Safe Interaction via Monte Carlo Linear-Quadratic Games

Benjamin A. Christie and Dylan P. Losey

Abstract— **Safety is critical during human-robot interaction. But — because people are inherently unpredictable — it is often difficult for robots to plan safe behaviors. Instead of relying on our ability to anticipate humans, here we identify robot policies that are robust to unexpected human decisions. We achieve this by formulating human-robot interaction as a zero-sum game, where (in the worst case) the human's actions directly conflict with the robot's objective. Solving for the Nash Equilibrium of this game provides robot policies that maximize safety and performance across a wide range of human actions. Existing approaches attempt to find these optimal policies by leveraging Hamilton-Jacobi analysis (which is intractable) or linear-quadratic approximations (which are inexact). By contrast, in this work we propose a computationally efficient and theoretically justified method that converges towards the Nash Equilibrium policy. Our approach (which we call *MCLQ*) leverages linear-quadratic games to obtain an initial guess at safe robot behavior, and then iteratively refines that guess with a Monte Carlo search. Not only does MCLQ provide real-time safety adjustments, but it also enables the designer to tune how conservative the robot is — preventing the system from focusing on unrealistic human behaviors. Our simulations and user study suggest that this approach advances safety in terms of both computation time and expected performance. See videos of our experiments here: https://youtu.be/KJuHeiWVuWY.**

I. Introduction

Interacting with people is challenging because humans are inherently unpredictable. Consider Figure 1 where an autonomous drone is flying near a human worker. This robot has some high-level task it wants to complete (e.g., environment monitoring), as well as a low-level controller which dictates how the robot should accomplish this task (e.g., circling the room). If the robot knew precisely what the human was going to do, it could anticipate the human's actions and choose behaviors that maintain a safe distance between agents. But because people often take unexpected actions, real human behavior will deviate from the robot's model. As a result, the robot's original plan — which it thought was safe — may actually be unsafe. Returning to our example, a drone that turns left (because it predicts the human will stop) actually puts both agents in danger (because the human unexpectedly keeps walking forwards).

To address this problem, today's robots recognize that they are often uncertain about the human's actions. Instead of assuming the human will follow an exact model, these robots search for plans that are safe across a distribution of human behaviors. There are two general approaches here. The first is a *precise* method: the robot reasons over the distribution of possible human trajectories, and then chooses the optimal
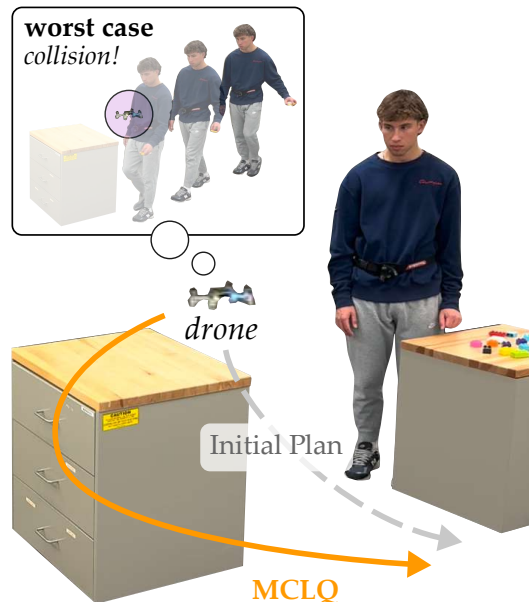
Fig. 1: Human and drone moving in a shared workspace. Under our proposed MCLQ safety filter, the drone reasons about worst case human actions within designer-specified bounds, and then selects robust behaviors. For instance, here the drone moves across the table to better prevent a collision.

control policy that maximizes safety across the distribution [1], [2]. The second approach is based on *approximations*: the robot simplifies the system dynamics and objectives, and then obtains a closed-form policy that maximizes safety under the simplified setting [3], [4]. Unfortunately, both types of approaches have significant limitations. Theoretically exact solutions are often computationally intractable; e.g., our autonomous vehicle could not apply these methods in real-time to adjust its behaviors. On the other hand, approximate solutions are imprecise, and may result in unsafe behaviors as the system becomes increasingly complex and nonlinear.

In this paper we propose a safety filter that robots can leverage to tune their real-time behavior and enhance safety around unpredictable humans. Our proposed filter achieves precise and tractable performance by combining both types of prior approaches. Specifically, our insight is that:

We can leverage approximate methods to seed safe robot behavior, and then apply a stochastic local search to guide that initial guess towards optimal solutions.

We show an example of our method in Figure 1. The robot iteratively approximates the interaction as a linear-quadratic (LQ) system, and finds an action sequence that maximizes robot performance under the worst case human response. In parallel, the robot refines this action sequence using sampling

methods that nosily converge towards the Nash Equilibrium of a zero-sum game. Robots that execute the resulting actions are robust to unpredictable humans within designer-specified bounds: even if the person suddenly changes direction, the autonomous drone has planned a safe trajectory.

Overall, we make the following contributions:

Identifying Robust Behavior. To maintain safety despite unpredictable humans, we frame interaction as a zero-sum game. Here the robot tries to minimize its cost while realizing an adversarial human might take actions to maximize that cost. The game's Nash Equilibrium defines a robot policy that is robust to unexpected human behavior; we develop a method (MCLQ) to tractably identify this robust policy.

Uniting Prior Works. Our approach to find robust robot policies blends aspects of approximate and precise methods. Specifically, we leverage LQ games to get an initial action sequence, and then refine that sequence with Metropolis-Hastings sampling. Our analysis proves that in the best case MCLQ converges to the Nash Equilibrium, and in the worst case our approach is as good as current approximations.

Adjusting Safety Margins. Our MCLQ framework enables practical advantages that go beyond existing methods. For example, designers can constrain the safety margin, i.e., the range of human actions the robot reasons over. Decreasing this safety margin causes the robot to rely more on its human predictions, preventing the robot from focusing on unrealistic human behaviors and becoming overly conservative.

