Multi-Agent Safe Planning with Gaussian Processes

Zheqing Zhu[1], Erdem Bıyık[2] and Dorsa Sadigh[2,3]

***Abstract*— Multi-agent safe systems have become an increasingly important area of study as we can now easily have multiple AI-powered systems operating together. In such settings, we need to ensure the safety of not only each individual agent, but also the overall system. In this paper, we introduce a novel multi-agent safe learning algorithm that enables decentralized safe navigation when there are multiple different agents in the environment. This algorithm makes mild assumptions about other agents and is trained in a decentralized fashion, i.e. with very little prior knowledge about other agents' policies. Experiments show our algorithm performs well with the robots running other algorithms when optimizing various objectives.**

I. Introduction

Safety in multi-agent systems is vital in collaborative tasks. Even when the agents can observe each other and know the dynamics of their environment, operating in a decentralized manner without knowing each other's policy makes it very challenging to guarantee safety. While it is a difficult problem, such safety-critical systems with multiple agents are commonly observed in autonomous driving [1], [2], collaborative quadrotor control [3]–[5], multi-robot coordination [6]–[8]. Ensuring the safety in these environments is extremely critical as the failures can cause damage not only to the agents themselves, but also to the environment.

We specifically study tasks where state-action pairs of the agents, and dynamics of the system are known, but agents are decentralized, i.e. they do not know other agents' policies. We decompose safety into two parts as *individual* and *joint* safety. There are many real-world use cases that fit into this framework. Consider a Mars exploration task where multiple rovers explore a region together, as visualized in Fig. 1, without explicit communication for increasing energy efficiency and avoiding delays due to communication. Individual safety could be each rover avoiding environment obstacles or steep terrain, and joint safety might refer to keeping the distance between rovers in a specific range to avoid collisions. The setting also works for autonomous cars that are trying to avoid collisions while optimizing each car's speed, and competitive robot teams where two teams are competing in a game setting, e.g. soccer game, but do not want to collide with one another to cause fatal damage.

While assuming a centralized controller enables us to formulate the problem as a single-agent problem, it is not realistic in practice, because such systems do not scale well with the number of agents in the environment: both state and action spaces grow exponentially with the number of agents.

Emails: {zheqzhu, ebiyik, dorsa}@stanford.edu
[1]Management Science & Engineering, Stanford University, CA, USA
[2]Electrical Engineering, Stanford University, CA, USA
[3]Computer Science, Stanford University, CA, USA

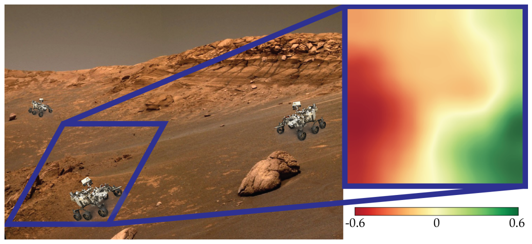

Fig. 1: Representative figure for three Mars rovers exploring a region on the surface of Mars. Colors represent the normalized altitudes of the terrain such that low altitude regions shown as red are unsafe for the rovers. It is also unsafe if two rovers operate in the same region due to the risk of collisions.

On the other hand, as opposed to many prior works, most real-world tasks require continuous state spaces. Therefore, we would like to *enable multiple decentralized agents to operate in an environment with a continuous state space without getting into individually or jointly unsafe states*.

Such a problem would naturally fit into a relaxed subset of decentralized partially observable Markov Decision Process (Dec-POMDP) framework [9] where the relaxation is due to full observability and independence of agents. However, not knowing other agents' policies makes the problem difficult. We attempt to learn other agents' policies, which eases the objective of avoiding jointly unsafe situations. Specifically, we use Gaussian Processes (GP) both for estimating other agents' actions, which then enables us to increase the probability joint safety, and for modeling the individual safety. We model the risk using confidence bounds, again both for other agents' possible actions, and for individual safety.

Our contributions in this paper are the following:

- We develop a novel decentralized multi-agent planning algorithm on continuous state space to achieve overall system safety more often than the existing algorithms.
- We show our algorithm has linear time complexity on the actions space size and polynomial time complexity on the number of visited states.
- Experiments show robots with our algorithm safely collaborate for exploration and exploitation with agents running standard planning and reinforcement learning algorithms.

