$

Equation 1.31 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.

Figure 1.3 plots the mean and standard deviation of the ISD as a function of σ , and N for 5000 randomly generated examples of two-dimensional 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 two-dimensional 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 Equation 1.27. Intuitively, $\sigma = 1$ minimizes the ISD, while $\sigma = 0$ most 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 .

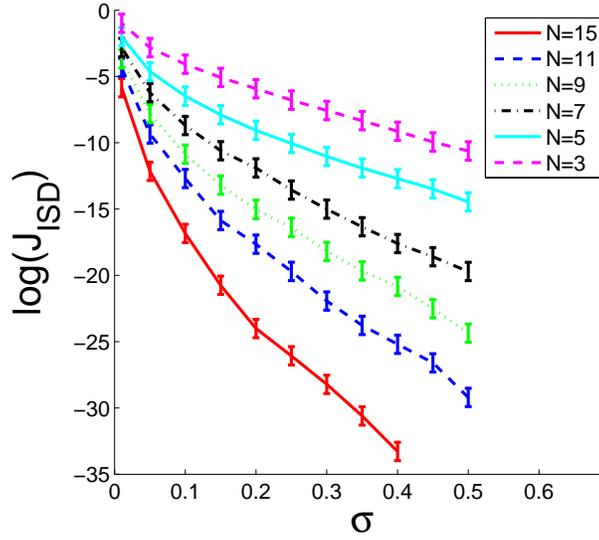


Figure 1.3: J_{ISD} as a function of N and σ , collected for 5000 randomly generated two-dimensional Gaussian distributions and splitting axes. The one-sigma error bars are shown

1.3.4 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 non-stationary growth model (UNGM): [49, 50]

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

$$\alpha = 0.3, \beta = 1, \gamma = 1 \tag{1.32}$$

and the second is a univariate cubic:

$$x_{k+1} = ax_k^3 + bx_k^2 + cx_k + d$$

$$a = 6, b = 1, c = 1, d = 1. \tag{1.33}$$

The UNGM is commonly used to evaluate the performance of nonlinear filters [49, 50]. Also, 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 (means uniformly distributed between -2 and 2, and variances uniformly distributed between 0 and 2) and propagated through the UNGM (Equation 1.32) and cubic (Equation 1.33) 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 to 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:

$$\text{KLD} = \int_{x_{k+1}} \log \left(\frac{\hat{p}(x_{k+1})}{p(x_{k+1})} \right) \hat{p}(x_{k+1}) dx_{k+1}. \quad (1.34)$$

Figure 1.4 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 to 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 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 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

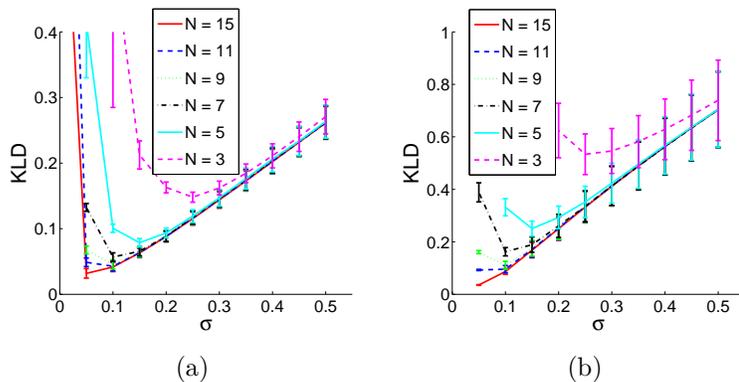


Figure 1.4: Accuracy of propagation for the UNGM model 1.4(a) and cubic model 1.4(b). For comparison, the mean and variance of the computed KLDs between a propagation with no splitting and the truth are 0.5977 and 0.0630 (UNGM model) and 1.4918 and 0.4259 (cubic model)

no-split solution.

These example problems can also be used to analyze how effective the linearity metric e_{res} (Equation 1.20) 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, 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 non-linear model. The choice of the linearity threshold ($e_{\text{res,max}}$) and of the split parameters N and σ are problem dependent, but can be explored off-line prior to implementation.

1.4 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 Runnalls [1] is used in the following examples.

1.4.1 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 reference [31]. Obstacle vehicles are assumed to drive on a known road network and to obey specified traffic regulations. Obstacle vehicles are modeled as 4-state bicycle robots, with the continuous state:

$$\mathbf{x}_k^C = \begin{bmatrix} x_k \\ y_k \\ v_k \\ \theta_k \end{bmatrix} \quad (1.35)$$

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 non-linear.

$$\mathbf{x}_{k+1}^C = f(\mathbf{x}_k^C, \mathbf{u}_k) = \begin{bmatrix} x_k + \Delta t \cdot \cos(\theta_k) \cdot v_k \\ y_k + \Delta t \cdot \sin(\theta_k) \cdot v_k \\ v_k + \Delta t \cdot (u_{1,k} + \mathbf{v}_{1,k}) \\ \theta_k + \Delta t \cdot l \cdot v_k \cdot (u_{2,k} + \mathbf{v}_{2,k}) \end{bmatrix} \quad (1.36)$$

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 $\Delta t = 0.1$ seconds 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 meters per second:

$$\mathbf{u}_k = \begin{bmatrix} u_{1,k} \\ u_{2,k} \end{bmatrix} = h(\mathbf{x}_k^C, \mathbf{x}_k^D). \quad (1.37)$$

This path following controller (Equation 1.37) is composed with the bicycle dynamics (Equation 1.36) to arrive at the continuous dynamics function f^C (Equation 1.2) required by the hGMM:

$$\mathbf{x}_{k+1}^C = f^C(\mathbf{x}_{k+1}^D, \mathbf{x}_k^C, \mathbf{v}_k) = f(\mathbf{x}_k^C, h(\mathbf{x}_k^C, \mathbf{x}_k^D)). \quad (1.38)$$

The anticipation algorithm is used with a time horizon of 3.5 seconds (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 to 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 $e_{\text{res,max}}$, and compared to the baseline of the hGMM algorithm with no splitting, which is equivalent to a single Gaussian UKF anticipation algorithm.

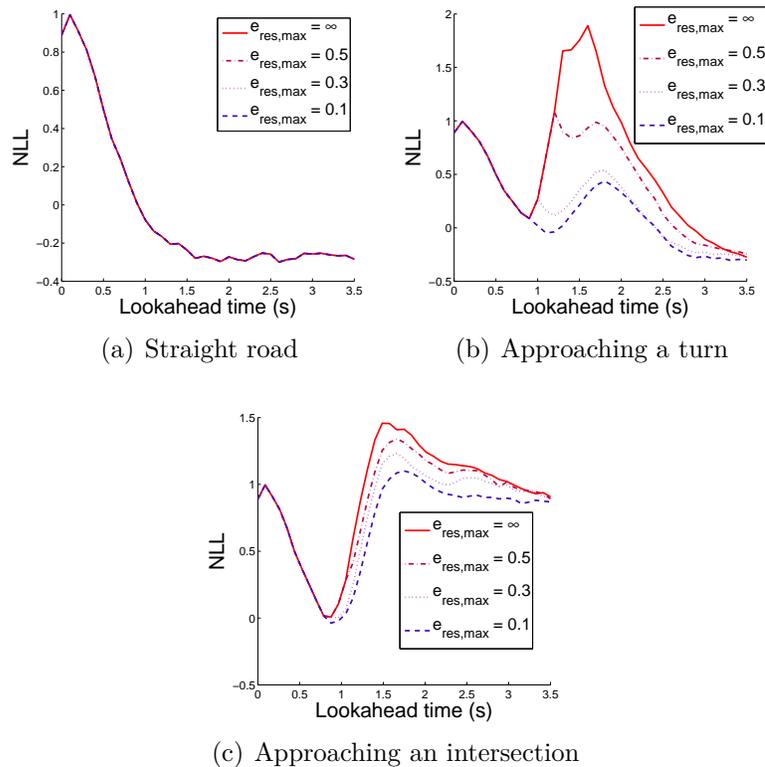


Figure 1.5: NLL between “truth” particle set and computed probability distributions plotted against the lookahead time (up to a 3.5 second anticipation horizon) for an obstacle vehicle on a straight road 1.5(a), approaching a turn 1.5(b), and approaching an intersection 1.5(c).

Figure 1.5 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 $e_{\text{res,max}}$, for each scenario: the hGMM with $e_{\text{res,max}} = 0.1$, the hGMM with $e_{\text{res,max}} = 0.3$, the hGMM with $e_{\text{res,max}} = 0.5$, and finally the hGMM with $e_{\text{res,max}} = \text{inf}$; the latter is equivalent to a single Gaussian UKF predictor. When the obstacle vehicle is traveling on a straight road (Figure 1.5(a)), all of the anticipation predictors show similar performance, because the dynamics model is very close to linear in this case. In Figures 1.5(b),1.5(c), the hGMM anticipation algorithm shows significant performance improvement over the no splitting predictor, as non-linearities 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 in the road, when the model is exhibiting a high degree of non-linear behavior. Once the turn is negotiated, and the vehicle is anticipated to be on the straight road after the turn, the non-linearities fade and the NLL decreases again.

Figure 1.6 shows the hGMM anticipated distributions of the rear-axle position for the intersection scenario, for $e_{\text{res,max}} = 0.1$. 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 Figure 1.5(c) has a peak just before 2 seconds 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 Figure 1.6(b).

The four-state obstacle vehicle simulation example is also used to compare the hGMM to Monte-Carlo (MC) propagation (particle filtering). MC methods are at-

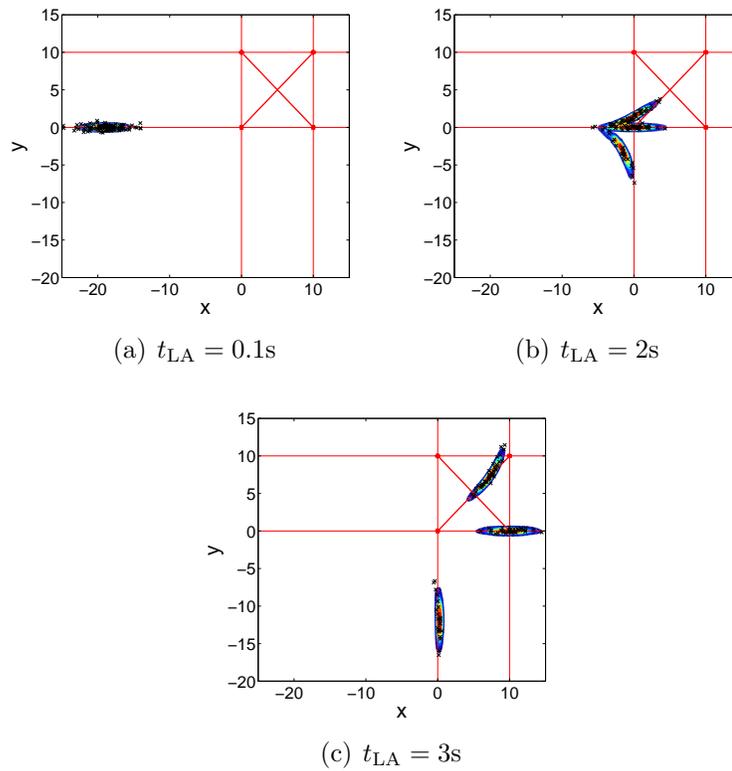


Figure 1.6: Anticipated distribution $p(x, y)$ (position of the rear axle) as obstacle traverses an intersection. The time at which the vehicle is observed is t , and the time into future the obstacle is anticipated is t_{LA} . The particle set used as truth is overlaid as black dots.

tractive because they inherently capture nonlinearities and multi-modal 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.

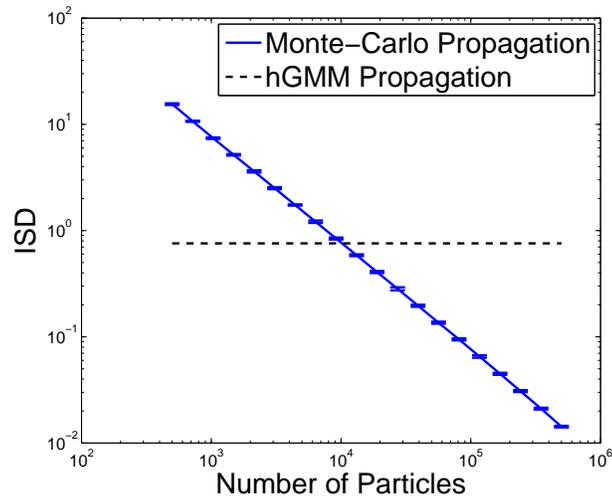


Figure 1.7: Performance of monte-carlo propagation of four-state obstacle vehicle state distribution

To compare MC propagation to the hGMM, the true distribution is computed by using a very large (six 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 four-dimensional grid. MC methods using between 500 and 500000 particles are used to propagate the distribution over the obstacle vehicle state forward in time, with multiple trials for each particle set size. Figure 1.7 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 for comparison. To match the accuracy of the hGMM, approximately 10,000 particles are required. In MATLAB, the hGMM propagation takes 1.23 seconds, while MC propagation for a 10,000 particle set takes 356 seconds. The hGMM enjoys an enormous compu-

tational advantage over MC methods for this problem, and this advantage enables a real-time implementation of the hGMM, discussed in Section 1.4.3.

1.4.2 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 data set used for this validation is the 2007 Columbus large image format (CLIF 2007) data set made available by the United States Air Force [51]. The data set 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 $e_{\text{res,max}}$, 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 1.4.1 (Equation 1.36) 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 to the anticipated distribution over the vehicle state to evaluate how well the hGMM algorithm anticipates the behavior of the tracked vehicle.

Figure 1.8 plots the log-likelihood (LL) of the hGMM anticipation agreeing with

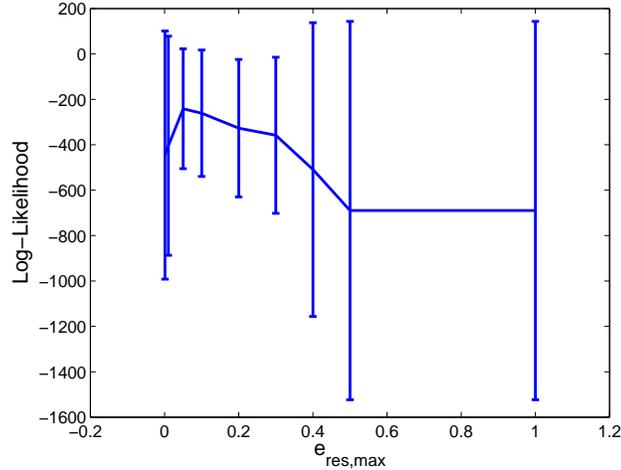


Figure 1.8: Log-likelihood and standard deviation of observed vehicles given by hGMM anticipation algorithm as a function of $e_{res,max}$

CLIF observations as a function of $e_{res,max}$. At high values of $e_{res,max}$, the LL is low with larger uncertainty bounds, indicating poorer predictive performance. The figure shows that smaller values for $e_{res,max}$ provide more accurate anticipation of observed cars, as well as much more consistent performance. At values for $e_{res,max}$ 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 $e_{res,max}$, performance also degrades due to errors introduced by repeated mixand splitting in the hGMM.

Figure 1.9 plots the p-values of t-test comparisons between the log-likelihood data for each linearity threshold $e_{res,max}$ to the log-likelihood data for the maximum linearity threshold ($e_{res,max} = 1$). This figure shows that the improved performance seen in Figure 1.8 for linearity thresholds $e_{res,max} = [0.05, 0.1, 0.2]$ 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 log-likelihood performance is not statistically differ-

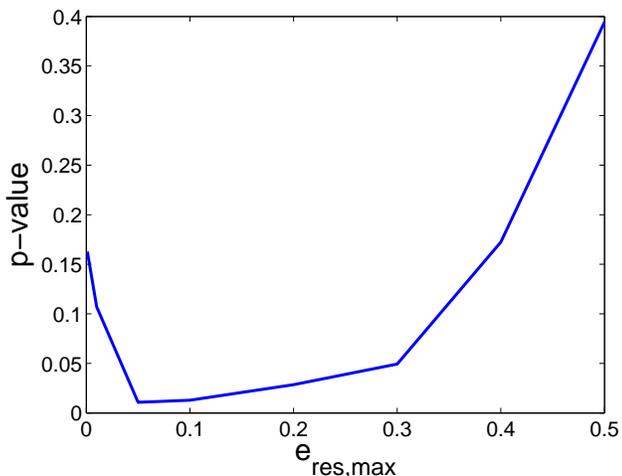


Figure 1.9: t-test results comparing Log-Likelihood data at different linearity thresholds $e_{\text{res,max}}$ to the data for $e_{\text{res,max}} = 1$

ent from the data taken with a linearity threshold high enough that no splitting occurs ($e_{\text{res,max}} = 1$).

The LL metric studied in Figures 1.8 and 1.9 considers only the likelihood that the observations agree with the anticipated obstacle state, and therefore only evaluates the hGMM at the observations. 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:

$$\text{EOTE} = \sum_{k=1}^K \int_{x_k} \int_{y_k} d(x_k, y_k) p(x_k, y_k) dy_k dx_k \quad (1.39)$$

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.

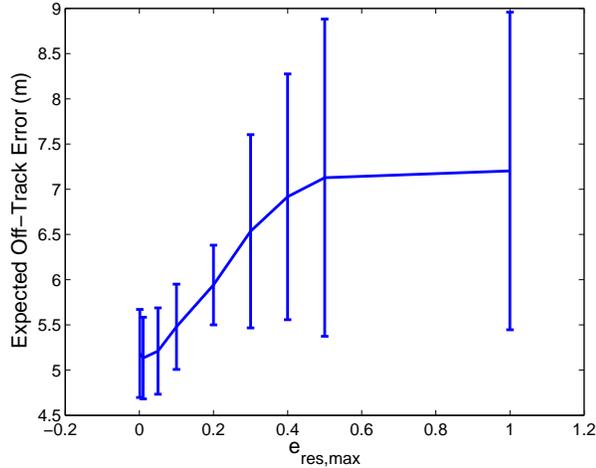


Figure 1.10: Expected off-track error and standard deviation of observed vehicles given by hGMM anticipation algorithm as a function of the linearity threshold $e_{res,max}$

Figure 1.10 plots the EOTE as a function of the linearity threshold $e_{res,max}$ used in the hGMM. At high values of $e_{res,max}$, the EOTE is large and has a large standard deviation, indicating poor anticipation performance. For small values of $e_{res,max}$, the EOTE and its standard deviation decrease, indicating that the hGMM predictions are more reasonable and more consistent. As with Figure 1.8, for $e_{res,max}$ above 0.5 the standard deviation of this metric becomes large as the mixand covariances become too large for the sigma-point approximation.

Figure 1.11 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 Figure 1.10 for small linearity thresholds ($e_{res,max} \leq 0.2$) is statistically significant.

Figure 1.12 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

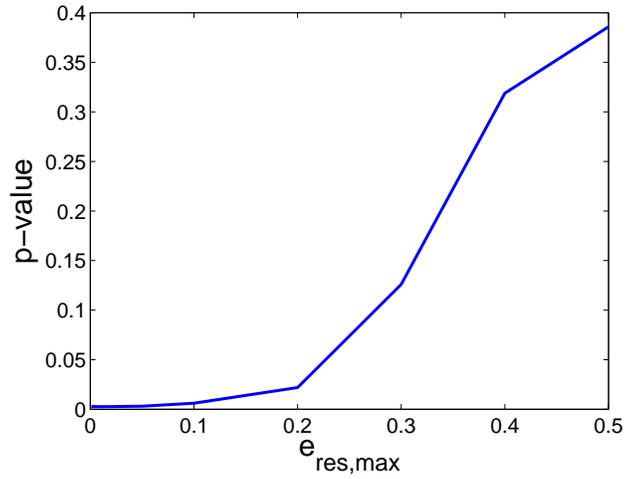


Figure 1.11: t-test results comparing EOTE data at different linearity thresholds $e_{\text{res,max}}$ to the data for $e_{\text{res,max}} = 1$

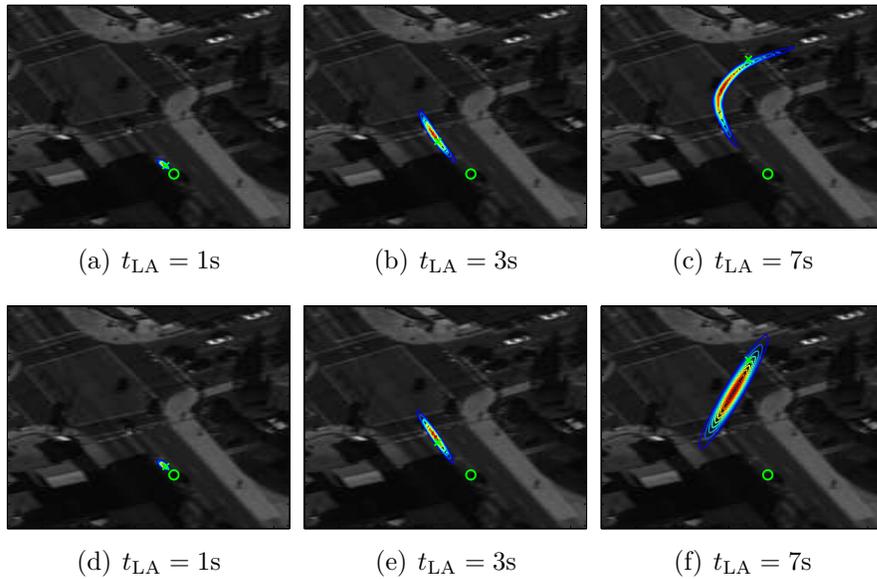


Figure 1.12: Anticipation of a real tracked vehicle using the hGMM (1.12(a),1.12(b), and 1.12(c)) and a single Gaussian predictor (1.12(d),1.12(e), and 1.12(f)). The vehicle state at time t is shown as a green circle. The observed vehicle state at the lookahead time is shown as a green cross.

predicts the measurements about as well as the single Gaussian method, according to the LL metric in Figure 1.8. However, Figures 1.12(c) and 1.12(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 Figure 1.12(f), and is quantified by the EOTE metric which shows clearly superior performance of the hGMM algorithm.

1.4.3 MIT-Cornell Collision Example



Figure 1.13: MIT-Cornell collision in the 2007 DUC

A motivating example for the importance of this work 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 Figure 1.13. Further details on the collision are available in Fletcher et al. [12].

To demonstrate the efficacy of the hGMM anticipation algorithm at improving safety, the scenario is re-visited 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 1.4.1). At the same time, Skynet simulates it’s 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 Hardy and Campbell [13], over a horizon of 3 seconds at 10 Hz.

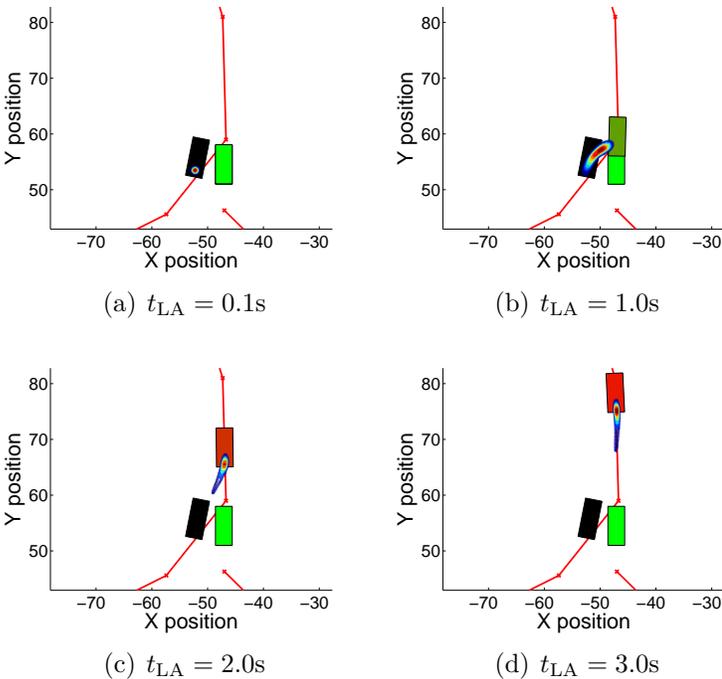


Figure 1.14: hGMM anticipation algorithm applied to the MIT-Cornell Collision

Figure 1.14 shows the results of the anticipation as a function of the lookahead time. Figure 1.14(a) shows the initial condition – Talos is shown as the black car, Cornell is shown as the green car. Figures 1.14(b), 1.14(c), and 1.14(d) show the

scenario as it evolves at three different lookahead times. The hGMM probability distribution prediction of Talos is plotted – in this case, the x-y 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). Figure 1.15 plots 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 seconds. These figures clearly show that even with a basic obstacle model, the hGMM could have predicted, and therefore prevented, the MIT-Cornell collision.