Conducting Experiments. We conduct simulations and a user study to compare our approach to state-of-the-art methods. Overall, we show that MCLQ solves for safe control policies more rapidly than HJB and iLQ methods, while also achieving higher performance than existing iLQ approximations. In addition, participants reported feeling "safer" when interacting with drones that leverage MCLQ.

II. RELATED WORK

Related research has introduced a variety of control algorithms to ensure safety during human-robot interaction [2]. Within this larger field, our approach is most connected to game-theoretic controllers. These methods frame the human and robot as two agents with their own objectives, and solve for robot policies that are robust (i.e., maximize performance) while accounting for the human's decisions.

Hamilton-Jacobi Methods. Game-theoretic works treat human-robot interaction as interconnected system: both agents have a task they are trying to accomplish, and the actions of each agent affect one another. We can formalize these interconnected dynamical games with Hamilton-Jacobi equations [5]. Precisely solving the Hamilton-Jacobi equations provides the optimal, game-theoretic behavior for each agent; e.g., a policy for the autonomous car that avoids collisions with the human. Hamilton-Jacobi analysis has therefore become a gold standard for safe interaction [1], [6]–[8]. Unfortunately, most Hamilton-Jacobi equations have no analytical solution, and numerical methods are often intractable — requiring both exponential time and computational memory [4], [9]. As such, within this paper we develop a real-time approximation that converges towards the ideal solution provided by the Hamilton-Jacobi framework.

Linear-Quadratic Approximations. We are not the first to try to approximate the game-theoretic solution. For example, [10]–[12] propose neural network models, and [4], [13], [14] outline open-loop methods. One particularly promising approximation recognizes that we can analytically solve the Hamilton-Jacobi equations in a special case: if the system dynamics are linear and the costs are quadratic [5]. For these linear-quadratic (LQ) games the optimal robot policy is closely connected to linear-quadratic regulators [15]. Robots can solve LQ games quickly, providing a tractable method to obtain safe behaviors that match the gold standard of Hamilton-Jacobi analysis. Indeed — even if the system is not linear-quadratic — recent works have tried to apply sequential linear-quadratic (iLQ) approximations [4], [16]. Under these iLQ approaches the robot iteratively linearizes the dynamics and quadraticizes the cost before solving the resulting LQ game to find the human and robot control gains. On the one hand, iLQ provides an increasingly prevalent method to tractably identify safe controllers. But on the other hand, this approximation is fundamentally limited by its reliance on linear-quadratic systems — as the real interaction diverges from the simplification, iLQ falls short. Overall, this gap motivates our work towards methods that combine theoretical exactness and practical implementation.

III. PROBLEM STATEMENT

Although our approach extends to k agents, we will describe interactions between one human and one robot. The robot has a task to perform, and to complete this task the robot must reason over the human (e.g., avoid colliding with the human worker). Without loss of generality, we assume the robot has an initial control policy for the task, as well as a nominal model for predicting the human's behavior. Our approach will introduce a safety filter that modifies the robot's policy to enhance safety even when the human deviates from the nominal model. To achieve this safety, we consider the worst-case interaction (within designer specified bounds); i.e., we identify which actions the human can take that will have the worst impact on the robot's performance.

Zero-Sum Game. Let the system have state $x \in \mathcal{X}$ (e.g., the position of both agents). The robot takes action $u \in \mathcal{U}$, and the human takes action $w \in \mathcal{W}$ (e.g., the human and robot velocities). At each timestep t, the system state transitions according to the deterministic, discrete-time dynamics:

$$x^{t+1} = f(x^t, u^t, w^t) \tag{1}$$

An interaction lasts for a total of T timesteps. The system begins the interaction in state x^0 and during the interaction it follows a trajectory $\xi = \{(x, u, w)^0, \cdots, (x, u, w)^{T-1}, x^T\}$. Since the dynamics are deterministic, we can abbreviate this trajectory as a sequence of robot and human actions $\xi(x^0) = \{(u, w)^0, \cdots, (u, w)^{T-1}\} = (u^{0:T}, w^{0:T})$.

The robot's goal is to select actions that will cause the system state to update in a way that completes the robot's task. More formally, the robot has a cost function it seeks to *minimize* across the interaction:

$$J\left(x^t, u^{t:T}, w^{t:T}\right) = \sum_{\tau=t}^{T} j(x^\tau, u^\tau, w^\tau) + D(x^T) \quad (2)$$
$$\text{s.t. } x^{\tau+1} = f\left(x^\tau, u^\tau, w^\tau\right)$$

Here J is the cumulative cost, $j(x, u, w)$ is the cost at a single timestep, and $D(x^T)$ is the bequest state cost. We emphasize that the robot's cost in Equation (2) depends on the human's actions w. For example, the autonomous drone will incur a significant penalty if the human moves into that agent. To optimize cost, the robot has an initial policy $\hat{\pi}_{\mathcal{R}}$ that determines how it will complete the task. The robot may also have some guess for how the human agent will behave: $\hat{\pi}_{\mathcal{H}}$. Both policies are mappings from states x to actions:

$$\hat{\pi}_{\mathcal{R}} : \mathcal{X} \mapsto \mathcal{U} \quad (3)$$
$$\hat{\pi}_{\mathcal{H}} : \mathcal{X} \mapsto \mathcal{W} \quad (4)$$

In practice, the real human will inevitably deviate from the robot's model. A worst-case human selects actions that *maximize* the cost in Equation (2). Formally, in this worst-case the human and robot are participating in a two-player *zero-sum* game [5]: the robot is trying to minimize its cost, whereas the antagonistic human is attempting to maximize that same cost, directly opposing the robot's objective.

