Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Fast Risk Assessment for Autonomous Vehicles Using Learned Models of Agent Futures

Allen Wang, Xin Huang, Ashkan Jasour, and Brian C. Williams

***Abstract*—This paper presents fast non-sampling based methods to assess the risk of trajectories for autonomous vehicles when probabilistic predictions of other agents' futures are generated by deep neural networks (DNNs). The presented methods address a wide range of representations for uncertain predictions including both Gaussian and non-Gaussian mixture models for predictions of both agent positions and controls. We show that the problem of risk assessment when Gaussian mixture models (GMMs) of agent positions are learned can be solved rapidly to arbitrary levels of accuracy with existing numerical methods. To address the problem of risk assessment for non-Gaussian mixture models of agent position, we propose finding upper bounds on risk using Chebyshev's Inequality and sums-of-squares (SOS) programming; they are both of interest as the former is much faster while the latter can be arbitrarily tight. These approaches only require statistical moments of agent positions to determine upper bounds on risk. To perform risk assessment when models are learned for agent controls as opposed to positions, we develop *TreeRing*, an algorithm analogous to tree search over the ring of polynomials that can be used to exactly propagate moments of control distributions into position distributions through nonlinear dynamics. The presented methods are demonstrated on realistic predictions from DNNs trained on the Argoverse and CARLA datasets and are shown to be effective for rapidly assessing the probability of low probability events.**

I. Introduction

In order for autonomous vehicles to drive safely on public roads, they need to predict the future states of other agents (e.g. human-driven vehicles, pedestrians, cyclists) and plan accordingly. Predictions, however, are inherently uncertain, so it is desirable to represent uncertainty in predictions of possible future states and reason about this uncertainty while planning. This desire is motivating ongoing work in the behavior prediction community to go beyond single mean average precision (MAP) prediction and develop methods for generating probabilistic predictions [1]–[4]. In the most general sense, this involves learning joint distributions for the future states of all the agents conditioned on their past trajectories and other context specific variables (e.g. an agent is at a stop light, lane geometry, the presence of pedestrians, etc). However, learning such a distribution can often be intractable, so current works use a wide variety of different simplified representations for probabilistic predictions. [3] trains a conditional Variational Autoencoder (CVAE) to generate samples of possible future trajectories. Other works use generative adversarial networks (GANs) to generate multiple trajectories with probabilities assigned to each of them [4], [5]. As a discrete alternative, [6], [7] train a DNN to generate a probabilistic occupancy grid map with a probability assigned to each cell. However, such grid-based approaches effectively treat possible agents' trajectories as belonging to a discrete space, while, in reality, agents may be at an uncountable number of points in continuous space. Many recent papers try to account for the continuous nature of uncertainty in space by learning (GMMs) for vehicle positions [1], [6], [8] or coefficients of polynomials in $\mathbb{R}^2$ that represent the vehicles' positions [9]. Since learning uncertain models for position or pose can also sometimes produce results that are inconsistent with basic kinematics, some recent works develop DNNs that predict future control inputs which are then propagated through a kinematic model to predict future positions [2], [10].

Given a probabilistic prediction, an autonomous vehicle still needs to be able to *rapidly* evaluate the probability of a given plan resulting in a collision or, more generally, a constraint violation. We will refer to this problem as *risk assessment* and it is particularly challenging in the context of autonomous driving as 1) autonomous vehicles need to reason about low probability events to be safer than human drivers and 2) there are hard real time constraints on algorithm latency. Latency is a critical consideration for safety and will be a major consideration motivating the methods presented in this paper. While an algorithm with a latency of, for example, one second would often be acceptable in other robotics applications, it would be unacceptable for an autonomous vehicle traveling at 20 m/s on public roads. This requirement of low latency while retaining the ability to reason about low probability events makes naive Monte Carlo computationally intractable. To address this problem, adaptive and importance sampling methods have been proposed to estimate these probabilities with fewer samples [11], [12]. However, such methods still do not usually simultaneously provide guarantees on both latency and error. Their performance can also be highly sensitive to algorithm parameters and proposal distributions.

Statement of Contributions: We present fast methods to assess the risk of trajectories for both Gaussian and non-Gaussian position and control models of other agents. In section IV, we begin by addressing the case when GMMs are used for agent position predictions. We show this particular case can be reduced to the problem of computing the CDF of a quadratic form in a multivariate Gaussian (QFMVG)- a well-studied problem in the statistics community for which methods exist that can rapidly solve it to arbitrary accuracy. To address the more general case when potentially non-Gaussian mixture models are used for agent position predictions, we

All authors are with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology {allenw, huangxin, jasour, williams} @ mit.edu

apply statistical moment-based approaches to determine upper bounds on risk, which we will refer to as *risk bounds*. Namely, we propose using Chebyshev's Inequality and a sums-of-squares (SOS) program that can be seen as a generalization of Chebyshev's Inequality; the former is faster, while the latter can provide arbitrarily tight risk bounds. These moment-based approaches have the feature of being *distributionally robust*, producing risk bounds that are true for all possible distributions that take on the value of the given moments. To address uncertain models for controls, in Section V, we develop *TreeRing*, a novel algorithm analogous to tree search, but over the ring of polynomials. Given a polynomial stochastic system, *TreeRing* can find polynomial expressions for moments of position in terms of moments of control random variables. This enables the application of our non-Gaussian position risk assessment methods to the problem of risk assessment when models are learned for agent controls. Figure 1 illustrates our framework in this case. In Section VI, we demonstrate our methods on realistic predictions generated by DNNs trained on the Argoverse and CARLA datasets [13], [14]. Source code can be found at https://github.com/allen-adastra/risk_assess.

II. Notation

Let $\mathcal{S}^n_{++}$ denote the set of $n \times n$ positive definite matrices. For any matrix $Q \in \mathcal{S}^n_{++}$ and vector $\mathbf{x} \in \mathbb{R}^n$, let $Q(\mathbf{x}) := \mathbf{x}^T Q \mathbf{x}$. Let Q_{ij} denote the element in the i_{th} row and j_{th} column of Q. For any $\theta \in \mathbb{R}$, let $R(\theta)$ be the 2D rotation matrix parameterized by θ. For a vector $\mathbf{x} \in \mathbb{R}^n$ and multi-index $\alpha \in \mathbb{Z}^n_+$, let $\mathbf{x}^\alpha = \prod_{i=1}^n x_i^{\alpha_i}$. Let $\mathbb{R}[\mathbf{x}]$ denote the ring of polynomials in $\mathbf{x}$ over $\mathbb{R}$. For $n \in \mathbb{N}$, let $[n] = \{k \in \mathbb{N} : k \leq n\}$. A polynomial $s(x) \in \mathbb{R}[x]$ is said to be sums-of-squares (SOS) if for some $l \in \mathbb{N}$, $\exists h_i(x) \in \mathbb{R}[x]$ for $i \in [l]$ s.t. $s(x) = \sum_{i=1}^l h_i(x)^2$. For a random vector $\mathbf{w}$ and any $d \in \mathbb{N}$, let $\mu_{\mathbf{w}_t}, \Sigma_{\mathbf{w}_t}$ denote its mean vector and covariance matrix respectively, and $\Phi_{\mathbf{w}}$ denote its characteristic function. For any set S, let $\mathcal{P}(S)$ denote the power set of S with the empty set removed, $|S|$ denote the cardinality of S, and S^n denote the *n-ary* Cartesian power. For a vector valued function f, f_i denotes the i_{th} component of f.