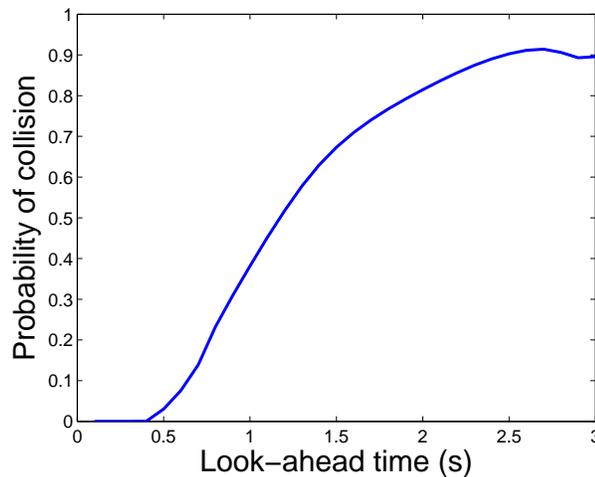


Figure 1.15: Anticipated probability of collision

Figure 1.16 plots the time required to predict an obstacle’s state forward 4.5 seconds 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 rackmount computer. The computation time grows with the number of allowed mixands and the inverse of the linearity threshold, as expected.

Figure 1.17 plots the prediction accuracy of the hGMM as a function of the

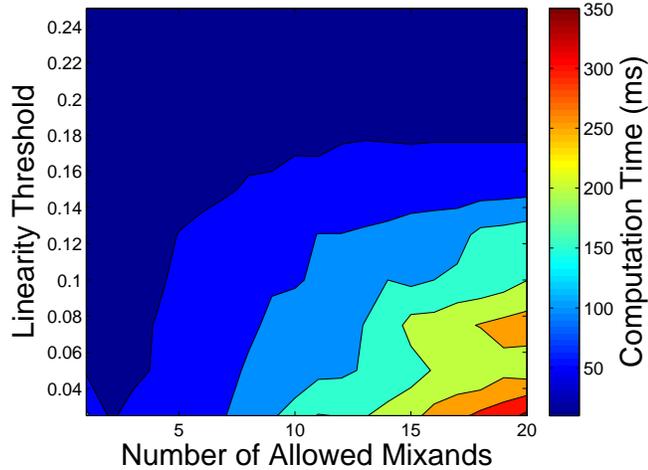


Figure 1.16: Computation time required to anticipate an obstacle to a 4.5 second horizon as a function of linearity threshold and maximum number of allowed mixands (enforced by Runnall’s reduction algorithm) [1]

linearity threshold and the maximum number of allowed mixands. Here the prediction accuracy is measured by computing the Log-Likelihood between the hGMM result and a large particle set propagated using the same vehicle dynamics model as the hGMM. Figure 1.17 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 Figures 1.16 and 1.17 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 seconds at greater than 3 Hz. Because the anticipation result is never older than 1/3 seconds, but extends to 4.5 seconds in the future, this is considered real-time performance.

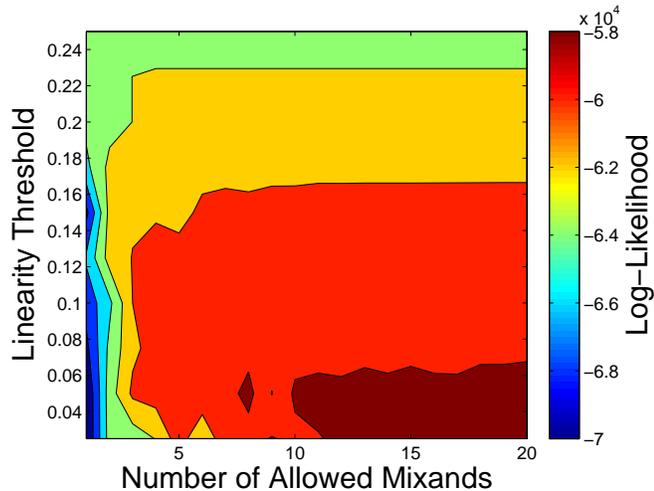


Figure 1.17: hGMM prediction accuracy as a function of linearity threshold and maximum number of allowed mixands

1.5 Conclusion

An anticipation algorithm is developed which uniquely recognizes and mitigates the impact of non-linearities 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 non-linear 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 non-linearities, the proposed algorithm is shown to have significant improvements in accuracy over standard propagation methods, such as the Unscented Transform.

The behavior of the non-linearity detection and the mixand splitting algorithms are explored using several simulated example problems, and compared to 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 accu-

racy of the anticipated probability distributions is compared across choices for the linearity threshold parameter in the hGMM anticipation algorithm, and compared to 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, defined as the negative log-likelihood between the particle set and the anticipated distribution.

The hGMM anticipation algorithm is validated on an experimental data set. Specifically, the hGMM algorithm is used to predict the behavior of vehicles in the CLIF 2007 data set provided by the USAF. The hGMM is compared to 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 log-likelihood 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 expected off-track error 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.

CHAPTER 2
PROBABILISTIC ANTICIPATION USING GAUSSIAN PROCESS
MODELS

This work was done in equal collaboration with Jason Hardy

2.1 Introduction

Gaussian Process (GP) regression modeling is a popular machine learning technique that is well suited for modeling complex, highly uncertain system behaviors such as those of intelligent agents, advanced controllers, or poorly understood system dynamics. GP regression modeling has been used to learn multi-step behaviors in reinforcement learning [52], time series forecasting [53], and human motion modeling [54].

A key challenge in performing multiple-step prediction using GP regression models is evaluating the GP over a distribution of input states. [55] show that a GP dynamics model can be used in standard Bayesian filter formulations including an Extended Kalman Filter (EKF), an Unscented Kalman Filter (UKF), and a particle filter. However, the GP Bayesian filtering methods presented in [55] approximate the propagated distribution by treating the GP as a point-to-point mapping, while evaluating the output uncertainty only at the mean of the input distribution. This can be an effective propagation strategy for sampling based filtering techniques such as the UKF and particle filter, but sampling over a GP is computationally challenging. [56] show that these approximations can be avoided for a certain class of kernel functions by analytically evaluating the first two moments of the output distribution. [57] also derive a closed form solution for the cross covariance between the GP input and output distributions for a general class of GPs. This cross covariance information is necessary for applying measurement

updates in a Bayesian filtering framework [58] and for capturing state dependent behaviors where feedback control is prevalent, such as lane keeping tendencies for human driven road vehicles.

Approximating the output distribution of a GP evaluated over an input distribution using the first and second moments can account for some degree of nonlinearity as long as the output distribution is well modeled by a Gaussian distribution[56]. When the output distribution is not well modeled by a Gaussian distribution, this approach produces a poor approximation of the true output distribution. Non-Gaussianity in the output distribution can arise due to complex nonlinearities in the GP data, such as multi-modal system behaviors. This paper proposes that if the nonlinearities in the GP mapping are smooth, the true output distribution can be better approximated by decomposing the input distribution into a Gaussian Mixture Model (GMM), where each mixture element has a smaller covariance than the original input distribution. This GMM approach has been applied to general nonlinear prediction and estimation problems [39, 59, 31, 60]. Here, the adaptive GMM approach is used in the problem of estimating the output of a GP regression model evaluated over an input distribution.

This paper presents an adaptive GMM GP prediction algorithm, referred to as the GP-aGMM algorithm, for performing multi-step prediction of highly nonlinear GP regression models. This algorithm extends the analytical GP prediction methods in [56], allowing for accurate prediction of highly nonlinear and multi-modal GP models. To provide insight into the behavior of the GP-aGMM algorithm, two metrics for splitting the input distribution are presented: 1) an analytical derivation of excess kurtosis of the GP output distribution which measures the non-Gaussianity, and 2) the error for a weighted least squares regression model which measures the local nonlinearity in the GP mapping. The prediction accu-

racy and computation of the two metrics are compared and discussed later in the paper. An on-the-fly data selection procedure is also developed that allows the analytical higher order moments to be computed using only a small subset of the most relevant data from the GP model. This procedure is independent of, and can be applied in addition to, global sparse GP strategies, such as [61], which seek to reduce the entire GP data set to a smaller representative data set.

The GP-aGMM algorithm can be applied to any problem that requires multi-step prediction of systems modeled by GPs. A motivating example used in this paper is anticipating the motion of a dynamic obstacle with an intelligent controller such as a road vehicle driven by a human or autonomous controller. Anticipating dynamic obstacle motion is a key enabling step in planning and collision avoidance systems for mobile robots in dynamic environments. Producing accurate motion predictions for complex goal-oriented dynamic obstacles such as road vehicles requires modeling obstacle intentions as well as the physical dynamics of the obstacle’s motion. Obstacle predictions based on inferred obstacle goals have been applied to mobile robots [18], autonomous road vehicles [22] [62], and air traffic control systems [63]. GP regression has been used before to model the motion of road vehicles. [64] and [65] both use GP regression to model the continuous state derivatives of a vehicle’s motion given the current vehicle state; [66] sample from a GP regression model, formed from the observed trajectories of traffic vehicles, to perform collision risk assessment.

The rest of the paper is organized as follows. Section 2.2 provides a brief overview of GP regression models. Section 2.3 details an analytical solution for the moments of a GP output evaluated over an input distribution as well as for the cross covariance terms between the GP output and input distributions; the cross covariance terms are critical to the proposed application of modeling a system with

feedback control. Section 2.4 presents a novel method for on-the-fly selection of the GP data set to improve computational efficiency. Section 2.5 presents the GP-aGMM algorithm for nonlinear and multimodal systems, including the two metrics for evaluating when to adapt the GMM. Section 2.6 provides prediction results for a bimodal one-dimensional function to gain insight into the performance of the GP-aGMM, and experimental results for a two stage driver-vehicle model trained using data collected from human subjects demonstrating empirical validity.

2.2 Gaussian Process Overview

A GP regression model is a probabilistic regression model that represents a distribution over functional mappings from an input space to an output space, where all outputs are jointly distributed as Gaussian. For a set of training data $\mathcal{D} = \{\mathcal{D}_x, \mathcal{D}_y\}$, where $\mathcal{D}_x = \{x_i\}_{i=1}^N$ is the set of input data points and $\mathcal{D}_y = \{y_i\}_{i=1}^N$ is the set of output data points, a GP regression model provides a distribution over the function $y_i = g(x_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$. A GP, given its data, is fully specified by its mean, $M(x)$, and covariance, $C(x_i, x_j)$, functions. For this paper, the standard zero mean function is used, $M(x) = 0$, and the covariance function is an autoregressive kernel function using the popular squared exponential form:

$$C(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(x_i - x_j)^T \Lambda^{-1}(x_i - x_j)\right) + \sigma_n^2 \cdot \delta(x_i, x_j) \quad (2.1)$$

where Λ is a diagonal weighting matrix, and σ_f^2 , Λ , and σ_n^2 are all covariance function parameters that are fit by maximizing the likelihood of the training data [67].

For a GP regression model with known parameters, the predictive distribution

at an arbitrary point in the input space, x^*

$$p(g(x^*)|x^*) \sim \mathcal{N}(\mu_y(x^*), \sigma_y^2(x^*))$$

is characterized by its mean and covariance:

$$\mu_y(x^*) = \mathbf{k}^{*T} \mathbf{K}^{-1} \mathbf{y}, \quad \sigma_y^2(x^*) = \mathbf{k}^{**} - \mathbf{k}^{*T} \mathbf{K}^{-1} \mathbf{k}^* \quad (2.2)$$

where $\mathbf{k}^* = [C(x^*, x_1), \dots, C(x^*, x_N)]^T$ represents the covariance function evaluated between the input point and each data point; $\mathbf{K}_{ij} = C(x_i, x_j)$ represents the covariance function evaluated for each pair of data points; $\mathbf{k}^{**} = C(x^*, x^*)$ is the covariance function evaluated between the input point and itself; and \mathbf{y} is a vector containing the output data points.

2.3 Analytical Evaluation of GPs at Uncertain Inputs

Multi-step prediction of a system model using GP regression requires evaluating the GP over an input distribution instead of an input point. [56] show that the first and second moments of the GP output given a normally distributed input have exact, closed-form solutions for certain Gaussian kernel functions, including the squared exponential kernel function given in Equation 2.1. Accurate prediction using GP regression models may also require computing the cross covariance between the input and output of the GP in order to capture autocorrelation or to define the full joint state distribution for systems where the GP models only one aspect of a hierarchical system's behavior. Subsections 2.3.1 and 2.3.2 provide an overview of the analytical moment evaluations for the mean, the output covariance, and the input-output cross covariance.

2.3.1 GP Mean and Covariance at Uncertain Input

If the GP input x^* is normally distributed such that $x^* \sim \mathcal{N}(\mu_x, \Sigma_x)$, the GP evaluation in Equation 2.2 becomes instead:

$$p(g(x^*)|\mu_x, \Sigma_x) = \int p(g(x^*)|x^*) \cdot \mathcal{N}(x^*|\mu_x, \Sigma_x) dx^* \quad (2.3)$$

This integral is intractable and can not be solved in closed form. However, the mean and covariance can be exactly calculated in closed-form, and the GP output distribution can be approximated by a normal distribution with the same mean and covariance as the true distribution:

$$p(g(x^*)|\mu_x, \Sigma_x) \approx \mathcal{N}(m(\mu_x, \Sigma_x), \text{var}(\mu_x, \Sigma_x))$$

where

$$\begin{aligned} m(\mu_x, \Sigma_x) &= E_{x^*} [E_{g(x^*)} [g(x^*)|x^*]] \\ &= E_{x^*} [\mu_y(x^*)] \\ \text{var}(\mu_x, \Sigma_x) &= E_{x^*} [\text{var}_{g(x^*)}(g(x^*)|x^*)] + \\ &\quad \text{var}_{x^*}(E_{g(x^*)} [g(x^*)|x^*]) \\ &= E_{x^*} [\sigma_y^2(x^*)] + \text{var}_{x^*}(\mu_y(x^*)) \end{aligned} \quad (2.4)$$

Quinonero-Candela et. al. provide exact, closed form expressions for the mean and covariance [57].

2.3.2 GP Input-Output Covariance

The full joint distribution of the GP input and output distributions is:

$$\begin{bmatrix} x^* \\ g(x^*) \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \mu_x \\ \text{m}(\mu_x, \Sigma_x) \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{xy}^T & \text{var}(\mu_x, \Sigma_x) \end{bmatrix} \right) \quad (2.5)$$

and the GP input-output covariance is defined as:

$$\begin{aligned} \Sigma_{xy} = \text{cov}(g(x^*), x^*) &= E_{x^*} \left[E_{g(x^*)} [g(x^*)|x^*] x^* \right] \\ &\quad - E_{x^*} \left[E_{g(x^*)} [g(x^*)|x^*] \right] E_{x^*} [x^*] \end{aligned} \quad (2.6)$$

As with the mean and covariance (Equation 2.4), a closed-form expression for the input-output covariance is available in [57]. This cross covariance term is critical for a variety of important applications, including: performing measurement updates using GP measurement models; capturing autocorrelation in time-series models; modeling a system with a feedback loop; and modeling systems using a GP composed with a parametric model. The later case is attractive for modeling systems that can be partially described by a parametric model, but include difficult-to-model processes that are more easily captured by a data driven GP. An example is a vehicle driven by a human or autonomous controller, where the vehicle dynamics are well-modeled by a parametric model, but the driver or autonomous controller is sufficiently complex that collecting data and using a GP to capture the closed loop driving behaviors is easier than attempting to construct a parametric model. In the proposed application, the cross-covariance is used to capture the correlation between the actuation commands of a human driver and the state of the vehicle being driven.

2.4 On-the-Fly Data Selection

Sections 2.3.1 and 2.3.2 describe an exact method for finding the mean, variance, and input-output cross covariance for a GP evaluated over a normally distributed input. Assuming that \mathbf{K}^{-1} is precomputed before performing any prediction, the computational complexity of the analytical calculation of the output variance is $\mathcal{O}(N^2)$, where N is the number of data points in the GP. This scaling with the square of the size of the data set means that evaluating the analytical solution can be much more computationally expensive than numerical methods such as the sigma point transform used in the UKF.

Existing GP compression approaches, such as [68], [69], and others summarized in [70], seek to reduce computation by replacing the training data set with a much smaller one that still captures the system behavior (thereby reducing N), and are performed as pre-processing steps offline. The proposed method here operates online, and works by using only the n most relevant points, where $n \ll N$, from the training data to calculate higher moments. The proposed method here could be used online on a GP that had already been compressed offline, for even greater computational savings.

The mean $m(\mu_x, \Sigma_x)$ in Equation 2.4 is given by [57] as:

$$m(\mu_x, \Sigma_x) = \mathbf{y}^T \mathbf{K}^{-1} \mathbf{l}$$

where $\mathbf{l} = [l_1, \dots, l_N]^T$ is given by:

$$l_i = \int C(x^*, x_i) \mathcal{N}(x^* | \mu_x, \Sigma_x) dx^* \quad (2.7)$$

The vector $\mathbf{l} = [l_1, \dots, l_N]^T$ is similar to \mathbf{k}^* in the standard GP evaluation in Equation 2.2, as both are vectors that measure the relevance each data point to the

input distribution or input point respectfully. In many systems, only a small fraction of the GP data is relevant to a given input. This implies that the output covariance can be well approximated using only the n most relevant data points, where $n \ll N$. This data selection step requires inverting the reduced data covariance matrix, $\mathbf{K}_{\text{red}}^{-1}$, which is $\mathcal{O}(n^3)$; the covariance calculation then scales as $\mathcal{O}(N + n^3)$ instead of $\mathcal{O}(N^2)$.

The selection of n is application-specific, and is discussed for the example applications in Section 2.6.

2.5 Adaptive Gaussian Mixture Formulation

A key component of the GP-aGMM algorithm is the ability to adaptively split the input distributions into appropriately sized GMMs, if needed. The two necessary elements for performing adaptive splitting are 1) a method for detecting when splitting is necessary, and 2) a vector along which to split the input distribution. Once the need for a split is detected, and a splitting direction is known, the input distribution can be normalized and a precomputed GMM approximation can be applied as in [31]. Section 2.5.1 presents a closed form analytical kurtosis metric for detecting non-Gaussianity in the GP output. Section 2.5.2 presents a numerical metric for detecting Local Linearity of the GP mapping. For both metrics, the eigenvector associated with the largest eigenvalue of the input covariance is chosen as the splitting direction as in [71]; this selection splits the Gaussian along the direction of its largest covariance. The two presented methods offer different trades between computation and accuracy, as described in the results of Section 2.6.

2.5.1 Analytical Kurtosis Evaluation for Non-Gaussianity Detection

The kurtosis of a distribution is an established metric of non-Gaussianity [72]. The excess kurtosis of a distribution is defined as

$$\text{kurt}(y) = \frac{E[y^4]}{(E[y^2])^2} - 3$$

where the excess kurtosis for a Gaussian distribution is zero. Typically, other metrics of non-Gaussianity such as negentropy are preferable due to concerns about sensitivity to outliers when computing the kurtosis based on a set of samples. However, the negentropy of a GP output cannot be evaluated in closed form while the excess kurtosis of the output distribution can be directly computed by analytically solving the required expectation integrals.

Using excess kurtosis as a measure of non-Gaussianity, splitting is performed whenever the magnitude of the excess kurtosis of the GP output surpasses a predefined threshold. The Appendix provides a derivation for analytically calculating the excess kurtosis of the GP output distribution. Note that the kurtosis calculation scales as $\mathcal{O}(N^4)$, which prevents computationally efficient application of the kurtosis metric for GP models with more than a small number of data points. The on-the-fly data selection method presented in Section 2.4 enables the kurtosis metric to be computed efficiently using only a small subset of the most relevant training data points, resulting in a scaling of $\mathcal{O}(N + n^4)$.

Selecting a threshold for the maximum allowed excess kurtosis, k_{\max} above which to split an input distribution depends on the application, available computing resources, and required performance. More specifically, this threshold directly trades accuracy with computation – small values of k_{\max} yield the most accurate

performance, but also cause the number of mixands in the GP-aGMM to become very large. Sections 2.6.1 and 2.6.3 study the selection of k_{\max} in detail for two example applications.

2.5.2 Numerically Evaluated Local Linearity

An alternative metric for determining when splitting is required to ensure accurate propagation is the linearity of the GP local to the input distribution. Local Linearity is defined here as the residual error of the least-squares fit to the GP data, weighted by relevance to the input distribution; this metric is similar to metrics that have been applied to adaptive GMM sigma-point filters [60, 31, 59]. The linearity metric for an input distribution $\mathcal{N}(\mu_x, \Sigma_x)$ is defined as:

$$\begin{aligned}
 e &= \|(\mathbf{y} - \mathbf{A}^* \hat{\mathbf{x}}) \cdot \mathbf{w}\| \\
 \hat{\mathbf{x}} &= \left[\begin{bmatrix} x_1 \\ 1 \end{bmatrix}, \dots, \begin{bmatrix} x_N \\ 1 \end{bmatrix} \right] \\
 \mathbf{w} &= \text{diag}([l_1^{1/2}, \dots, l_N^{1/2}])
 \end{aligned} \tag{2.8}$$

where $\mathbf{y} = [y_1, \dots, y_N]$ is the GP output data, and \mathbf{A}^* is the least-squares optimal affine fit to the data, weighted by the relevance of each data point to the input query. The relevance of a data point is taken here to be \mathbf{l} , given in Equation 2.7. The matrix \mathbf{A}^* can be found efficiently using the pseudo-inverse:

$$\mathbf{A}^* = (\mathbf{y} \cdot \mathbf{w} \cdot \mathbf{w} \cdot \hat{\mathbf{x}}) \cdot (\hat{\mathbf{x}} \cdot \mathbf{w} \cdot \mathbf{w} \cdot \hat{\mathbf{x}})^{-1} \tag{2.9}$$

The Local Linearity metric is a direct measure of how affine the GP is local to the input distribution. If $e = 0$, the GP data is exactly affine local to the input

distribution, and the GP output is therefore well-described by a single Gaussian; therefore, no splitting is required. As e becomes larger, the GP is nonlinear near the input distribution, and the GP output poorly approximates the process by a single Gaussian; thus, splitting is required.

As with the excess kurtosis metric, selecting a threshold e_{\max} above which to split an input distribution is application dependant, and is studied in Sections 2.6.1 and 2.6.3.

2.5.3 GP-aGMM

Summarizing, the GP-aGMM prediction algorithm accurately propagates a GMM through a dynamics model defined fully or partially by GP models. Such models are applicable to any system with dynamics that are either poorly understood, or computationally expensive to model, but for which data is available. The GP-aGMM can be applied to systems with dynamics purely modeled by GPs, as in Section 2.6.1, or to systems with dynamics that are modeled as GPs composed with parametric models, as in Section 2.6.2. Algorithm `refalg:ch2:overall` describes the GP-aGMM prediction algorithm for the latter case; for the former case, the algorithm is similar, but does not include the parametric model (steps 11-40). Algorithm `refalg:ch2:overall` uses sigma points to propagate the distribution through the parametric model for generality [26], but in cases where the parametric model is easily differentiable, the first-order Taylor Series approximation could also be used (as in the Extended Kalman Filter).