II. Related Work

Safe Exploration. Single-agent safe exploration has been extensively studied. Turchetta *et al.* [10] established the SafeMDP algorithm for deterministic Markov Decision Processes (MDP) with discrete state spaces by assuming the risk value of each state is under some regularity that allows the use of GPs. Wachi *et al.* [11], again employing GPs, extended the work to both exploration and exploitation. Using a similar idea, Berkenkamp *et al.* [12] established a parameter exploration algorithm under multiple constraints, that would safely tune robots' parameters, as an extension to earlier work [13]. More recently, Bıyık *et al.* [14] leveraged continuity assumptions to deterministically guarantee safety for efficient exploration in unknown environments. Bajcsy *et al.* [15] and Fridovich-Keil *et al.* [16] developed reachability-based frameworks for safe navigation, again in unknown environments. While we employ many similar ideas, all of these works focused only on single-agent settings, whereas ensuring safety in decentralized multi-agent systems require modeling the other agents in the environment.

Safe Reinforcement Learning. On the safe reinforcement learning (RL), Basu *et al.* [17] developed a learning algorithm to handle risk-sensitive cost. Geibel and Wysotzki [18] formulated the risk as a second criterion based on cumulative return. Moldovan and Abbeel [19] proposed an algorithm that constrains the attention to the guaranteed safe policies. Fisac *et al.* [20] proposed a framework using reachability methods for guaranteeing safety during learning. Berkenkamp *et al.* [21] developed a framework for model-based RL using Lyapunov stability verification. However these works, too, focused only on single-agent settings. We refer to [22] for a comprehensive survey on safe RL.

Multi-Agent Reinforcement Learning. Many recent works studied how to train multiple robots that will operate in a decentralized manner [23]. They developed various techniques and got successful results in several different tasks, such as simulated navigation [24], video games [25], target tracking [26], soccer [27], etc. However, these tasks are either not safety-critical, or safety is hard-coded, which requires careful analysis and design. Fisac *et al.* [28] proposed an extension of Hamilton-Jacobi methods on reach-avoidance problem. While their approach specifies conventions agents typically follow, we make only very weak assumptions about other agents.

Intent Inference in Multi-Agent Settings. Predicting other agents' actions has been studied in the context of intent inference (and theory of mind [29], [30]). Bai *et al.* [31] demonstrated it is possible to autonomously drive in a crowd by estimating the intentions of pedestrians. Sadigh *et al.* [32] developed a method to enable the learning of other agents' internal states by actively probing them. Both of these works rely on the assumptions about the models of other agents' policies. Recently, Song *et al.* [33] proposed a multi-agent extension of generative adversarial imitation learning, which can help learn other agents' policies after observing a few instances. However, this is mostly limited to offline settings as it requires large computation powers to learn the policies.

III. Problem Definition

In this section, we formalize the multi-agent safe exploration problem and our key assumptions. In our setting, multiple robots interact with each other by *simultaneously* taking actions that explore a shared environment. Each robot should take only safe actions that not only satisfy the individual safety constraints, but also avoid moving the overall system to a jointly undesirable configuration.

The key challenge is that each robot needs to act simultaneously in a decentralized manner. Without prior knowledge of other agents' policies, they make decisions based on their own policies after observing the previous states and actions of all the agents. *Our goal is to develop such a decentralized strategy to safely navigate in the environment.*

We model this system as a Markov Decision Process with multiple agents (MDP-MA), where the agents share the same environment but their transitions are factorized.

Definition III.1. (MDP-MA) An MDP-MA with N agents is defined by a tuple $(\mathcal{S}, \mathcal{A}, f, r)$. $\mathcal{S}$ is a continuous set of states, where $\mathbf{s}_t = (s_t^1, s_t^2, ..., s_t^N) \in \mathcal{S}^N$ represents the state of all N agents at time t. We note the state of each agent s_t^i lies in $\mathcal{S}$. Similarly, $\mathcal{A}$ is the discrete set of actions, i.e., $\mathbf{a}_t = (a_t^1, a_t^2, ..., a_t^N) \in \mathcal{A}^N$. f is a probability distribution such that $f(s_{t+1}|s_t, a_t)$ is the probability of reaching s_{t+1} from s_t with action a_t. All agents act synchronously, so $\mathbf{s}_{t+1} \sim f(\cdot|\mathbf{s}_t, \mathbf{a}_t) = [f(\cdot|s_t^1, a_t^1), \ldots, f(\cdot|s_t^N, a_t^N)]$. Our formulations can be generalized to the settings where the action spaces are state- or agent-dependent. $r : \mathcal{S} \to \mathbb{R}$ is the unknown reward function shared by all the agents. In terms of rewards, agent i can only observe $r(s_t^i) + w_t^i$ at time step t, where $w_t^i \sim \mathcal{N}(0, {\eta^i}^2)$.

