

Interactive multi-modal motion planning with Branch Model Predictive Control

Yuxiao Chen, *Member, IEEE*, Ugo Rosolia, *Member, IEEE*, Wyatt Ubellacker, Noel Csomay-Shanklin, and Aaron D. Ames, *Fellow, IEEE*

Abstract—Motion planning for autonomous robots and vehicles in presence of uncontrolled agents remains a challenging problem as the reactive behaviors of the uncontrolled agents must be considered. Since the uncontrolled agents usually demonstrate multimodal reactive behavior, the motion planner needs to solve a continuous motion planning problem under these behaviors, which contains a discrete element. We propose a branch Model Predictive Control (MPC) framework that plans over feedback policies to leverage the reactive behavior of the uncontrolled agent. In particular, a scenario tree is constructed from a finite set of policies of the uncontrolled agent, and the branch MPC solves for a feedback policy in the form of a trajectory tree, which shares the same topology as the scenario tree. Moreover, coherent risk measures such as the Conditional Value at Risk (CVaR) are used as a tuning knob to adjust the tradeoff between performance and robustness. The proposed branch MPC framework is tested on an autonomous vehicle planning problem in simulation, and on an autonomous quadruped robot alongside an uncontrolled quadruped in experiments. The result demonstrates interesting human-like behaviors, achieving a balance between safety and performance.

I. INTRODUCTION

Motion planning is one of the central modules of autonomous robots and vehicles. In particular, as the autonomous agent shares the environment with uncontrolled agents, the reactive behaviors of the uncontrolled agents need to be taken into account in the motion planning. Take autonomous vehicles as an example: as they share the road with uncontrolled vehicles, pedestrians, and cyclists, and all of whom would adjust their behaviors based on other agents around them, the motion planner for the autonomous vehicle then needs to plan trajectories that are safe yet not overly conservative in presence of other road users. The challenges of achieving safe interactive motion planning in presence of uncontrolled agents are twofold: (i) the reactive behavior of the uncontrolled agents need to be properly modelled (ii) the reactive behaviors need to be leveraged in the motion planning.

The modelling of reactive behaviors is a challenging problem that has attracted increasing attention recently. Many uncontrolled agents encountered by autonomous agents exhibit highly nondeterministic and multimodal behaviors. For instance, a human driver may choose totally different behaviors under the same situation at different times (swerve left or right, yield or not yield). Therefore, accurately modeling these nondeterministic behaviors is almost impossible. Stochastic models, naturally, have been proposed to model the nondeterministic reactive behavior, such as Markovian models [1]–[3],

The authors are with the Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA, USA. chenyx,urosolia,wubellac,noelcs,ames@caltech.edu

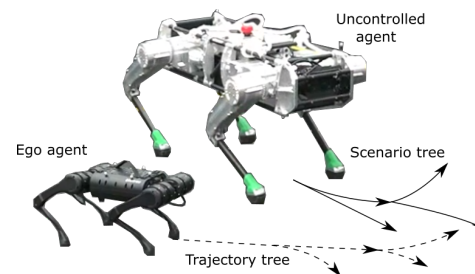


Fig. 1: Branch MPC on quadrupeds

and generative models [4], [5]. Another class of approaches is the set-based method that models the set of possible behaviors, including the GAN-based prediction [6], and classifier-based approaches [7]–[10].

Once a predictive model is obtained, in most cases, the motion planner is at the downstream of the predictive model. The typical strategy is to use some form of robust motion planning algorithm to let the autonomous agent avoid the reachable set of the uncontrolled agent [11] or the set of all possible trajectories [8]. Obviously, the robust formulations lead to conservative motion plans and compromised performance as it ignores the reactivity of the uncontrolled agent within the prediction horizon. In particular, a typical predictive model takes the scenario description as input and outputs the prediction of the uncontrolled agent’s behavior, and these behaviors are assumed to be fixed by the motion planner under the robust planning setup. This is not a deliberate choice, but rather a compromise due to the difficulty of reasoning about future reactive behaviors. Since the future scenario is nondeterministic as the behavior of the uncontrolled agent is nondeterministic, the reasoning of the behavior of the uncontrolled agent in future prediction steps is usually convoluted and intractable. On the other hand, when the input and state spaces are discrete, as in Markov decision processes (MDPs) and Partially Observed Markov Decision Processes (POMDPs), future scenarios can be enumerated and the planning can be solved with dynamic programming [12].

Inspired by policy planning and disturbance feedback techniques in the MPC literature [13], [14], we propose a branching Model Predictive Control (MPC) framework that combines the continuous motion planning with discrete modes representing the multimodal reactive behaviors of the uncontrolled agent. To obtain tractability, the continuous spectrum of possible behaviors of the uncontrolled agent is simplified with a finite set of policies, which are feedback control laws that represent the multimodality of the reactive behavior. On top of this finite set of policies, we assume that a predictive

model is given that outputs the probability of each policy given the scenario. We then construct a scenario tree by forward propagating the policies of the uncontrolled agent and solve for a trajectory tree with the same topology. Furthermore, we use coherent risk measures such as the Conditional Value at Risk (CVaR) in presence of the possibly inaccurate predictive model to improve the robustness of the policy. The branching idea for MPC can date back to [14]. In [14], the authors showed that optimizing over a tree of trajectories is equivalent to optimizing over feedback policies, when the goal is to minimize the worst case cost and the uncertainties acting on the system are uni-modal.

The multi-modal case was studied in [15]–[18], where the authors leveraged the optimization over a tree of trajectories to optimize over a set of feedback policies. However in these works the interaction between the controlled agent and the environment was not considered, meaning that the probability associated with the uncertainty modes determining the different tree branches is fixed. The advantage of the proposed branch MPC framework are (i) the reactive behavior of the uncontrolled agent in future steps is leveraged via the branching structure (ii) the MPC formulation can handle highly dynamic systems and generate high-quality motion plans (iii) the risk level acts as a convenient tuning knob reflecting the confidence of the reactive model and enables tradeoff between robustness and performance.

II. PRELIMINARIES

We first review the major tools used in our framework.