Algorithm 1 The GP-aGMM prediction algorithm

Require: $\mathbf{x}_k \sim \sum_{i=1}^{N_k} w_k^i \cdot \mathcal{N}(\mu_k^i, \Sigma_k^i)$

Require: $\mathbf{x}_{k+1} = f(\mathbf{x}_k, [g_1(\mathbf{x}_k), \dots, g_M(\mathbf{x}_k)]^T)$ where f is a parametric function and g_1 through g_M are GPs

```

1: for  $i = 1 \rightarrow N_k$  do
2:   Evaluate the GPs over the  $i^{\text{th}}$  mixand:
3:   for  $j = 1 \rightarrow M$  do
4:      $[\mu_j, \sigma_j^2, \Sigma_{xj}, k] \leftarrow g_j(\mu_k^i, \Sigma_k^i)$ 
5:     or
6:      $[\mu_j, \sigma_j^2, \Sigma_{xj}, e] \leftarrow g_j(\mu_k^i, \Sigma_k^i)$ 
7:     if  $e > e_{\max}$  or  $k > k_{\max}$  then
8:       Split  $w_k^i \cdot \mathcal{N}(\mu_k^i, \Sigma_k^i)$  and go to 2
9:     end if
10:  end for
11:   $\boldsymbol{\mu}_u \leftarrow [\mu_1, \dots, \mu_M]^T$ 
12:   $\Sigma_u \leftarrow \text{diag}([\sigma_1^2, \dots, \sigma_M^2])$ 
13:   $\Sigma_{xu} \leftarrow [\Sigma_{x1}, \dots, \Sigma_{xM}]$ 
14:  Perform the sigma point propagation:
15:  Find  $\mathbf{S}_x = [\mathbf{S}_x^1, \dots, \mathbf{S}_x^{n_x}]$  such that  $\mathbf{S}_x \cdot \mathbf{S}_x^T = \Sigma_x^i$ 
16:  Find  $\mathbf{S}_u = [\mathbf{S}_u^1, \dots, \mathbf{S}_u^M]$  such that  $\mathbf{S}_u \cdot \mathbf{S}_u^T = \Sigma_u$ 
17:  Given sigma point spread parameter  $\lambda$ 
18:  for  $l = 0 \rightarrow 2n_x + 2M$  do
19:    if  $l \in [0, 2n_x + 1, \dots, 2n_x + 2n_u]$  then
20:       $\chi_k^l \leftarrow \mu_x$ 
21:    else if  $l \in [1, \dots, n_x]$  then
22:       $\chi_k^l \leftarrow \mu_x + \lambda \cdot \mathbf{S}_x^l$ 
23:    else
24:       $\chi_k^l \leftarrow \mu_x - \lambda \cdot \mathbf{S}_x^{l-n_x}$ 
25:    end if
26:    if  $l \in [0, \dots, 2n_x]$  then
27:       $\nu_k^l \leftarrow \boldsymbol{\mu}_u + \Sigma_{xu}^T \cdot \Sigma_x^{-1} \cdot (\chi_k^l - \mu_k)$ 
28:    else if  $l \in [2n_x + 1, \dots, 2n_x + M]$  then
29:       $\nu_k^l \leftarrow \boldsymbol{\mu}_u + \lambda \cdot \mathbf{S}_u^{l-2n_x}$ 
30:    else
31:       $\nu_k^l \leftarrow \boldsymbol{\mu}_u - \lambda \cdot \mathbf{S}_u^{l-2n_x-M}$ 
32:    end if
33:    Evaluate the parametric model
34:     $\bar{\chi}_{k+1}^l \leftarrow f(\chi_k^l, \nu_k^l)$ 
35:  end for
36:  Find first two moments of propagated state
37:  Given sigma point weights  $[w_m^1, \dots, w_m^{2n_x+2n_u}]$  and  $[w_c^0, \dots, w_c^{2n_x+2n_u}]$ 
38:   $\mu_{k+1} \leftarrow \sum_l w_m^l \cdot \bar{\chi}_{k+1}^l$ 
39:   $\Sigma_{k+1} \leftarrow \sum_l w_c^l \cdot (\bar{\chi}_{k+1}^l - \mu_{k+1}) \cdot (\bar{\chi}_{k+1}^l - \mu_{k+1})^T$ 
40:   $w_{k+1}^i \leftarrow w_k^i$ 
41: end for
42:  $\mathbf{x}_{k+1} \sim \sum_{i=1}^{N_{k+1}} w_{k+1}^i \cdot \mathcal{N}(\mu_{k+1}^i, \Sigma_{k+1}^i)$ 

```

2.6 GP-aGMM Probabilistic Prediction Examples

Two examples are used to evaluate prediction accuracy and computation using the GP-aGMM algorithm. Section 2.6.1 presents prediction results for the Univariate Nonstationary Growth Model (UNGM); the UNGM is a bimodal one dimensional system that is commonly used as a benchmark problem in nonlinear estimation [49, 50]. Section 2.6.2 presents a two stage prediction model for vehicle behavior at road intersections where GP regression is used to model driver behavior and a parametric dynamics model is used to model vehicle motion. Section 2.6.3 presents prediction results for this driver-vehicle model with driving behavior GPs trained using driving behavior data collected from human volunteers in a driving simulation.

2.6.1 Univariate Nonstationary Growth Model

A prediction example using the Univariate Nonstationary Growth Model (UNGM) is presented here to provide clarity and insight into how the GP-aGMM algorithm works on a system with clearly observable bimodal behavior. The UNGM also serves to demonstrate the performance of the GP-aGMM using an established benchmark problem. The UNGM is defined as:

$$y = f(x) = \alpha x + \beta \frac{x}{1+x^2} + \gamma \cos(1.2(t-1)) \quad (2.10)$$

For this paper, the following parameter values are used: $\alpha = 1.5, \beta = 15, \gamma = 8, t = 1$.

Figure 2.1(a) shows the UNGM with noise, $y = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, which maps a Gaussian distribution over the input space into a bimodal distribution in the output space. In Figure 2.1(b), samples from this noisy UNGM model

are used to train a GP regression model and the GP output is approximated using the analytical mean and covariance given in Equation 2.4; this is the ‘no splitting’ case.

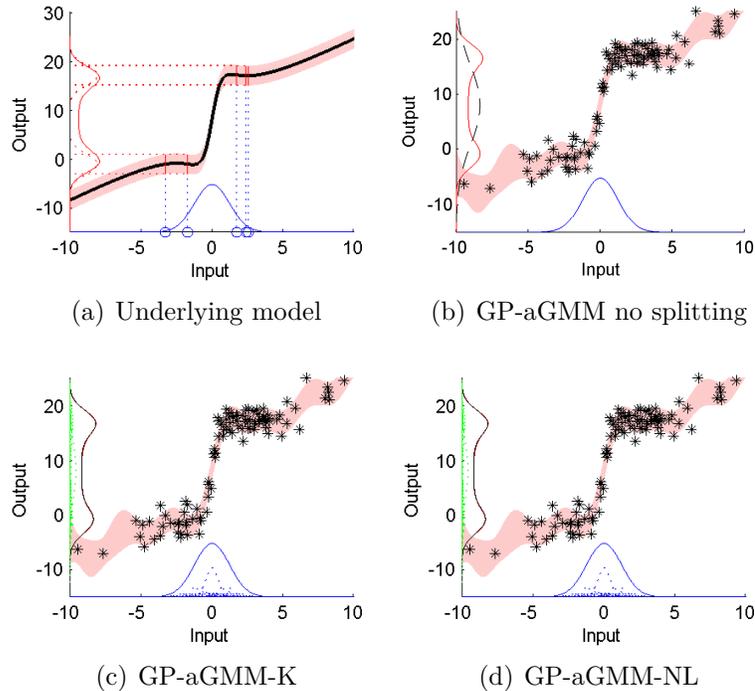


Figure 2.1: Analytical prediction of a GP model trained from a simulated noisy UNGM model. The true output distribution is shown as a red line. The predicted distribution for the different methods is shown as a black dashed line. Adaptive splitting in (c) and (d) is based on the analytical kurtosis (k) metric and the numerical local linearity (NL) metric. A kurtosis threshold of $k_{\max} = 0.2$ and a local linearity threshold of $e_{\max} = 3$ are used.

The two adaptive splitting approaches outlined in Section 2.5 are applied using a kurtosis threshold of $k_{\max} = 0.2$ and a local linearity threshold of $e_{\max} = 3$. When an unacceptable level of non-Gaussianity or nonlinearity is detected, the input distribution is split into a mixture of three component distributions. This process is repeated recursively until all output distributions satisfy the splitting threshold or until a predefined recursion depth is reached. The recursion depth for this example is $n_{\max}^{\text{rec}} = 4$.

Predictions using adaptive splitting shown in Figures 2.1(c) for the kurtosis metric and 2.1(d) for the numerical local linearity metric both produce predictions that closely match the true output distribution (shown in red). The Kullback-Leibler divergence (KLD) between the predicted distribution and the true output distribution is $KLD_{\text{kurt}} = 0.022$ for the adaptive splitting approach using the kurtosis splitting metric, $KLD_{\text{NL}} = 0.015$ for the adaptive splitting approach using the numerical local linearity metric, and $KLD_{\text{no_split}} = 0.300$ for the non-splitting approach. The adaptive splitting approaches required 65 and 79 mixture elements for the kurtosis and local linearity based splitting metrics respectively. The KLD results indicate that both splitting approaches produce a much better estimate of the true output solution as compared to the non-splitting approach, but do so at the cost of requiring additional mixands.

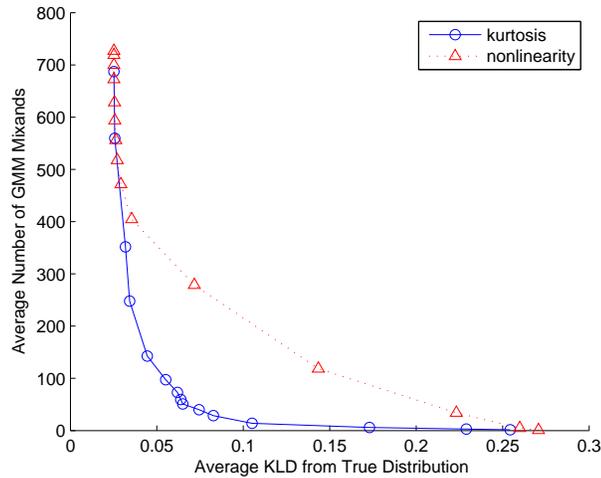


Figure 2.2: Average KLD from true predicted distribution for $n_{\text{sim}} = 100$ random input distributions propagated through the UNGM function.

The propagation results in Figure 2.1 show the performance of the GP-aGMM-K and GP-aGMM-NL algorithms for the case of a highly bimodal predicted distribution with a limited recursion depth of $n_{\text{max}}^{\text{rec}} = 4$. To evaluate the gen-

eral performance of proposed GP-aGMM approach and to compare the splitting efficiency of the kurtosis and nonlinearity based splitting metrics, the simulation was repeated for $n_{\text{sim}} = 100$ randomly sampled Gaussian input distributions using a large range of splitting threshold values and a higher recursion depth of $n_{\text{max}}^{\text{rec}} = 8$. Randomly sampled input distributions ensure that splitting is not always necessary or uniform and the increased recursion depth provides a better comparison of the two splitting metrics by removing artificial means of controlling splitting efficiency. The kurtosis splitting metric was tested for the values $k_{\text{max}} = \{5, 4, 3, 2, 1, 0.75, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.01, 0.001\}$ and the nonlinearity splitting metric was tested for the values $e_{\text{max}} = \{5, 4.75, 4.5, 4.25, 4, 3.75, 3.5, 3.25, 3, 2.75, 2.5, 2.25, 2, 1.75, 1.5\}$.

Figure 2.2 presents the average KLD from the true predicted distribution as a function of the average number of GMM mixands for each of the tested threshold values. These results show that the kurtosis splitting metric is able to provide a better estimate of the true predicted distribution while requiring far fewer GMM mixands over the tested range of splitting threshold values. The performance of the two splitting metrics converges at the extremes of the splitting threshold ranges when either no splitting occurs or when splitting is performed exhaustively, limited only by the recursion depth.

2.6.2 Two Stage Driver-Vehicle Model

To evaluate the performance of the GP-aGMM algorithm in a practical application, the algorithm is applied to the case of predicting the combination of driving behavior and vehicle motion of a road vehicle at a four way intersection. The underlying driver-vehicle model used for prediction consists of two stages, where the first stage models the driver decision making process and the second stage models

the vehicle dynamics, as shown in Figure 2.3. In the first stage, the driver takes as inputs information about the environment, the state of the vehicle, and internal driving goals and outputs a set of vehicle commands controlling the steering and forward acceleration of the vehicle. This stage represents a complex, highly uncertain decision making process and is modeled using GP regression. In the second stage, the driver commands are applied to a model of the vehicle dynamics and the vehicle state is propagated forward in time. This stage represents a known physical process and is well modeled using a parametric model.

A common approach when using GP regression to model complex systems with known underlying dynamics is to include the dynamics in the mean function used by the GP [73]. While this approach allows a GP regression model to track the underlying dynamics of a system, it does not allow for the inclusion of parametric constraints on the system dynamics. Implicit linear and quadratic constraints can be enforced on the mean of the GP output distribution [74], however, these mean-only constraints are insufficient for ensuring that a predicted distribution obeys the underlying dynamics of the system. Including system dynamics through the mean function is also a poor approximation for hierarchical systems in which there exists a clean separation between the states modeled by the GP and the states used in the known dynamics model.

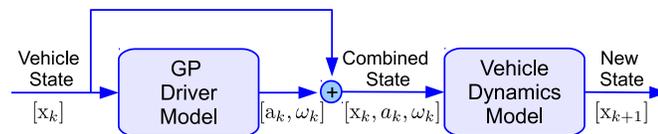


Figure 2.3: Block diagram of the two stage driver-vehicle model consisting of a GP driver behavior model combined with a parametric vehicle dynamics model.

The vehicle state description used in this model is a four dimensional state vector which includes the two dimensional position of the center of vehicle’s back

axle along with the vehicle’s two dimensional velocity vector:

$$\mathbf{x}_k = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T \quad (2.11)$$

At the current timestep k , the vehicle state \mathbf{x}_k is assumed to be normally distributed with a known mean and covariance, $\mathbf{x}_k \sim \mathcal{N}(\mu_{\mathbf{x}}, \Sigma_{\mathbf{x}})_k$. The driver behavior model consists of two independently trained GP regression models for throttle commands expressed as forward acceleration, a_k , and steering commands expressed as turn rate, ω_k :

$$a_k \sim \text{GP}_a(\mathbf{x}_k, \mathcal{D}_x, \mathcal{D}_a) \quad (2.12)$$

$$\omega_k \sim \text{GP}_\omega(\mathbf{x}_k, \mathcal{D}_x, \mathcal{D}_\omega) \quad (2.13)$$

where \mathcal{D}_x is the set of vehicle states used as input data in the GP regression model, and \mathcal{D}_a and \mathcal{D}_ω are the set of output data points corresponding to forward acceleration commands and steering commands respectfully.

The driving commands, (a, ω) are used as control inputs in the vehicle dynamics model, as shown in Figure 2.3. Parametric models for the dynamics of road vehicles are well studied; for this paper, the vehicle dynamics are represented by a constant curvature, constant acceleration bicycle model. [75] show that despite its simplicity, this model is as effective as more sophisticated vehicle dynamics models for predicting vehicle motion. The vehicle state dynamics are modeled as:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, a_k, \omega_k, \delta t) \quad (2.14)$$

This equation can be rewritten with the function $f(\mathbf{x}_k, a_k, \omega_k, \delta t)$ explicitly defined as:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ v_{x,k+1} \\ v_{y,k+1} \end{bmatrix} = \begin{bmatrix} x_k + \Delta x \cos(\theta_k) - \Delta y \sin(\theta_k) \\ y_k + \Delta y \sin(\theta_k) + \Delta x \cos(\theta_k) \\ (v_k + a_k \delta t) \cos(\theta + \Delta\theta) \\ (v_k + a_k \delta t) \sin(\theta + \Delta\theta) \end{bmatrix} \quad (2.15)$$

where

$$\begin{aligned}\theta &= \tan\left(\frac{v_{y,k}}{v_{x,k}}\right) \\ v_k &= \sqrt{v_{y,k}^2 + v_{x,k}^2} \\ \rho_k &= \frac{v_k}{\omega_k} \\ d_k &= \frac{1}{2}a_k\delta t^2 + v_k\delta t \\ \Delta\theta &= d_k \frac{1}{\rho_k} \\ \Delta x &= \rho_k \sin(\Delta\theta) \\ \Delta y &= \rho_k(1 - \cos(\Delta\theta))\end{aligned}$$

and δt is the prediction timestep. The intermediate value ρ_k represents the radius of constant curvature for the vehicle and the intermediate value d_k represents the distance the vehicle travels over δt .

Given the combination of a GP regression model of the human driver, and a parametric model of the physics of the vehicle, the vehicle state is predicted forward in time using Algorithm

refalg:ch2:overall.

2.6.3 Experimental Evaluation of Two Stage Driver-Vehicle Model

A four way road intersection scenario is used to evaluate the GP-aGMM algorithm using the two stage driver-vehicle model. Driving behavior data from three human test subjects was collected using the open source driving simulation TORCS [76]. Each test subject was asked to perform 10 runs each for four distinct intersection traversal maneuvers: turn left, turn right, drive straight, and stop-at-line, for a total of 40 trials per subject. Position and velocity data was recorded for each

maneuver at a rate of 5 Hz. Acceleration and steering commands were computed using the inverse of the constant curvature constant acceleration vehicle dynamics model specified in Equation 2.14.

Section 2.6.3 presents prediction results for the unimodal turn left, turn right, and drive straight driving maneuvers using the analytical propagation portion of the GP-aGMM algorithm, but with no adaptive splitting. The intent of this study is to evaluate the performance of the analytical propagation step with and without on-the-fly data selection, compared to prediction results for benchmark GP Bayesian propagation strategies studied in [55]. Section 2.6.3 presents prediction results for multimodal combined maneuvers using the GP-aGMM algorithm with adaptive splitting, and compares the kurtosis and numerical nonlinearity splitting thresholds over a range of threshold values. Section 2.6.3 presents computational scaling results for the on-the-fly data selection procedure outlined in Section 2.4.

Benchmark Comparison for Unimodal Maneuvers

Open loop prediction was performed over a prediction horizon of 20 timesteps to evaluate the propagation accuracy for the turn left, turn right, and drive straight maneuvers. Prediction trials were conducted using a leave-one-out cross validation approach, where the initial state estimate of the left out trajectory is used as the initial state distribution for prediction. The results in this section compare the propagation performance of four algorithms: 1) the GP-aGMM algorithm with no adaptive splitting and no data reduction; 2) the GP-aGMM with no adaptive splitting using on-the-fly data selection with a reduced data size of $n = 30$; 3) the GP-EKF algorithm presented in [55]; 4) a sigma point algorithm, similar to the GP-UKF presented in [55], where each sigma point is propagated through the entire two stage driver-vehicle model and the covariance of the GP outputs is evaluated only at the mean of the input distribution. An open loop Monte Carlo

prediction was also run using 1000 samples to provide a close approximation of the true predicted distribution. The performance of each algorithm is evaluated by computing the KLD between the predicted distribution at each step and the normal distribution given by the first two moments of the Monte Carlo result.

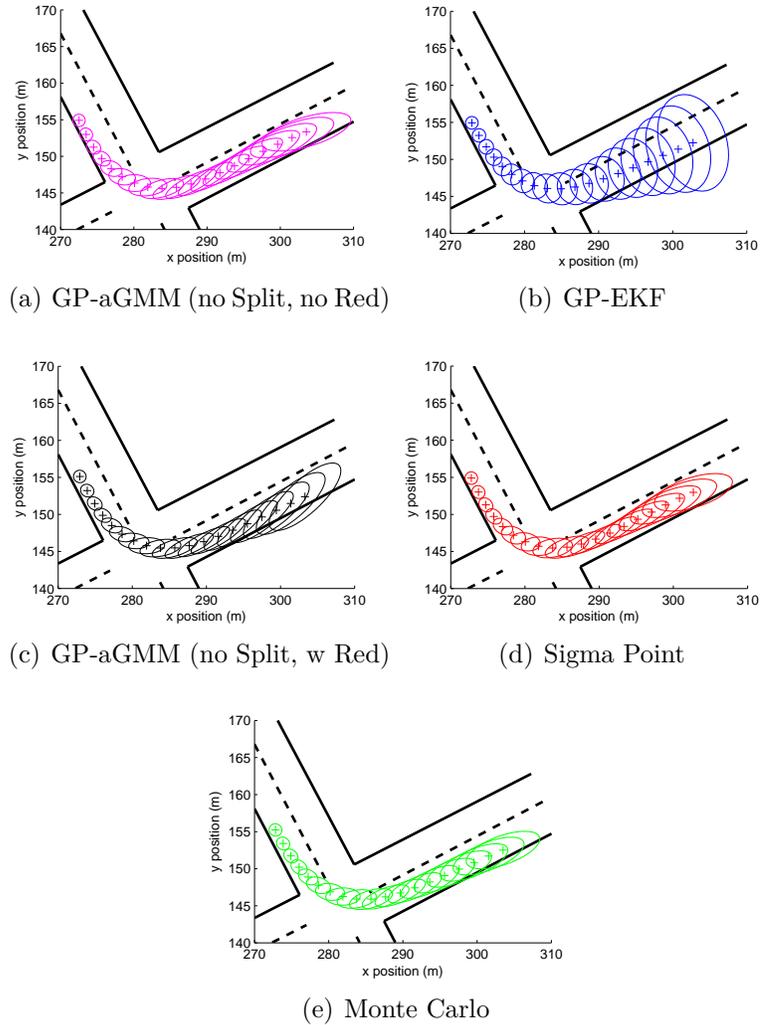


Figure 2.4: Turn left prediction results for each algorithm over 20 timesteps. Ellipses represent the prediction error uncertainty at 50% confidence intervals.

Figure 2.4 shows typical predicted vehicle trajectory distributions for each of the tested algorithms using GP models trained on the turn left maneuver data. The prediction results for the analytical GP-aGMM with on-the-fly compression

shown in Figure 2.4(c) are qualitatively very similar to the results for the GP-aGMM without compression shown in Figure 2.4(a). Both are also similar to the results for the sigma point approach shown in Figure 2.4(d) and the Monte Carlo solution shown in Figure 2.4(e). These results suggest that the analytical approach with cross covariance information and the sigma point approach are both capable of capturing lane keeping behavior and are both good approximations of the Monte Carlo result. Figure 2.4(b) shows the predicted vehicle trajectory using the GP-EKF. The prediction results for the GP-EKF fail to capture the lane-keeping behavior of the vehicle because the GP-EKF does not account for the cross covariance information between the input and output of the GP driver model. More specifically, the GP-EKF assumes that the driver control actions are independent of the vehicle state for a given timestep. As a result, state-dependent control behaviors, such as lane keeping and other closed loop decision making, are absent from the prediction. The on-the-fly compression uses $n = 30$, and this selection is discussed in detail in Section 2.6.3.