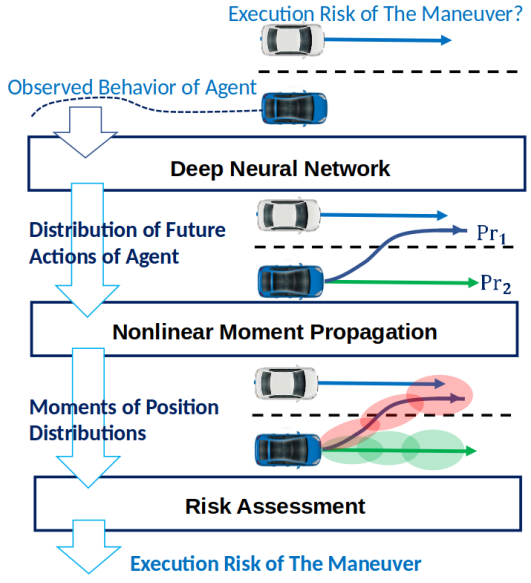

Fig. 1. Illustration of our risk assessment framework when control distributions are used for predictions. When position distributions are used, the nonlinear moment propagation step is skipped.

III. Problem Statement

We define *risk* as the probability of an agent entering an ellipsoid around the ego vehicle. Thus, we are interested in computing the probability of other agents entering:

$$\left\{\mathbf{x} \in \mathbb{R}^2 : Q(\mathbf{x}) \leq 1\right\}, \quad Q \in \mathcal{S}^2_{++} \tag{1}$$

We argue ellipsoids are a useful representation as: 1) they can be fit relatively tightly to the profiles of vehicles, and 2) the sizes of both the ego vehicle and agent can be accounted for by properly scaling the size of the ellipsoid around the vehicle. Throughout the paper, agent positions at each time step are always defined in the frame of the planned future poses of the ego vehicle unless stated otherwise; Section IV-A shows how moments of distributions can be expressed in different frames. Given this formulation, the ellipsoid is parameterized by a constant matrix $Q \in \mathcal{S}^2_{++}$ in the ego vehicle frame. In practice, multiple ellipsoids can be defined around the vehicle and an appropriate one selected at run-time. We restrict our focus to the single agent case and note that the risk in a multi-agent setting can be upper bounded by summing the risk associated with each agent.

If $\mathbf{x}_t = [x_t, y_t]^T$ is some random vector for the position of the agent at time t, then the risk associated with an agent across the whole T step time horizon is:

$$\mathcal{R} := \mathbb{P}\left(\bigcup_{t=1}^{T} \{Q(\mathbf{x}_t) \leq 1\}\right) \tag{2}$$

By the inclusion-exclusion principle, the probability (2) can be computed as the sum of the probabilities of the marginal events and the probabilities of all possible intersections of events:

$$\mathcal{R} = \sum_{J \in \mathcal{P}([T])} (-1)^{|J|+1} \mathbb{P}\left(\bigcap_{j \in J} \{Q(\mathbf{x}_j) \leq 1\}\right) \tag{3}$$

In many works, the random variables are assumed to be independent across time or can be made to be independent across time by conditioning on a discrete mode [1], [6], [8]. If there is dependence across time, one would need the conditional distributions of the events which require additional information to be learned. As most work on behavior prediction currently assumes independence across time, this paper restricts its focus to the time independent case, and so:

$$\mathcal{R} = 1 - \prod_{t \in [T]} \left(1 - \mathbb{P}(Q(\mathbf{x}_t) \leq 1)\right) \tag{4}$$

Thus, the problem of risk assessment along the trajectory can be solved by computing the marginals at each time step t, so the rest of the paper restricts its focus to the marginals.

IV. Risk Assessment

In this section, we present solutions for both Gaussian and non-Gaussian risk assessment when moments of the random vector for agent position $\mathbf{x}_t$ are known. We begin by addressing the problem of determining moments of agent positions in different frames to account for the ego vehicles planned trajectory. We then present our solution for the GMM case using numerical approximations of the CDFs of QFMVGs. To address the non-Gaussian case, we present methods based off Chebyshev's Inequality and SOS programming. We assume basic knowledge of SOS programming; We refer the reader to [15], [16] for an overview of SOS programming and [16]–[21] for moment-SOS based planning. Throughout this section, we assume the necessary moments of $\mathbf{x}_t$ are known.

A. Changing Frames

Predictions are usually given in a global frame, so this section provides a method for transforming the global frame distribution moments into the ego vehicle frame. More generally, we are concerned with computing moments of $\mathbf{x}_t$ in a new frame offset by $\mathbf{v} \in \mathbb{R}^2$ and rotated by $-\theta \in \mathbb{R}$. As shown in the appendix, if $\mathbf{x}_t$ is a mixture model, its moments can be computed in terms of moments of its components, so, in this section, let $\mathbf{x}_t$ be a component of a mixture model. We propose only translating the moments and then accounting for the rotation by using $Q^* = R(\theta)^T Q R(\theta)$ instead of Q. The rotation can be accounted for by using Q^* instead of Q because:

$$\mathbf{x}_t^T Q^* \mathbf{x}_t = \mathbf{x}_t^T R(\theta)^T Q R(\theta) \mathbf{x}_t \tag{5}$$

$$= (R(\theta)\mathbf{x}_t)^T Q (R(\theta)\mathbf{x}_t) \tag{6}$$

The translated moments can be computed by applying the binomial theorem to $(\mathbf{x}_t - \mathbf{v})^n$ (here, the power is applied element-wise). Note that applying the binomial theorem to $(\mathbf{x}_t - \mathbf{v})^n$ requires moments of $\mathbf{x}_t$ up to order n.