A. Model predictive control

Model Predictive Control (MPC) is a well-established control methodology that leverages forecast to compute actions [19], [20]. While MPC has been used for servo-level control in robotics [21], [22], we use MPC for the trajectory-level planning. In MPC, at each time t the controller plans a sequence of open-loop actions to minimize an objective function, and then the first predicted action is applied to the system. More formally, at each time t given the state of the system $x(t)$ the action is computed solving the following *Finite Time Optimal Control Problem (FTOCP)*:

$$\begin{aligned} \min_{u_0, \dots, u_{N-1}} & \sum_{k=0}^{N-1} h(x_k, u_k) + V(x_N) \\ \text{s.t.} & \quad x_{k+1} = f(x_k, u_k), \forall k \in \{0, \dots, N-1\}, \\ & \quad x_k \in \mathcal{X}, u_k \in \mathcal{U}, \forall k \in \{0, \dots, N-1\}, \\ & \quad x_0 = x(t), x_N \in \mathcal{X}_N, \end{aligned}$$

where the discrete time update $x_{k+1} = f(x_k, u_k)$ describes the evolution of the state $x_k \in \mathbb{R}^n$ when the input $u_k \in \mathbb{R}^d$ is applied to the system. The stage cost $h : \mathbb{R}^n \times \mathbb{R}^d \rightarrow \mathbb{R}$ and the terminal cost $V : \mathbb{R}^n \rightarrow \mathbb{R}$ are defined by the control designer, $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{U} \subset \mathbb{R}^d$, and $\mathcal{X}_N \subset \mathbb{R}^n$ are the state, input, and terminal constraint sets, respectively. At time t , let

$$[u_0^*(x(t)), \dots, u_{N-1}^*(x(t))]$$

be the optimal input sequence to the above FTOCP under the current systems's state $x(t)$. Then, the first optimal action is applied, resulting in the following MPC policy:

$$\pi^{\text{MPC}}(x(t)) = u_0^*(x(t)).$$

B. Risk-aware decision making

Risk-aware decision making concerns an optimization problem with stochastic outcomes [23]. Rather than optimizing over the expectation of the cost function, risk-aware optimization optimizes over a risk measure, which is a functional of the probability distributions over the cost function. Formally, consider a probability space (Ω, \mathcal{F}, P) where Ω is a Borel measurable space with a σ -algebra \mathcal{F} and probability function P . The cost function then can be understood as a function $X : \Omega \rightarrow \mathbb{R}$. We let \mathcal{X} denote the linear space of all \mathcal{F} -measurable functions. A risk measure $\rho(\cdot)$ is a functional that maps from \mathcal{X} to the extended real line $\{-\infty\} \cup \mathbb{R} \cup \{+\infty\}$. In particular, one useful class of risk measures are the *coherent risk measures*, defined below.

Definition 1. A risk measure ρ is a *coherent risk measure* if it satisfies the following properties

- **Convexity:** $\forall X, Y \in \mathcal{X}, \forall \lambda \in [0, 1], \rho(\lambda X + (1-\lambda)Y) \leq \lambda \rho(X) + (1-\lambda)\rho(Y)$
- **Monotonicity:** $\forall X, Y \in \mathcal{X}$ with $Y \geq X, \rho(Y) \geq \rho(X)$.
- **Translation equivariance:** For all $c \in \mathbb{R}$ and $X \in \mathcal{X}$, $\rho(X + c) = \rho(X) + c$
- **Positive homogeneity:** For all $t > 0, X \in \mathcal{X}, \rho(tX) = t\rho(X)$.

As a counterexample, value at risk (VaR), which is defined as $\text{VaR}_{1-\alpha}(X) := \min_{z \in \mathbb{R}: P(X \geq z) \leq \alpha} z$ is not a coherent risk measure as it is not convex. Examples of coherent risk measures include the conditional value at risk (CVaR), entropic value at risk (EVaR) [24], [25], and expectation.

Definition 2. For a random variable $X : \Omega \rightarrow \mathbb{R}$, the conditional value at risk is defined as

$$\begin{aligned} \text{CVaR}_{1-\alpha}(X) &:= \frac{1}{\alpha} \int_0^\alpha \text{VaR}_{1-\gamma}(X) d\gamma \\ &= \inf_{z \in \mathbb{R}} \left\{ z + \frac{1}{\alpha} \int_{-\infty}^\infty [x - z]_+ dP(x) \right\}, \end{aligned} \quad (1)$$

where $dP(x)$ is the probability density function of X .

CVaR is the expectation of the α percent worst outcomes of X , $\alpha \in [0, 1]$ is a parameter that determines how risk-averse the CVaR is and CVaR is monotonically decreasing w.r.t. α . Moreover, $\lim_{\alpha \rightarrow 0} \text{CVaR}_{1-\alpha}(X) = \text{ess sup } X$ and $\text{CVaR}_0(X) = \mathbb{E}[X]$. We chose to use CVaR due to its nice interpretability and computation properties.

It is shown in [26], [27] that any coherent risk measure has a dual representation as the maximization of expectation over a probability ambiguity set, which often turns out more convenient in computation, i.e., $\rho(X) = \max_{Q \in \mathcal{A}} \mathbb{E}_Q[X]$.

\mathcal{A} is the ambiguity set, which is a set of probability distributions. For CVaR, the ambiguity set is

$$\mathcal{A} = \left\{ Q \left| \begin{array}{l} \forall x \in \mathbb{R}, dQ(x) \geq 0, \\ \int dQ(x) dx = 1, dQ(x) \leq \frac{1}{\alpha} dP(x) \end{array} \right. \right\}. \quad (2)$$

When the distribution is discrete, the ambiguity set reduces to $\mathcal{A} = \{Q|Q(x_i) \geq 0, \sum Q(x_i) = 1, Q(x_i) \leq \frac{1}{\alpha}P(x_i)\}$. Risk-aware decision making then looks for the decision variable that minimizes some risk measure. The typical solution is via dualization, which we present in detail in Section IV.

III. BRANCH MPC

To fully utilize the reactivity of the uncontrolled agent, the control strategy needs to adapt to the behavior of the uncontrolled agent and reason about its future reaction to the environment. The classic model predictive control formulation optimizes over a single trajectory with a finite horizon, which is not able to leverage the reactivity. [14] proposed a min-max model predictive control which, instead of optimizing over a single trajectory, searches for a policy that reacts to different disturbance signals via enumerating the extreme cases of the disturbance signal. The enumeration generates a branching tree structure that grows exponentially with the planning horizon, and a min-max problem is solved to obtain the robust MPC policy. The branching enumeration bears some similarity to the forward dynamic programming method used in many decision-making problems such as the Markov decision processes and Partially observed Markov decision processes. The difference is that in Markov models, a transition probability is associated with each branch, and the goal is the expectation of the reward instead of the worst-case performance.

The branch MPC proposed in this paper extends the branch enumeration strategy proposed in [14] and associates it with a probabilistic characterization of the branches via a predictive model. To be specific, a finite set of policies are propagated forward to generate a scenario tree representing possible future behaviors of the uncontrolled agent. The branch MPC then optimizes over feedback policies in the form of a trajectory tree, which shares the same topology as the scenario tree. Each branch in the trajectory tree is the instantiation of the feedback policy under the uncontrolled agent's behavior characterized by the corresponding branch in the scenario tree.