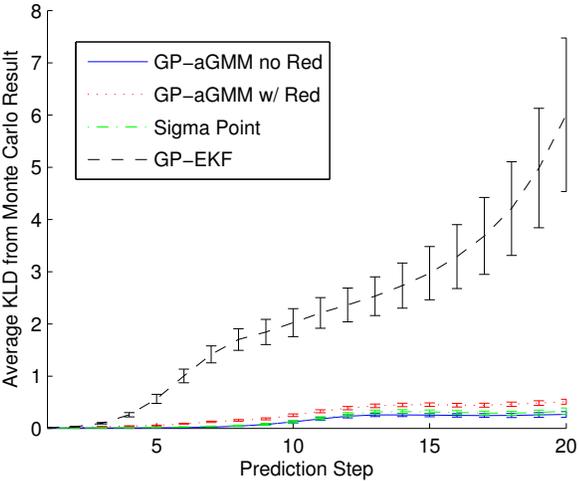


Figure 2.5: Average KLD from the Monte Carlo solution for the turn left maneuver.

Figure 2.5 plots the average KLD of each prediction method as compared to the

Monte Carlo result (an approximation of truth) as a function of timestep for the turn left maneuver. As expected, the GP-aGMM results with and without data selection and the sigma point prediction are all good approximations of the Monte Carlo result over the entire range of prediction timesteps. The use of the on-the-fly data selection procedure causes no significant loss in prediction accuracy, while offering significant improvements in computation time. This trade off between accuracy and computation for the data selection procedure is presented in more detail in Section 2.6.3. The GP-EKF approach performs much worse at later prediction timesteps when the lack of lane keeping causes large errors in the shape of the vehicle state distribution. This result matches the qualitative analysis of the typical trajectories in Figure 2.4. The results for the right turn maneuver showed similar trends and therefore plots have been omitted.

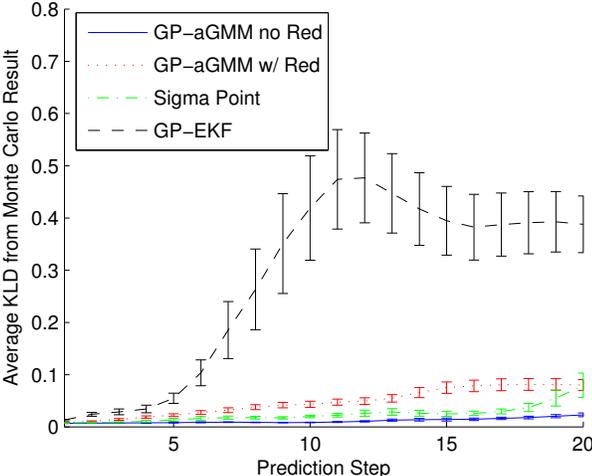


Figure 2.6: Average KLD from the Monte Carlo solution for the drive straight maneuver. Note: the axes for this figure have been rescaled compared to the left turn maneuver in Figure 2.5 for clarity.

Figure 2.6 shows the KLD results for the drive straight maneuver for each of the predictors. The variations are much smaller (as compared with the left turn maneuver in Figure 2.5) across the entire prediction window for all predictors,

Table 2.1: Computation time statistics (sec)

Algorithm	Left Turn		Straight		Right Turn	
	μ_{comp}	σ_{comp}	μ_{comp}	σ_{comp}	μ_{comp}	σ_{comp}
GP-aGMM w Red	0.109	0.003	0.111	0.015	0.114	0.013
GP-aGMM no Red	7.382	0.036	7.375	0.054	7.422	0.056
GP-EKF	0.420	0.034	0.464	0.027	0.460	0.074
Sigma Point	1.051	0.015	1.072	0.020	1.074	0.016

although the GP-EKF is still the largest of the set. This suggests that all prediction algorithms closely match the Monte Carlo result for the drive straight maneuver. This is primarily due to the fact that the drive straight maneuver requires little to no lane-keeping, so the behavior is largely absent from the training set. During the turn left and turn right maneuvers, the driver must make constant steering corrections to stay in the correct driving lane. During the drive straight maneuver, drivers can align the vehicle with the driving lane and traverse the intersection with negligible additional steering input. This absence of steering correction information results in a GP steering model which in all cases predicts a zero mean noisy steering output.

Table

reftab:ch2:comptimesGP provides computation time statistics for each algorithm for a single prediction step. Predictions were run in Matlab on a Core 2 Duo mobile processor running Windows 7. These results show that the GP-aGMM evaluation with on-the-fly data selection is nearly an order of magnitude faster than the Sigma Point algorithm and nearly two orders of magnitude faster than the GP-aGMM evaluation without data selection for all three maneuvers. The GP-aGMM with data selection is also faster than the GP-EKF due to the extra computation required to find the Jacobian matrix at each prediction step.

Comparison of Adaptive Splitting Metrics for Prediction of Multimodal Behavior

Identifying and enumerating distinct driving behaviors from the raw data can be challenging and time consuming. As a result, training data for a specific maneuver model might contain elements of multiple basic maneuver types. One common situation in which complex multimodal driving behavior occurs is when a driver is occasionally forced to stop and wait before executing a turn at an intersection. This behavior is a compound behavior that includes both a distinct stopping maneuver and a distinct turning maneuver.

To evaluate the performance of the GP-aGMM algorithm for combined driving maneuvers, GP models for acceleration and turn rate were trained using training data from both stop-at-line and turn right or turn left maneuvers. Open loop prediction was then performed over 20 timesteps using the same leave-one-out cross validation approach as in Section 2.6.3. This prediction was performed for a range of maximum kurtosis splitting threshold values, $k_{\max} \in \{2, 1, 0.75, 0.5, 0.3, 0.1\}$, and nonlinearity splitting threshold values, $e_{\max} \in \{6, 5, 4, 3, 2, 1\}$. In an effort to retain a tractable number of mixands, the recursion depth is limited to $n_{\max}^{\text{rec}} = 2$, and Gaussian mixture reduction is performed at the end of each prediction step using the mixture reduction approach proposed in [1]. This mixture reduction phase reduces the mixture to either a maximum of 15 elements, or until the Integral Squared Difference [48] between the reduced distribution and the original unreduced distribution falls below a predefined threshold, whichever yields the smallest number of mixture elements. The ISD threshold used here is $\text{ISD}_{\max} = 1 \times 10^{-5}$.

Figure 2.7 shows the final predicted driver position distributions of the GP-aGMM-K algorithm for the $k_{\max} \in \{2, 0.75, 0.1\}$ cases, as well as the Monte Carlo prediction result using 1000 samples for a typical combined stop-at-line and turn

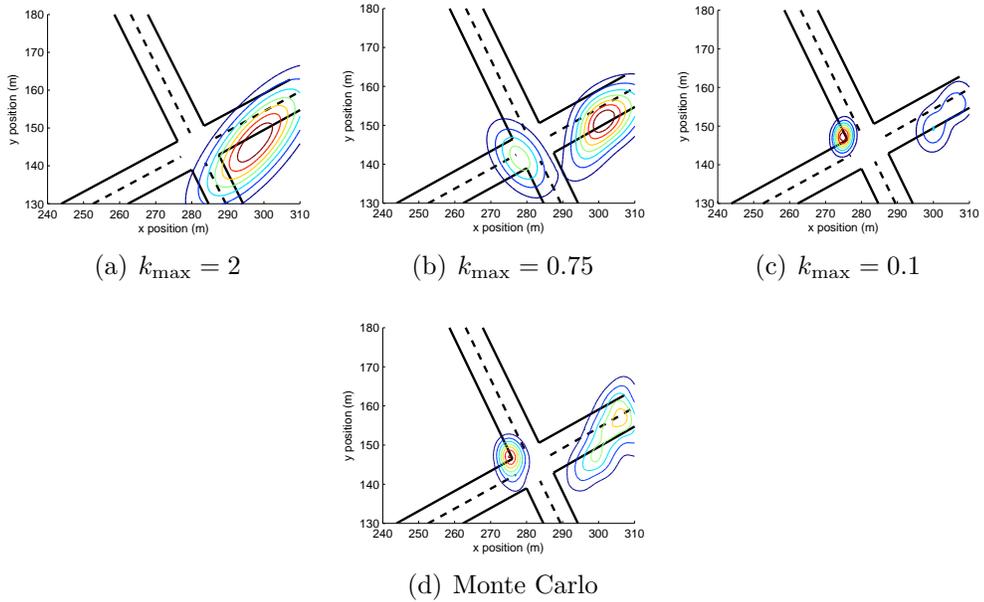


Figure 2.7: Predicted distribution of the GP-aGMM-K predictor at step 20 using $k_{\max} = 2$, $k_{\max} = 0.75$ and $k_{\max} = 0.1$.

left maneuver prediction trial. The $k_{\max} = 2$ case in Figure 2.7(a) experiences no adaptive splitting and is unable to capture the bimodal nature of the system as shown in the Monte Carlo result in Figure 2.7(d). The $k_{\max} = 0.75$ case in Figure 2.7(b) captures the bimodal behavior but is unable to closely approximate the shape of the true distribution. The $k_{\max} = 0.1$ case in Figure 2.7(c) captures the bimodal system behavior and provides a solution that closely approximates shape of the final distribution in Figure 2.7(d).

Figures 2.8–2.11 show the average KLD from the Monte Carlo result and the average number of mixture elements before reduction at each prediction step for the combined stop-at-line and turn left maneuver over the range of tested k_{\max} and e_{\max} values. In each figure, error bars represent one Standard Error of the mean, $SE = \frac{\sigma}{\sqrt{n_{\text{trial}}}}$. The KLD between the Monte Carlo sample set and the GP-aGMM-K solution is computed using a Monte Carlo estimate, $D_{KL}(P||Q) \approx$

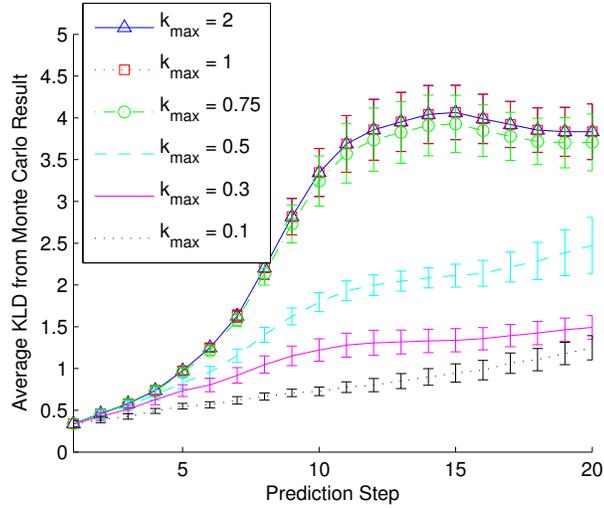


Figure 2.8: Average KLD from the Monte Carlo result at prediction step 20 for combined stop-at-line and turn left maneuvers using the GP-aGMM-K algorithm with the kurtosis splitting metric.

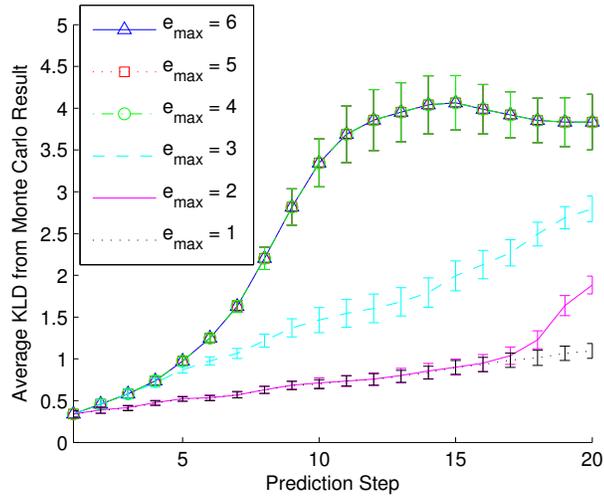


Figure 2.9: Average KLD from the Monte Carlo result at prediction step 20 for combined stop-at-line and turn left maneuvers using the GP-aGMM-NL algorithm with the nonlinearity splitting metric.

$\sum_i \log(P(x_i)/Q(x_i))$, where $P(x_i)$ is approximated using Gaussian kernel density estimation. The $k_{\max} = \{2, 1\}$ and $e_{\max} = \{6, 5, 4\}$ cases experience no adaptive splitting and provide the poorest prediction performance. Prediction performance

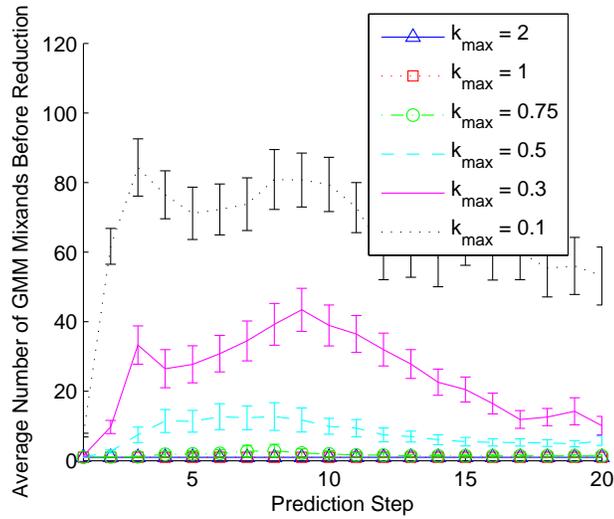


Figure 2.10: Average number of mixture elements before reduction for combined stop-at-line and turn left maneuvers using the GP-aGMM-K algorithm with the kurtosis splitting metric.

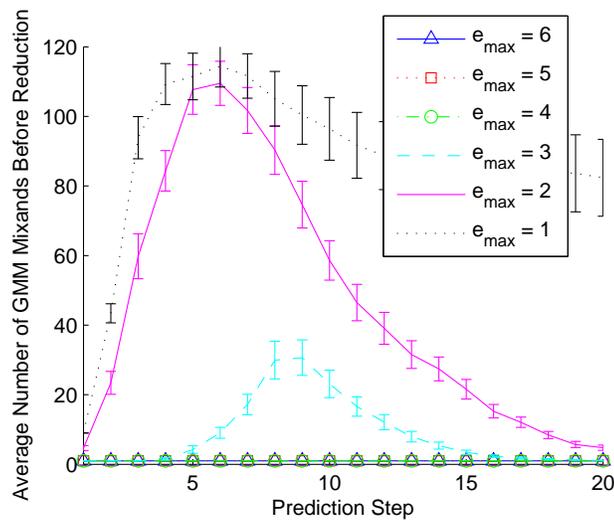


Figure 2.11: Average number of mixture elements before reduction for combined stop-at-line and turn left maneuvers using the GP-aGMM-NL algorithm with the nonlinearity splitting metric.

increases as the splitting threshold is lowered, with a minimum KLD provided by the $k_{\max} = 0.1$ and $e_{\max} = 1$ cases.

Figures 2.10 and 2.11 show the number of mixture elements for each case as a

function of the prediction step. Predictably, the number of mixands peaks when the predicted vehicle distribution is midway through the turn, which also correlates with the largest nonlinearities in the system. This improvement in prediction performance comes at the cost of increased computational complexity, with the $k_{\max} = 0.1$ and $e_{\max} = 1$ cases requiring a significantly higher number of mixture elements than the $k_{\max} = 0.3$ and $e_{\max} = 2$ cases for a comparably small improvement in prediction performance. Similar trends were obtained for the stop-at-line and turn right combined maneuver and result plots for this case have been omitted.

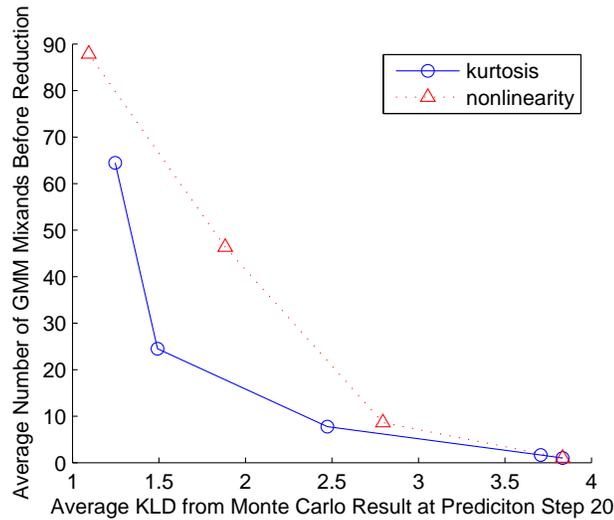


Figure 2.12: Average KLD from Monte Carlo result at prediction step 20 as a function of the average number of mixture elements before reduction for combined stop-at-line and turn left maneuvers.

Figure 2.12 shows a comparison of the KLD of both the GP-aGMM-K and GP-aGMM-NL algorithms as compared to the Monte Carlo result at the final prediction step as a function of the average number of mixands used in propagation. The data points in this plot correspond to the tested splitting threshold values shown in Figures 2.8–2.11. The $k_{\max} = \{2, 1\}$ and $e_{\max} = \{6, 5, 4\}$ cases experience no adaptive splitting and all map to the same point on the far right side of the

figure. The remaining data points suggest that the kurtosis splitting metric is able to make more efficient splitting decisions than the nonlinearity splitting metric, allowing for better prediction performance using a smaller number of mixands.

Computational Scaling with Data Selection

The on-the-fly data selection method proposed in Section 2.4 is studied for several maneuvers. Figure 2.13 presents the KLD statistics of the GP-aGMM solution for the pure left turn maneuver, without using splitting, as a function of the GP compression size n , where $N = 300$. The trend is as expected: for small values of n , performance is poor, but for $n > 30$, there is no performance difference from the GP-aGMM solution not using compression. Performance is similar for the straight and right-turn maneuvers.

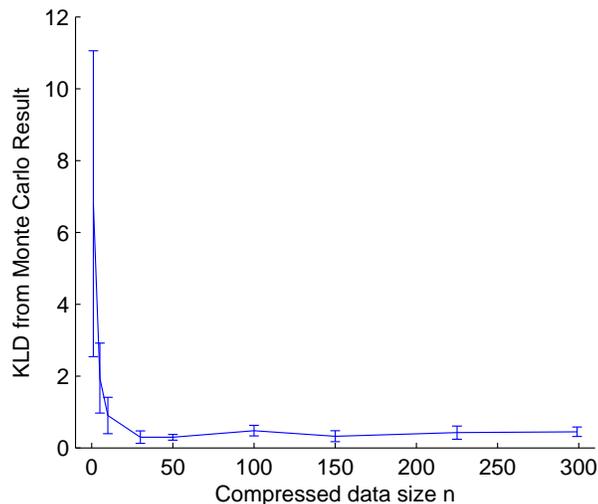


Figure 2.13: KLD between the GP-aGMM solution and the Monte Carlo solution as a function of the on-the-fly data selection size n for the left turn maneuver, without using splitting.

Figure 2.14 plots the computation time for each prediction step as a function of the compressed data size n . For the range of data used in this paper, the prediction computation time scales approximately with n^2 , which reflects the computational

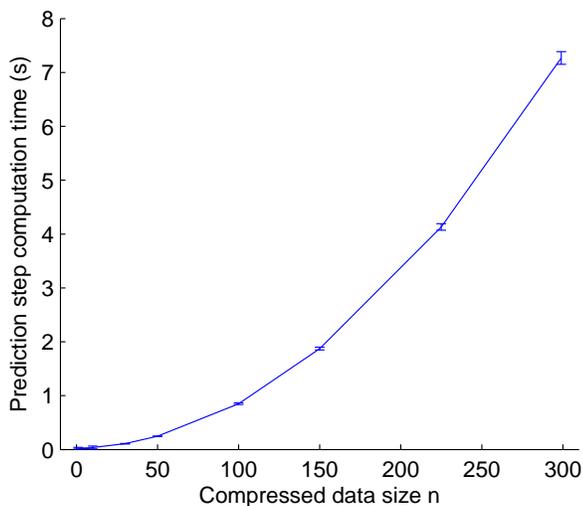


Figure 2.14: Computation time per prediction step as a function of the on-the-fly data selection size n for the left turn maneuver, without using splitting

complexity of the covariance calculation. For very high values of n , this scaling is strongly influenced by the $\mathcal{O}(n^3)$ matrix inverse required to compute $\mathbf{K}_{\text{red}}^{-1}$.

Figures 2.13 and 2.14 show that there is a large range of values for n such that computation time is greatly reduced without sacrificing propagation accuracy. For the value of $n = 30$, the impact of the data selection on propagation accuracy is negligible while the computation time is reduced by nearly two orders of magnitude. The selection of n is problem dependant, but a study comparing the output of a GP using compression to the output not using compression, as a function of n , provides insight on how to select an appropriate value for a given GP model.

Figure 2.15 presents the kurtosis computation time as a function of the on-the-fly data selection size n . Note that n only ranges to 30 out of 300 in this figure. The $\mathcal{O}(n^4)$ scaling of the kurtosis calculation makes its application to the full GP data set intractable. On-the-fly data selection allows the excess kurtosis to be efficiently estimated using only a small set of the most relevant data points. The adaptive GMM trials in Section 2.6.3 used a data selection size of $n = 10$ to compute the

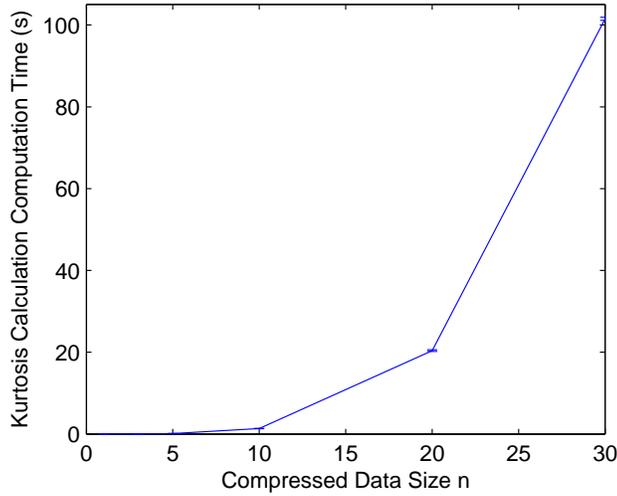


Figure 2.15: Computation time of the excess kurtosis calculation as a function of the on-the-fly data selection size n .

excess kurtosis splitting metric. In comparison, the average computation time for the GP-aGMM-NL using the nonlinearity metric was 0.01 ± 0.002 seconds using all 300 data points. The computation time when performing predictions using the nonlinearity splitting metric is dominated by the moment evaluations of the mixand propagation step, as shown in Figure 2.14. For both the driver-vehicle system and the benchmark UNGM presented in Section 2.6.1, the kurtosis splitting metric performs better at determining when to split mixands to accurately propagate the state distribution. However, for both of these systems, the computational cost of evaluating the kurtosis metric, even with aggressive on-the-fly data compression, is prohibitive. Selecting which splitting metric to use is a function of application. The kurtosis metric is better for cases where computation time is not critical or for the special case where the GP portion of the prediction model is not the dominant computational cost. For example, if the vehicle dynamics stage of the two stage driver-vehicle model shown in Figure 2.3 required significant computation time per GMM mixand it would be beneficial to use the excess kurtosis metric to make

the most efficient use of the available GMM mixands. The nonlinearity metric is typically the best metric to use for cases where computation time is a concern, such as a real-time implementation.

2.7 Conclusion

This paper presents an adaptive Gaussian mixture model (GMM) formulation for performing multi-step probabilistic prediction using a Gaussian Process (GP) regression model with underlying nonlinearities, possibly composed with a parametric model. The presented adaptive GMM formulation relies on existing work to analytically evaluate GPs over an input distribution, while accounting for the correlation between inputs and outputs of the GP as proposed by [56]. A novel on-the-fly data selection method is also presented which greatly reduces the required computation time of the analytical GP moment calculations. Two adaptive splitting metrics are presented based on detecting non-Gaussianity in the GP output and on detecting nonlinearity in the GP mapping; each allow a Gaussian mixture to be adaptively refined to better capture nonlinearities and multimodal system behavior in the GP training data.

The adaptive GP GMM formulation (GP-aGMM) performance is studied using the benchmark univariate nonstationary growth model (UNGM). Results show that the GP-aGMM output closely approximates truth using both splitting metrics. The results also indicate that the kurtosis metric uses GMM mixands more efficiently, but is expensive enough that the nonlinearity metric allows for faster predictions.