B. Risk Assessment for GMM Position Models

In this section, we provide a method to solve the risk assessment problem when the uncertain prediction is represented as a sequence of GMMs, $\mathbf{x}_t$, of the agents position with discrete modes determined by the Multinoulli Z_t. Many works currently learn GMMs for vehicle position as they express both multi-modal and continuous uncertainty [1], [6], [8]. As shown in Figure 2, they provide an intuitive representation of uncertainty in both the drivers high level decisions and low level execution. With time independence, the risk is:

$$\sum_{z=1}^{n} \left(1 - \prod_{t \in [T]} 1 - \mathbb{P}(Q(\mathbf{x}_t) \leq 1 : Z_t = z) \right) \mathbb{P}(Z_t = z) \tag{7}$$

Note that the above expression can be easily modified for the case when there is a single Multinoulli random variable that is constant across all time, an assumption used in, for example, [1]. The probabilities $\mathbb{P}(Z_t = z)$ are learned parameters of GMMs, so the problem of risk assessment can be solved by computing $\mathbb{P}\left(Q(\mathbf{x}_t) \leq 1 : Z_t = z\right)$ for each agent, time step, and mode. Note that this is exactly the CDF of $Q(\mathbf{x}_t)$ conditioned on $Z_t = z$ which is a quadratic form in a multivariate Gaussian (QFMVG). Unfortunately, there does not exist a known closed form solution to exactly evaluate the CDF of QFMVGs, but fast approximation methods with bounded errors have been studied within the statistics community [22]–[26]. Several of these methods have been implemented in the R package CompQuadForm [27]. Of particular interest is the method of Imhof, which produces results with bounded approximation error by numerical inversion of the characteristic function of the QFMVG [26]. A faster, but less accurate, alternative is the method of Liu-Tang-Zhang which involves approximating the CDF of the QFMVG with the CDF of a non-central chi square distribution with parameters chosen to minimize the difference in kurtosis and skew between the approximate and target distributions [22].

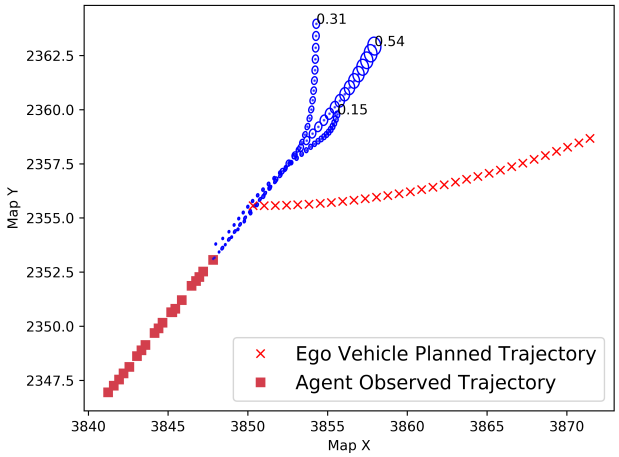

Fig. 2. An example risk assessment scenario. One standard deviation confidence ellipses (in blue) of a multi-modal GMM prediction are shown with mode probabilities. The observed agent trajectory and planned ego vehicle trajectory are also shown in red with different markers.

C. Non-Gaussian Risk Assessment with Chebyshevs Inequality

As a consequence of the one-tailed Chebyshev's Inequality, for any measurable function g, whenever $\mathbb{E}[g(\mathbf{x}_t)] > 0$, we have that:

$$\mathbb{P}(g(\mathbf{x}_t) \leq 0) \leq \frac{\mathbb{E}[g(\mathbf{x}_t)^2] - \mathbb{E}[g(\mathbf{x}_t)]^2}{\mathbb{E}[g(\mathbf{x}_t)^2]} \tag{8}$$

That is, the first two moments of $g(\mathbf{x}_t)$ are sufficient to establish a bound on the risk that the constraint $g(\mathbf{x}_t) \leq 0$ is violated. We note that the requirement $\mathbb{E}[g(\mathbf{x}_t)] > 0$ is not particularly restrictive because $\mathbb{E}[g(\mathbf{x}_t)] \leq 0$ means the average case involves collision, thus corresponding to what is usually an unacceptable level of risk.

1) Applying Chebyshev's Inequality to the Quadratic Form: To apply Chebyshev's inequality to $\mathbb{P}(Q(\mathbf{x}_t) - 1 \leq 0)$, we would need the first two moments of $Q(\mathbf{x}_t) - 1$ which can be expressed in terms of the first two moments of $Q(\mathbf{x}_t)$. The first moment can be expressed in terms of the mean vector and covariance matrix of $\mathbf{x}_t$ [28]:

$$\mathbb{E}[Q(\mathbf{x}_t)] = \mathrm{Tr}(Q\Sigma_{\mathbf{x}_t}) + \mu_{\mathbf{x}_t}^T Q \mu_{\mathbf{x}_t} \tag{9}$$

We can determine an expression for $\mathbb{E}[(Q(\mathbf{x}_t)^2]$ via an alternate representation for the quadratic form:

$$\mathbb{E}[Q(\mathbf{x}_t)^2] = \sum_{(i,j,k,l)\in[2]^4} Q_{ij}Q_{kl}\mathbb{E}\left[x_{t_i}x_{t_j}x_{t_k}x_{t_l}\right] \quad (10)$$

Thus, to compute the second moment of $Q(\mathbf{x}_t)$, we would need the moments of $\mathbf{x}_t$ of order up to four.

2) Conservative Approximation with Half-Spaces: It's possible to reduce the order of the moments that need to be propagated to two by instead approximating the ellipsoid as the intersection of n_h half-spaces parameterized by $\mathbf{a}_i \in \mathbb{R}^2$ and $b_i \in \mathbb{R}$. The approximated set is thus:

$$\mathcal{X}_{Approx} = \cap_{i=1}^{n_h}\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{a}_i^T\mathbf{x} + b_i \leq 0\} \quad (11)$$

Since the probability of any individual event is greater than the probability of the intersection of events, we have that:

$$\mathbb{P}(\cap_{i=1}^{n_h}\{\mathbf{a}_i^T\mathbf{x}_t + b_i \leq 0\}) \leq \min_{i\in[n_h]} \mathbb{P}(\mathbf{a}_i^T\mathbf{x}_t + b_i \leq 0) \quad (12)$$

So if we determine an upper bound on the probability of each $\mathbb{P}(\mathbf{a}_i^T\mathbf{x}_t + b_i \leq 0)$ with Chebyshev's Inequality, the minimum of the Chebyshev bounds will be an upper bound on our risk. Since $\mathbf{a}_i^T\mathbf{x}_t + b_i$ is an affine transformation of $\mathbf{x}_t$, its mean and variance can be expressed with the mean vector and covariance matrix of $\mathbf{x}_t$.

D. Non-Gaussian Risk Assessment with SOS Programming

When tighter risk bounds are desired than those obtained via Chebyshev's Inequality, for any measurable function g, an *univariate* SOS program can be used to upper-bound $\mathbb{P}(g(\mathbf{x}_t) \leq 0)$ – the SOS program is univariate in the sense that it searches for a polynomial in a single indeterminant, not in the sense that there is only one decision variable [19]. The fact that the SOS program is univariate is significant because the key disadvantages of SOS, scalability and conservatism, are not as limiting for univariate SOS because: 1) the number of decision variables in the resulting SDP scales quadratically w.r.t. the order of the polynomial we are searching for and 2) the set of non-negative univariate polynomials is equivalent to the set of univariate SOS polynomials, allowing univariate SOS to explore the full space of possible solutions.