We consider the scenario with one ego agent under control and one uncontrolled agent. A scenario tree is constructed by enumerating the behavior of the uncontrolled agent, and the branching probability is a function of the state of the ego agent and the uncontrolled agent.

Remark 1. When there are multiple uncontrolled agents, one needs to consider the product space of their behaviors. Due to the exponential complexity, some pruning protocol is needed, which is beyond the scope of this paper. We plan to tackle the case with multiple uncontrolled agents in future works.

We let $x \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ denote the state of the ego agent and $z \in \mathcal{Z} \subseteq \mathbb{R}^{n_z}$ denote the state of the uncontrolled agent, following known dynamics:

$$x^+ = f(x, u), \quad z^+ = g(z, d), \quad (3)$$

where $u \in \mathcal{U}$ is the ego agent input, $d \in \mathcal{D}$ is the uncontrolled agent input.

Policies for the uncontrolled agent. In order to enable branching within the prediction horizon, the behavior of the uncontrolled agent needs to be enumerable. However, using

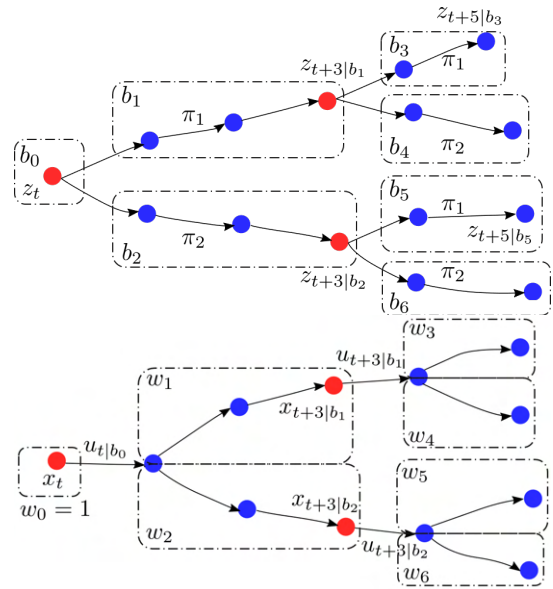


Fig. 2: Example scenario tree with two policies, branching nodes are in red and non-branching nodes are in blue

discrete actions, even motion primitives, can cause the behavior of the uncontrolled agent to be stiff and unnatural. Motivated by the fact that human driving behaviors demonstrate strong multimodality, a finite set of policies $\Pi = \{\pi_i\}_{i=1}^m$ is used to represent the possible behaviors of the uncontrolled agent, which consists of feedback policies $\pi_i : \mathcal{Z} \rightarrow \mathcal{D}$. Take highway driving as an example: the set of policies can include *maintain fixed speed*, *slow down*, *left lane change* and *right lane change*. Note that the feedback policies only depend on z , making it possible to be propagated forward independently of other agents in the scene.

Predictive model and scenario tree. Given a policy set, z can be propagated forward with a selected policy. A scenario tree is constructed by enumerating the policies. To be specific, a scenario tree starts at the root, which is the current state of the uncontrolled agent z and propagates forward. Certain nodes in the tree are selected as branching nodes, which would have m children, each following one of the policies in Π . A non-branching node only has one child, following its current policy. In particular, the root is always a branching node, and we use a simple strategy where branching happens every M steps. Ideally, one wants all the nodes in the scenario tree to be branching nodes. However, this naive choice may lead to an unnecessarily large number of nodes, causing computation issues. A subset of consecutive nodes following the same policy is called a branch. A branch in the scenario tree ends at a branching node or a leaf node (nodes with no children) and contains all its non-branching predecessors up to its first predecessor that is a branching node.

The top figure of Fig. 2 shows an example of a scenario tree with $m = 2, M = 3$ (each branching node has 2 children, branching happens every 3 time steps), the dashed squares show the branches in the tree. The branch MPC would construct another tree, denoted as the trajectory tree, of the ego agent trajectory whose topology is similar to the scenario tree, each node contains the planned future state of the ego

agent, which are variables of the MPC FTOCP, as shown in the bottom figure of Fig. 2. b_0 is the root branch with children b_1 and b_2 , and b_1 has two children, b_3 and b_4 , and so on. \mathbf{I} denotes the set of indices of all the branches in the scenario tree. For the one in Figure 2, $\mathbf{I} = \{0, 1, 2, 3, 4, 5, 6\}$.

Remark 2. Due to the causality constraint, before the uncontrolled agent demonstrate which policy it executes, the MPC shall not take different actions. Therefore, the node directly following a branching node in the trajectory tree is shared by multiple branches.

For a branch b_i , w_i denotes its weight, the root always has weight $w_0 = 1$, and at every branching point, a predictive model then assigns weight to the children of the branching node based on the x and z predictions. Take the branching from b_1 to b_3 and b_4 as an example,

$$\begin{aligned} w_3 &= w_1 P(\pi_1 | x_{t+3|b_1}, z_{t+1|b_1}), \\ w_4 &= w_1 P(\pi_2 | x_{t+3|b_1}, z_{t+1|b_1}), \end{aligned} \quad (4)$$

where $x_{t+3|b_1}$ is the state prediction at $t+3$ in branch b_1 , $P(\pi|x, z)$ is the probability of the uncontrolled agent taking π under the scenario described by x and z , which is given by the predictive model, and $\sum_{i=1}^m P(\pi_i, x, z) = 1$ always holds. An evaluation of the scenario tree refers to a path from the root to one of the leaf nodes. For example, $b_0 \rightarrow b_1 \rightarrow b_3$ is an evaluation of the scenario tree shown in Fig. 2.