Nash Equilibria. Similar to prior works [3], [17], [18], we have formulated human-robot interaction as a zero-sum game. The advantage of this formulation is — if we can find robot behavior that is safe in the worst case — then we can ensure safety across the board, regardless of what the human does. Let $\pi_{\mathcal{R}}(x)$ be the robot's filtered policy, and let $\pi_H(x)$ be the human's true policy. In a zero-sum game, the optimal policies for both agents form a Nash Equilibrium [19], where neither agent can unilaterally deviate from their policy without negatively impacting performance:

$$\pi_{\mathcal{R}} = \underset{u^{0:T} \in \mathcal{U}^T}{\arg\min} \; \underset{w^{0:T} \in \mathcal{W}^T}{\max} J(x^t, u^{0:T}, w^{0:T}) \quad (5)$$

$$\pi_{\mathcal{H}} = \underset{w^{0:T} \in \mathcal{W}^T}{\arg\max} \; \underset{u^{0:T} \in \mathcal{U}^T}{\min} J(x^t, u^{0:T}, w^{0:T}) \quad (6)$$

In a zero-sum game, the Nash Equilibrium is unique [19]: there is a single set of (mixed) policies that satisfy Equations (5)–(6). By definition, the robot policy of Equation (5) is maximally robust to changes in the human policy. This policy enables us to maintain safe interactions — even when we do not know what actions the real human will take.

Solving for Nash Equilibria. Accordingly, it is desirable to shift the robot's initial policy $\hat{\pi}_{\mathcal{R}}$ towards the Nash Equilibrium policy. But how do we solve for this equilibrium? The standard approach in finite horizon, discrete time settings is

to apply the Bellman Equation:

$$V(x, t) = \min_u \max_w \left[j(x, u, w) + V^{t+1} \right]$$
$$V^{t+1} = V(x^{t+1}, t+1) \quad (7)$$
$$\text{s.t.} \quad x^{t+1} = f(x, u, w)$$

Applying Equation (7) in continuous time settings results in the Hamilton-Jacobi-Isaacs (HJI) partial differential equation. However, this formula can only be applied tractably in special cases such as linear-quadratic systems [5]; in general, the Nash Equilibrium policy must be approximated.

Objective. Given that the robot has some initial policy $\hat{\pi}_{\mathcal{R}}$, our goal is to develop a real-time safety filter that adjusts the robot's policy so that it converges towards the Nash Equilibrium solution. Because we cannot tractably identify this ideal policy, we must develop an approximation that the robot can leverage efficiently in real-time.

IV. Monte Carlo Linear-Quadratic Games

In this section we propose our real-time safety filter based on zero-sum games. We recognize that there are special cases where we can find analytical solutions to Equation (7): when the dynamics are linear and the cost is quadratic. Accordingly, we first leverage this linear-quadratic case to obtain an initial guess of the Nash Equilibrium policies (Section IV-A). We next refine that guess by stochastically solving the true optimization problem (Section IV-B). Our overall approach — Monte Carlo Linear-Quadratic Games (**MCLQ**) — combines rapid LQ solutions with a parallel local search. We theoretically demonstrate that MCLQ improves upon existing LQ baselines, and converges towards the ideal Nash Equilibrium policy (Section IV-C). We conclude with the practical advantages of our MCLQ framework, including a designer-specified safety margin that prevents the robot from becoming overly conservative (Section IV-D).

A. Obtaining an Initial Action Trajectory

Our method attempts to find the sequence of actions $u^{0:T}$ that optimize Equation (5). To get an initial estimate of these actions, we iteratively simplify the actual system into an LQ approximation. Starting with f, we linearize the dynamics:

$$f(x^t, u^t, w^t) \approx Ax^t + Bu^t + Dw^t \quad (8)$$

and quadraticize the state-action cost function j:

$$j(x^t, u^t, w^t) \approx x^{t\prime} Q x^t + u^{t\prime} R_u u^t + w^{t\prime} R_w w^t \quad (9)$$

where $\circ'$ represents transposition. Under this LQ approximation the Nash Equilibrium policies exhibit linear state-feedback behavior [5]. These optimal feedback policies are captured by the gain matrices:

$$u^t = -K^t x^t$$
$$w^t = -L^t x^t \quad (10)$$

To find the gain matrices, we employ the discrete-time algebraic Riccati Equation (DARE, a version of the Bellman Equation) as detailed in [15]. For a particular robot gain

matrix K, the optimal human gain matrix L can be found in closed form by solving the DARE:

$$\begin{aligned}
\boldsymbol{P_{K,L(K)}} &= \boldsymbol{Q} + \boldsymbol{K'R_uK} + \\
&\quad (\boldsymbol{A} - \boldsymbol{BK})'\widetilde{\boldsymbol{P}}_{\boldsymbol{K,L(K)}}(\boldsymbol{A} - \boldsymbol{BK}) \\
\text{s.t.} \quad &\Re\left(\|\boldsymbol{P_{K,L(K)}}\|\right) \geq 0 \\
&\Re\left(\|\boldsymbol{R_w} - \boldsymbol{D'P_{K,L(K)}D}\|\right) > 0
\end{aligned} \quad (11)$$

where $\widetilde{\boldsymbol{P}}_{\boldsymbol{K,L(K)}}$ is condensed for brevity:

$$\begin{aligned}
\widetilde{\boldsymbol{P}}_{\boldsymbol{K,L(K)}} &\equiv \boldsymbol{P_{K,L(K)}} + \boldsymbol{P_{K,L(K)}D} \\
&\quad \cdot (\boldsymbol{R_w} - \boldsymbol{D'P_{K,L(K)}D})^{-1}\boldsymbol{D'P_{K,L(K)}}
\end{aligned} \quad (12)$$

and the terminal condition is $\boldsymbol{P_{K,L(K)}} = \boldsymbol{Q}$. The bolded matrices shown in Equation (11) are augmented forms of the LQ matrices from Equations (8) and (9):

$$\begin{aligned}
\boldsymbol{A} &\equiv \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \mathrm{diag}(A^{t:(T-1)}) & \boldsymbol{0} \end{bmatrix} \\
\boldsymbol{B} &\equiv \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \mathrm{diag}(B^{t:(T-1)}) & \boldsymbol{0} \end{bmatrix} \\
\boldsymbol{D} &\equiv \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \mathrm{diag}(D^{t:(T-1)}) & \boldsymbol{0} \end{bmatrix} \\
\boldsymbol{Q} &\equiv \mathrm{diag}(Q^{t:T}) \\
\boldsymbol{R_u} &\equiv \mathrm{diag}(R_u^{t:T}), \boldsymbol{R_w} \equiv \mathrm{diag}(R_w^{t:T})
\end{aligned} \quad (13)$$