We begin by noting that the probability of constraint violation is equivalent to the expectation of the indicator function on the sub-level set of g:

$$\mathbb{P}(g(\mathbf{x}_t) \leq 0) = \mathbb{E}[\mathbf{1}_{g(\mathbf{x}_t)\leq 0}] \quad (13)$$

The expectation of the indicator function, however, is not necessarily easily computable. To solve this problem, we find some polynomial with a more easily computable expectation that upper bounds the indicator function. If we can find some univariate polynomial, $p : \mathbb{R} \to \mathbb{R}$ of order d in some indeterminant $x \in \mathbb{R}$ with coefficients $c_k, k = 0, ..., d$ that upper bounds the indicator function, then clearly the following implication holds by substitution:

$$p(x) := \sum_{k=0}^{d} c_k x^k \geq \mathbf{1}_{x\leq 0} \Rightarrow \sum_{k=0}^{d} c_k g(\mathbf{x}_t)^k \geq \mathbf{1}_{g(\mathbf{x}_t)\leq 0} \quad (14)$$

Given the coefficients c_k, if we apply the expectation w.r.t. the density function of $\mathbf{x}_t$ to both sides, then we can reduce the problem of finding an upper bound on $\mathbb{P}(g(\mathbf{x}_t) \leq 0)$ to that of computing moments of the random variable $g(\mathbf{x}_t)$:

$$\sum_{k=0}^{d} c_k\mathbb{E}[g(\mathbf{x}_t)^k] \geq \mathbb{E}[\mathbf{1}_{g(\mathbf{x}_t)\leq 0}] = \mathbb{P}(g(\mathbf{x}_t) \leq 0) \quad (15)$$

The moments of $g(\mathbf{x}_t)$, in turn, are computable in terms of moments of $\mathbf{x}_t$ by expanding out the polynomial power and applying the linearity of expectation. For example, if $g(\mathbf{x}_t) = x_t^2 + y_t^2$, then:

$$\mathbb{E}[g(\mathbf{x}_t)^3] = \mathbb{E}[x_t^6] + 3\mathbb{E}[x_t^4y_t^2] + 3\mathbb{E}[x_t^2y_t^4] + \mathbb{E}[y_t^6] \quad (16)$$

In the case that $\mathbf{x}_t$ is a multivariate Gaussian, the higher order central moments and central cross moments can be computed in close form given the mean vector and covariance matrix. In the non-Gaussian case, there may not be a convenient way to determine the desired moments, but we note that the moments can often be computed using moment generating functions or sampling based approaches. In this section, we assume that we know the necessary moments of $\mathbf{x}_t$ to compute $\mathbb{E}[g(\mathbf{x}_t)^k], \forall k \in [d]$. We also normalize the moments, as doing so improves the numerical conditioning of the problem [1].

Now consider the following univariate SOS program in the indeterminant x which can search for the polynomial which minimizes the upper bound on risk.

$$\min_{p,s_1,s_2} \quad \sum_{k=0}^{d} c_k\mathbb{E}[g(\mathbf{x}_t)^k] \quad (17a)$$

$$p(x) - 1 = s_1(x) - xs_2(x) \quad (17b)$$

$$p(x), s_1(x), s_2(x) \quad SOS \quad (17c)$$

If the order of the polynomial is chosen to be $d = 2n$ for some $n \in \mathbb{N}$, then we should have that $\deg(s_1) = d$ and $\deg(s_2) = d - 2$. If $d = 2n + 1$ for some $n \in \mathbb{N}$, then we should have that $\deg(s_1) = 2n$ and $\deg(s_2) = 2n$. Note that the constraint (17b) enforces:

$$p(x) \geq 1 \quad \forall x \in (-\infty, 0] \quad (18)$$

And since $p(x)$ is constrained to be SOS, it is also globally non-negative so $p(x) \geq \mathbf{1}_{x\leq 0}, \forall x \in \mathbb{R}$. Thus, we can see that the optimal objective value of this SOS program yields an upper bound on $\mathbb{P}(g(\mathbf{x}_t) \leq 0)$.

V. Moment Propagation

While directly learning distributions for agents future positions can be an effective strategy, one major disadvantage is it can produce physically unrealistic predictions. [2], [10] address this by learning distributions for control inputs and then propagating samples through a kinematic model. While the Kalman filter and its variants, such as the extended and unscented Kalman filters, can be used to propagate mean and covariance, they are not exact and do not immediately apply to higher order moments [29]–[31].

[1] normalization is valid because $\mathbb{P}(X \leq 0) = \mathbb{P}(cX \leq 0)$ for $c > 0$

In this section, we provide an approach for nonlinear moment propagation that can, in principle, work for moments up to arbitrary order [20]. Given a nonlinear model and a random vector for control, $\mathbf{w}_t$, this section is concerned with the problem of computing statistical moments of $\mathbf{x}_t$ s.t. the non-Gaussian risk assessment methods presented in Section IV can be applied. More precisely, we are looking for moments of the form $\mathbb{E}[x_t^\alpha y_t^\beta]$ where $\alpha, \beta \in \mathbb{N}$. The only requirements we impose on $\mathbf{w}_t$ is that its entries have 1) bounded moments and 2) computable characteristic functions. In the case that mixture models are used, we show in the appendix that it is sufficient for the components of the mixture models to satisfy the requirements as the moments and characteristic functions of mixtures of random variables can be computed in terms of those of their components. We use a stochastic version of the discrete-time Dubin's car to both demonstrate the general approach and to address the problem of agent risk assessment:

$$x_{t+1} = x_t + v_t \cos(\theta_t) \tag{19a}$$
$$y_{t+1} = y_t + v_t \sin(\theta_t) \tag{19b}$$
$$v_{t+1} = v_t + w_{v_t} \tag{19c}$$
$$\theta_{t+1} = \theta_t + w_{\theta_t} \tag{19d}$$

Above, the control vector is $\mathbf{w}_t = [w_{v_t}, w_{\theta_t}]$ where w_{v_t} and w_{θ_t} are random variables describing the agent's acceleration and steering at time t and are assumed to be independent. $\mathbf{x}_t = [x_t, y_t]$ is the position of some reference point on the agent in a fixed frame, v_t is its speed, and θ_t is the angle of its velocity vector with respect to the fixed frame. The time steps Δt for discretization are omitted for brevity; the values of the variables can simply be scaled accordingly.