The GP-aGMM is also demonstrated on the problem of predicting the motion of a multiple maneuver driver-vehicle model for a four-way road intersection. For unimodal vehicle maneuvers, prediction using analytical moment evaluations with

on-the-fly data selection is shown to provide propagation accuracy similar to sigma point propagation, but with significant computational savings. For multimodal maneuvers, such as the case where the observed driver behavior includes both stopping and turning, the GP-aGMM allows for accurate predictions that capture both distinct driving behavior modes. As with the UNGM, experimental results indicate that the kurtosis based splitting metric allows for more efficient use of GMM mixands, however, the nonlinearity based splitting metric is significantly easier to compute and allows for faster prediction times despite requiring more mixands to achieve the same level of prediction accuracy.

CHAPTER 3
LEARNING GAUSSIAN PROCESS MODELS IN LANE FEATURE
SPACES

3.1 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 [2, 3, 4]. 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 hours) [5, 6, 7]. Continued development in the field has led to autonomous cars beginning to drive in real urban environments alongside civilian traffic [8, 9, 10, 11].

Early autonomous cars used primarily reactionary planners that relied on rapid re-planning in order to respond to the dynamic environments in which they operated [5, 77]. A collision between the MIT and Cornell entries was one of several examples in the 2007 DUC that raised safety concerns about reactionary planning for autonomous driving [12]. Recently, ‘anticipation’, or the explicit reasoning about how the dynamic environment will evolve in the future, has been identified as one of the enabling technologies for safer autonomous vehicles [59, 78].

Anticipation, in the context of autonomous driving, can be broken down into two problems: first, how to capture the complex behavior of an object like a human-driven car in a tractable model that can be used to make predictions; and second, how to effectively predict the state of a tracked object forward in time. The prediction problem is well-studied in the literature. Chapter 1 explores the literature and presents a novel algorithm for making probabilistic predictions of

tracked object motion.

The first component of anticipation – building an accurate and tractable model of human-driven vehicle – has received less attention in the literature. A natural simplification is to decompose the human-driven vehicle model into separate driver and vehicle models. A relatively simple parametric model, such as the four-state bicycle model, can accurately capture the dynamics of a tracked object [75]. The human driver is much more difficult to capture in a simple model [79, 80, 81]. One approach is to assume a simple parametric form of the human driver, such as a lane-keeping and speed-keeping controller [31]. The accuracy of a parametric model can be improved by fitting the parameters to human driver data [59]. However, a parametric controller can only capture driver behavior that is explicitly modeled by the controller [82].

The Gaussian Process (GP) is nonparametric regression model that is well suited for capturing complex, highly uncertain systems such as human drivers. GP regression models have been used to model the behavior of road vehicles (vehicle and driver dynamics together) [64, 66, 65]. Hardy et al. [82] propose using a GP regression model to capture the driver, composed with a parametric model to capture the vehicle dynamics. In each of these approaches, the GP must be trained on a dataset that includes all of the expected state-space of the vehicle. Practically, this means that the GP dataset must include examples of drivers operating on every lane of every road of the operating environment. While quite accurate for modeling the behavior of drivers on the roads which the model has training data for, this becomes intractable due to both the difficulty of capturing a dataset that includes training examples from all roads and conditions and the computational cost of a GP regression with such a large dataset.

This work proposes using a lane-feature space, describing the relationship be-

tween the current vehicle state and the local environment, to learn the driver model. By learning a GP driver model in a lane feature space, the model generalizes outside of its training data, i.e. the GP model can be used to anticipate the behavior of a driver in other environments outside of the training set; thus, the training data no longer requires data from the complete operating environment. This generalization enables the practical use of GP driver models in large environments.

This chapter is organized as follows. Section 3.2 describes the decomposition of the tracked vehicle dynamics model into a vehicle model and a driver model, and the Gaussian Process regression models used to capture driver behavior. Section 3.3 introduces the data set used to validate the GP models developed in this work. Section 3.4 presents the lane feature space proposed by this work, including the various features used, and presents a method for automatically selecting the most relevant features. Section 3.5 presents the lane feature GP driver model anticipation algorithm (LFGPAA). Section 3.6 studies the LFGPA algorithm performance using a real human driver data set.

3.2 Two-Stage Driver Vehicle Modeling

The dynamics model of a human-driven vehicle is naturally decomposed into a hierarchical model in which the driver and vehicle dynamics are modeled with two different methods, but jointly connected. The decomposed model used in this chapter is very similar to the model presented in Chapter 2.6.2, but uses a different parameterization of the vehicle state and model. Figure 3.1 shows the model decomposition. For the vehicle dynamics, a relatively simple parametric model is efficient and accurate [75]. The driver dynamics, on the other hand, are complex and not fully understood, and thus difficult to capture with a physics based model.

The driver is a feedback controller, so capturing the correlation between input and output of the driver model is key to making accurate probabilistic predictions of the driver-vehicle system. In addition, while the driver dynamics can be approximated using a parametric model fit to data, any behavior not explicitly included in the model is lost [59]. Adding complexity to an assumed parametric model can improve predictive power, but because anticipation typically requires many hundreds of model evaluations per second, this approach does not scale well. Using a GP driver model allows arbitrarily complex driver behavior to be modeled, so long as observations of that behavior are included in the the training data set. Fast approximate GP regression algorithms such as the SPGP make GP regression models attractive for computational reasons as well.

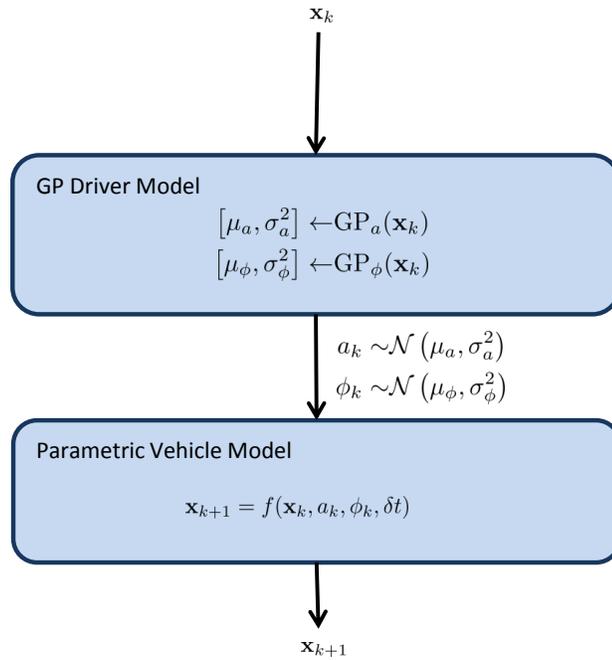


Figure 3.1: The two stage driver/vehicle model.

3.2.1 Parametric Vehicle Model

The parametric vehicle model used in this work is the four-state constant curvature, constant acceleration bicycle model. The vehicle state (\mathbf{x}_k) at time k has four states: the two dimensional position of the center of the rear axle (x_k and y_k), the heading of the vehicle (θ_k), and the speed of the vehicle (v_k). This discrete-time model assumes the vehicle acceleration (a_k) and the vehicle steering angle (ϕ_k) to be constant during time steps. The predictive state model of the vehicle is given by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, a_k, \phi_k, \delta t) \quad (3.1)$$

where $f(\mathbf{x}_k, a_k, \phi_k, \delta t)$ explicitly defined as:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + \Delta x \cos(\theta_k) - \Delta y \sin(\theta_k) \\ y_k + \Delta x \sin(\theta_k) + \Delta y \cos(\theta_k) \\ \theta_k + \Delta\theta \\ v_k + a_k \delta t \end{bmatrix} \quad (3.2)$$

and

$$\begin{aligned} \rho_k &= \frac{l}{\phi_k} \\ d_k &= \frac{1}{2} a_k \delta t^2 + v_k \delta t \\ \Delta\theta &= d_k \frac{1}{\rho_k} \\ \Delta x &= \rho_k \sin(\Delta\theta) \\ \Delta y &= \rho_k (1 - \cos(\Delta\theta)). \end{aligned}$$

and δt is the prediction time step and l is the length between the front and rear axles. The intermediate value ρ_k represents the radius of constant curvature for the vehicle, which is determined by the length between front and rear axles, l , and the

steering angle, ϕ_k . The intermediate value d_k represents the distance the vehicle travels over δt . The two control inputs in this model are the vehicle acceleration a_k and the vehicle steering angle ϕ_k . This model is simple, fast, and effective for predicting vehicle motion.

3.2.2 Gaussian Process Driver Model

For a detailed introduction to Gaussian Process regression models, refer to Chapter 2.2.

The vehicle model described in Equation 3.1 has two control inputs: a_k , the acceleration of the vehicle; and ϕ_k , the steering angle of the vehicle. Given a data set \mathcal{D} that pairs vehicle state observations $\mathcal{D}_x \sim \mathbf{x}_{k,i}$ with driver control action observations $\mathcal{D}_a \sim a_{k,i}$ and $\mathcal{D}_\phi \sim \phi_{k,i}$, a GP model is learned for each control input:

$$\begin{aligned} [\mu_{a,k}, \sigma_{a,k}^2] &= \text{GP}_a(\mathbf{x}_k) \\ [\mu_{\phi,k}, \sigma_{\phi,k}^2] &= \text{GP}_\phi(\mathbf{x}_k). \end{aligned} \tag{3.3}$$

In summary, the driver-vehicle model can be written as

$$\begin{aligned} a_k &\sim \text{GP}_a(\mathbf{x}_k) \\ \phi_k &\sim \text{GP}_\phi(\mathbf{x}_k) \\ \mathbf{x}_{k+1} &= f(\mathbf{x}_k, a_k, \phi_k, \delta t) \end{aligned} \tag{3.4}$$

where the GP's deliver distributions on the controls. This model is summarized in Figure 3.1. The GP driver model (Equation 3.3 and Figure 3.1) allows complex human driver behavior to be included in the overall human driven vehicle model

(Equation 3.4) in a general and tractable framework. Because the inputs to the GP driver model (Equation 3.3) are the vehicle states \mathbf{x}_k , the training data set must include data points from the entire environment in which the model is to be applied. For example, in order to perform vehicle anticipation with this driver-vehicle model in a city, the training set would need to include data samples from every road in that city. The logistical expense and technical challenges (training a GP is $\mathcal{O}(N^3)$ in the size of the data set) caused by such a large data set make this approach practical only on small scales. The remainder of this Chapter develops an alternate input space that generalizes the learned GP driver models, and thus eliminates the need for the data set to cover the entire environment.

3.2.3 Fast Approximate GPs

Due to the superlinear scaling of GPs with respect to the size of the dataset for both training and regression, faster approximate methods are very popular [68, 83, 84, 70]. In general, these methods involve identifying a subset of the dataset on which to perform inference. Snelson and Ghahramani [68] propose a method which approximates the dataset with pseudo-input points, rather than a subset of the dataset. The SPGP allows a GP to be well approximated using only a handful of pseudo-inputs, giving large reductions in computational cost. Quiñonero-Candela and Rasmussen [70] describe the SPGP and other approximate methods in detail.

Due to their computation advantages, the work here uses SPGPs to model human drivers. Additionally, SPGPs bypass a failure case for GPs with data sets that have a large number of input points that are nearly or exactly identical. In these cases, optimizing the likelihood of the training data can become numerically unstable, making training and regression impossible. The transformation of the

data into a lane feature space, described in Section 3.4, creates a data set with very tight clusters, and training a standard GP is not reliable. SPGPs can be reliably trained on these data sets, however, because the pseudo-inputs naturally spread to cover the input space, while still incorporating the full data set into training. Section 3.6.1 includes studies over the number of pseudo-inputs used by the SPGP driver models.

3.3 hciLab Driving Data Set

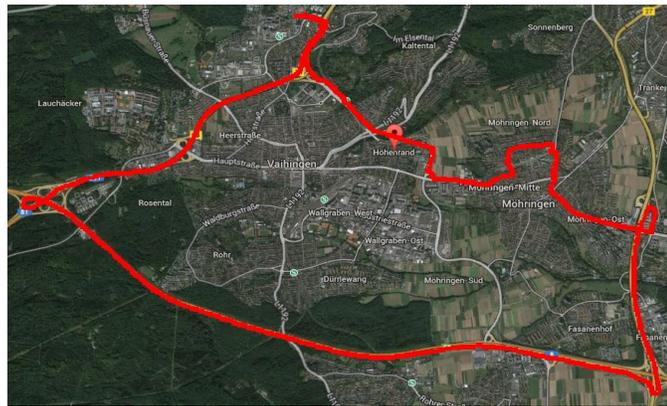


Figure 3.2: The route driven by subjects in the hciLab data set.

For training and validation, the human driver data set published by the hciLab at the University of Stuttgart is used. The hciLab data set consists of data for ten subjects driving in Stuttgart, Germany [85]. The data set was collected in order to evaluate driver load and stress in different driving conditions, and therefore consists largely of biometric data (for example, heart rate), in addition to GPS data. The route driven by the subjects is shown in Figure 3.2. The data set does not contain direct measurements of vehicle state, steering inputs, or throttle commands, but using the GPS data these values can be inferred. Inference is performed using an Unscented Kalman Smoother [86], an assumed vehicle model, and an assumed

measurement model. The assumed dynamics model of the vehicle is very similar to the parametric vehicle model used to perform anticipation, Equation 3.1, but here an additional state (ϕ_k) is included to model the steering angle as a red noise process instead of a white noise process, as there is no explicit driver model for this inference problem.

$$\mathbf{x}_{k+1}^A = f^A(\mathbf{x}_k^A, \mathbf{u}_k^A, \delta t) \quad (3.5)$$

where \mathbf{x}_k^A is the vehicle state augmented by the steering angle

$$\mathbf{x}_k^A = \begin{bmatrix} \mathbf{x}_k \\ \phi_k \end{bmatrix}.$$

The augmented dynamics function becomes

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \phi_{k+1} \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_k, a_k, \phi_k + \frac{1}{2}\Delta\phi_k, \delta t) \\ \phi_k + \Delta\phi_k \end{bmatrix} \quad (3.6)$$

where $\Delta\phi_k$ is the change in steering angle. The measurements used in inference are the GPS position and heading from the data set, in units of meters East, meters North, and radians from East. The measurement model is:

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{v}_k)$$

$$\begin{bmatrix} z_k^3 \\ z_k^3 \\ z_k^3 \end{bmatrix} = \begin{bmatrix} x_k + v_k^3 \\ y_k + v_k^3 \\ \theta_k + v_k^3 \end{bmatrix} \quad (3.7)$$

For inference, the time step used is either the time until the next GPS measurement from the data set, or 0.1sec, whichever is shorter. Processing the data set using these dynamics and measurement models results in smooth vehicle state estimates ($\hat{\mathbf{x}}_k$) paired with control inputs ($\hat{a}_k, \hat{\phi}_k$) which can then be used to learn the

driver GP models (Equation 3.3). Qualitatively, the processing results in smooth vehicle trajectories, intuitive estimates of driver actions, and vehicle trajectories in reasonable driving lanes when overlaid on aerial images of the city.

In order to transform the data into a lane feature space, the driving lane must also be known. Here, the driving lane is a series of way points defined by hand using both Google Maps and the GPS data from the data set. The combination of driving lane and vehicle state/control data input pairs yields the data required to train the proposed lane feature GP driver models (Section 3.4).

3.4 Human Driver Models in Lane Feature Space

The lane feature space proposed in this work is a feature space that generalizes the relationship between the vehicle’s state and the road. By learning a driver model GP in this space, instead of in the vehicle’s inertial space, the driver model GP can be applied driving in other environments, not just those where the training data set was collected.

In order for a driver model GP to be learned in a lane feature space, the lane feature space must be defined such that it contains informative features for both the steering and throttle behavior of the driver. Formally, a transformation is defined that takes the vehicle state \mathbf{x}_k in inertial space and returns a lane feature vector $\tilde{\mathbf{x}}_k$

$$\tilde{\mathbf{x}}_k = \mathcal{H}(\mathbf{x}_k, \mathcal{M}) \tag{3.8}$$

where \mathcal{M} is a map of the road environment, and \mathcal{H} transforms the vehicle state, using the map, into a set of lane features $\tilde{\mathbf{x}}_k$. A key part of this work is defining $\tilde{\mathbf{x}}_k$, and thus \mathcal{H} . Equation 3.8 shows the difference between the inertial state \mathbf{x}_k

and the lane feature space $\tilde{\mathbf{x}}_k$, which uses the map \mathcal{M} to generalize the vehicle state to a set of features that capture the local relationship between the vehicle and the driving lane.

The approach here, given the the non-physical characteristics, is to define a super-set of lane features, and through analysis identify which are most important to the anticipation problem. Intuitively, a human driver operating a vehicle on a road acts to remain in the lane of travel, and to maintain a speed that is both legal and safe. In order to capture the relationships between vehicle and road that inform these driver actions, the following classes of lane feature are defined (see Figure 3.3):

1. \mathbf{e}_t^y is the off-track error of the vehicle t seconds in the future, assuming heading and speed remain constant. Intuitively, this feature influences the steering behavior of the driver.
2. \mathbf{e}_t^θ is the heading error of the vehicle t seconds in the future, assuming heading and speed remain constant. This feature also influences the steering behavior of the driver.
3. κ_t is the unsigned route curvature ahead of the vehicle, computed to a distance of $t \cdot v_k$. This feature influences the throttle behavior of the driver.
4. Δv_t is the difference between the vehicle speed and the nominal speed of the road (taken as the average speed of vehicles on the road, but could also be the known speed limit of the road), calculated at the projected closest point on the road from the vehicle's origin, assuming the heading and speed remain constant. Intuitively, this feature captures driver throttle and brake behavior.

Figure 3.3 illustrates the different classes of lane features. Each lane feature is calculated at nine distinct times in the future, $t \in [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4]$ sec

resulting in a 36 dimensional lane feature space. In addition, the Δv_t feature is averaged over the nine time steps to yield a 37th feature. The vector of all potential lane features is defined:

$$\tilde{\mathbf{x}}_k = \begin{bmatrix} [\mathbf{e}_{0,0.5,1,1.5,2,2.5,3,3.5,4}^y]^T \\ [\mathbf{e}_{0,0.5,1,1.5,2,2.5,3,3.5,4}^\theta]^T \\ [\kappa_{0,0.5,1,1.5,2,2.5,3,3.5,4}]^T \\ [\Delta v_{0,0.5,1,1.5,2,2.5,3,3.5,4}]^T \\ \Delta v_{\text{avg}} \end{bmatrix} \quad (3.9)$$

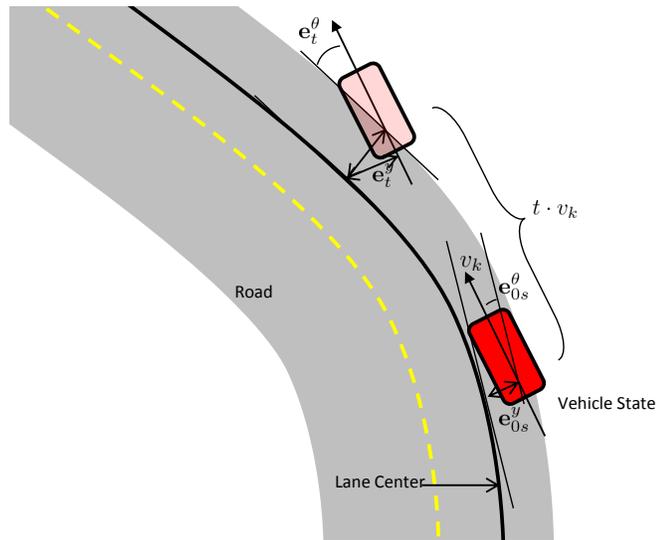


Figure 3.3: Illustration of lane features at the current vehicle state and the projected vehicle state t seconds in the future.

3.4.1 Lane Feature Selection for Anticipation

The full lane feature vector (Equation 3.9) is simply a list of potential features; using all 37 is both computationally infeasible and potentially numerically unstable because the problem is over parametrized. Because many of the features are highly correlated with each other (for example, \mathbf{e}_{0s}^y and $\mathbf{e}_{0.5s}^y$), there is a large amount

of redundancy in the lane feature space which can lead to over fitting. For least-squares regression, including a regularizing term in the cost function that penalizes the L_1 or L_2 norm of the weights is a common strategy to prevent such over fitting [87].

Here, the least angle regression (LARS) algorithm presented by Fraley and Hesterberg [88] is used to identify the best, most informative subsets of lane features. LARS is an iterative algorithm that solves the regularized least squares problem by iteratively selecting the most relevant features first. LARS starts with no features in the active set, and adds (or removes) features one at a time until the full solution (using all the features) is reached.

An iteration of the LARS algorithm scores the features by how correlated they are to the residual errors of the solution at the previous iteration. The feature most highly correlated with the residual error is included in the active set, and least-squares is used to find the optimal linear solution. If a feature in the active set is found to have very small weights after the least-squares solution, it is removed from the active set. The algorithm iterates until the full set of features is included in the active set.

The optimal subset of $N < 37$ lane features for use by the lane feature GP driver models is selected by using the same features that the best intermediate LARS solution with N features uses. This method is similar to McClelland and Campbell [89], who defined a path feature space and used LARS to identify relevant features in a study of the effect of time delays on humans tele-operating robots.

Tables 3.1 and 3.2 show the first six lane features identified by LARS for both the steering model and the throttle model on the hciLab data set, described in Section 3.3. As expected, the features identified for the steering model are dominated by the off-track and heading error of the vehicle at different look-aheads.

Table 3.1: First six optimal feature sets for lane feature steering model

Number of Features N					
1	2	3	4	5	6
$\mathbf{e}_{2.5s}^\theta$	$\mathbf{e}_{2.5s}^\theta$	$\mathbf{e}_{2.5s}^\theta$	$\mathbf{e}_{2.5s}^\theta$	$\mathbf{e}_{2.5s}^\theta$	$\mathbf{e}_{2.5s}^\theta$
	\mathbf{e}_{0s}^θ	\mathbf{e}_{0s}^θ	\mathbf{e}_{0s}^θ	\mathbf{e}_{0s}^θ	\mathbf{e}_{0s}^θ
		\mathbf{e}_{3s}^y	\mathbf{e}_{3s}^y	\mathbf{e}_{3s}^y	\mathbf{e}_{3s}^y
			\mathbf{e}_{4s}^y	\mathbf{e}_{1s}^y	\mathbf{e}_{1s}^y
				\mathbf{e}_{1s}^θ	\mathbf{e}_{1s}^θ
					$\kappa_{2.5s}$

Table 3.2: First six optimal feature sets for lane feature throttle model

Number of Features N					
1	2	3	4	5	6
Δv_{3s}	Δv_{3s}	Δv_{3s}	Δv_{3s}	Δv_{3s}	Δv_{3s}
	Δv_{2s}	Δv_{2s}	Δv_{2s}	Δv_{2s}	Δv_{2s}
		$\mathbf{e}_{3.5s}^y$	$\mathbf{e}_{3.5s}^y$	$\mathbf{e}_{2.5s}^y$	$\mathbf{e}_{1.5s}^y$
			\mathbf{e}_{0s}^y	\mathbf{e}_{0s}^y	\mathbf{e}_{0s}^y
				$\kappa_{2.5s}$	$\kappa_{2.5s}$
					\mathbf{e}_{3s}^θ

For the throttle model, the first several features included are the speed difference between the vehicle’s speed and the nominal speed of the road, also as expected.

Figure 3.4 shows the residual error against the number of features used for the LARS regression for a driver steering model fit to the hciLab data set. Based on the clear knee at $N = 4$, $N = 4$ is a natural selection for the number of features to use in the steering model for this data set.