Branch MPC setup. For notational clarity, we let t_i^0 and t_i^f denote the time instances of the first and last node in b_i , and let $\mathbf{x}_i = [x_{t_i^0|b_i}, \dots, x_{t_i^f|b_i}]$, $\mathbf{z}_i = [z_{t_i^0|b_i}, \dots, z_{t_i^f|b_i}]$, $\mathbf{u}_i = [u_{t_i^0|b_i}, \dots, u_{t_i^f|b_i}]$ be the ego agent trajectory, uncontrolled agent trajectory, and ego agent control input sequence of b_i . $\text{Pre}(\cdot)$ denotes the parent of a branch and $\text{Ch}(\cdot)$ denote the set of children of a branch, e.g., $\text{Pre}(b_1) = b_0$, $\text{Ch}(b_1) = \{b_3, b_4\}$. Sometimes with a slight abuse of notation, we use $\text{Ch}(\cdot)$ to denote the set of indices of the children branches. When the context is clear, we also use $P(b_i | \mathbf{x}_{\text{Pre}(i)}, \mathbf{z}_{\text{Pre}(i)})$ to denote the branching probability of b_i given the \mathbf{x} and \mathbf{z} of its parent branch. For each branch, a cost function $J_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i)$ and a set of constraints $\mathcal{C}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i) \leq 0$ are considered. With slack variables, the extended local cost function is:

$$\bar{J}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i) = J_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i) + \beta \mathbf{1}^\top [\mathcal{C}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i)]_+,$$

where $[\cdot]_+ = \max\{0, \cdot\}$, β is the slack penalty. Given the ego agent state x_t and the predicted trajectory \mathbf{z} for all $i \in \mathbf{I}$, the overall branch MPC FTOCP is then:

$$\begin{aligned} \min_{\{\mathbf{x}_i, \mathbf{u}_i\}} \quad & \sum_{i \in \mathbf{I}} w_i(\mathbf{x}, \mathbf{z}) \bar{J}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i) \\ \text{s.t.} \quad & x_{t|b_i} = f(x_{t-1|b_i}, u_{t-1|b_i}), \forall i \in \mathbf{I}, \forall t = t_i^0 + 1, \dots, t_i^f, \\ & (5a) \end{aligned}$$

$$x_{t_k^0|b_k} = f(x_{t_k^0-1|b_k}, u_{t_k^0-1|b_k}), \forall k \in \mathbf{I} \setminus \{0\}, \quad (5b)$$

$$w_0 = 1, w_i = w_{\text{Pre}(i)} P(b_i | \mathbf{x}_{\text{Pre}(i)}, \mathbf{z}_{\text{Pre}(i)}) \quad (5c)$$

$$x_{t_0^0} = x_t, z_{t_0^0} = z_t, \mathbf{x}_i \in \mathcal{X}, \mathbf{u}_i \in \mathcal{U} \quad (5d)$$

where (5a) enforces the dynamics within the branches, (5b) enforces the dynamics on the connection between any branch and its parent. We use $w_i(\mathbf{x}, \mathbf{z})$ to indicate that the branch

weight depends on the planned trajectory \mathbf{x} and the predicted \mathbf{z} . (5c) encodes the weights of the branches with the predictive model, (5d) is the constraint on the initial condition, state and input bounds. As previously noted in Remark 1, for causality, since all children share the same input at every branching point, the first node of all children branches must share the same state.

The optimization in (5) tries to minimize the expected total cost over the prediction horizon, which is the weighted sum of the cost over all branches in the scenario tree. In general, (5) is nonconvex, and we use Sequential Quadratic Program (SQP) to solve it by linearizing the dynamics, the constraints, and the weights of the branches. To be specific, at every iteration, the solution from the last iteration is used as the linearization point. The dynamics after linearization is simply

$$x_{t+1|b_i} = A_{t|b_i} x_{t|b_i} + B_{t|b_i} u_{t|b_i} + C_{t|b_i},$$

where $A_{t|b_i} = \frac{\partial f(x, u)}{\partial x} |_{\hat{x}_{t|b_i}, \hat{u}_{t|b_i}}$, $B_{t|b_i} = \frac{\partial f(x, u)}{\partial u} |_{\hat{x}_{t|b_i}, \hat{u}_{t|b_i}}$, $C_{t|b_i} = f(\hat{x}_{t|b_i}, \hat{u}_{t|b_i}) - A_{t|b_i} \hat{x}_{t|b_i} - B_{t|b_i} \hat{u}_{t|b_i}$. State and input values shifted backwards from the last iteration are given by $\hat{x}_{t|b_i}, \hat{u}_{t|b_i}$, i.e., $\hat{x}_{t|b_i}$ is the solution of the state associated with its children in the scenario tree from the last iteration. The constraints are linearized and enforced as linear inequality constraints (with slack). The cost function follows a Linear Quadratic format as $(x - x_{ref})^\top Q (x - x_{ref}) + u^\top R u$ with PSD matrices Q and R . Although the quadratic cost does not need to be linearized, the product of the cost and the weight w_i on every branch needs to be linearized:

$$w_i(\mathbf{x}_i, \mathbf{z}_i) \bar{J}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i) \approx \frac{\partial w_i}{\partial \mathbf{x}} |_{\hat{\mathbf{x}}_i} \bar{J}_i(\hat{\mathbf{x}}_i, \mathbf{z}_i, \hat{\mathbf{u}}_i) + w_i(\hat{\mathbf{x}}_i, \mathbf{z}_i) \bar{J}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i), \quad (6)$$

where $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{u}}_i$ are the solution from the last iteration shifted backwards. The first term on the RHS of (6) represents the change of the weighted cost function achieved by changing w_i , and the second term represents the cost with the weight calculated with the motion plan from the last iteration. This decomposition of cost encourages the ego agent to not only minimize the weighted cost by minimizing \bar{J} , but also put more weight on branches with smaller costs via influencing the weights given by the predictive model. This phenomenon is shown later in the vehicle highway simulation where the ego vehicle would ‘nudge’ forward before an overtaking to reduce the probability that the uncontrolled vehicle changes lane and blocks the overtaking.

The SQP after linearization for branch MPC is then

$$\begin{aligned} \min_{\mathbf{x}_i \in \mathcal{X}, \mathbf{u}_i \in \mathcal{U}} \quad & \sum_{i \in \mathbf{I}} \frac{\partial w_i}{\partial \mathbf{x}} |_{\hat{\mathbf{x}}_i} \bar{J}_i(\hat{\mathbf{x}}_i, \mathbf{z}_i, \hat{\mathbf{u}}_i) + w_i(\hat{\mathbf{x}}_i, \mathbf{z}_i) \bar{J}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i) \\ \text{s.t.} \quad & x_{t|b_k} = A_{t-1|b_k} x_{t-1|b_k} + B_{t-1|b_k} u_{t-1|b_k} + C_{t-1|b_k}, \\ & \quad \quad \quad \forall i \in \mathbf{I}, \forall t = t_i^0 + 1, \dots, t_i^f, \\ & x_{t'_k|b_k} = A_{t'|b_j} x_{t'|b_j} + B_{t'|b_j} u_{t'|b_j} + C_{t'|b_j}, \\ & \quad \quad \quad \forall k \in \mathbf{I} \setminus \{0\}, b_j = \text{Pre}(b_k), t' = t_k^0 - 1, \end{aligned} \quad (7)$$

After each iteration, only the first input is used as control command and the MPC solver replans at every iteration. However, note that the root is the ancestor of the whole scenario tree and the first input influences all the subsequent branches, indicating that the branch MPC policy takes all the subsequent branches in the scenario tree into account and leverages the reactive behavior in future predictions steps.