A. Motivating Example

To motivate *TreeRing*, we begin by showing how the dynamics of the moment $\mathbb{E}[x_{t+1}y_{t+1}]$ for the system (19) can be found manually. *TreeRing* is essentially an automated version of this process. By substituting the equations (19) in and applying the linearity of expectation, we arrive at the dynamics of our moment:

$$\begin{aligned}\mathbb{E}[x_{t+1}y_{t+1}] = \mathbb{E}[x_t y_t] + \mathbb{E}[v_t^2 \sin(\theta_t)\cos(\theta_t)] \\ + \mathbb{E}[x_t v_t \sin(\theta_t)] + \mathbb{E}[y_t v_t \cos(\theta_t)]\end{aligned} \tag{20}$$

Notice above that the term $\mathbb{E}[x_t y_t]$ shows up; if we find some way to compute the other three terms, we will arrive at an update relation with which we can compute $\mathbb{E}[x_t y_t]$ recursively. Turning our attention to the second term, we have that v_t is independent of θ_t since we assumed w_{v_t} and w_{θ_t} are independent at each time step. So the second term above can be factored by independence:

$$\mathbb{E}[v_t^2 \sin(\theta_t)\cos(\theta_t)] = \mathbb{E}[v_t^2]\mathbb{E}[\sin(\theta_t)\cos(\theta_t)] \tag{21}$$

In the appendix, we show how the quantities $\mathbb{E}[v_t^2]$, which is the sum of independent random variables, and $\mathbb{E}[\sin(\theta_t)\cos(\theta_t)]$ can be computed in terms of the characteristic functions of w_{v_t} and w_{θ_t}, which we assumed are known. In the last two terms, there are dependencies between the variables, so there is not a clear way to factor them down into computable moments. Our main idea is to derive the dynamics of $\mathbb{E}[x_{t+1}v_{t+1}\sin(\theta_{t+1})]$ and $\mathbb{E}[y_{t+1}v_{t+1}\cos(\theta_{t+1})]$ by substituting in the equations (19) much in the same way we did so for $\mathbb{E}[x_{t+1}y_{t+1}]$. We then simulate the dynamics of these additional moments in addition to $\mathbb{E}[x_{t+1}y_{t+1}]$; in a sense, we are producing a new dynamical system in terms of moments. In this case, recursively repeating this process of adding new moments to our dynamics produces a closed form set of equations that can recursively compute $\mathbb{E}[x_{t+1}y_{t+1}]$ in terms of moments of an initial state distribution and uncertainty at each time step. This process, however, is tedious and is easily subject to human error, especially for larger expressions. To address these issues, we developed *TreeRing* to algorithmically derive such expressions for moment propagation.

B. TreeRing: An Algorithm for Moment Propagation

TreeRing is an algorithm for polynomial stochastic systems, but, in many cases, we are interested in a nonlinear system f that is not polynomial, as is the case for (19). However, nonlinear systems can always be approximated as a polynomial system by applying taylor expansions. In the case of (19), the system can be made to be polynomial by applying the change of variables $c_t = \cos(\theta_t)$, $s_t = \sin(\theta_t)$, $c_{w_t} = \cos(w_t)$, and $s_{w_t} = \sin(w_{\theta_t})$:

$$x_{t+1} = x_t + v_t c_t \tag{22a}$$
$$y_{t+1} = y_t + v_t s_t \tag{22b}$$
$$v_{t+1} = v_t + w_{v_t} \tag{22c}$$
$$c_{t+1} = c_t c_{w_t} - s_t s_{w_t} \tag{22d}$$
$$s_{t+1} = s_t c_{w_t} + c_t s_{w_t} \tag{22e}$$

above, update relations for c_t and s_t were arrived at by using the trigonometric sums formulas to expand out $\cos(\theta_t + w_{\theta_t})$ and $\sin(\theta_t + w_{\theta_t})$ into the expressions shown.

Throughout this section, for any set of multi-indices, $\mathcal{Z} \subset \mathbb{N}^n$, let $\mathcal{Z}(\mathbf{b_t}) = \{\mathbb{E}[\mathbf{b_t}^\xi] : \xi \in \mathcal{Z}\}$ denote the set of corresponding moments. Given any $\xi \in \mathbb{N}^{n_b}$, suppose we want to be able to compute $\mathbb{E}[\mathbf{b}_t^\xi]$ for all t in some finite horizon. The key idea is to find some set of multi-indices corresponding to moments $\mathcal{Z} \subset \mathbb{N}^{n_b}$ and some set of scalar-valued polynomials $\mathcal{F} \subset \mathbb{R}[\mathbf{b}_t]$ s.t. $\xi \in \mathcal{Z}$ and $\forall m \in \mathcal{Z}(\mathbf{b}_{t+1})$, one of the following is true:

1) The value of m is known
2) $\exists f \in \mathcal{F}$ s.t. $m = f(\mathcal{Z}(\mathbf{b}_t))$

That is, every moment at time $t+1$ is either known or is a polynomial in the moments at time t. Thus, given $\mathcal{Z}(\mathbf{b}_t)$, we can directly compute $\mathcal{Z}(\mathbf{b}_{t+1})$ with basic mathematical operations (addition, multiplication, exponentiation). By induction, if $\mathcal{Z}(\mathbf{b}_0)$ is known, as is the case when the moments of the initial distribution are known or the initial state is deterministic, we can compute all the moments for the entire time horizon. The key function of *TreeRing* is to search for the appropriate set of moments, $\mathcal{Z}$ and corresponding set of dynamics equations $\mathcal{F}$.

An important property of the ring of polynomials is that it is closed under multiplication and addition. Thus, if we wanted to express the moment $\mathbb{E}[\mathbf{b}_{t+1}^{\xi}]$, for some $\xi \in \mathbb{N}^{n_b}$, in terms of quantities at time t, we have that $\exists p \in \mathbb{R}[\mathbf{b}_t]$ s.t:

$$\mathbb{E}[\mathbf{b}_{t+1}^{\xi}] = \mathbb{E}[p(\mathbf{b}_t)] \tag{23}$$

Letting $\mathcal{A} \subset \mathbb{N}^{n_b}$ denote the set of monomial exponents of the terms of p and $C_{\mathcal{A}} = \{c_\alpha \in \mathbb{R} : \alpha \in \mathcal{A}\}$ denote the set of coefficients, we have by the linearity of expectation that:

$$\mathbb{E}[\mathbf{b}_{t+1}^{\xi}] = \sum_{\alpha \in \mathcal{A}} c_\alpha \mathbb{E}[\mathbf{b}_t^{\alpha}] \tag{24}$$

Both $\mathcal{A}$ and $C_{\mathcal{A}}$ can be found by using compute algebra packages such as SymPy [32]. So we already have that $\mathbb{E}[\mathbf{b}_{t+1}^{\xi}]$ can be expressed as a polynomial in moments at time t. To further reduce the problem, we factor each $\mathbb{E}[\mathbf{b}_t^{\alpha}]$ into the product of moments of smaller expressions (for example, this is clearly doable in the case when the variables in $\mathbf{b}_t$ are all independent of each other). To provide a computational method to search for these factorization, we introduce the following definitions which are used in Proposition 1. Proposition 1 essentially states that factorization can be found by simple search algorithms on the dependence graph. In TreeRing, once a decomposition has been found, moments of the form $\mathbf{b}_t^{\alpha[c]}$ that are not already accounted for in $\mathcal{Z}$ are simply added. The expand subroutine is then called with $\alpha[c]$ input in the first argument. Algorithm 1 summarizes the algorithm.