The selection of $N < 37$ intuitively is a trade between performance and the ability to capture increasingly complex driver behavior as N becomes large with the increased computational costs, training data set requirements, and risks of overfitting. The active set of N features is denoted by adding the indicator N to the lane feature transform in Equation 3.8. GP driver models using N lane

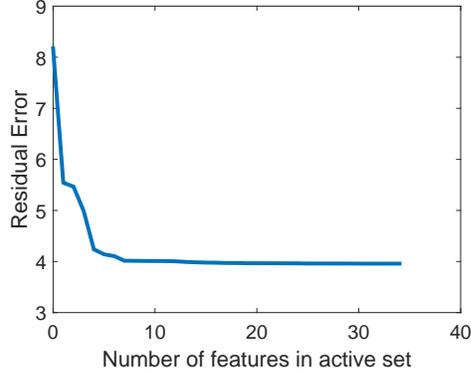


Figure 3.4: Residual error of LARS solution as a function of the number of features in the active set for the steering model.

features are defined using three components:

- 1) the transformation from vehicle state to N lane features:

$$\begin{aligned}\tilde{\mathbf{x}}_{a,k}^N &= \mathcal{H}_a^N(\mathbf{x}_k) \\ \tilde{\mathbf{x}}_{\phi,k}^N &= \mathcal{H}_\phi^N(\mathbf{x}_k).\end{aligned}\tag{3.10}$$

- 2) the GP driver models, mapping N lane features to control input distributions

$$\begin{aligned}a_k &\sim [\mu_{a,k}, \sigma_{a,k}^2] = \text{GP}_a(\tilde{\mathbf{x}}_{a,k}^N) \\ \phi_k &\sim [\mu_{\phi,k}, \sigma_{\phi,k}^2] = \text{GP}_\phi(\tilde{\mathbf{x}}_{\phi,k}^N).\end{aligned}\tag{3.11}$$

- 3) the vehicle model

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, a_k, \phi_k, \delta t)\tag{3.12}$$

3.5 Lane Feature GP Anticipation (LFGPA) Algorithm

The LFGPA algorithm uses the composed parametric vehicle model / lane feature GP driver model described in Equations 3.10-3.12 to make probabilistic predictions of the states of tracked vehicles in the environment. At the current time step k , it is assumed that an existing tracking filter provides an estimate of the state of the tracked vehicle $\mathbf{x}_k \sim \mathcal{N}(\mu_k, \Sigma_k)$. The LFGPA algorithm uses the composed model to make probabilistic predictions about the states of the tracked vehicle at future times out to some anticipation horizon H , i.e. ($\mathbf{x}_{k+1} \sim \mathcal{N}(\mu_{k+1}, \Sigma_{k+1}), \dots$, $\mathbf{x}_{k+H} \sim \mathcal{N}(\mu_{k+H}, \Sigma_{k+H})$).

Figure 3.5 gives a block diagram of the LFGPA algorithm while Algorithm 2 gives a detailed summary. The sigma point transform [26] is used to propagate state distributions through the composed model. The algorithm proceeds as follows. First, the lane feature GP driver models are evaluated at the mean of the current vehicle state, in order to calculate the distribution over the control inputs ($\mathbf{u} = [a, \phi] \sim \mathcal{N}(\mu_u, \Sigma_u)$). Next, sigma points (χ_k^l and ν_k^l) are deterministically sampled from the vehicle state and control input distributions. For sigma points relating to the control noise ($\chi_k^l = \mu_k$), the control sigma points are deterministically sampled from the already computed control input distribution ($\nu_k^l \sim \mathcal{N}(\mu_u, \Sigma_u)$). For sigma points related to the state uncertainty ($\chi_k^l \neq \mu_k$), the LFGPA algorithm slightly differs from the traditional sigma point transform. Traditionally, the control sigma points would just be the precomputed mean ($\nu_k^l = \mu_u$). Using this structure would fail to capture local dependence of the control on vehicle state, so any closed-loop stabilizing behaviors would not be captured by the transform. Here, the lane feature GP driver models are re-evaluated at each sigma point to calculate the relevant control ($\nu_k^l = [\mu_a^l, \mu_\phi^l]^T$).

Given the full set of state (χ_k^l) and control (ν_k^l) sigma points, the parametric ve-

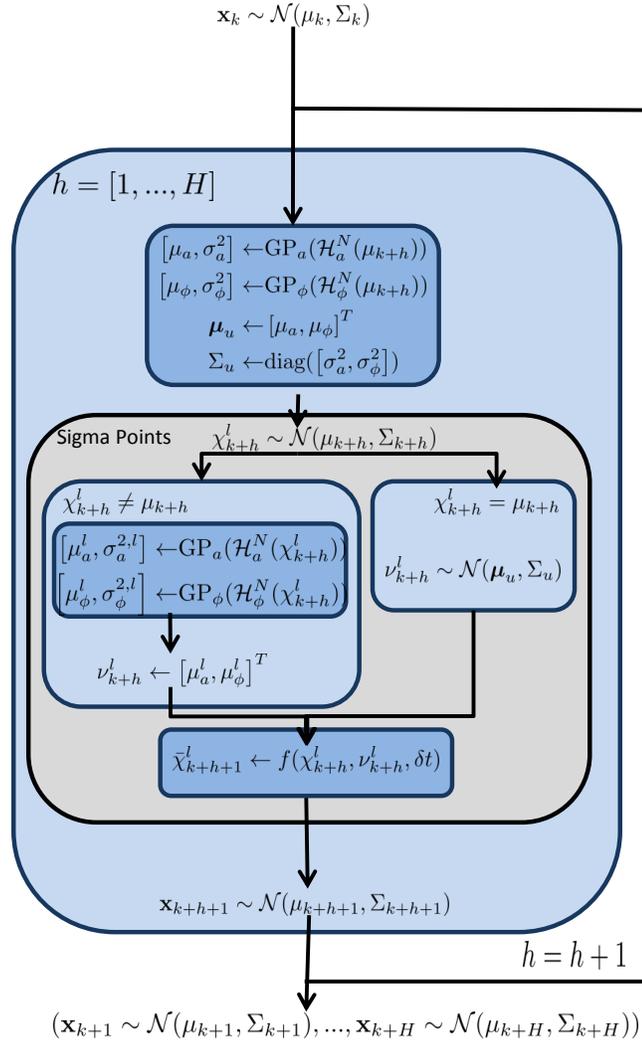


Figure 3.5: Block diagram showing the LFGPA algorithm.

hicle model is used to find the propagated set of sigma points ($\chi_{k+1}^l = f(\chi_k^l, \nu_k^l, \delta t)$). The propagated sigma points define the distribution over the next state ($\mathbf{x}_{k+1} \sim \mathcal{N}(\mu_{k+1}, \Sigma_{k+1})$), completing a single anticipation step forward in time. These is repeated until the horizon H is reached.

A more detailed explanation of the LFGPA algorithm is given in Algorithm 2, and includes the lane feature GP driver model training. Note that the prediction step is similar to the GP-UKF presented by Ko and Fox [55], except that work used purely GP dynamics models.

3.5.1 Map and GPS Free Anticipation

The LFGPA algorithm presented in this work assumes a known map, and instrumented (low frequency GPS measurements) vehicles in the training data set. The GPS measurements are used to reconstruct the training vehicle’s states and control inputs, and the map is required for the transformation from inertial vehicle states to lane features.

A sufficiently sophisticated autonomous vehicle with effective vehicle tracking and road detection algorithms would be able to relax these assumptions. The states of tracked vehicles could be saved, and processed like the vehicle tracks in the training set. Eventually, the observed vehicle tracks could supplant the need for an existing data set containing measured vehicle tracks from instrumented cars.

Relaxing the requirement for a map is more difficult – the autonomous vehicle’s road detector must be sophisticated enough to not only estimate the road in front of the autonomous vehicle, but also to estimate the road in front of all vehicles being tracked. Because the autonomous car often will not be able to directly observe the road in front of tracked vehicles (a vehicle traveling in the same direction of the autonomous vehicle will occlude the road in front of it, or a tracked vehicle turning

Algorithm 2 The LFGPA algorithm

A training data set consisting of paired vehicle state and control action observations:

Require: $\mathcal{D} = (\mathbf{x}_k^i \rightarrow \mathbf{u}_k^i)$

Require: N , M , and H

Train the GP driver models:

Use LARS to find the active feature sets \mathcal{H}_a^N and \mathcal{H}_ϕ^N

Train $\text{GP}_a(\tilde{\mathbf{x}}_{k,a,N})$ and $\text{GP}_\phi(\tilde{\mathbf{x}}_{k,\phi,N})$ using M pseudo-inputs

Use sigma-point transform to perform anticipation

Require: $\mathbf{x}_k \sim \mathcal{N}(\mu_k, \Sigma_k)$

for $h = 1 \rightarrow H$ **do**

Evaluate the GPs at the mean to find the control statistics

$$[\mu_a, \sigma_a^2] \leftarrow \text{GP}_a(\mathcal{H}_a^N(\mu_{k+h}))$$

$$[\mu_\phi, \sigma_\phi^2] \leftarrow \text{GP}_\phi(\mathcal{H}_\phi^N(\mu_{k+h}))$$

$$\boldsymbol{\mu}_u \leftarrow [\mu_a, \mu_\phi]^T$$

$$\Sigma_u \leftarrow \text{diag}([\sigma_a^2, \sigma_\phi^2])$$

Find $\mathbf{S}_x = [\mathbf{S}_x^1, \dots, \mathbf{S}_x^{n_x}]$ such that $\mathbf{S}_x \cdot \mathbf{S}_x^T = \Sigma_{k+h}$

Find $\mathbf{S}_u = [\mathbf{S}_u^1, \dots, \mathbf{S}_u^M]$ such that $\mathbf{S}_u \cdot \mathbf{S}_u^T = \Sigma_u$

for $l = 0 \rightarrow 2n_x + 2n_u$ **do**

if $l \in [0, 2n_x + 1, \dots, 2n_x + 2n_u]$ **then**

$$\chi_{k+h}^l \leftarrow \mu_x$$

else if $l \in [1, \dots, n_x]$ **then**

$$\chi_{k+h}^l \leftarrow \mu_x + \lambda \cdot \mathbf{S}_x^l$$

else

$$\chi_{k+h}^l \leftarrow \mu_x - \lambda \cdot \mathbf{S}_x^{l-n_x}$$

end if

if $l = 0$ **then**

$$\nu_{k+h}^l \leftarrow \boldsymbol{\mu}_u$$

else if $l \in [1, \dots, 2n_x]$ **then**

Evaluate the GP driver models at this sigma point

$$[\mu_a^l, \sigma_a^{2,l}] \leftarrow \text{GP}_a(\mathcal{H}_a^N(\chi_{k+h}^l))$$

$$[\mu_\phi^l, \sigma_\phi^{2,l}] \leftarrow \text{GP}_\phi(\mathcal{H}_\phi^N(\chi_{k+h}^l))$$

$$\nu_{k+h}^l \leftarrow [\mu_a^l, \mu_\phi^l]^T$$

else if $l \in [2n_x + 1, \dots, 2n_x + M]$ **then**

$$\nu_{k+h}^l \leftarrow \boldsymbol{\mu}_u + \lambda \cdot \mathbf{S}_u^{l-2n_x}$$

else

$$\nu_{k+h}^l \leftarrow \boldsymbol{\mu}_u - \lambda \cdot \mathbf{S}_u^{l-2n_x-M}$$

end if

Evaluate the parametric model at each sigma point

$$\bar{\chi}_{k+h+1}^l \leftarrow f(\chi_{k+h}^l, \nu_{k+h}^l, \delta t)$$

end for

Find first two moments of propagated state

Given sigma point weights $[w_m^1, \dots, w_m^{2n_x+2n_u}]$ and $[w_c^0, \dots, w_c^{2n_x+2n_u}]$

$$\mu_{k+h+1} \leftarrow \sum_l w_m^l \cdot \bar{\chi}_{k+h+1}^l$$

$$\Sigma_{k+h+1} \leftarrow \sum_l w_c^l \cdot \left(\bar{\chi}_{k+h+1}^l - \mu_{k+h+1} \right) \cdot \left(\bar{\chi}_{k+h+1}^l - \mu_{k+h+1} \right)^T$$

end for

onto a previously un-observed side road).

If no map is available, the autonomous vehicle could construct one over time. Increasingly, however, high-quality maps are commercially available (such as from Nokia HERE), and basic road maps are widely available from route navigation suppliers (such as Garmin’s NAVIGON system).

3.6 Experimental Validation

3.6.1 Study of the Size of the Feature Space and Size of the SPGP

The first set of evaluations are done to study the performance implications of the number of features used (N) and the number of pseudo-inputs used (M). LFGP driver models trained using the full hciLab data set, and used to predict the motion of a driver-vehicle operating on a simulated roads. Using the LFGPA algorithm, the state of a vehicle is propagated forward in time 200 time steps along either a straight road or a curved road. The number of features in the feature space (N) and the number of pseudo-inputs used by the SPGP (M) are varied to study how they effect the performance of the LFGPA algorithm, and provide insight into their selection. Three metrics are used to evaluate performance: 1) the expected value of the distance of the vehicle from the driving lane center ($e_1(\mathbf{x}_k)$); 2) the expected value of the difference between the vehicle heading and the lane heading ($e_2(\mathbf{x}_k)$); 3) the time required to predict the vehicle state forward to a horizon of 200 time steps.

Figure 3.6 plots the performance metrics as a function of N and M for the straight road case. In general, the SPGPs with the best performance use larger

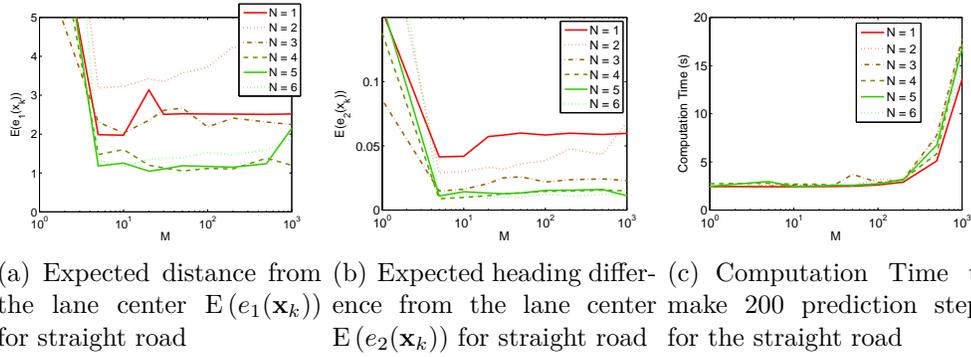


Figure 3.6: Expected value of the performance metrics $e_1(\mathbf{x}_k)$, $e_2(\mathbf{x}_k)$, and computation time for a vehicle driving on a straight road.

feature sets ($N \in \{4, 5, 6\}$), while the SPGPs with smaller feature sets perform worse. This behavior suggests that when $N < 4$ features are used, the data set is not enough to capture the behavior of the driver. This is intuitive because at least four features are required in the lane feature space to capture the behavior of drivers in the hciLab data set. When less than four features are used, vehicle-environment relationships that are distinct and result in different control actions cannot be distinguished from each other, resulting in poor and noisy predictions. Figure 3.6(a) also shows LFGPA algorithm performance improving as M becomes larger. Finally, Figure 3.6(c) plots the time required to make the full 200 prediction steps as a function of N and M . The number of features used (N) has minimal impact on the time required to make predictions. For $M < 500$, the computation time is nearly flat, and the computation time associated with the SPGP is small enough that computational time is dominated by other aspects of anticipation. The prediction time step used is 0.1sec, so Figure 3.6(c) shows that for most choices of M and N , it takes 3 seconds to predict the behavior of a tracked vehicle 20 seconds into the future. The complexity of the SPGP is not the only variable that effects the computational time, however, as the number of lane features computed and the complexity of the environment used to compute the lane features also effects

computation times. For $M = 500$ and larger, however, the SPGP cost dominates, and computation becomes very expensive.

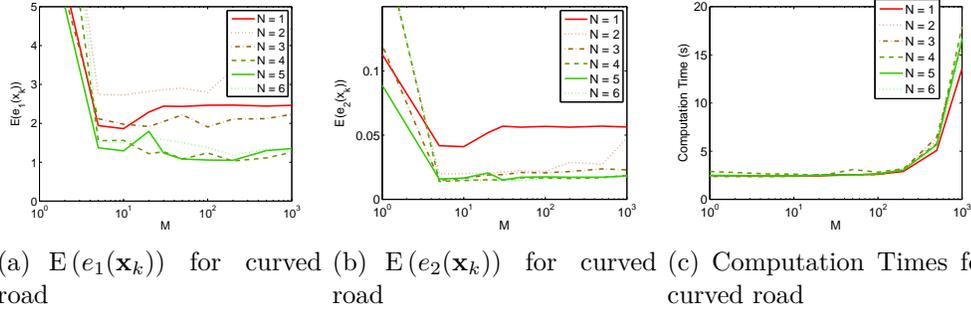


Figure 3.7: Expected value of the performance metrics $e_1(\mathbf{x}_k)$ and $e_2(\mathbf{x}_k)$ for a vehicle driving on a curved road.

Figure 3.7 shows the results from the same studies performed on a curved simulated road. The general performance trends are the same as for the straight road study (Figure 3.6), with $N = 4, 5, 6$ having the best performance, and a weak trend showing performance improving as M becomes larger. The similarity between Figures 3.7(c) and 3.6(c) suggest that the differing environment has no impact on the computation time of the algorithm.

Based on these results for the hciLab data set, $N = 4$ is used in the rest of the evaluations, as compressing the data any farther has a negative impact on performance (given Figure 3.4, this result matches expectations). In addition, due to the trend that performance improves as M becomes larger, $M = 200$ is chosen. For these simple roads, smaller values for M perform reasonably well, but for more complex environments better coverage of the training space might be required. As noted, $M = 200$ is also the largest M can become before severely increasing computation times.

For these values of N and M , the computational time of about 3 seconds to predict the vehicle state to an anticipation horizon of 200 time steps, or 20 seconds,

in MATLAB; a C/C++ implementation would see an order of magnitude increase in speed. This is considered real-time performance, as the anticipation horizon is much larger than the required time to compute the predictions.

3.6.2 Generalization of LFGPA Algorithm Across Data Sets

Sets

The goal of the next set of studies is to evaluate how general the proposed LFGPA algorithm is for predicting the future motion of tracked vehicles. Specifically, the ability of the LFGPA algorithm to be trained on a subset of the data, and evaluated in a variety of scenarios (some from regions in the training set, some from outside the training set).

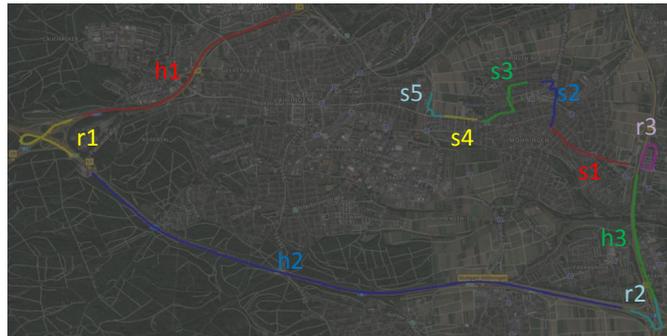


Figure 3.8: Region partitioning of the hciLab data set. Much of the route after the last region defined here ($s5$) consists of a tunnel, so GPS data is not available.

The hciLab training data set is partitioned into regions by road type, resulting in three highway regions ($h1$, $h2$, $h3$), three highway ramp regions ($r1$, $r2$, $r3$), and five surface street regions ($s1$, $s2$, $s3$, $s4$, $s5$). Figure 3.8 shows the region partitions. The performance of the LFGPA algorithm trained with different subsets of the data (e.g. using only data from the highway regions) predicting the behavior of driver-vehicle systems operating in the different regions. All LFGPA algorithms

trained here using $N = 4$ features and $M = 200$ pseudo-inputs.

Intra-Region Studies

This section studies the performance of an LFGPA algorithm trained using data from one region, and used for anticipation in another region *of the same type*. For a given region, 100 vehicle states within that region are sampled from the hciLab data set. Each state is predicted 40 steps forward in time using each of the LFGPA algorithms being evaluated. The LFGPA algorithms are compared to the actual vehicle state history from the data set by evaluating the log-likelihood of the actual vehicle state trajectory from the data set being drawn from the predicted vehicle state distributions (higher is better).

For a set of vehicle states from the training data set $\{\mathbf{x}_k, \dots, \mathbf{x}_{k+H}\}$, the LFGPA algorithm being evaluated is used to predict the motion of the driver-vehicle from the first state, $\mathbf{x}_k \rightarrow \{(\mu_{k+1}, \Sigma_{k+1}), \dots, (\mu_{k+H}, \Sigma_{k+H})\}$. The log-likelihood is the log of the likelihood of that actual driver-vehicle trajectory having been sampled from the anticipated trajectory:

$$\begin{aligned} \text{Log-Likelihood} &= \log \left(\prod_{j=1}^H \mathcal{N}(\mathbf{x}_{k+j} | \mu_{k+j}, \Sigma_{k+j}) \right) \\ &= \sum_{j=1}^H \log (\mathcal{N}(\mathbf{x}_{k+j} | \mu_{k+j}, \Sigma_{k+j})) \end{aligned} \quad (3.13)$$

Figure 3.9 plots the study results for the highway regions ($h1$, $h2$, $h3$). LFGPA algorithms are trained using data from each of the three regions ($h1$, $h2$, $h3$), as well as the combined highway regions ($hAll$) and the full data set ($dAll$). When considering training using data from one region only ($h1$, $h2$, $h3$), the LFGPA algorithm trained in that region does better than the models trained in either of the other two regions, although not by a large margin. This shows that LFGPA

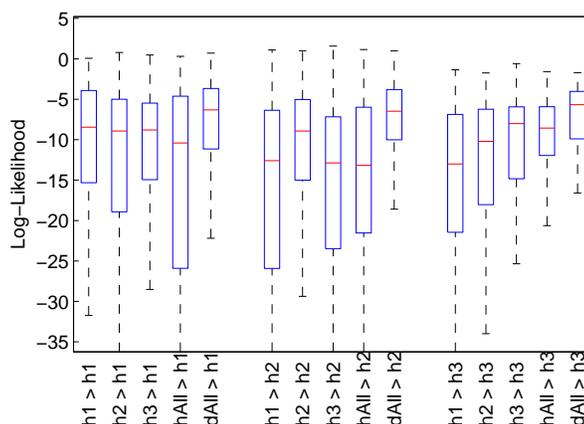


Figure 3.9: Anticipation log-likelihood of LFGPA algorithms learned using different highway regions ($h1, h2, h3, hAll$) and the entire data set ($dAll$) anticipating the behavior of vehicles driving in the three highway regions ($h1, h2, h3$).

algorithms trained with data from *geographically different* regions can make qualitatively competitive predictions to an LFGPA algorithm trained on data from the region where predictions are being made.

Figure 3.10 plots the study results for the highway ramp regions ($r1, r2, r3$). In general, the LFGPA algorithms trained with data from $r1$ and $r3$ perform poorly, except when anticipating the behavior of vehicles in their training region. Here, the LFGPA trained with data from $r2$ works well not only inside its training region, but also in the distinct regions $r1$ and $r3$. The poor performance of the LFGPA algorithms trained in $r1$ and $r3$ is discussed in Section 3.6.2.