Once the DARE has converged, the unique human gain matrix $\boldsymbol{L(K)}$ that *maximizes* the cumulative cost is:

$$\begin{aligned}
\boldsymbol{L(K)} &= (-\boldsymbol{R_w} + \boldsymbol{D'P_{K,L(K)}D})^{-1}\boldsymbol{D'} \\
&\quad \cdot \boldsymbol{P_{K,L(K)}}(\boldsymbol{A} - \boldsymbol{BK})
\end{aligned} \quad (14)$$

Note that $\boldsymbol{L(K)}$ is the *worst-case* gain matrix given the robot's policy and objective; we have found the human actions that maximize the robot's cost. To develop the correct response to this worst-case, we use another DARE to compute the robust robot gain matrix $\boldsymbol{K(L)}$:

$$\begin{aligned}
\boldsymbol{K(L)} &= (\boldsymbol{R_u} + \boldsymbol{B'P_{K(L),L}B})^{-1}\boldsymbol{B'} \\
&\quad \cdot \boldsymbol{P_{K(L),L}}(\boldsymbol{A} - \boldsymbol{DL})
\end{aligned} \quad (15)$$

By iteratively recomputing the DARE for both Equation (14) and Equation (15), we converge to feedback policies that form the Nash Equilibrium for zero-sum LQ games. Expanding these matrices provides our initial guess of the optimal action trajectory $\xi(x^0) = (u^{0:T}, w^{0:T})$.

B. Refining the Action Trajectory

The action trajectory $\xi(x^0) = (u^{0:T}, w^{0:T})$ we have obtained is optimal if the system is linear-quadratic. But often this is not the case; solutions that assume linear dynamics and quadratic costs may fall apart when faced with nonlinear, realistic interactions, or when the necessary Eigenvalue conditions are not met [15]. We therefore propose to use the solution from Section IV-A as a *first pass approximation* that we will refine with stochastic gradient descent.

As a reminder, our ultimate goal is to reach the robot's Nash Equilibrium policy. This policy is defined by Equation (5), which can be treated as a nested optimization problem. In the outer loop the robot proposes a sequence of actions $u^{0:T}$ to minimize its cost, and in the inner loop the antagonistic human responds with actions $w^{0:T}$ that maximize cost. We therefore apply a nested gradient descent algorithm to iteratively improve the initial action trajectory $\xi(x^0) = (u^{0:T}, w^{0:T})$ and optimize Equation (5). More specifically, we perform nested Metropolis-Hastings sampling (i.e., stochastic gradient descent). Other Monte Carlo methods can be leveraged within our general framework; we here selected the Metropolis-Hastings sampler due to its ergodic properties and avoidance of local optima [20].

Given the seed $\xi(x^0) = (u^{0:T}, w^{0:T})$ provided by Section IV-A, this sampler has two parts [21]: an *inner loop* that updates the antagonistic human actions $\xi_w = w^{0:T}$, and an *outer loop* that improves the robot's actions $\xi_u = u^{0:T}$. An outline of our resulting Monte Carlo Linear-Quadratic Games approach is included in Algorithm 1.

Inner Loop. The inner loop seeks to identify worst-case human actions that increase the robot's cost. Given the robot's current action trajectory ξ_u, this inner loop modifies ξ_w by proposing random perturbations $\Delta\xi_w$. These perturbations are accepted if they increase the total cost, or if they satisfy the acceptance rate:

$$\exp \beta \left(J\left(x^0, \xi_u, \xi_w + \Delta\xi_w\right) - J\left(x^0, \xi_u, \xi_w\right) \right) \quad (16)$$

where β is a design parameter that regulates the level of noise in the stochastic gradient ascent (as $\beta \to 0$, the sampler explores Ξ more and is less likely to remain in local minima). The inner loop repeats N times before the terminating with a new human action sequence ξ_w'.

Outer Loop. The outer loop takes the updated choice of human actions ξ_w', and seeks to find robot actions that are robust to the human. Similar to the inner loop, the outer loop modifies ξ_u by proposing random perturbations $\Delta\xi_u$. These perturbations are accepted if they decrease total cost, or if they satisfy the acceptance rate:

$$\exp \beta \left(J\left(x^0, \xi_u, \xi_w'\right) - J\left(x^0, \xi_u + \Delta\xi_u, \xi_w'\right) \right) \quad (17)$$

Importantly, the inner loop (i.e., the human) is *maximizing* the cost, while here the outer loop (i.e., the robot) is *minimizing* that same cost. This outer loop terminates after M iterations, and outputs a final robot action trajectory ξ_u'.

Summary. Taken together, the LQ game in Section IV-A provides a first guess at the robot's action trajectory ξ. This action trajectory is then refined by nested Metropolis-Hastings samplers that update ξ to solve for the true Nash Equilibrium policy in Equation (5). The resulting trajectory ξ' contains the sequence of actions a robust robot should execute over the next T timesteps; this robust action sequence can be recomputed online (i.e., at each timestep t).