Definition 1. Let $V_{\mathbf{b}}$ be the set of variables in $\mathbf{b}$, $n_b = |V_{\mathbf{b}}|$, and $E_{\mathbf{b}}$ be the set of undirected edges s.t. $(b_i, b_j) \in E_{\mathbf{b}}$ i.f.f. b_i and b_j are dependent. Then:

- The graph $G_{\mathbf{b}} = (V_{\mathbf{b}}, E_{\mathbf{b}})$ is the *dependence graph* of $\mathbf{b}$.
- Given any multi-index $\alpha \in \mathbb{N}^{n_b}$, *the subgraph of* $G_{\mathbf{b}}$ *induced by* α, denote it $G_{\mathbf{b}}[\alpha] = (V_{\mathbf{b}}[\alpha], E_{\mathbf{b}}[\alpha])$, is the sub-graph of $G_{\mathbf{b}}$ with the variables with non-zero power in $\mathbf{b}^{\alpha}$ as its vertex set.
- Let $\text{Comps}(G_{\mathbf{b}})$ denote the *set of vertices of connected components of* $G_{\mathbf{b}}$.
- For any component $c \in \text{Comps}(G_{\mathbf{b}})$, let $\alpha[c] \in \mathbb{N}^{n_b}$ denote the multi-index s.t. the i_{th} element of $\alpha[c]$ equals α_i if its corresponding $v \in V_{\mathbf{b}}$ is in c and zero otherwise.

Proposition 1. Given a vector of variables $\mathbf{b}$ with dependence graph $G_{\mathbf{b}}$ and given a multi-index $\alpha \in \mathbb{N}^{n_b}$:

$$\mathbb{E}[\mathbf{b}^{\alpha}] = \prod_{c \in \text{Comps}(G_{\mathbf{b}}[\alpha])} \mathbb{E}\left[\mathbf{b}^{\alpha[c]}\right] \tag{25}$$

Proof: First, note that $\sum_{c \in \text{Comps}(G_{\mathbf{b}}[\alpha])} \alpha[c] = \alpha$, so we have that:

$$\mathbb{E}[\mathbf{b}^{\alpha}] = \mathbb{E}\left[\prod_{c \in \text{Comps}(G_{\mathbf{b}}[\alpha])} \mathbf{b}^{\alpha[c]}\right] \tag{26}$$

Letting $c \in \text{Comps}(G_{\mathbf{b}}[\alpha])$, every node in c is independent of every node in $V_{\mathbf{b}}[\alpha]/c$ as the definition of connected components does not allow for the existence of edges from c to $V_{\mathbf{b}}[\alpha]/c$. Thus, the product above can be moved outside of the expectation operator. ■

Algorithm 1 TreeRing

```
procedure EXPAND(ξ, b, g, G_b, Z, F)
    A, C_A ← Represent b^ξ as a polynomial in b using g
    Add ξ to Z and add A, C_A to F
    for all α ∈ A do
        for all c ∈ Comps(G_b[α]) do
            if α[c] ∉ Z then
                Expand(α[c], b, g, G_b, Z, F)
            end if
        end for
    end for
end procedure
```

C. Solving the Motivating Problem with TreeRing

We return to the motivating problem of deriving an update relation for $\mathbb{E}[x_t y_t]$ by using *TreeRing*. We begin by transforming (19) into the polynomial system (22) by making the substitutions

The set of indeterminants, without the time index, is chosen to be:

$$V_{\mathbf{b}} = \{x, y, v, c, s, w_v, s_w, c_w\} \tag{27}$$

In the appendix, we show how the moments of all of the indeterminant variables except x and y can be computed to arbitrary order, so we initialize $\mathcal{Z}$ to contain the multi-indices corresponding to, for example, c^n, up to some large $n \in \mathbb{N}$. The edge set for this system under our assumptions is:

$$E_{\mathbf{b}} = \{(x,y),(x,v),(y,v),(x,s),(x,c),(y,s),(y,c)\} \tag{28}$$

After running algorithm (1), we arrive at the following $\mathcal{Z}(\mathbf{b})$:

$$\mathcal{Z}(\mathbf{b}) = \{xy, xs, ys, xc, yc, xvs, xvc, yvs, yvc\} \tag{29}$$

The result is that if we derive the dynamics for each moment in $\mathcal{Z}(\mathbf{b})$, that set of equations can be used to recursively compute the moments $\mathcal{Z}(\mathbf{b})$ at each time step using only moments of the initial state distribution and moments of s_w and c_w.

VI. Experiments

In this section, we demonstrate the performance of our system through two learning-based predictors that predict stochastic position and control for the target agent. For each predictor, we describe its network architecture and training procedure, before presenting risk assessment results. All computations were performed on a desktop with an Intel Core i9-7980XE CPU at 2.60 GHz. All Monte Carlo (MC) methods are implemented with vectorized NumPy operations to have a realistic assessment of run times for naive MC.

A. GMM Position Predictor

1) Model Description: To obtain probabilistic trajectory distributions of agents, we trained a simple DNN to generate Gaussian mixture model parameters for $\mathbf{x}_{1:T}$ over a fixed horizon $T = 30$ given a sequence of observed positions over 20 time steps. Although our framework works with any prediction model that outputs GMM parameters for positions, we aim to generate accurate and realistic predictions by selecting an encoder-decoder-based predictor that utilizes long short-term memory (LSTM) units because of the recent success of recurrent neural networks in trajectory prediction on different benchmarks [4], [8], [10], [33].

As shown in Figure 3, the encoder is a sequence of LSTM units taking observed trajectories of the target agent as input and outputs a latent vector encoding agent hidden state. The decoder is also a sequence of LSTM units that takes the latent vector and generates a set of GMM position parameters from each LSTM unit. For simplicity, we used three component mixture models. For each component, we generate a mean position vector and a covariance matrix representing uncertainties of predictions. The model is trained and validated on a subset of the Argoverse dataset [13].

2) Experiments: On a dataset of 500 scenarios similar to that shown in Figure 2, predictions were made and the risk was evaluated along a predefined trajectory for the ego vehicle. To evaluate QFMVG's, we tested both the methods of Imhof and Liu-Tang-Zhang. The methods proposed are much faster than naive Monte Carlo with far lower error. The method of Imhof with an error tolerance of 10^{-10} was used as ground truth [26]. Only 170 scenarios were used for error computation as results from scenarios with computed ground truth errors within tolerances (i.e: 10^{-10}) were neglected for error computation. We note that the method of Liu-Tang-Zhang empirically produces results with very small errors while being several times faster than the method of Imhof, which may prove useful in certain contexts.

B. GMM Control Predictor

1) Model Description: We use a similar DNN as the GMM position predictor, but the output becomes instead a set of GMM parameters for control signals defined in (19). Again, we assume a fixed number of three components in the mixture model for each control signal for the sake of simplicity. During model training, we obtain the ground truth control sequence by differentiating future positions of the target vehicle. Instead of using the Argoverse dataset, which has noisy differentiated control data due to noise in the perception system used to collect the dataset, we use our own data collected from a naturalistic driving simulator called CARLA [14] that provides accurate ground truth control values. The model is trained and validated on 10k samples collected in CARLA.