IV. BRANCH MPC WITH RISK MEASURE OBJECTIVES

The previous setup of branch MPC is essentially minimizing the expectation of the cost over the scenario tree, which is risk-neutral. Suppose one is not very confident about the predictive model, or one is more cautious, i.e., cares more about the bad outcome of the scenario tree; then, a risk-averse policy is preferred. By optimizing over a risk measure of the cost function, more focus is put on the worse outcomes. We use CVaR, which is the mean of the α -percentile worst outcomes. The risk-averseness is conveniently tuned by the parameter α , which is used as a tuning knob for the tradeoff between performance and robustness in our MPC setup.

As mentioned in Section II-B, the dual representation of the risk measure gives another interpretation of the CVaR optimization. Take a discrete random variable X as an example, and suppose there are N outcomes ξ_1, \dots, ξ_N with probabilities p_1, \dots, p_N and $\sum p_i = 1$. The expectation is $\mathbb{E}[X] = \sum_i p_i \xi_i$, and using the dual representation, the CVaR of X is

$$\text{CVaR}_{1-\alpha}(X) = \max_{P \in \mathcal{A}} \mathbb{E}_P[X],$$

$$\mathcal{A} = \{q \in \mathbb{R}^N \mid q_i \geq 0, \sum_i q_i = 1, q_i \leq \frac{1}{\alpha} p_i\}.$$

The ambiguity set \mathcal{A} can be viewed as a set of probability distributions similar to the assumed distribution of X , and α determines the maximum allowed difference. From this view point, the CVaR optimization is a robust optimization over distributions close to the one given by the predictive model.

The stochasticity of the branch MPC comes from the predictive model, which determines the weights of the branches. Since we work with a scenario tree with multiple stages of branching, we adopt the nested risk measure discussed in [15], [28]. In particular, let ϕ_i denote the cost incurred by all the subsequent branches of b_i , which is a random variable whose value depends on the evaluation of the scenario tree, i.e., which policy the uncontrolled agent choose in reality. For example, for b_0 in Fig. 2, $\phi_0 = \bar{J}_1 + \bar{J}_3$ if the evaluation is $b_0 \rightarrow b_1 \rightarrow b_3$, and the probability associated with this evaluation is w_3 . The risk measure is then defined in a nested way. Let ρ_i be the risk measure of ϕ_i and let \mathcal{A}_i denote the ambiguity set corresponding to the risk measure, then

$$\rho_i(\phi_i) = \max_{Q \in \mathcal{A}_i} \mathbb{E}_Q[\bar{J}_{\text{Succ}(i)} + \rho_{\text{Succ}(i)}(\phi_{\text{Succ}(i)})], \quad (8)$$

where $\text{Succ}(i)$ is the succeeding branch of b_i , which is a random variable, and \mathcal{A}_i is a set of probability distributions over $\text{Succ}(i)$, determined by the risk measure. Take the scenario tree in Fig. 2 as an example: it contains 3 layers, each branching node has two possible successors. Then

$$\begin{aligned} J &= \bar{J}_0 + \rho_0(\phi_0) \\ \rho_0(\phi_0) &= \max_{Q \in \mathcal{A}_0} Q(b_1)(\bar{J}_1 + \rho_1(\phi_1)) + Q(b_2)(\bar{J}_2 + \rho_2(\phi_2)) \\ \rho_1(\phi_1) &= \max_{Q \in \mathcal{A}_1} Q(b_3)\bar{J}_3 + Q(b_4)\bar{J}_4 \\ \rho_2(\phi_2) &= \max_{q \in \mathcal{A}_2} Q(b_5)\bar{J}_5 + Q(b_6)\bar{J}_6. \end{aligned} \quad (9)$$

If ρ_i is taken to be $\text{CVaR}_\alpha(\cdot)$, the ambiguity set is simply $\mathcal{A}_i = \{Q \mid \forall j \in \text{Ch}(b_i), Q(b_j) \geq 0, \sum_{j \in \text{Ch}(i)} Q(b_j) =$

$1, Q(b_j) \leq \frac{1}{\alpha} P(b_j \mid \mathbf{x}_i, \mathbf{z}_i)\}$, where $P(b_j \mid \mathbf{x}_i, \mathbf{z}_i)$ is the branching probability at b_i . Note that ρ_1 and ρ_2 are nested in the definition of ρ_0 .

Under the dual formulation, the optimization over risk measures is essentially minimizing the worst-case expectation over distributions in the ambiguity set, which is a distributionally robust optimization (DRO) [29]. It turns out that CVaR has a convenient conic representation [15] and with Lagrange duality, one can enforce the nested risk measure as convex constraints via the epigraph representation.

Theorem 1. *The risk-averse branch MPC with a CVaR cost function can be solved with the following optimization.*

$$\begin{aligned} \min_{\mathbf{x}_i, \mathbf{u}_i, \gamma_i, \mu_i^+, \mu_i^-, \sigma_i} & \bar{J}_0(\mathbf{x}_0, \mathbf{z}_0, \mathbf{u}_0) + \gamma_0 \\ \text{s.t.} & \forall i \in \mathbf{I}, \forall t = t_i^0 + 1, \dots, t_i^f, x_{t|b_i} = f(x_{t-1|b_i}, u_{t-1|b_i}) \\ & \forall i \in \mathbf{I} \setminus \{0\}, x_{t_i^0|b_i} = f(x_{t_i^0-1|b_i}, u_{t_i^0-1|b_i}), \\ & \forall i \in \mathbf{I}, \gamma_i = -\sigma_i + \frac{1}{\alpha} p_{\pi_j|b_i}^\top \mu_i^-, \mu_i^+ \geq 0, \mu_i^- \geq 0, \quad (10a) \\ & \forall j \in \text{Ch}(b_i), \bar{J}_j(\mathbf{x}_j, \mathbf{z}_j, \mathbf{u}_j) \leq -\sigma_i - \mu_{i,j}^+ + \mu_{i,j}^- - \gamma_j. \quad (10b) \end{aligned}$$

The proof can be found in the extended version at [arXiv:2109.05128](https://arxiv.org/abs/2109.05128). We omit the linearized branch MPC with risk measure cost as the derivation follows exactly as (7). When \bar{J} is quadratic, (10) is solved as a second order cone programming after linearization.

V. BRANCH MPC FOR HIGHWAY DRIVING

This section presents some implementation details of the branch MPC on the highway driving example.