C. Theoretical Analysis

Here we theoretically compare MCLQ to both the ideal solution (i.e., the Nash Equilibrium) and relevant approximations (i.e., LQ games). We consider two cases. First, when

Algorithm 1 Monte Carlo Linear-Quadratic Games (MCLQ)

Require: f, J, β, M, N ▷ Dynamics and Cost Functions
1: **procedure** LQ APPROXIMATION(x, f, J)
2:     $A, B, D \leftarrow$ Linearize(f)
3:     $Q, R_u, R_w \leftarrow$ Quadraticize(J)
4:     $K, L \leftarrow$ DARE Solution
5:     $\xi_u \leftarrow -Kx$
6:     $\xi_w \leftarrow -Lx$
7:     **return** ξ_u, ξ_w
8: **end procedure**   ▷ Provides initial guess of ξ_u, ξ_w
9: **procedure** MONTE CARLO SEARCH(x, ξ_u, ξ_w, f, J)
10:     $\Delta_u \leftarrow 0, \Delta_w \leftarrow 0$
11:     $J_0 \leftarrow J(\xi_u, \xi_w)$
12:     **for** $m = 1 \ldots M$ **do**
13:       **for** $n = 1 \ldots N$ **do**
14:         $\Delta_n \leftarrow$ Perturb(ξ_w, Δ_w)
15:         $J_w \leftarrow J(\xi_u + \Delta_u, \xi_w + \Delta_n)$
16:         **if** $\exp(\beta(J_w - J_0)) > \eta, \ \eta \sim U[0, 1]$ **then**
17:           $J_0 \leftarrow J_w$
18:           $\Delta_w \leftarrow \Delta_n$
19:         **end if**
20:       **end for**   ▷ Updates worst-case human actions
21:       $\Delta_m \leftarrow$ Perturb(ξ_u, Δ_u)
22:       $J_u \leftarrow J(\xi_u + \Delta_m, \xi_w + \Delta_w)$
23:       **if** $\exp(\beta(J_0 - J_u)) > \eta, \ \eta \sim U[0, 1]$ **then**
24:         $J_0 \leftarrow J_u$
25:         $\Delta_u \leftarrow \Delta_m$
26:       **end if**   ▷ Updates robot response to human
27:     **end for**
28:     **return** $\xi'_u = \xi_u + \Delta_u, \ \xi'_w = \xi_w + \Delta_w$
29: **end procedure**

---

the actual system has linear dynamics f and a quadratic cost J, and second, when the system is *not* linear-quadratic.

LQ Systems. For a linear-quadratic system the Bellman Equation is equivalent to the DARE presented in Equation (11) [22]. Hence, the trajectory produced by the DARE is the Nash Equilibrium of the zero-sum game. This means that — for our MCLQ method — the initial trajectory is the global optimum of Equation (5), and Monte Carlo sampling is unable to improve upon this initial guess. Each method therefore converges to the same sequence of robot actions.

Non-LQ Systems. If the system is not linear-quadratic, then MCLQ improves upon the LQ approximation. Consider the case where $\beta \to \infty$; here our sampler only accepts perturbations Δ_u and Δ_w if they move towards the Nash Equilibrium in Equation (5). In the worst case, the initial trajectory provided by the DARE is a local optima, and MCLQ performs on par with LQ games. In any other case, the robot is not seeded in a local optima, and nested gradient descent moves towards the Nash Equilibrium. Hence, MCLQ matches or outperforms standard LQ approximations.

D. Balancing Performance and Safety

One key advantage of our approach is that — as we will show — it can be implemented in real-time. But another core aspect is that the designer can tune the *safety margin*, i.e., how conservative the robot's behavior is. Recall that $\hat{\pi}_{\mathcal{H}}$ is a predictive model of the human's actions. We recognize that the real human will inevitably deviate from $\hat{\pi}_{\mathcal{H}}$; but how different should we expect the human's actions to be? Within a zero-sum game formulation, if the human can take any action at any time, then the robot is forced to overly conservative behaviors (e.g., the autonomous drone always moving away from the human). Unlike LQ approximations, our approach is structured to resolve this conflict between performance and safety. Specifically, during Monte Carlo sampling we can impose hard constraints on the perturbations Δ_w so that they are close to the predictions of our human model. Let $\hat{\xi}_w \sim \hat{\pi}_{\mathcal{H}}$ be a predicted human trajectory; we can limit the range of considered human actions such that:

$$\|\hat{\xi}_w - (\xi_w + \Delta_w)\|^2 \leq \lambda \tag{18}$$

where λ is a design parameter that determines conservatism. As $\lambda \to \infty$, the search space approaches the original trajectory space $\Xi_{\mathcal{H}}$, and as $\lambda \to 0$, the robot is increasingly confident in its human model. Importantly, Equation (18) does not determine *if* the human's behavior aligns with our nominal model: it simply restricts the search space to trajectories that are *similar* to our nominal model.

V. SIMULATIONS

In theory, our proposed MCLQ method converges towards robust, game-theoretic behaviors while offering real-time performance. Here we test both of these theoretical claims by comparing our approach to exact Hamilton-Jacobi methods and tractable LQ approximations. More specifically, we perform controlled experiments where simulated agents interact in three environments: point-mass, driving, and manipulator (see Figure 2). Each environment contains two agents. The simulated robot is attempting to complete its task (e.g., reaching a goal) while simulated humans move in its proximity. The robot must avoid collisions with these simulated humans — even when the humans take unexpected or noisy actions. A repository of our code is available here.

Independent Variables. We vary the robot's controller across four levels. To test LQ approximations, we include **NPG** [15] and **ILQ** [4] approaches. These methods iteratively approximate the dynamics and cost as a LQ system and solve for the resulting Nash Equilibrium. To test exact solutions, we next solve for the true Nash Equilibrium (**NE**) using the Bellman Equation in Equation (7). Finally, we evaluate our **MCLQ** approach from Algorithm 1.

Environments. Below we describe the three simulated environments. All simulations were performed on an AMD Ryzen 7000 Series 5 CPU.