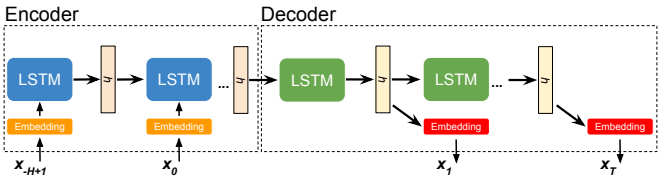

Fig. 3. Architecture diagram of GMM position predictor, including an LSTM-based encoder and an LSTM-based decoder. We introduce extra embedding layers to process the observed positions in the encoder and process the predicted hidden states before generating predicted parameters in the decoder.

TABLE I
RESULTS FROM EVALUATING RISKS IN 500 SCENARIOS. MEAN TIME IS FOR EVALUATING A THIRTY TIME STEP THREE MODE GMM PREDICTION. ERRORS CORRESPOND TO THE TIME STEP WITH THE MAXIMUM ERROR.

Method	**Mean Time (ms)**	**Mean Max. Absolute Error**	**Mean Max. Relative Error**
Imhof	91.21	0.0	0.0
Liu-Tang-Zhang	26.67	2.7×10^{-6}	2.3×10^{-4}
MC 10^4	106.9	6.7×10^{-4}	0.38
MC 5×10^4	422.5	2.7×10^{-4}	0.13
MC 10^5	1329	1.9×10^{-4}	0.12

2) TreeRing + Chebyshev Experiments: When deriving expressions for position moments up to order four, *TreeRing* returns 92 polynomial expressions while only 11 are needed to propagate the mean vector and covariance matrix. As these expressions currently require manual transcription into code, a process that is prone to human error, the half-space approximation method with 12 half-spaces was tested as it requires fewer expressions. The initial state of the agent vehicles was assumed to be known and deterministic. Random variables for control from the DNN and expressions from *TreeRing* were then used to compute the mean and covariance matrix of position at each time step. Over 50 scenarios, the mean time to evaluate the risk for a given trajectory for the Chebyshev method was 80ms while the Monte Carlo method with 10^6 samples took 140 seconds. The average worst-case conservatism of the Chebyshev risk estimate for a given time step along a trajectory was 0.012 (assuming the Monte Carlo results represent ground truth). Figure 4 shows the risk for both methods.

3) Comparing SOS + Chebyshev: Experiments were run to test and compare the Chebyshev and SOS methods described in (IV-C1) and (IV-D). For this experiment, higher order moments were obtained by automatic differentiation of the MVG moment generating function and the resulting moments of $Q(\mathbf{x}_t) - 1$ were normalized. YALMIP was used to transcribe the SOS programs into Semidefinite programs, and SeDuMi was used to solve the resulting semidefinite programs [34],

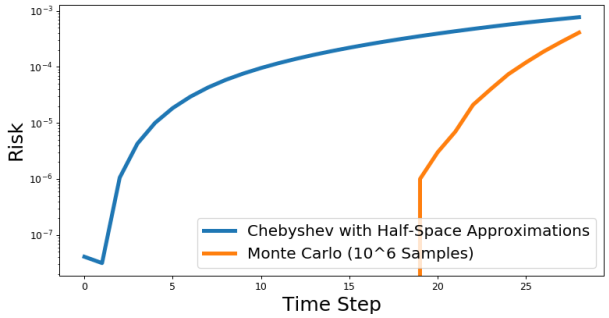

Fig. 4. Risk estimates across time for an example scenario using random variables from the GMM control predictor.

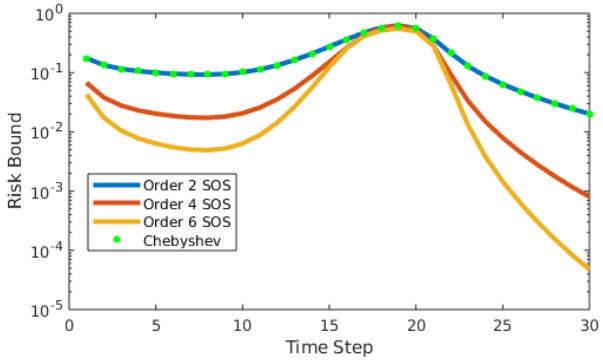

Fig. 5. Risk bounds computed with the SOS formulation from section IV-D compared with Chebyshev's inequality without half-space approximations.

[35]. We observe that 1) Chebyshev bound produces nearly the same result as the second order SOS program and 2) the SOS program with higher order moments can yield significantly better bounds, especially in the tails. The solve times for each time step only marginally increased for the higher order SOS programs; the mean solve times were 42, 44, and 49 ms for the second, fourth, and sixth order SOS formulations respectively. While these solve times are much better than those often encountered with SOS programs, further advances in performance are needed for this to be used online.

VII. Conclusions

Our experimental results with the methods for GMM position models and the Chebyshev method for non-Gaussian position and control models suggest that high performance implementations would be immediately practical for use in online applications. Future work should incorporate risk assessment into motion planning algorithms, but we note that it may be easily incorporated into standard algorithms with "collision check" primitives such as RRTs and PRMs [16]. The SOS method does significantly improve upon the risk bounds from the Chebyshev method at the cost of additional computation; future research should further develop methods to make the SOS step offline to improve runtimes for online applications [19]. Future work in both prediction and risk assessment should also work towards relaxing assumptions such as time independence.

Acknowledgements

This work was supported in part by Boeing grant MIT-BA-GTA-1 and by the Masdar Institute grant 6938857. Allen Wang was supported in part by a NSF Graduate Research Fellowship.

VIII. Appendix

A. Moments and Characteristic Functions of Mixture Models

Letting f_X denote the pdf of a K component mixture model, X, with component pdfs $f_{X_i}, \forall i \in [K]$ and letting f_Z denote the pdf of the K category Multinoulli, by definition $f_X(x) = \sum_{i=1}^{m} f_{X_i}(x) f_Z(i)$. For any measurable function, g, by interchanging the order of integration and summation

$$\mathbb{E}[g(X)] = \int g(x) f_X(x) dx \tag{30}$$

$$= \sum_{i=1}^{K} f_Z(i) \int g(x) f_{X_i}(x) dx \tag{31}$$

By letting $g(X) = X^n$ or $g(X) = e^{itX}$, we have that the moments and characteristic function of X can both be computed as the weighted sum of those of their components.