Predictive model. We use a simple predictive model inspired by the backup control barrier function work in [30] where the idea is to forward integrate the dynamics following a policy, and check collision along the trajectory. To be specific, for every branching node in the scenario tree, let x and z denote the ego agent and the uncontrolled agent states, respectively. Given a fixed horizon T and a set of policies $\Pi = \{\pi_i\}_{i=1}^m$, we propagate the uncontrolled agent state z under all policies in Π , and let \mathbf{z}_{π_i} be the trajectory propagated under π_i for the uncontrolled agent. Let \mathbf{x}_{π_i} denote the planned trajectory in the branch MPC corresponding to the child branch under π_i , both \mathbf{x}_{π_i} and \mathbf{z}_{π_i} last for T steps. Then given safety constraints \mathcal{C} including collision avoidance and lane boundary constraints for the uncontrolled agent, a safety function $h : \mathcal{X}^T \times \mathcal{Z}^T \rightarrow \mathbb{R}$ is defined. h is defined such that $h(\mathbf{x}, \mathbf{z}) \geq 0$ indicates that the uncontrolled agent satisfies \mathcal{C} , otherwise it violates \mathcal{C} , and the smaller h is, the more \mathbf{z} violates \mathcal{C} . the predictive model is simply defined with a softmax function:

$$P(\pi_i | x, z) = \frac{\exp(\min\{h(\mathbf{x}_{\pi_i}, \mathbf{z}_{\pi_i}), \eta\})}{\sum_j \exp(\min\{h(\mathbf{x}_{\pi_j}, \mathbf{z}_{\pi_j}), \eta\})}. \quad (11)$$

where η is a parameter that saturates h such that all safe scenarios would have similar probability.

Implementation with automatic differentiation. To speed up the computation, our implementation heavily relies on automatic differentiation to obtain the gradient. In particular, we

use CasADi [31] as the computation engine for differentiating the dynamics, the constraints, and the branching probability. In particular, to obtain the branching probability, the scenario tree is propagated forward under Π in closed-form and the safety function h is defined with the softmin function instead of the min function over multiple constraints so that CasADi can perform automatic differentiation. Since we wrote all these functions as compositions of basic functions and lookup tables (which can also be handled by CasADi), the computation for the values and gradients is negligible comparing to the solving time of the SQP. For the branch MPC with expectation cost function in (5), OSQP [32] is used to solve the SQP after linearization; for the branch MPC using CVaR cost function in (10), ECOS [33] is called to solve the SOCP. The code can be found on <https://github.com/chenyx09/belief-planning>.

Overtaking and lane change. Overtaking and lane change is a common task for autonomous vehicles where the ego vehicle needs to overtake the uncontrolled vehicle and perform a lane change to cut in front. A unicycle model is used as the dynamic model for both the ego vehicle and the uncontrolled vehicle:

$$x = \begin{bmatrix} X \\ Y \\ v \\ \psi \end{bmatrix} \quad \dot{x} = \begin{bmatrix} v \cos(\psi) \\ v \sin(\psi) \\ a \\ r \end{bmatrix}, \quad (12)$$

where X , Y are the longitudinal and lateral coordinates, v is the velocity, ψ is the heading angle. a and r are the acceleration and yaw rate, which are the control input. The dynamics for the uncontrolled vehicle is exactly the same. The continuous-time dynamics is discretized to obtain the discrete-time model $x^+ = f(x, u)$. The cost function is a typical LQR type quadratic cost, i.e.,

$$J = (x - x_{ref})^T Q (x - x_{ref}) + u^T u,$$

where no penalty is put on X . The state constraints are

$$\begin{aligned} Y_{\min} \leq Y \leq Y_{\max}, \quad \psi_{\min} \leq \psi \leq \psi_{\max}, \\ \frac{\Delta X e^{\kappa \Delta X} + \Delta Y e^{\kappa \Delta Y}}{e^{\kappa \Delta X} + e^{\kappa \Delta Y}} \geq 1, \end{aligned} \quad (13)$$

where $\kappa > 0$ is a constant, $\Delta X = \frac{|X - X_z|}{\Delta X_{\max}}$, $\Delta Y = \frac{|Y - Y_z|}{\Delta Y_{\max}}$, X_z, Y_z are the coordinates of the uncontrolled vehicle, ΔX_{\max} and ΔY_{\max} are the longitudinal and lateral clearance. We use the softmax function instead of the max function so that the expression can be differentiated by automatic differentiation.

The uncontrolled vehicle has three policies, *maintain fixed speed*, *slow down*, and *lane change*, where the direction of *lane change* is towards the ego vehicle as lane change towards the other direction is of no danger to the ego vehicle. During the simulation, the uncontrolled vehicle would update its policy with a fixed frequency, and we tested two update rules: (i) the uncontrolled vehicle randomly sample its policy according to the predictive model in (4) (ii) the uncontrolled vehicle always pick the most likely policy, and the simulation results are similar.

Fig. 3 shows the snapshots from the simulation of the overtaking and lane change task under the branch MPC controller with the CVaR objective function. α is chosen

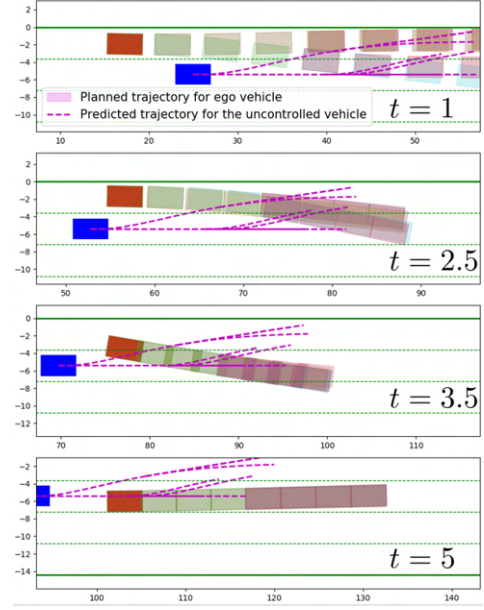


Fig. 3: Simulation of the overtaking and lane change under branch MPC

to be 0.9 ($\text{CVaR}_{0.1}(\cdot)$), which is relatively risk-neutral. As shown in the snapshots, the branch MPC plans a tree of trajectories, corresponding to the multiple possible behaviors in the scenario tree. Fig. 4 shows the weight changing of b_1, b_2 , and b_3 changing over time, which corresponds to the *maintain fixed speed*, *slow down*, and *lane change* policies, respectively (they all have children branches, but their children branches' weights are omitted in the plot). The ego vehicle demonstrates an interesting ‘nudging’ behavior at the beginning since there is a substantial probability that the uncontrolled vehicle may change to the left lane. Since the branch corresponding to the left lane change has a high cost, the gradient of the branching probability in (7) motivates the ego vehicle to nudge forward so that the weight on b_3 reduces. After 2.5 seconds, by nudging forward, the branch MPC determines that the probability of a lane change from the uncontrolled vehicle is very low and eventually it performs the overtaking and cut in front of the uncontrolled vehicle. After the overtaking, the different branches in the trajectory tree converge since all branches of the scenario tree pose little to no danger of collision.