Figure 3.11 plots the anticipation performance for lane feature GP driver models used on surface streets ($s1, s2, s3, s4, s5$). Similar to the highway and ramp region studies (Figures 3.9 and 3.10), a number of the LFGPA algorithms work well both within and without their training data ($s2, s3$, and $s5$). The LFGPA algorithms that perform poorly ($s1$ and $s4$) share similarities with the LFGPA algorithms that perform poorly for the highway ramp regions ($r1$ and $r3$), and are

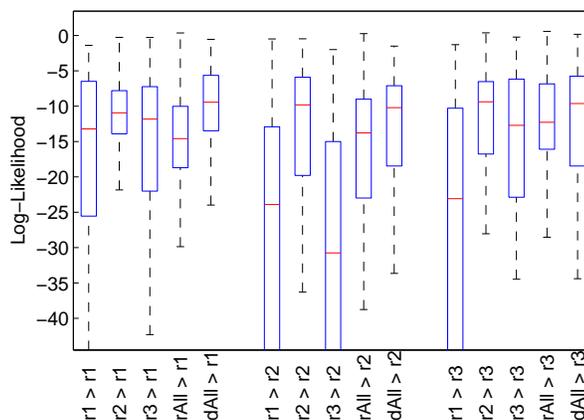


Figure 3.10: Anticipation log-likelihood of LFGPA algorithms learned using different highway ramp regions ($r1$, $r2$, $r3$, $rAll$) and the entire data set ($dAll$) anticipating the behavior of vehicles driving in the three highway ramp environments ($r1$, $r2$, $r3$).

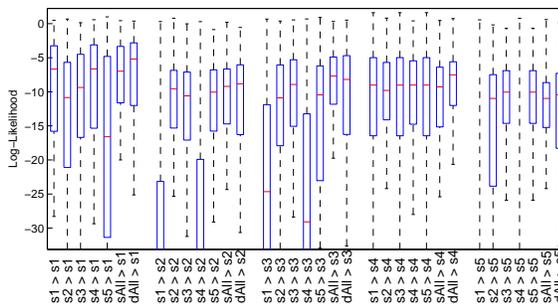


Figure 3.11: Anticipation log-likelihood of LFGPA algorithms learned using different surface street regions ($s1$, $s2$, $s3$, $s4$, $s5$, $sAll$) and the entire data set ($dAll$) anticipating the behavior of vehicles driving in the five surface street environments ($s1$, $s2$, $s3$, $s4$, $s5$).

discussed in Section 3.6.2.

This section is the first demonstration of the ability of LFGPA algorithms to be used outside of their training data sets (with a few exceptions). The ability to train an anticipation algorithm on a data set consisting of humans driving on some selected roads, and being able to use it to predict the motion of driver-vehicles operating on *general* roads is the primary goal of the LFGPA algorithm.

Training Data Selection

Section 3.6.2 shows that LFGPA algorithms trained in one region can make accurate predictions of driver-vehicle motion in another, but also that not all training data sets result in effective LFGPA algorithms ($r1$, $r3$, $s1$, and $s4$, for example). This section analyzes various training data sets to provide insight into training effective LFGPA algorithms.

Starting with data sets $r1$ and $r3$, Figure 3.8 provides some indication as to why the LFGPA algorithms trained with data from these regions were unable to make accurate predictions in $r3$ – both $r1$ and $r3$ consist entirely of right-hand turns. Intuitively, it makes sense that an LFGPA algorithm trained on these regions would not be able to predict that a driver would turn left to follow a left-hand turn, as drivers must do in $r2$. More concretely, the total range of observed steering angles in the training data sets from $r1$ and $r3$ are $\phi \in [-0.1457, 0.0420]$ and $\phi \in [-0.3534, 0.0742]$, respectively, while the left-hand turn in $r2$ requires up to $\phi = 0.141$. Because this steering angle is outside of the range of the training data sets from $r1$ and $r3$, LFGPA algorithms trained on these regions cannot produce this control, and therefore cannot accurately predict the behavior of drivers in $r2$.

This analysis also explains why the LFGPA algorithm trained on $r3$ is able to predict the motion of driver-vehicles in $r1$, but not the other way around. $r3$ includes a much sharper left-hand turn than $r1$ (the required control is $\phi =$

-0.3534), which is outside of the training data set for $r1$.

Similarly, Figure 3.8 shows that regions $s1$ and $s4$ consist of primarily long straight roads, and intuitively could not therefore be used to train LFGPA algorithms to predict the motion of driver-vehicles negotiating corners. The steering control ranges for $s1$ and $s4$ are $\phi \in [-0.1329, 0.0991]$ and $\phi \in [-0.1285, 0.1035]$, respectively. The range of steering control angles observed in $s5$ is $\phi \in [-0.7122, 0.5120]$, well outside the range of angles that can be produced by LFGPA algorithms trained with data sets from $s1$ or $s4$.

This analysis can be used to predict the efficacy of LFGPA algorithms trained on different subsets of the hciLab data set at making driver-vehicle motion predictions on general roads. A data set comprised of $h1$, $r2$, $s2$, and $s5$ is expected to make good predictions *anywhere* in the hciLab data set, as the subset of regions is representative of highways, highway ramps, and surface streets. Likewise, a data set comprised of $h1$, $r2$, $s2$, and $s5$ is expected to perform worse, as the regions selected are not as representative of the scenarios faced by drivers. Section 3.6.2 studies the overall predictive power of several LFGPA algorithms.

This section provides insight into the selection of an appropriate training data set for a given application. To train an LFGPA algorithm that can effectively predict the behavior of a driver-vehicle operating on surface streets in a city, for example, the training set must include representative examples of drivers operating on surface streets. To capture highway behavior the training set must include examples of drivers operating on highways. To ensure good prediction performance from an LFGPA algorithm in a given region, the *types* of driving in that region must be identified, and the training set must include examples of drivers operating in these types of situations. While the LFGPA algorithm does not enable a model trained using a small data set of drivers on a highway to predict the behavior of

drivers on winding surface streets, it does enable a driver model trained on a mixed data set of driving conditions to accurately predict the behavior of drivers operating in similar conditions anywhere in the world.

General Prediction Performance Studies

This section evaluates the overall effectiveness of various LFGPA algorithms in predicting the behavior of driver-vehicle systems in the hciLab data set. To study the effectiveness of a given LFGPA algorithm, the full state history for one of the drivers' route traversals in the hciLab data set is used for validation. LFGPA algorithms are trained using subsets of the data from the other nine drivers. A 40 time step prediction (4 seconds) is made every tenth state in the route traversal (1 second). A prediction is *accurate* if the true vehicle state stays within the 95% confidence bound of the anticipated vehicle state distribution for the entire prediction window. Additionally, the *precision* of an anticipation algorithm is given by the average trace of the covariance matrix of accurate predictions. The ideal anticipation algorithm is both accurate and precise. An algorithm that performs accurately, but predicts very large covariance matrices leads to safe, but very conservative planning. Of course, an algorithm that performs poorly on accuracy also leads to unsafe behavior.

An LFGPA algorithm trained with data from all of the regions is evaluated first as a benchmark for the other LFGPA algorithms used. An LFGPA algorithm trained with the subset of data likely to perform well given the analysis in Section 3.6.2($\{h1, r2, s2, s5\}$) is evaluated and compared to the benchmark. Finally, two LFGPA algorithms trained on data sets likely to result in poor performance ($\{h1, r2, s2, s5\}$ and $\{s1, s2, s3, s4, s5\}$) are evaluated.

The benchmark LFGPA algorithm is trained using the entire training set, i.e. this is the most complete training set, in terms of diversity of data. Figure 3.12

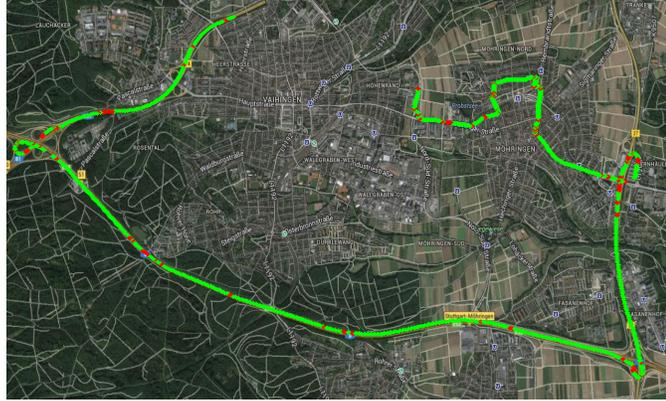


Figure 3.12: Prediction performance using a lane feature GP driver model trained with the entire data set. Points in green are initial conditions from which the anticipations were accurate. Points in red are initial conditions from which the anticipation was inaccurate. Here, 93.4% of the predictions were accurate.

plots the initial conditions used in anticipation from the route of the driver, color coded by the accuracy of the prediction from that initial condition (green is accurate, red is inaccurate). For this LFGPA algorithm, 93.4% of predictions were accurate. The precision for this driver model is 3.766.

Figure 3.13 shows the accuracy of an LFGPA algorithm trained with data from regions $\{h1, r2, s2, s5\}$. The accuracy of 92.3% and precision of 5.884 is very close to the benchmark set by the LFGPA algorithm trained with the complete data set. The good performance of this LFGPA algorithm is expected because its training set is chosen based on the analysis in Section 3.6.2. As a result, the LFGPA algorithm trained with this data set can make accurate predictions of driver-vehicle motion *outside of its training set* and *on any type of road* from the hciLab data set. Note that a primary area where this LFGPA algorithm performs worse than the LFGPA algorithm trained with the full data set is $r1$, which is a high-speed, tight, right-hand ramp from one highway to another, which is outside the training set for this LFGPA algorithm.

Figure 3.14 shows the accuracy of the model trained using regions

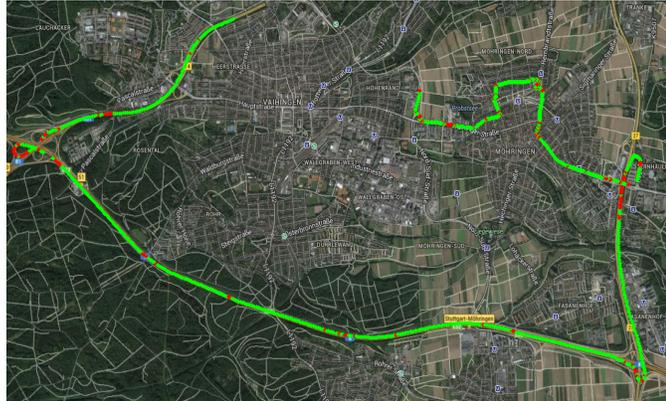


Figure 3.13: Prediction performance using a lane feature GP driver model trained with the data from regions $\{h1, r2, s2, s5\}$. Points in green are initial conditions from which the anticipations were accurate. Points in red are initial conditions from which the anticipation was inaccurate. Here, 92.3% of the predictions were accurate.

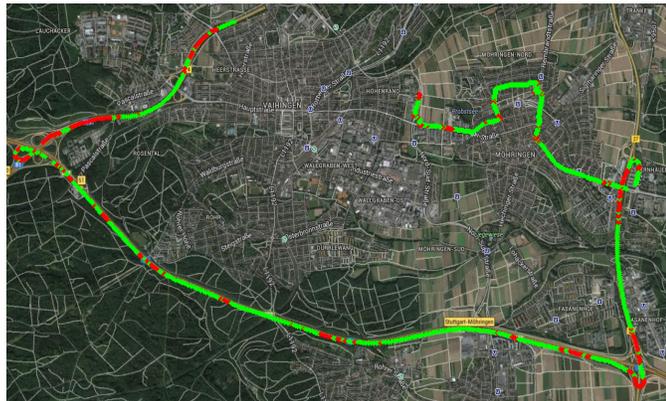


Figure 3.14: Prediction performance using a lane feature GP driver model trained with the data from regions $\{h1, r2, s2, s5\}$. Points in green are initial conditions from which the anticipations were accurate. Points in red are initial conditions from which the anticipation was inaccurate. Here, 73.9% of the predictions were accurate.

$\{h1, r2, s2, s5\}$. This model has an accuracy of 73.9%, and a precision of 7.530, both significantly worse than the benchmark LFGPA algorithm, matching the low expectations for an LFGPA algorithm trained with this data set.

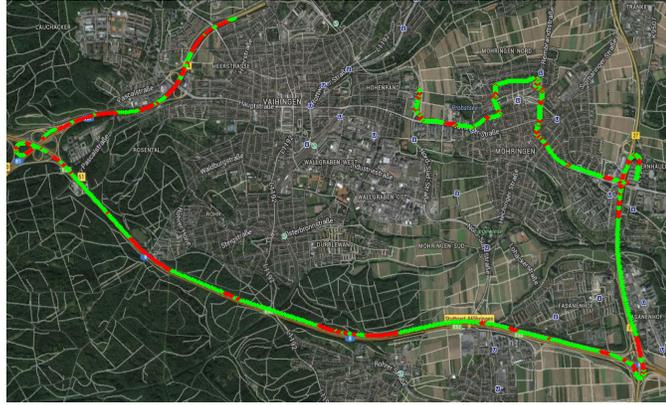


Figure 3.15: Prediction performance using a lane feature GP driver model trained with only surface street data ($\{s1, s2, s3, s4, s5\}$). Points in green are initial conditions from which the anticipations were accurate. Points in red are initial conditions from which the anticipation was inaccurate. Here, only 69.2% of the predictions were accurate, and accuracy on the highway regions is especially bad.

Finally, an LFGPA algorithm trained using only data from surface streets ($\{s1, s2, s3, s4, s5\}$) is compared to the benchmark. Figure 3.15 shows the prediction accuracy of the LFGPA algorithm matches expectations by performing poorly overall and particularly so on highway and ramp regions, neither of which are included in the training set for this LFGPA algorithm.

Table 3.3: Anticipation Accuracy and Precision

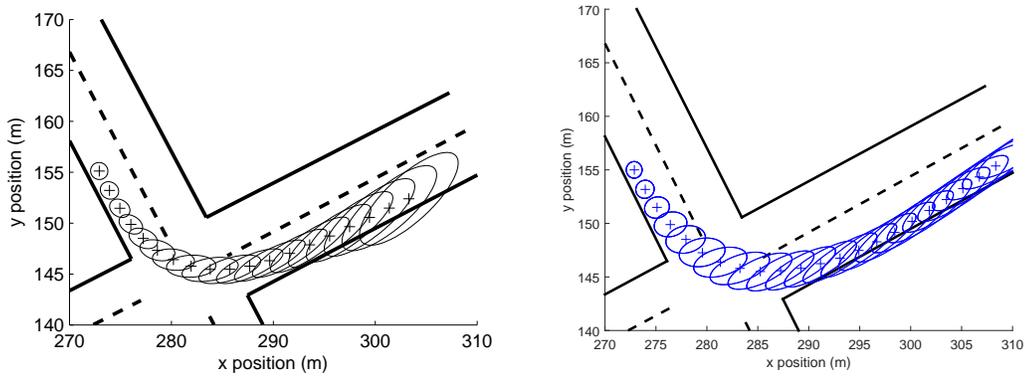
	Accuracy	Precision	training set length (km)
All	93.4%	3.766	18.13
$\{h1, r2, s2, s5\}$	92.3%	5.884	5.08
$\{h3, r3, s4, s5\}$	73.9%	7.530	3.24
Highway	78.3%	4.393	11.32
Ramp	81.0%	4.993	3.10
Surface	69.2%	18.36	3.72

Table 3.3 summarizes the results in this section. The success of the LFGPA algorithm trained using the diverse subset of data ($\{h1, r2, s2, s5\}$) shows that the LFGPA algorithm can successfully predict the motion of human-driven vehicles anywhere in the hciLab data set, including regions not included in the training data set for the model. Here, an LFGPA algorithm trained using data from only 5.08 kilometers of roads is able to accurately predict the behavior of driver-vehicles operating throughout the 18.13 kilometer environment. Also noteworthy, an LFGPA algorithm using only 200 data points in the GP input space is able to effectively anticipate the behavior of a human driven vehicle operating on over 18 kilometers of road, which would require many thousands of data points if trained in the vehicle’s inertial state space instead of the lane feature space presented here.

Lane-Feature GP Driver Models Across Data Sets

To primary advantage of the LFGPA algorithm is that it allows a driver-vehicle model to be trained on a relatively small data set and used to predict driver-vehicle motion in general environments. Section 3.6.2 demonstrates the ability of LFGPA algorithms (in this section, the LFGPA algorithm is trained with the $\{h1, r2, s2, s5\}$ data set) to make accurate predictions of driver-vehicle motion outside of the training data sets used. This section further studies the ability of an LFGPA algorithm to work generally by using an LFGPA algorithm trained using a subset of the hciLab data to predict the motion of driver-vehicles from the data set collected and used in Section 2.6.

Figure 3.16 shows the prediction of a human-driven vehicle making a left turn in the data set from Chapter 2 using a driver modeled trained on the data set from Chapter 2, and an LFGPA algorithm trained on the hciLab data set from the same initial conditions. Qualitatively, the LFGPA algorithm prediction looks similar to the GP-aGMM prediction.



(a) GP-aGMM driver model trained on same data set (b) LFGPA algorithm trained on hciLab data set

Figure 3.16: Comparison of predictions for left turn maneuver between a GP-aGMM driver model trained on the same data set (3.16(a)) and an LFGPA algorithm trained on the hciLab data set(3.16(b))

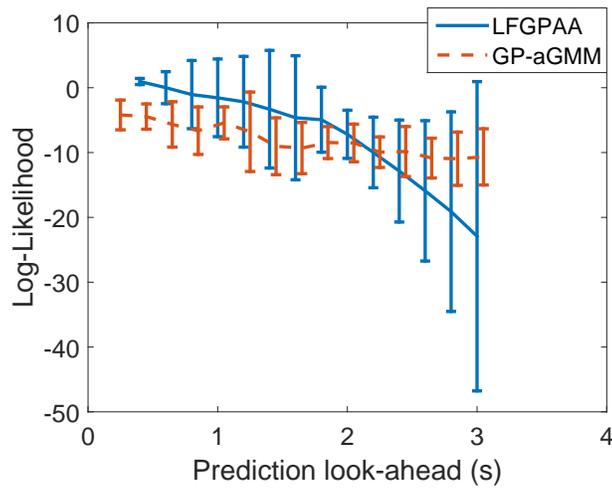


Figure 3.17: Log-likelihood comparison (smaller is better) of an LFGPA algorithm trained on data from the hciLab data set, and a GP-aGMM trained on the local data set.

A more quantitative analysis is performed by using the LFGPA algorithm and the GP-aGMM to predict the behavior of all of the examples of a human driver making the left turn in the data set from Section 2.6. Figure 3.17 plots the log-likelihood (NLL) of the actual trajectories of the human driven vehicles in the data set having been sampled from the predicted trajectories calculated by the two algorithms. The LFGPA algorithm is competitive to the GP-aGMM, and even scores higher on average for short horizons. The better performance over short horizons, and the growing error and standard deviation for longer horizons, is due to the fact that the LFGPA algorithm predicts smaller covariance matrices for the driver-vehicle states than the GP-aGMM, which can be seen in Figure 3.16.

3.7 Conclusion

This chapter presents a novel process for generalizing Gaussian Process driver models outside of their training data set. The generalization works by training Gaussian Process driver models in a lane feature space rather than the vehicle’s inertial state space. The lane feature space captures the relationship between the vehicle state and its local environment, so the GP driver model learned in this space can be used to anticipate the behavior of drivers operating in any environment, not just those included in the training data set. This generalization alleviates the need to collect a training data set that fully covers the environment in which the algorithm will be used, making the application of GP driver models for anticipation in larger environments feasible.

A method for automatically selecting a relevant subset from a large lane feature space during driver model training is also presented. A high-dimensional lane feature space is defined, with partially redundant features, and optimal subsets for the GP driver models are automatically extracted before training. This system

can easily handle the inclusion of new types of features. For the application presented here, a subset of four features is found to be the optimal choice for trading performance with computational and training complexity.

The presented algorithm is evaluated on a large data set consisting of human drivers operating in Stuttgart, Germany. Lane feature GP driver models are trained using the data set, and the accuracy and precision of the models are evaluated in simulation and on examples from the data set. A large-scale evaluation of the proposed anticipation algorithm is performed by anticipating the behavior of one of the hciLab drivers using data from the other nine. The key result from this evaluation is that a lane feature GP driver model trained using data from a subset of the regions in the environment shows similar anticipation performance to a driver model trained using the full data set, even when making anticipations outside of its training regions.

Although the presented algorithm allows a small training set to be used to train a widely applicable driver model, the training set must still contain examples of the types of scenarios the driver model is expected to encounter. The process of selecting a training data set is explored, and a driver model trained with the selected set is shown to have good general performance. However, the selection of a training set depends on the intended application of the driver model – although the driver models here are shown to perform well in a mix of city and highway driving, they would not be able to accurately anticipate the behavior of a driver on, for example, a winding mountain road without examples from such roads included in the training set.

Finally, the presented algorithm is used with a training set from the hciLab data to make predictions about driver behavior in an entirely different data set, and is shown to make qualitatively similar predictions to a GP driver model trained

on data from that same data set. The power of training a GP driver model in the presented lane feature space to generalize driver models is demonstrated by a driver model trained on data collected by a different lab, in a different city, under different conditions making reasonable predictions about the behavior of human drivers operating the simulator used to collect data for Chapter 2.

CONCLUSION

Chapter 1 presents a method for probabilistically anticipating the future states of tracked objects with nonlinear dynamics. Gaussian Mixture Models are used to represent the probability distribution over the object's state, allowing for non-Gaussian beliefs. A novel method for evaluate the accuracy of propagation of individual mixands through the dynamics model is developed, which gives the algorithm the ability to recognize when nonlinearities in the dynamics function are negatively impacting the accuracy of belief propagation. The algorithm then adapts the Gaussian Mixture Model by splitting the innacurately propagated mixand into several mixands with reduced covariances, to mitigate the impact of nonlinearities. The mixand splitting uses pre-computed optimal splits calculated for a one dimensional case, and applies the splits online using a series of simple transformations. The algorithm is evaluated in simulation and on a publically available data set. In addition, an anecdotal example is provided, showing that the algorithm would have been able to predict the collision that occured between Cornell's and MIT's DARPA Urban Challenge entries. Additionally, the algorithm is implemented in real hardware and operates in real time, and can be used by planning algorithm's on Cornell's autonomous vehicle.

Chapter 2 addresses the problem of modeling a human-driven vehicle. The presented algorithm uses a traditional parametric model to capture the vehicle dynamics, and a non parametric data driven Gaussian Process regression model to model the human driver. An efficient anticipation algorithm is constructed using this model. Due to the computational cost of Gaussian Process evaluations, a closed form approximation is used that evaluates the Gaussian Process over a normal distribution, instead of a point, so the Gaussian Process only needs to be evaluated once per mixand per timestep. However, because the Gaussian Process

is no longer being evaluated using the sigma point transform, the nonlinearity detection work in Chapter 1 will no longer effectively respond to nonlinearities in the driver model. Chapter 2 presents two novel metrics for evaluating the accuracy of a Gaussian Process evaluated over an input distribution: first, a measure of how Gaussian the output of the Gaussian Process is by analytically computing the excess kurtosis of the output distribution; and second, a measure of how linear the training data of the Gaussian Process is local to the input distribution. To further address the computational cost of Gaussian Process models, a novel on-the-fly compression scheme is presented. The overall algorithm is evaluated using simple one dimensional simulation problems and data collected from human drivers operating in a driving simulator. The algorithm is shown to provide accurate predictions using Gaussian Process models at a significant computational advantage over existing works.

A major drawback to the work in Chapter 2 when applied to the problem of anticipating the behavior of human drivers is the logistical and computational challenge presented by the required training data set for the Gaussian Process driver model. In order to be useable, the training data set would need to fully cover the environment with reasonable density — this makes applying these methods on a city-scale infeasible. Chapter 3 presents a method for generalizing the state of tracked objects for anticipation by transforming the object’s state into a feature space that captures the relationship between the object and its local environment. In the resulting feature space, Gaussian Process driver models trained using data from one environment can be used to make effective predictions in any similar environment. Using a Gaussian Process compression scheme, a Gaussian Process driver model with only 200 training data points is shown to make accurate predictions about a tracked vehicle’s future behavior over a 20 kilometer route.