The goal of each agent is to take actions that optimize its own reward while *safely* planning in the environment. We now formalize the notion of *safety* in this MDP-MA.

Definition III.2. (Safety) A state s is *individually safe* if and only if $r(s) \geq h$, for some safety threshold h. In addition to individual safety, a set $U \subset S^N$ defines *jointly unsafe states*.

We want to note two important points. First, it is possible to have $(s_t^1, \ldots, s_t^N) \in U$, even though $r(s_t^i) \geq h$ for all $i \in \{1, \ldots, N\}$. For example, a specific location might be individually safe for drones, but having multiple drones in that location might cause catastrophic collisions. Second, as the reward is a function of individual states, jointly unsafe states are not induced by the reward function.

Assumption III.1. (Observability Assumptions) Following [10] and [11], we assume:

- All agents know the set U, the safety threshold h, the initial state $\mathbf{s}_0 \in S^N \setminus U$, and the dynamics f.
- At any time step t, all agents observe the states $\mathbf{s}_t$ and the actions $\mathbf{a}_t$.

Since f, the transition distribution, is known at all times to all agents, any agent could estimate the next state $\mathbf{s}_{t+1}$ based on its information about the MDP-MA if it could predict the actions of other agents accurately. However, the agents do not have prior information about other agents' policies.

Assumption III.2. (Reward Function Assumptions) With no assumption on r, an agent cannot learn other agents' policies without repeatedly observing all possible state configurations. Therefore,

- We assume that $\mathcal{S}$ is endowed with a positive definite kernel function $k_r(s, s')$ and that $r(s)$ has bounded norm in the associated Reproducing Kernel Hilbert Space (RKHS).
- We also assume L-Lipschitz continuity of the reward function r with respect to some metric $d(\cdot, \cdot)$ on $\mathcal{S}$. This

is guaranteed by many commonly used kernels with high probability [34], [35].

Objective. Given the problem definition and assumptions, our goal is to achieve a predefined objective, e.g. maximizing the number of states explored or maximizing cumulative reward, while avoiding individually and jointly unsafe states.

IV. Multi-Agent SafeMDP Algorithm

We formalize individual and joint safety separately and use confidence bounds to determine whether or not a constraint is satisfied. Sec. IV-A introduces a method to find the states that satisfy individual safety with high confidence and Sec. IV-B explains our approach for modeling other agents' policies to achieve joint safety.

A. Iterative SafeMDP

Our algorithm finds the most desirable one-step reachable state with high probability of being individually safe, as well as being returnable to the previously found safe states within one step.

We use a Gaussian Process (GP) to model the reward function on the state space for the agent running our algorithm [35]. Given Assumption III.2, we model the reward function using a GP as

$$r \sim \mathcal{GP}_r(\mu_r, k_r), \tag{1}$$

where $\mu_r(s)$ is the mean function and $k_r(s, s')$ is the covariance function. We assume the prior mean μ_r of all states is 0. The variance $\sigma_r^2(s) = k_r(s, s)$ encodes the noise of the environment from observations and our uncertainty. Our algorithm updates $\mathcal{GP}_r$ after every reward observation and utilizes it to estimate the reward of the next state. We refer to [36] for GP posterior update with new observations. We define the confidence bounds of $\mathcal{GP}_r$ as $C_{rt}(s) = [\mu_{rt}(s) \pm \beta_r(t)\sigma_{rt}(s)]$ for some $\beta_r(t) \geq 0$. We denote

$$\begin{aligned} r_t^\mu(s) &= \mu_{r_t}(s), \\ r_t^u(s) &= \mu_{r_t}(s) + \beta_r(t)\sigma_{rt}(s), \\ r_t^l(s) &= \mu_{r_t}(s) - \beta_r(t)\sigma_{rt}(s). \end{aligned} \tag{2}$$

Given a state-action pair at time step t, the lower bound on the expected reward of the next state is:

$$lr_t(s, a) = \int_{\mathcal{S}} r_t^l(\varsigma) f(\varsigma|s, a) d\varsigma. \tag{3}$$