B. Moments of Trigonometric Variables

In this section, we show how moments of the form $\mathbb{E}[\cos^n(X)]$, $\mathbb{E}[\sin^n(X)]$, and $\mathbb{E}[\cos^m(X)\sin^n(X)]$ can be computed in terms of the characteristic function of the random variable X, Φ_X. We begin by applying Euler's Identity to the definition of the characteristic function:

$$\begin{aligned} \Phi_X(t) &= \mathbb{E}[e^{itX}] \\ &= \mathbb{E}[\cos(tX)] + i\mathbb{E}[\sin(tX)] \end{aligned} \tag{32}$$

Thus, we have that $\mathbb{E}[\cos(tX)] = \text{Re}(\Phi_X(t))$ and $\mathbb{E}[\sin(tX)] = \text{Im}(\Phi_X(t))$. This immediately gives us the ability to compute the first moments of our trigonometric random variables. For higher moments, the trigonometric power formulas can be used to express quantities of the form $\cos^n(X)$ as the sum of quantities of the form $\cos(mX)$ where $m \in \mathbb{N}$ and similarly for $\sin^n(X)$ [36]. Thus, higher moments of $\sin(X)$ and $\cos(X)$ can be computed using $\Phi_X(t)$. Moments of the form:

$$\mathbb{E}[\cos^m(X)\sin^n(X)] \tag{33}$$

can also ultimately be computed in terms of $\Phi_X(t)$. This can be seen if we make the substitutions $\cos(X) = \frac{1}{2}(e^{ix} + e^{-ix})$ and $\sin(X) = \frac{1}{2i}(e^{ix} - e^{-ix})$, then (33) can be expressed as:

$$\mathbb{E}\left[\frac{1}{i^n 2^{m+n}}(e^{iX} + e^{-iX})^m (e^{iX} - e^{-iX})^n\right] \tag{34}$$

By applying the binomial theorem to both expressions in parentheses, and multipying the resulting expressions, we find the entire expression in the expectation operator can be expressed as a polynomial in e^{iX} and e^{-iX}. Thus, the entire expression can be written as the sum of terms of the form $\mathbb{E}[e^{itX}]$ for $t \in \mathbb{Z}$ which is in the definition of $\Phi_X(t)$. In practice, computer algebra packages were used to derive these expressions when needed.

C. Sum of Independent Random Variables

In the case that our random variable X is a sum of independent random variables Y_i and a constant c, $X = c + \sum_{i\in[n]} Y_i$, the characteristic function Φ_X can be expressed as such:

$$\Phi_X(t) = e^{itc} \prod_{i\in[n]} \Phi_{Y_i}(t) \tag{35}$$

REFERENCES

[1] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *Conference on Robot Learning (CoRL)*, 2019.

[2] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2P2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 772–788.

[3] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 336–345.

[4] X. Huang, S. G. McGill, J. A. DeCastro, B. C. Williams, L. Fletcher, J. J. Leonard, and G. Rosman, "Diversity-aware vehicle motion prediction via latent semantic sampling," *arXiv preprint arXiv:1911.12736*, 2019.

[5] J. Li, H. Ma, and M. Tomizuka, "Interaction-aware multi-agent tracking and probabilistic behavior prediction via adversarial learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6658–6664.

[6] J. Hong, B. Sapp, and J. Philbin, "Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8454–8462.

[7] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *Robotics: Science and Systems*, 2019.

[8] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1179–1184.

[9] X. Huang, S. G. McGill, B. C. Williams, L. Fletcher, and G. Rosman, "Uncertainty-aware driver trajectory prediction at urban intersections," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9718–9724.

[10] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley, and N. Djuric, "Deep kinematic models for physically realistic prediction of vehicle trajectories," *arXiv preprint arXiv:1908.00219*, 2019.

[11] E. Schmerling and M. Pavone, "Evaluating trajectory collision probability through adaptive importance sampling for safe motion planning," in *Robotics: Science and Systems*, 2017.

[12] J. Norden, M. O'Kelly, and A. Sinha, "Efficient black-box assessment of autonomous vehicle safety," *arXiv preprint arXiv:1912.03618*, 2019.

[13] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8748–8757.

[14] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[15] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical programming*, vol. 96, no. 2, pp. 293–320, 2003.

[16] A. Jasour, "Risk aware and robust nonlinear planning," *Course Notes for MIT 16.S498, rarnop.mit.edu*, 2019.

[17] A. Jasour and B. C. Williams, "Risk contours map for risk bounded motion planning under perception uncertainties," in *2019 Robotics: Science and System (RSS)*, Germany, 2019.

[18] ——, "Sequential convex chance optimization for flow-tube based control of probabilistic nonlinear systems," in *2019 IEEE Conference on Decision and Control (CDC)*, France, 2019.

[19] A. Jasour, A. Hofmann, and B. C. Williams, "Moment-sum-of-squares approach for fast risk estimation in uncertain environments," in *2018 IEEE Conference on Decision and Control (CDC)*, 2445–2451, 2018.

[20] A. Jasour, "Convex approximation of chance constrained problems: Application in systems and control," in *Dissertation in School of Electrical Engineering and Computer Science, The Pennsylvania State University*, 2016.

[21] A. Jasour, N. S. Aybat, and C. Lagoa, "Semidefinite programming for chance constrained optimization over semialgebraic sets," in *SIAM Journal on Optimization*, 25(3), 1411–1440, 2015.

[22] H. Liu, Y. Tang, and H. H. Zhang, "A new chi-square approximation to the distribution of non-negative definite quadratic forms in non-central normal variables," *Computational Statistics & Data Analysis*, vol. 53, no. 4, pp. 853–856, 2009.

[23] H. Solomon and M. A. Stephens, "Distribution of a sum of weighted chi-square variables," *Journal of the American Statistical Association*, vol. 72, no. 360a, pp. 881–885, 1977.

[24] S. Kotz, N. L. Johnson, and D. Boyd, "Series representations of distributions of quadratic forms in normal variables II. Non-central case," *The Annals of Mathematical Statistics*, vol. 38, no. 3, pp. 838–848, 1967.

[25] P. Duchesne and P. L. De Micheaux, "Computing the distribution of quadratic forms: Further comparisons between the liu–tang–zhang approximation and exact methods," *Computational Statistics & Data Analysis*, vol. 54, no. 4, pp. 858–862, 2010.

[26] J.-P. Imhof, "Computing the distribution of quadratic forms in normal variables," *Biometrika*, vol. 48, no. 3/4, pp. 419–426, 1961.

[27] P. L. De Micheaux, "Compquadform: distribution function of quadratic forms in normal variables," *R package version*, vol. 1, no. 3, 2017.

[28] S. B. Provost and A. Mathai, *Quadratic forms in random variables: theory and applications*. M. Dekker, 1992.

[29] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee, 2000, pp. 153–158.

[30] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," 1961.

[31] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[32] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh *et al.*, "Sympy: symbolic computing in python," *PeerJ Computer Science*, vol. 3, p. e103, 2017.

[33] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961–971.

[34] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[35] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization methods and software*, vol. 11, no. 1-4, pp. 625–653, 1999.

[36] D. Zwillinger, *CRC standard mathematical tables and formulae*. Chapman and Hall/CRC, 2002.