Point-Mass. Here the simulated human and robot move in a 2D plane. The initial positions and velocities of both agents
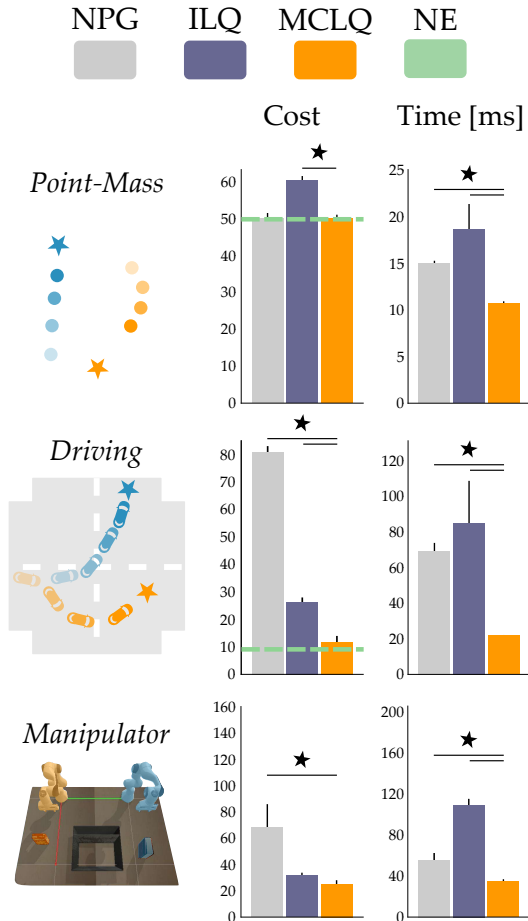
Fig. 2: Simulation results across *point-mass*, *driving*, and *manipulator* environments. (Left) We plot the cost and (Right) computation time averaged over 100 simulations. Computation time is the number of milliseconds per robot action (normalized by the number of timesteps per trajectory). In non-LQ settings the computation time for NE is prohibitively high; e.g., in *driving* the **NE** computation time exceeded one hour. We could not calculate **NE** in the 26-dimensional *manipulator* environment. Error bars show standard deviation and an ∗ denotes statistical significance.

are randomized; across an interaction of $T = 30$ timesteps, the robot attempts to reach a fixed goal location. Both agents have linear dynamics such that $x^{t+1} = x^t + u^t + w^t$, and the robot's cost is quadratic (considering both distance to goal and distance to human). Because this environment is a linear-quadratic game, we anticipate that LQ approximations should find the Nash Equilibrium.

Driving. Our driving environment is taken from the iLQ baseline [4]. As before, one simulated human and robot move in a 2D plane with randomized initial configurations. The dynamics of each agent follow the nonlinear bicycle model from [23]. Here the robot's cost function is not quadratic:

$$j(x, u, w) = \|x_{\mathcal{R}}^t - g\|^2 + \eta \exp\left(\frac{\|x_{\mathcal{R}}^t - x_{\mathcal{H}}^t\|^2}{-\eta}\right) + u'R_u u$$

where g is the goal position the robot is trying to reach, $x_{\mathcal{R}}$ is the robot's position, $x_{\mathcal{H}}$ is the human's location, and η
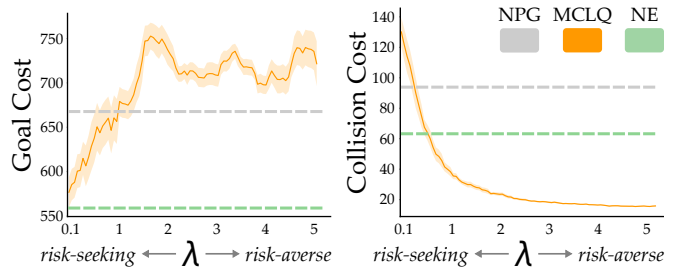

Fig. 3: Simulation results for a modified point-mass environment where we adjust the safety margin λ in MCLQ. Increasing λ causes the MCLQ robot to consider a wider range of worst case human actions, resulting in more conservative behavior. Conversely, decreasing λ causes the MCLQ robot to increasingly rely on its nominal human model. Unlike LQ approximations, our proposed method gives designers the flexibility to tune λ and adjust the safety margin.

scales the cost of approaching the human. Each interaction lasts a total of $T = 30$ timesteps.

Manipulator. Our final environment contains two 7-DoF robot arms in PyBullet. We treat one of these arms as the simulated human, and the other as the robot. Both agents are trying to pick up and place objects within a shared workspace. The dynamics of the arms and the task are nonlinear, and the system state $x \in \mathcal{X}$ is 26-dimensional (including manipulators and objects). We leverage a cost function similar to *driving*. The robot moves to reach, grasp, and place items while avoiding collisions with the other arm.

Simulated Human. We simulate humans as bounded-rational agents that noisily optimize their cost function J_H:

$$\pi_{\mathcal{R}}\left(w^t \mid x^t\right) \propto \exp\left(-\alpha \cdot J_H(x^t, w_i^t)\right)$$
$$\text{s.t. } x^{t+1} = f(x^t, u^{t_0}, w^t)$$

Increasing $\alpha \to \infty$ causes the human to always take the optimal action, while decreasing $\alpha \to 0$ causes the human to act randomly [7]. In our simulations we set $\alpha = 7.5$. When computing the cost for future timesteps the simulated human assumes the robot will repeat its most recent action u^{t_0} (e.g., the robot will keep moving with the same velocity). In each environment the human had a task that was independent of the robot's objective — for instance, in *driving* the human tried to reach their own goal position.

Results. The results from our first simulation are summarized in Figure 2. Across all three environments, **MCLQ** achieved costs that were closest to the Nash Equilibrium (**NE**) while also requiring the least amount of computation time. In *point-mass* we highlight that all methods reached similar cost; this matches our expectations, because *point-mass* was an LQ system. By contrast, in non-LQ systems (*driving* and *manipulator*) the LQ approximations made by **NPG** and **ILQ** fell short, leading to suboptimal performance. Computing the exact solution with **NE** was time-consuming and not always feasible: indeed, in the 26-dimension *manipulator* task, we were unable to compute the true **NE** because of the high-dimensional and continuous state-action space.