Definition IV.1. (Returnability) At any time step t, given an initial safe state set $S_0 \supseteq \{s_0\}$, we define $S_j = S_{j-1} \cup \{s \in S \mid \exists a \in A, \int_{\varsigma \in S_{j-1}} f(\varsigma|s, a) d\varsigma \geq \tau \wedge lr_t(s, a) \geq h\}$ and $\bar{S}_t = \lim_{j\to\infty} S_j$, for some $\tau \in [0, 1]$. And the returnability of a state-action pair (s, a) is defined as

$$return_t(s, a) = \int_{\bar{S}_t} f(\varsigma|s, a) d\varsigma. \tag{4}$$

We emphasize that τ is a threshold on the returnability property. Given the lower bound on the expected reward and the Definition IV.1, we say a state-action pair (s_t, a_t) can be safely realized if and only if $lr_t(s_t, a_t) \geq h$ and $return_t(s_t, a_t) \geq \tau$. We use this definition to restrict the actions our agent can take.

B. Multi-agent Modeling

To avoid jointly unsafe states, our algorithm predicts other agents' actions. To do so, we make the following assumption to account for both exploration and exploitation [37].

Assumption IV.1. (General Policy Assumptions) Agent i (any other agent in the environment) follows a policy that is a combination of commonly used exploration strategies, namely Optimism in the face of Uncertainty (OFU) and Boltzmann policy, which we describe in detail below. This is a mild assumption, because combining these strategies leads to a very general and inclusive class of policies.

We define a function g^i, which given the combined policy, computes the probability of taking an action that would transition from s_t^i to s_{t+1}^i. We now describe the elements of the combined policy.

Q-Function with GP. Before we derive OFU and Boltzmann, we first introduce the Q-functions (action-value functions) for each agent. Given the GP reward model, we leverage Q-learning to find the mean of the Q-function:

$$Q_t^\mu(s, a) = \int_{\mathcal{S}} \left(r_t^\mu(\varsigma) + \gamma \max_{a'} Q_t^\mu(\varsigma, a')\right) f(\varsigma|s, a) d\varsigma \tag{5}$$

where $\gamma \in (0, 1)$ is a discount factor. We similarly define Q^u and Q^l as the confidence bounds of the Q-function by using r^u and r^l instead of r^μ, respectively. We denote $C_{Q_t}(s, a) = [Q_t^l(s, a), Q_t^u(s, a)]$. These definitions rely on an independence assumption for faster computation and good results both empirically and theoretically (see Sec. V and VI). In fact, the posterior distribution above assuming independence is more conservative due to reward correlation between states.

To learn a Q-function in a continuous domain, we adapt temporal difference error (TD-error) learning. For Q^μ, Q^u and Q^l, which are parameterized by θ^μ, θ^u and θ^l, respectively,

$$\theta \leftarrow \arg\min_\theta \|r_t^\mu(s') + \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s_t, a_t)\| \tag{6}$$

where $s' \sim f(\cdot|s_t, a_t)$, and θ^- is the set of parameters that are updated after every Δ time steps by copying θ (as in [38]). In this way, Q values can be approximated.

Optimism in the face of Uncertainty. OFU is a classic exploration strategy that favors the actions with high upper bound in potential return [39], [40]. Given a Q-function distribution, OFU can be written as

$$\pi_o(s, a) = \frac{\exp(Q^u(s, a)/T_o)}{\sum_{a'} \exp(Q^u(s, a')/T_o)}, \tag{7}$$

which outputs the probability of taking action a at state s, where $T_o > 0$ is the unknown temperature parameter. To derive an upper bound of the probability that a state-action pair is observed, we get the upper bound of π_o^i as

$$\begin{aligned} \pi_o^{iu}(s, a) &= \frac{\exp(Q^{iu}(s, a)/T_o^i)}{\exp(Q^{iu}(s, a)/T_o^i) + \sum_{a' \neq a} \exp(Q^{i\mu}(s, a')/T_o^i)} \\ &\geq \pi_o^i(s, a) \end{aligned} \tag{8}$$

Boltzmann Policy. Similar to OFU, the Boltzmann exploration strategy [41] is:

$$\pi_b(s,a) = \frac{\exp(Q^{\mu}(s,a)/T_b)}{\sum_{a'} \exp(Q^{\mu}(s,a')/T_b)} \tag{9}$$

where $T_b > 0$ is the unknown temperature parameter. The upper confidence bound is:

$$\begin{aligned}\pi_b^{iu}(s,a) &= \frac{\exp(Q^{i\mu}(s,a)/T_b^i)}{\exp(Q^{i\mu}(s,a)/T_b^i) + \sum_{a' \neq a} \exp(Q^{il}(s,a')/T_b^i)} \\ &\geq \pi_b^i(s,a).\end{aligned} \tag{10}$$

Exploitation is implicitly covered within the Boltzmann policy formulation with a right choice of T_b.