As a benchmark, we tested the same task under a robust MPC controller, which only optimizes over one trajectory and tries to avoid all possible trajectories of the uncontrolled vehicle. Due to this limitation, the vehicle is stuck behind the uncontrolled vehicle and is afraid to perform the overtake, as shown in Fig. 5.

To demonstrate the influence of α , we run the same simulation with $\alpha = 0.1$, which is very risk-averse, and the result is similar to that under the robust MPC, shown in Fig. 6. Since the ambiguity set is much larger under the small α , the worst-case (maximum) probability of the uncontrolled vehicle performing a lane change to the left is larger, preventing the branch MPC from performing the overtake.

Another task we tested the branch MPC on was merging. However, due to the page limit, the results are omitted

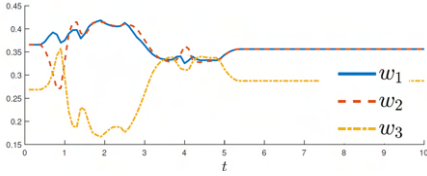


Fig. 4: Weights of b_1, b_2 , and b_3 over time

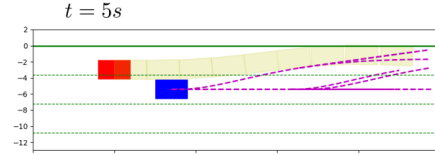


Fig. 5: Simulation of overtaking under robust MPC

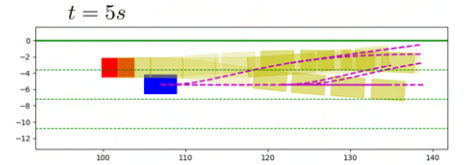


Fig. 6: Simulation of overtaking under CVaR with $\alpha = 0.1$

here. Interested readers please check out the full version at [arXiv:2109.05128](https://arxiv.org/abs/2109.05128).

Computation. In the overtaking and lane change example, $m = 3, M = 8$, i.e., every branching node has 3 children, branching happens every 8 time steps. The full scenario tree contains 3 layers, with 1, 3 and 9 branches, respectively. The ECOS solver is able to solve the branch MPC within 100 ms on an Intel Xeon CPU @ 2.4 GHz.

VI. BRANCH MPC FOR QUADRUPED MOTION PLANNING

The branch MPC controller is also tested on a quadruped platform with hardware experiments. In particular, we consider two quadruped robots, one remotely controlled (uncontrolled robot), the other under the branch MPC controller (ego robot). the operator would give waypoints for the ego robot to reach, and the ego robot would navigate itself to the waypoints while interacting with the uncontrolled robot.

Platform setup. A Unitree A1 quadruped was used as the ego robot, with locomotion performed via an inverse dynamics-based trotting controller built off the work in [34]. The trot controller tracked body-frame forward, lateral, and angular velocity commands from the branch MPC controller using the motion primitive framework in [35].

To get global location of the two quadrupeds, an OptiTrack motion capture system was used with 6 cameras and 5 markers per robot. The position and orientation data was streamed over a ROS node at 125 Hz to the branch MPC controller on the off-board computer.

Dynamic model and setup for branch MPC. The actual quadruped model is a 36 dimensional hybrid highly nonlinear model, which is not suitable for the interactive motion planning. We use a modified unicycle model on the high-level planning layer and sends command to the low-level controller which tracks the desired motion. In particular, the following unicycle model is used:

$$x = \begin{bmatrix} X \\ Y \\ \psi \end{bmatrix} \quad \dot{x} = \begin{bmatrix} v_x \cos(\psi) - v_y \sin(\psi) \\ v_x \sin(\psi) + v_y \cos(\psi) \\ r \end{bmatrix}, \quad (14)$$

where X, Y, ψ are the global coordinates and heading angle of the robot, v_x, v_y , and r are the longitudinal velocity, lateral velocity, and yaw rate, which are the inputs of the high-level branch MPC controller. Compared to the vehicle case, the quadruped robots have much smaller masses and thus can accelerate and decelerate much faster. Therefore, we directly use velocity as input and rely on the low-level controller to track the desired velocity. Lateral velocity is added as an input

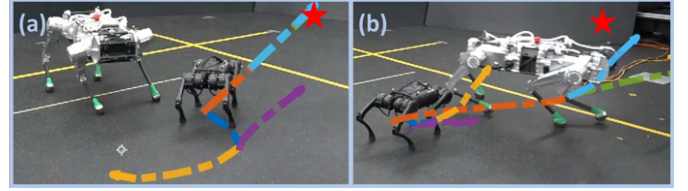


Fig. 7: Snapshots of the quadruped experiment

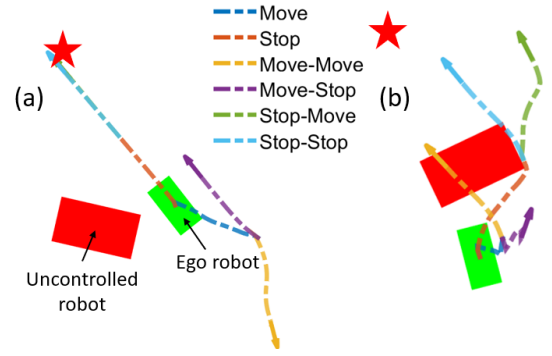


Fig. 8: Top-down view of the motion plan

since the quadruped has the ability to sidestep, yet it incurs a much larger cost as it is not the desired motion.

The uncontrolled quadruped is remotely controlled by a human operator. The branch MPC equips the uncontrolled robot with two policies: moving forward with constant speed and stopping, corresponding to the two operation modes, not yielding and yielding. The video of the experiment can be found at <https://youtu.be/W3jzoMjAZsQ>. Fig. 7 shows two snapshots of the experiment where the star shows the position of the waypoint, and the trajectory tree is plotted with one color for each branch. In both snapshots, the ego robot plans different trajectories in preparation for the uncontrolled robot to either keep moving forward, or to stop. In the top figure, the ego robot would keep moving straight to the goal should the uncontrolled robot stop, and side step to the right should the uncontrolled robot move forward. In the bottom figure, the ego robot would wait until the uncontrolled robot moves away and cross from behind should the uncontrolled robot move forward, and cross in front of the uncontrolled robot should the uncontrolled robot stop.