To summarize, this thesis presents the following novel contributions:

- A method for detecting nonlinearities in a dynamics model during sigma point propagation
- A method for computing optimal splits of a Gaussian distribution
- A method for applying pre-computed optimized splits to arbitrary problems in a real time system
- Analysis of the anticipation algorithm on real and simulated data
- The decomposition of the model of a human driven vehicle into a parametric vehicle model and a Gaussian Process driver model
- An on-the-fly compression algorithm that reduces the computational cost of evaluating Gaussian Processes over uncertain states
- A metric that evaluates how Gaussian the output of a Gaussian Process evaluated at an uncertain state is by calculating analytically the excess kurtosis
- A metric that evaluates how linear the training data of a Gaussian Process is local to an uncertain state input
- The generalization of object states by transforming them into a feature space
- Applying feature space transformations to make using Gaussian Process driver models feasible in large scale environments
- A method for automatically identifying important features in the feature space
- Analysis of Gaussian Process driver models trained in feature spaces on real driver data

APPENDIX A

KURTOSIS DERIVATION

The kurtosis of a distribution is defined as $\text{kurt}(y) = E[y^4] - 3(E[y^2])^2$. Using this definition, finding the kurtosis of the output of a GP is equivalent to finding the fourth order cumulant of the output, $\kappa_4 = \mu_4 - 3\mu_2^2$. The fourth order cumulant of a GP output evaluated over a Gaussian input distribution, $x^* \sim \mathcal{N}(\mu_x, \Sigma_x)$, is:

$$\begin{aligned} \kappa_4(g(x^*)) = & \tag{A.1} \\ & \kappa(\kappa_4(g(x^*)|x^*)) + 4\kappa(\kappa_3(g(x^*)|x^*), \kappa(g(x^*)|x^*)) \\ & + 3\kappa(\kappa_2(g(x^*)|x^*), \kappa_2(g(x^*)|x^*)) \\ & + 6\kappa(\kappa_2(g(x^*)|x^*), \kappa(g(x^*)|x^*), \kappa(g(x^*)|x^*)) \\ & + \kappa_4(\kappa(g(x^*)|x^*)) \end{aligned}$$

where $\kappa_4(g(x^*)|x^*) = 0$, $\kappa_3(g(x^*)|x^*) = 0$ since the output of a GP at a known input point is Gaussian. This allows Equation A.1 to be simplified to:

$$\begin{aligned} \kappa_4(g(x^*)) = & \tag{A.2} \\ & 3\kappa(\sigma^2(x^*), \sigma^2(x^*)) + 6\kappa(\sigma^2(x^*), \mu(x^*), \mu(x^*)) \\ & + \kappa_4(\mu(x^*)) \end{aligned}$$

The terms in Equation A.2 can then be expressed in terms of expectation integrals of the GP functions evaluated over x^* :

$$\kappa(\sigma^2(x^*), \sigma^2(x^*)) \tag{A.3}$$

$$\begin{aligned} &= \text{var}(\sigma^2(x^*)) \\ &= E[\sigma^2(x^*)^2] - E[\sigma^2(x^*)]E[\sigma^2(x^*)] \end{aligned}$$

$$\kappa(\sigma^2(x^*), \mu(x^*), \mu(x^*)) \tag{A.4}$$

$$\begin{aligned} &= E[\mu(x^*)^2 \sigma^2(x^*)] - 2E[\mu(x^*) \sigma^2(x^*)]E[\mu(x^*)] \\ &\quad - E[\sigma^2(x^*)]E[\mu(x^*)^2] + 2E[\sigma^2(x^*)]E[\mu(x^*)]^2 \end{aligned}$$

$$\kappa_4(\mu(x^*)) = \mu_4[\mu(x^*)] - 3\mu_2[\mu(x^*)]^2 \tag{A.5}$$

where

$$\mu_4(\mu(x^*)) = E[\mu(x^*)^4] - 4E[\mu(x^*)^3]E[\mu(x^*)] \tag{A.6}$$

$$+ 6E[\mu(x^*)^2]E[\mu(x^*)]^2 - 3E[\mu(x^*)]^4$$

$$\mu_2(\mu(x^*)) = E[\mu(x^*)^2] - E[\mu(x^*)]^2 \tag{A.7}$$

This means that computing the fourth order cumulant of the GP output requires the evaluation of the following additional expectation integrals that are not required for the evaluation of the mean, covariance, and cross-covariance:

$$E[\mu(x^*)\sigma^2(x^*)]$$

$$E[\mu(x^*)^3]$$

$$E[\sigma^2(x^*)^2]$$

$$E[\mu(x^*)^2\sigma^2(x^*)]$$

$$E[\mu(x^*)^4]$$

of which $E[\mu(x^*)\sigma^2(x^*)]$ and $E[\mu(x^*)^3]$ are $\mathcal{O}(N^3)$ and the rest are $\mathcal{O}(N^4)$. These expectation integrals can be simplified by substituting in the GP equations for

$\mu(x^*)$ and $\sigma^2(x^*)$:

$$E[\mu(x^*)\sigma^2(x^*)] = \sigma_n^2 \sum_i \beta_i l_i - \sum_i \sum_j \sum_k \mathbf{K}_{ij}^{-1} \beta_k O_{ijk} \quad (\text{A.8})$$

$$E[\mu(x^*)^3] = \sum_i \sum_j \sum_k \beta_i \beta_j \beta_k O_{ijk} \quad (\text{A.9})$$

$$E[\sigma^2(x^*)^2] = (\sigma_n^2)^2 - 2\sigma_n^2 \sum_i \sum_j \mathbf{K}_{ij}^{-1} L_{ij} + \quad (\text{A.10})$$

$$\sum_i \sum_j \sum_k \sum_d \mathbf{K}_{ij}^{-1} \mathbf{K}_{kd}^{-1} W_{ijkd}$$

$$E[\mu(x^*)^2 \sigma^2(x^*)] = \sigma_n^2 \sum_i \sum_j \beta_i \beta_j L_{ij} - \quad (\text{A.11})$$

$$\sum_i \sum_j \sum_k \sum_d \mathbf{K}_{ij}^{-1} \beta_k \beta_d W_{ijkd}$$

$$E[\mu(x^*)^4] = \sum_i \sum_j \sum_k \sum_d \beta_i \beta_j \beta_k \beta_d W_{ijkd} \quad (\text{A.12})$$

where $\boldsymbol{\beta} = [\beta_1, \dots, \beta_N]^T = \mathbf{K}^{-1} \mathbf{y}$ and O_{ijk} and W_{ijkd} are defined as:

$$O_{ijk} = \quad (\text{A.13})$$

$$\int C(x^*, x_i) C(x^*, x_j) C(x^*, x_k) \mathcal{N}(x^* | \mu_x, \Sigma_x) dx^*$$

$$W_{ijkd} = \quad (\text{A.14})$$

$$\int C(x^*, x_i) C(x^*, x_j) C(x^*, x_k) C(x^*, x_d) \mathcal{N}(x^* | \mu_x, \Sigma_x) dx^*$$

Since the GP Covariance function, $C(x_i, x_j)$, is of the squared exponential form, the integral expressions for O_{ijk} and W_{ijkd} can be expressed as the integral over the product of four and five Gaussian distributions respectively, with the appropriate normalization terms. Since the product of Gaussian distributions is itself a

Gaussian, the above expressions simplify to:

$$O_{ijk} = z_O \int \mathcal{N}(x^* | \bar{\mu}_O, \bar{\Sigma}_O) dx^* \quad (\text{A.15})$$

$$= z_O$$

$$W_{ijkd} = z_W \int \mathcal{N}(x^* | \bar{\mu}_W, \bar{\Sigma}_W) dx^* \quad (\text{A.16})$$

$$= z_W$$

where $\bar{\Sigma}_O = (3\Lambda^{-1} + \Sigma_x^{-1})^{-1}$ and $\bar{\Sigma}_W = (4\Lambda^{-1} + \Sigma_x^{-1})^{-1}$. The normalization terms, z_O and z_W , are defined as:

$$z_O = (\sigma_f^2)^3 |\bar{\Sigma}_O|^{\frac{1}{2}} |\Sigma_x|^{-\frac{1}{2}} \quad (\text{A.17})$$

$$\prod_{x_a \in \{x_i, x_j, x_k\}} \exp\left(-\frac{1}{2}(x_a - \mu_x)^T B_1^O (x_a - \mu_x)\right)$$

$$\prod_{x_b \in \{x_i, x_j, x_k\} | b < a} \exp\left(-\frac{1}{2}(x_a - x_b)^T B_2^O (x_a - x_b)\right)$$

$$z_W = (\sigma_f^2)^4 |\bar{\Sigma}_W|^{\frac{1}{2}} |\Sigma_x|^{-\frac{1}{2}} \quad (\text{A.18})$$

$$\prod_{x_a \in \{x_i, x_j, x_k, x_d\}} \exp\left(-\frac{1}{2}(x_a - \mu_x)^T B_1^W (x_a - \mu_x)\right)$$

$$\prod_{x_b \in \{x_i, x_j, x_k, x_d\} | b < a} \exp\left(-\frac{1}{2}(x_a - x_b)^T B_2^W (x_a - x_b)\right)$$

where $B_1^O = \Lambda^{-1} \bar{\Sigma}_O \Sigma_x^{-1}$, $B_1^W = \Lambda^{-1} \bar{\Sigma}_W \Sigma_x^{-1}$, $B_2^O = \Lambda^{-1} \bar{\Sigma}_O \Lambda^{-1}$, $B_2^W = \Lambda^{-1} \bar{\Sigma}_W \Lambda^{-1}$,

BIBLIOGRAPHY

- [1] Andrew R Runnalls. Kullback-leibler approach to gaussian mixture reduction. *Aerospace and Electronic Systems, IEEE Transactions on*, 43(3):989–999, 2007.
- [2] T.C. Folsom. Social ramifications of autonomous urban land vehicles. In *IEEE International Symposium on Technology and Society, May*, 2011.
- [3] R. Bishop. Intelligent vehicle applications worldwide. *Intelligent Systems and Their Applications, IEEE*, 15(1):78–81, 2000.
- [4] Z. Juan, J. Wu, and M. McDonald. Socio-economic impact assessment of intelligent transport systems. *Tsinghua Science & Technology*, 11(3):339–350, 2006.
- [5] Isaac Miller, Mark E. Campbell, Dan Huttenlocher, Frank-Robert Kline, Aaron Nathan, Sergei Lupashin, Jason Catlin, Brian Schimpf, Pete Moran, Noah Zych, Ephraim Garcia, Mike Kurdziel, and Hikaru Fujishima. Team cornell’s skynet: Robust perception and planning in an urban environment. *J. Field Robotics*, 25(8):493–527, 2008.
- [6] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher R. Baker, Robert Bittner, M. N. Clark, John M. Dolan, Dave Duggins, Tugrul Galatali, Christopher Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matthew McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul E. Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar,

- Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *J. Field Robotics*, 25(8):425–466, 2008.
- [7] Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles Reinholtz, Dennis Hong, Al Wicks, Thomas Alberi, David Anderson, et al. Odin: Team victortango’s entry in the darpa urban challenge. *Journal of Field Robotics*, 25(8):467–492, 2008.
- [8] John Markoff. Collision in the making between self-driving cars and how the world works, Jan 2012.
- [9] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J.Z. Kolter, D. Langer, O. Pink, V. Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168. IEEE, 2011.
- [10] Autonomous Labs. Autonomous car navigates the streets of berlin.
- [11] John Markoff. Google cars drive themselves, in traffic. *New York Times*, Oct 2010.
- [12] 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. *Journal of Field Robotics*, 25(10):775–807, 2008.
- [13] Jason Hardy and Mark Campbell. Contingency planning over probabilistic hybrid obstacle predictions for autonomous road vehicles. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2237–2242. IEEE, 2010.

- [14] B.D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J.A. Bagnell, M. Hebert, A.K. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3931–3936. IEEE, 2009.
- [15] Stephane Petti, Thierry Fraichard, Inria Rocquencourt, and Ecole Des Mines De Paris. Safe motion planning in dynamic environments. pages 3726–3731, 2005.
- [16] J.S. Choi, G. Eoh, J. Kim, Y. Yoon, J. Park, and B.H. Lee. Analytic collision anticipation technology considering agents’ future behavior. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1656–1661. IEEE.
- [17] T. Ohki, K. Nagatani, and K. Yoshida. Collision avoidance method for mobile robot considering motion and personal spaces of evacuees. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1819–1824. IEEE, 2010.
- [18] J. Miura and Y. Shirai. Probabilistic uncertainty modeling of obstacle motion for robot motion planning. *Journal of Robotics and Mechatronics*, 14(4):349–356, 2002.
- [19] C. Fulgenzi, A. Spalanzani, and C. Laugier. Probabilistic rapidly-exploring random trees for autonomous navigation among moving obstacles. In *Workshop on safe navigation, IEEE International Conference on Robotics and Automation (ICRA)*, 2009.
- [20] N.E. Du Toit and J.W. Burdick. Robotic Motion Planning in Dynamic, Clut-

- tered, Uncertain Environments. In *2010 IEEE Conference on Robotics and Automation*, pages 966–973, 2010.
- [21] Noel E Du Toit and Joel W Burdick. Robot motion planning in dynamic, uncertain environments. *Robotics, IEEE Transactions on*, 28(1):101–115, 2012.
- [22] D. Ferguson, M. Darms, C. Urmson, and S. Kolski. Detection, prediction, and avoidance of dynamic obstacles in urban environments. In *2008 IEEE Intelligent Vehicles Symposium*, pages 1149–1154, 2008.
- [23] D. Alspach and HW Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *Automatic Control, IEEE Transactions on*, 17(4):439–448, 1972.
- [24] S.J. Julier and J.K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, volume 3, page 26. Spie Bellingham, WA, 1997.
- [25] J.K. Uhlmann. *Dynamic map building and localization: New theoretical foundations*. PhD thesis, University of Oxford, 1995.
- [26] S. Julier and J.K. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. *Robotics Research Group, Department of Engineering Science, University of Oxford, Oxford, OC1 3PJ United Kingdom, Tech. Rep*, 1996.
- [27] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. IEEE, 2000.

- [28] D Musicki and Robin J Evans. Measurement gaussian sum mixture target tracking. In *Information Fusion, 2006 9th International Conference on*, pages 1–8. IEEE, 2006.
- [29] M. Šimandl and J. Dunk. Sigma point gaussian sum filter design using square root unscented filters. In *Preprints of the 16th IFAC World Congress*. IFAC. Prague, 2005.
- [30] H.W. Sorenson and D.L. Alspach. Recursive bayesian estimation using gaussian sums. *Automatica*, 7(4):465–479, 1971.
- [31] F. Havlak and M. Campbell. Discrete and continuous, probabilistic anticipation for autonomous robots in urban environments. In *Proceedings of SPIE*, volume 7833, page 78330H, 2010.
- [32] M.F. Huber. Adaptive gaussian mixture filter based on statistical linearization. 2011.
- [33] Rudolph Van Der Merwe. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis, University of Stellenbosch, 2004.
- [34] M.F. Huber and U.D. Hanebeck. Progressive Gaussian mixture reduction. In *Information Fusion, 2008 11th International Conference on*, pages 1–8. IEEE, 2008.
- [35] D.J. Salmond. Mixture reduction algorithms for point and extended object tracking in clutter. *Aerospace and Electronic Systems, IEEE Transactions on*, 45(2):667–686, April 2009. ISSN 0018-9251. doi: 10.1109/TAES.2009.5089549.

- [36] V. Garcia, F. Nielsen, and R. Nock. Levels of details for gaussian mixture models. *Computer Vision–ACCV 2009*, pages 514–525, 2010.
- [37] C. Hennig. Methods for merging gaussian mixture components. *Advances in Data Analysis and Classification*, 4(1):3–34, 2010.
- [38] Simo Ali-Löytty and Niilo Sirola. Gaussian mixture filter in hybrid navigation. In *Proceedings of The European Navigation Conference GNSS 2007*, pages 831–837, 2007.
- [39] Mark L Psiaki, Jonathan R Schoenberg, and Isaac T Miller. Gaussian mixture approximation by another gaussian mixture for” blob” filter re-sampling. *AIAA Paper*, (2010-7747):2–5, 2010.
- [40] Naonori Ueda, Ryohei Nakano, Zoubin Ghahramani, and Geoffrey E. Hinton. Smem algorithm for mixture models. *NEURAL COMPUTATION*, pages 200–0, 1999.
- [41] F. Faubel, J. McDonough, and D. Klakow. The split and merge unscented gaussian mixture filter. *Signal Processing Letters, IEEE*, 16(9):786–789, 2009.
- [42] Zhihua Zhang, Chibiao Chen, Jian Sun, and Kap Luk Chan. Em algorithms for gaussian mixtures with split-and-merge operation, 2003.
- [43] Mauro J Caputi and Richard L Moose. A modified gaussian sum approach to estimation of non-gaussian signals. *Aerospace and Electronic Systems, IEEE Transactions on*, 29(2):446–451, 1993.
- [44] Gabriel Terejanu, Puneet Singla, Tarunraj Singh, and Peter D Scott. Uncertainty propagation for nonlinear dynamic systems using gaussian mixture models. *Journal of guidance, control, and dynamics*, 31(6):1623, 2008.

- [45] S. Kullback. *Information theory and statistics*. Dover, 1968.
- [46] J. Goldberger and H. Aronowitz. A distance measure between gmms based on the unscented transform and its application to speaker recognition. In *Ninth European Conference on Speech Communication and Technology*. Citeseer, 2005.
- [47] J.R. Hershey and P.A. Olsen. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2007. ICASSP 2007*, volume 4, 2007.
- [48] J.L. Williams and P.S. Maybeck. Cost-function-based gaussian mixture reduction for target tracking. In *Proc. 6th Int. Conf. Inform. Fusion*, volume 2, pages 1047–1054, 2003.
- [49] J.H. Kotecha and P.M. Djuric. Gaussian sum particle filtering. *Signal Processing, IEEE Transactions on*, 51(10):2602–2612, 2003.
- [50] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.
- [51] United States Air Force. 2007 cliff data set. 2007. URL <https://www.sdms.afrl.af.mil/index.php?collection=clif2007>.
- [52] Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. 2011.
- [53] A. Girard, C.E. Rasmussen, J. Quinonero-Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs? application to multiple-step ahead time series forecasting. 2003.

- [54] Raquel Urtasun, David J Fleet, and Pascal Fua. 3d people tracking with gaussian process dynamical models. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 238–245. IEEE, 2006.
- [55] J. Ko and D. Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009.
- [56] J.Q. Candela, A. Girard, J. Larsen, and C.E. Rasmussen. Propagation of uncertainty in bayesian kernel models-application to multiple-step ahead forecasting. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2, pages II–701. IEEE, 2003.
- [57] J. Quinonero-Candela, A. Girard, and C.E. Rasmussen. *Prediction at an Uncertain Input for Gaussian Processes and Relevance Vector Machines Application to Multiple-Step Ahead Time-Series Forecasting*. IMM, Informatik og Matematisk Modelling, DTU, 2003.
- [58] Marc Peter Deisenroth. *Efficient reinforcement learning using gaussian processes*. KIT Scientific Publishing, 2009.
- [59] F. Havlak and M. Campbell. Discrete and continuous, probabilistic anticipation for autonomous robots in urban environments. *Robotics, IEEE Transactions on*, PP(99):1–14, 2013. ISSN 1552-3098. doi: 10.1109/TRO.2013.2291620.
- [60] Marco F Huber. Adaptive gaussian mixture filter based on statistical linearization. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–8. IEEE, 2011.

- [61] P. Groot, P. Lucas, and P. van den Bosch. Multiple-step time series forecasting with sparse gaussian processes. In *Causmaecker, P. De (ed.), Proceedings of the 23rd Benelux Conference on Artificial Intelligence*, page 1. SI: sn, 2011.
- [62] M. Althoff, O. Stursberg, and M. Buss. Model-based probabilistic collision detection in autonomous driving. *Intelligent Transportation Systems, IEEE Transactions on*, 10(2):299–310, 2009.
- [63] I. Hwang and CE Seah. Intent-Based Probabilistic Conflict Detection for the Next Generation Air Transportation System. *Proceedings of the IEEE*, 96(12):2040–2059, 2008.
- [64] K. Kim, D. Lee, and I. Essa. Gaussian process regression flow for analysis of motion trajectories. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1164–1171. IEEE, 2011.
- [65] J. Joseph, F. Doshi-Velez, A.S. Huang, and N. Roy. A bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31(4):383–400, 2011.
- [66] C. Laugier, I.E. Paromtchik, C. Tay, K. Mekhnacha, G. Othmezzouri, and H. Yanagihara. Collision risk assessment to improve driving safety.
- [67] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, MA, 2006.
- [68] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18:1257, 2006.
- [69] Neil Lawrence, Matthias Seeger, Ralf Herbrich, et al. Fast sparse gaussian

- process methods: The informative vector machine. *Advances in neural information processing systems*, pages 625–632, 2003.
- [70] Joaquin Quiñero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.
- [71] Wing Ip Tam, KN Plataniotis, and D Hatzinakos. An adaptive gaussian sum algorithm for radar tracking. *Signal processing*, 77(1):85–104, 1999.
- [72] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430, 2000.
- [73] J. Wang, D. Fleet, and A. Hertzmann. Gaussian process dynamical models. *Advances in neural information processing systems*, 18:1441, 2006.
- [74] Mathieu Salzmann and Raquel Urtasun. Implicitly constrained gaussian process regression for monocular non-rigid pose estimation. *Advances in Neural Information Processing Systems*, 23:2065–2073, 2010.
- [75] R. Schubert, C. Adam, M. Obst, N. Mattern, V. Leonhardt, and G. Wanielik. Empirical evaluation of vehicular models for ego motion estimation. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 534–539. IEEE, 2011.
- [76] E. Espié, C. Guionneau, B. Wymann, C. Dimitrakakis, R. Coulom, and A. Sumner. Torcs-the open racing car simulator, 2005. URL <http://torcs.sourceforge.net/>.
- [77] Mark Campbell, Magnus Egerstedt, Jonathan P How, and Richard M Murray. Autonomous driving in urban environments: approaches, lessons and challenges. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4649–4672, 2010.

- [78] Benjamin Johnson, Frank Havlak, Mark Campbell, and Hadas Kress-Gazit. Execution and analysis of high-level tasks with dynamic obstacle anticipation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 330–337. IEEE, 2012.
- [79] Charles C Macadam. Understanding and modeling the human driver. *Vehicle System Dynamics*, 40(1-3):101–134, 2003.
- [80] Alex Pentland and Andrew Liu. Modeling and prediction of human behavior. *Neural computation*, 11(1):229–242, 1999.
- [81] Dario D Salvucci. Modeling driver behavior in a cognitive architecture. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 48(2):362–380, 2006.
- [82] Jason Hardy, Frank Havlak, and Mark Campbell. Multiple-step prediction using a two stage gaussian process model. In *American Control Conference (ACC), 2014*. IEEE, 2014.
- [83] Alex J Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. 2000.
- [84] Matthias Seeger, Christopher KI Williams, and Neil D Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Workshop on AI and Statistics*, volume 9, page 2003, 2003.
- [85] Stefan Schneegass, Bastian Pfleging, Nora Broy, Frederik Heinrich, and Albrecht Schmidt. A data set of real world driving to assess driver workload. In *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '13, pages 150–

157, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2478-6. doi: 10.1145/2516540.2516561. URL <http://doi.acm.org/10.1145/2516540.2516561>.

- [86] M.L. Psiaki and M. Wada. Derivation and Simulation Testing of a Sigma-Points Smoother. *Journal of Guidance Control and Dynamics*, 30(1):78, 2007. ISSN 0731-5090.
- [87] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.
- [88] Chris Fraley and Tim Hesterberg. Least angle regression and lasso for large datasets. *Statistical Analysis and Data Mining*, 1(4):251–259, 2009.
- [89] M. McClelland and M. Campbell. Probabilistic modeling of anticipation in human controllers. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 43(4):886–900, July 2013. ISSN 2168-2216. doi: 10.1109/TSMCA.2012.2220541.