Adjusting Safety Margins. As discussed in Section IV-D, one feature of our approach is the designer-selected safety margin. MCLQ robots reason over the worst-case human action within bounds λ. By increasing λ in Equation (18) the designer makes the robot more risk-averse (i.e., the robot considers larger deviations from the nominal human model). Conversely, decreasing λ makes the robot more risk-seeking (i.e., the robot increasingly relies on its human model). Within our simulations from Figure 2 we left this value of λ fixed at risk-neutral behavior. Now we explore how increasing and decreasing λ changes the robot's performance in a modified *point-mass* environment.

Our extended *point-mass* environment includes one to ten humans that are navigating to goal positions. The robot seeks to reach to its own goal while avoiding these randomly generated agents. To find an initial plan, the robot is equipped with a nominal human model — it assumes each human will move in a straight line towards their goal. In practice, however, our simulated humans deviate from this model when nosily optimizing their cost function. Figure 3 plots the robot's performance as a function of λ. When λ increases the robot becomes more risk-averse: the MH sampler accounts for a wider range of adversarial human actions. This causes the robot agent to stay farther from humans (reducing collisions), but also leads to longer paths (increasing distance). Conversely, lower values of λ cause to risk-seeking behavior where the robot relies on its human model. If humans stick to this model, the robot avoids collisions while minimizing distance. But when humans deviate, the number of collisions increase. Overall, this simulation supports our theoretical description of λ and demonstrates how designers can leverage MCLQ to tune the safety margin.

VI. USER STUDY

Our simulations from Section V support our theoretical analysis, and suggest that MCLQ converges towards maximally-robust behaviors while minimizing computation time. We next evaluate our method in a real-world setting with $N = 24$ in-person human participants. Here users walk around a room to complete an assembly task while an autonomous drone circles that room to inspect the parts (see Figure 1). The robot modifies its high-level trajectory to avoid getting to close to the human workers. We compare two *real-time* safety filters for adjusting the drone's behavior: **ILQ** [4] and **MCLQ**. We selected **ILQ** here because it was the best performing real-time baseline from our simulations. Videos of our user study are available here.

Experimental Setup. Participants interacted with a Crazyflie 2.1+ (Bitcraze) during the assembly task. The participant's objective was to construct a Lego tower at the central station. The blocks needed for building that tower were scattered in three other stations located along the perimeter of the workspace. Accordingly, users needed to move back and forth through the workspace to acquire blocks and build their tower. The drone's objective was to monitor the workspace during the task by completing as many revolutions around

the central station as possible. As the drone moved around the central station it repeatedly intersected with the human worker: at these intersections the drone should to take actions to help avoid the moving participant.

Participants and Procedure. We recruited 24 participants (6 female, average age 24.5 ± 4.5) for this user study. Of the 24 participants, 8 had not previously used drones and 5 did not have prior experience with robotics. Participants received monetary compensation for their time and provided written consent according to university guidelines (IRB #23-1237).

We leveraged a between-subjects design where every participant interacted with **ILQ** for five minutes and **MCLQ** for five minutes. To prevent the participants from always going back and forth between the same stations, we instructed users to move according to three movement patterns: clockwise, counter-clockwise, and random. Both the order of the methods and movement patterns were counterbalanced (i.e., half of the participants started with **MCLQ**). Participants were never told which algorithm the drone was using.

Dependent Measures — Objective. We using tracking devices (VIVE) to measure the states and actions of the drone and human at each timestep of the interaction. To assess objective performance, we considered two metrics. For our first metric (*collisions*) we counted the number of times the drone was within a radius of 0.5m from the human. Our second metric (*revolutions*) is the number of revolutions the drone completed within the five minute trial. Lower values for *collisions* indicate that the drone is maintaining safety, while higher values for *revolutions* show that the drone is performing the task more efficiently.

Dependent Measures — Subjective. After interacting with each control algorithm, participants completed a 7-point Likert scale survey. This survey assessed the user's subjective preferences along three multi-item scales. We asked users:

1) If they felt *safe* during the interaction,
2) If they thought the drone was *attentive* to their position,
3) If they thought the drone's movements were *predictable*.

Hypotheses. We had two hypotheses for this user study:

> **H1.** MCLQ *will modify the drone's actions to reduce the number of collisions and improve subjective feelings of safety.*
>
> **H2.** MCLQ *will complete a similar number of revolutions as the iLQ baseline.*

Results. The results from our in-person user study are summarized in Figure 4. To assess **H2**, we measured the number of revolutions that each method completed per interaction: a higher number indicates more efficient performance. One-way ANOVA tests show that the performance difference between the methods is trending towards significance ($F(2, 86) = 1.50$, $p = 0.14$), where **MCLQ** completes more revolutions than **ILQ**. This suggests that the safety improvements made by **MCLQ** are not coming at the expense of overly conservative behavior — the **MCLQ** drone is still performing its high-level task of monitoring the workspace.
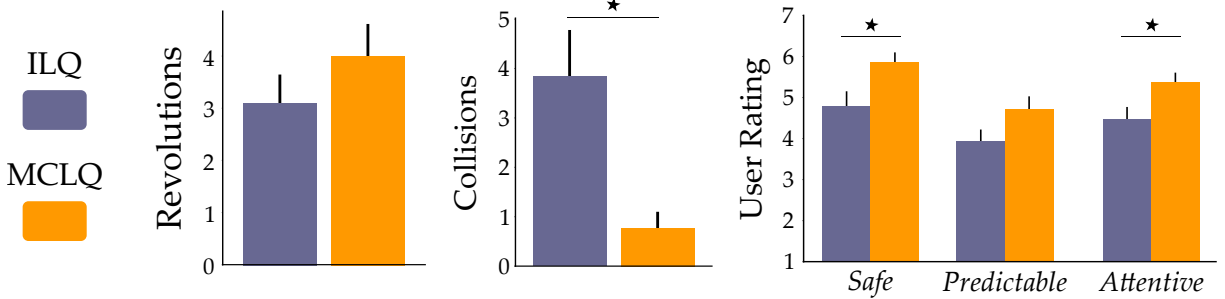
Fig. 4: Results from our user study in Section VI. Participants walked around a room to assemble a tower; a drone completed revolutions around the same workspace to monitor the human's progress (also see Figure 1). (Left) The average number of revolutions the drone completed and the average number of collisions. Here "collisions" occurred when the drone was within 0.5 meters of the human. The proposed **MCLQ** algorithm adjusts the robot's behavior to increase safety (fewer collisions) while also enhancing performance (more revolutions). (Right) After interacting with each algorithm participants answered survey questions about how safe, predictable, and attentive the robot was. Ratings suggest that participants perceived **MCLQ** to be a safer system. Error bars show standard deviation and an $*$ denotes statistical significance ($p < .05$).