Fig. 8 shows the same two moments in Fig. 7 from top down. The dashed lines represent the branches corresponding to the assumed motion of the uncontrolled vehicle. For instance, in the first plot, the blue dashed line shows the branch in the trajectory tree corresponding to the uncontrolled agent moving forward, in which case the ego robot would move back to avoid collision; the orange branch shows the plan should the

uncontrolled robot stop, and the ego robot would move past the uncontrolled robot towards the waypoint. Both branches have two children branches corresponding to the subsequent motion of the uncontrolled robot.

Proximal stabilization of the SQP. The nonlinear, nonconvex MPC was solved by linearizing around the solution from the last time step and formulate a quadratic approximation of the problem. The SQP strategy may cause stability issues as the linearization point may change drastically between time steps. Instability was not observed in the highway example due to the lane boundary. In the quadruped example, we were able to maintain stability with a proximal cost, i.e., $J_{\text{prox}} = \|\mathbf{x} - \hat{\mathbf{x}}\|$, which penalizes the deviation from the linearization point.

VII. CONCLUSION

We present a branch Model Predictive Control framework for reactive motion planning for autonomous agents in the presence of uncontrolled agents. The reactive behavior of the uncontrolled agent is approximated with a finite set of policies to build a scenario tree, and a feedback policy is then solved in the form of a trajectory tree that shares the same topology as the scenario tree. Furthermore, risk measures such as the Conditional Value at Risk (CVaR) are used to tune the tradeoff between performance and robustness. The simulation and experiments results on a highway example and a quadruped example show that the branch MPC is able to capture the reactivity of the uncontrolled agent in future prediction steps and plan human-like behaviors for the autonomous agent that balances liveness and safety. Several limitations of the current algorithm are (i) the reactive model is assumed given, which is critical to the performance (ii) the timing of branching is chosen heuristically, which can be improved. We plan to improve these limitations in future works.

REFERENCES

- [1] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.
- [2] T. Kumagai and M. Akamatsu, "Prediction of human driving behavior using dynamic bayesian networks," *IEICE Transactions on Information and Systems*, vol. 89, no. 2, pp. 857–860, 2006.
- [3] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 IEEE international conference on robotics and automation (ICRA)*, 2015, pp. 454–460.
- [4] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2375–2384.
- [5] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.
- [6] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1534–1539.
- [7] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covertnet: Multimodal behavior prediction using trajectory sets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 074–14 083.
- [8] Y. Chen, U. Rosolia, C. Fan, A. D. Ames, and R. Murray, "Reactive motion planning with probabilistic safety guarantees," in *Conference on Robot Learning (CoRL)*. PMLR, 2021.
- [9] Y. Chen, N. Sohani, and H. Peng, "Modelling of uncertain reactive human driving behavior: a classification approach," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 3615–3621.
- [10] Y. Chen, S. Dathathri, T. Phan-Minh, and R. M. Murray, "Counter-example guided learning of bounds on environment behavior," in *Conference on Robot Learning (CoRL)*. PMLR, 2020, pp. 898–909.
- [11] S. Kousik, S. Vaskov, M. Johnson-Roberson, and R. Vasudevan, "Safe trajectory synthesis for autonomous driving in unforeseen environments," in *ASME 2017 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2017.
- [12] H. Kurniawati, D. Hsu, and W. S. Lee, "Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces," in *Robotics: Science and systems*, 2008.
- [13] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski, "Optimization over state feedback policies for robust control with constraints," *Automatica*, vol. 42, no. 4, pp. 523–533, 2006.
- [14] P. O. Scokaert and D. Q. Mayne, "Min-max feedback model predictive control for constrained linear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 8, pp. 1136–1142, 1998.
- [15] P. Sopasakis, D. Herceg, A. Bemporad, and P. Patrino, "Risk-averse model predictive control," *Automatica*, vol. 100, pp. 281–288, 2019.
- [16] J. P. Alsterda and J. C. Gerdes, "Contingency model predictive control for linear time-varying systems," *arXiv preprint arXiv:2102.12045*, 2021.
- [17] J. P. Alsterda, M. Brown, and J. C. Gerdes, "Contingency model predictive control for automated vehicles," in *2019 American Control Conference (ACC)*, 2019, pp. 717–722.
- [18] I. Batkovic, U. Rosolia, M. Zanon, and P. Falcone, "A robust scenario mpc approach for uncertain multi-modal obstacles," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 947–952, 2020.
- [19] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.
- [20] B. Kouvaritakis and M. Cannon, "Model predictive control," *Switzerland: Springer International Publishing*, p. 38, 2016.
- [21] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, 2021.
- [22] M. Neunert, M. Stäuble, M. Giffthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.
- [23] A. Ruszczyński and A. Shapiro, "Optimization of convex risk functions," *Mathematics of operations research*, vol. 31, no. 3, pp. 433–452, 2006.
- [24] A. Ahmadi-Javid, "Entropic value-at-risk: A new coherent risk measure," *Journal of Optimization Theory and Applications*, vol. 155, no. 3, pp. 1105–1123, 2012.
- [25] A. Dixit, M. Ahmadi, and J. W. Burdick, "Risk-sensitive motion planning using entropic value-at-risk," *arXiv preprint arXiv:2011.11211*, 2020.
- [26] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, "Coherent measures of risk," *Mathematical finance*, vol. 9, no. 3, pp. 203–228, 1999.
- [27] R. T. Rockafellar, S. P. Uryasev, and M. Zabarankin, "Deviation measures in risk analysis and optimization," *University of Florida, Department of Industrial & Systems Engineering Working Paper*, no. 2002-7.
- [28] S. Singh, Y. Chow, A. Majumdar, and M. Pavone, "A framework for time-consistent, risk-sensitive model predictive control: Theory and algorithms," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2905–2912, 2018.
- [29] H. Rahimian and S. Mehrotra, "Distributionally robust optimization: A review," *arXiv preprint arXiv:1908.05659*, 2019.
- [30] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: a control barrier function approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 127–132, 2020.
- [31] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.
- [32] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, pp. 1–36, 2020.
- [33] A. Domahidi, E. Chu, and S. Boyd, "Ecos: An sdp solver for embedded systems," in *2013 European Control Conference (ECC)*. IEEE, 2013, pp. 3071–3076.
- [34] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant quadruped locomotion over rough terrain," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 814–820.
- [35] W. Ubellacker, N. Csomay-Shanklin, T. G. Molnar, and A. D. Ames, "Verifying safe transitions between dynamic motion primitives on legged robots," *arXiv preprint arXiv:2106.10310*, 2021.