Given that the drone is completing a similar number of revolutions with each method, the key question becomes the *safety* of the robot's behavior. We analyzed **H1** along two levels: objective safety (maintaining a minimum distance between agents) and subjective safety (the user's perception of the system). For objective safety, ANOVA tests revealed that **MCLQ** leads to significantly fewer collisions than **ILQ** ($F(2, 86) = 4.16$, $p < 0.001$). The participants' responses to our Likert scale survey align with these objective results: users perceived the **MCLQ** robot to be safer ($F(2, 24) = 2.33$, $p < 0.05$) and more attentive ($F(2, 24) = 2.37$, $p < 0.05$). In addition, users thought that drones following the **MCLQ** algorithm had more predictable actions, with differences trending towards significance ($F(2, 24) = 1.84$, $p = 0.073$). In their free response comments, participants stated that with the **MCLQ** drone they "*felt safer*" and that the drone was "*more reactive and predictable*."

VII. CONCLUSION

In this paper we presented a real-time safety filter for human-robot interaction. Our approach ensures that the robot is robust to noisy and unexpected human behaviors within designer-specified bounds. To achieve this game-theoretic safety with tractable computation, we combined linear-quadratic approximations with stochastic local searches. Our theoretical analysis showed the resulting MCLQ algorithm converges towards the ideal Nash Equilibrium policy while providing flexible and efficient implementation. Across multiple simulations and a user study, we observed that MCLQ advances both objective and subjective safety measures when compared to state-of-the-art control alternatives.

REFERENCES

[1] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *IEEE Annual Conference on Decision and Control*, 2017, pp. 2242–2253.

[2] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, 2023.

[3] R. Tian, L. Sun, A. Bajcsy, M. Tomizuka, and A. D. Dragan, "Safety assurances for human-robot interaction via confidence-aware game-theoretic human models," in *IEEE International Conference on Robotics and Automation*, 2022, pp. 11 229–11 235.

[4] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 1475–1481.

[5] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. SIAM, 1998.

[6] S. Bansal, A. Bajcsy, E. Ratner, A. D. Dragan, and C. J. Tomlin, "A Hamilton-Jacobi reachability-based framework for predicting and analyzing human motion for safe planning," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 7149–7155.

[7] D. Fridovich-Keil, A. Bajcsy, J. F. Fisac, S. L. Herbert, S. Wang, A. D. Dragan, and C. J. Tomlin, "Confidence-aware motion prediction for real-time collision avoidance," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 250–265, 2020.

[8] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin, "An efficient reachability-based framework for provably safe autonomous navigation in unknown environments," in *IEEE Conference on Decision and Control*, 2019, pp. 1758–1765.

[9] A. W. Starr and Y.-C. Ho, "Nonzero-sum differential games," *Journal of Optimization Theory and Applications*, vol. 3, pp. 184–206, 1969.

[10] F. Jiang, G. Chou, M. Chen, and C. J. Tomlin, "Using neural networks to compute approximate and guaranteed feasible Hamilton-Jacobi-Bellman PDE solutions," *arXiv preprint arXiv:1611.03158*, 2016.

[11] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, "A causality-free neural network method for high-dimensional Hamilton-Jacobi-Bellman equations," in *American Control Conference*, 2020, pp. 787–793.

[12] ——, "Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman equations," *SIAM Journal on Scientific Computing*, vol. 43, no. 2, pp. A1221–A1247, 2021.

[13] M. Jones and M. M. Peet, "Polynomial approximation of value functions and nonlinear controller design with performance bounds," *arXiv preprint arXiv:2010.06828*, 2020.

[14] C. M. Chilan and B. A. Conway, "Optimal nonlinear control using Hamilton–Jacobi–Bellman viscosity solutions on unstructured grids," *Journal of Guidance, Control, and Dynamics*, 2020.

[15] J. Wu, A. Barakat, I. Fatkhullin, and N. He, "Learning zero-sum linear quadratic games with improved sample complexity," in *IEEE Conference on Decision and Control*, 2023, pp. 2602–2609.

[16] F. Laine, D. Fridovich-Keil, C.-Y. Chiu, and C. Tomlin, "The computation of approximate generalized feedback Nash equilibria," *SIAM Journal on Optimization*, vol. 33, no. 1, pp. 294–318, 2023.

[17] O. Slumbers, D. H. Mguni, S. B. Blumberg, S. M. Mcaleer, Y. Yang, and J. Wang, "A game-theoretic framework for managing risk in multi-agent systems," in *International Conference on Machine Learning*, 2023, pp. 32 059–32 087.

[18] M. Wang, N. Mehr, A. Gaidon, and M. Schwager, "Game-theoretic planning for risk-aware interactive agents," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[19] J. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, no. 2, 1951.

[20] C. Andrieu and É. Moulines, "On the ergodicity properties of some adaptive MCMC algorithms," *The Annals of Applied Probability*, vol. 16, no. 1, pp. 1462–1505, 2006.

[21] J. Hoegerman and D. Losey, "Reward learning with intractable normalizing functions," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7511–7518, 2023.

[22] K. Zhang, X. Zhang, B. Hu, and T. Basar, "Derivative-free policy optimization for linear risk-sensitive and robust control design: Implicit regularization and sample complexity," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2949–2964, 2021.

[23] R. Rajamani, *Vehicle Dynamics and Control*. Springer Science & Business Media, 2011.