Combining OFU and Boltzmann. The agent i will follow OFU with probability ϵ^i, and Boltzmann with probability $1-\epsilon^i$ for some unknown $0 \leq \epsilon^i \leq 1$. When $T_o^i \to \infty$ and $T_b^i \to 0$, the policy reduces to a pure ϵ-greedy policy. This combination completes the definition of g^i, the transition probability between states given estimated policies and ϵ^i (see Assumption IV.1), and allows it to model all of OFU, Boltzmann and ϵ-greedy strategies.

C. Inference of Joint Policy Parameters

Having described the strategies, we now explain how to jointly estimate T_o^i, T_b^i and ϵ^i. By assuming uniform prior over $P(\epsilon^i, T_b^i, T_o^i | Q^i)$, after a series of observations ξ^i, Bayes' rule gives

$$P(\epsilon^i, T_b^i, T_o^i | \xi^i, Q^i) \propto P(\xi^i | \epsilon^i, T_b^i, T_o^i, Q^i). \tag{11}$$

We find the maximum likelihood estimate of $(\epsilon^{i*}, T_b^{i*}, T_o^{i*})$:

$$\epsilon^{i*}, T_b^{i*}, T_o^{i*} = \underset{\epsilon^i, T_b^i, T_o^i}{\arg\max} \log(p(\xi^i | \epsilon^i, T_b^i, T_o^i, Q^i)). \tag{12}$$

Because the form of g^i is known, we use $g^i(\epsilon^{i*}, \beta_r^i, T_b^{i*}, T_o^{i*}, \pi_o^{iu}, \pi_b^{iu}, s_t^i, s_{t+1}^i)$ as an upper confidence bound estimate of $g^i(\epsilon^i, \beta_r^i, T_b^i, T_o^i, \pi_o^i, \pi_b^i, s_t^i, s_{t+1}^i)$.

We now have a full loop of inference and belief update. In the inference stage, each agent running our algorithm would keep a GP for r and a corresponding estimated Q-function distribution. The agent would also keep track of policy parameters for each of the other agents. Given the expression above, the lower confidence bound of not entering any jointly unsafe states for our agent is

$$lc(s_t^1, a_t^1) = \int_{\mathcal{S}} \left(1 - \int_{u \in U : u^1 = \varsigma} \prod_{i=2}^{N} g^i(\dots) du \right) f(\varsigma | s_t^1, a_t^1) d\varsigma, \tag{13}$$

where $g^i(\dots)$ is short for $g^i(\epsilon^i, \beta_r^i, T_o^i, T_b^i, \pi_o^{iu}, \pi_b^{iu}, s_t^i, u^i)$.

D. Overall Algorithm

Algorithm 1 introduces the overall method. We compute the expected lower bound reward according to Eq. (4), and then select the set of individually safe actions $A_{\text{hi-rew}}$ (lines 6-7). We then compute the set of actions, $A_{\text{joint-safe}}$, with low risk of joint unsafety as defined in Definition III.2 of Sec. IV-B (lines 8-9), and also compute the set of actions, A_{safe} that satisfies both constraints (line 10). Finally the algorithm selects the action with the lowest probability of joint unsafety if the available action set is empty, or selects the action that optimizes the agent's objective Obj within the available action set (line 11-13). We update the Q-function and the parameters following Eq. (5) and Sec. IV-C (line 15-17).

Algorithm 1 Multi-agent Safe Q-Learning

```
1: Input: $\mathcal{S}, \mathcal{A}, f, S_0, c, h, \tau, \beta_r, Obj$
2: Initialize $Q$.
3: Initialize $\mathcal{GP}_r$ for reward estimate.
4: Initialize $\boldsymbol{\epsilon}$, $\mathbf{T_b}$ and $\mathbf{T_o}$.
5: for $t = 1, 2, ...$ do
6:     Compute $lr(s_t^i, a)$ and $return(s_t^i, a)$ for $\forall a \in \mathcal{A}$
7:     $A_{\text{hi-rew}} \leftarrow \{a | lr(s_t^i, a) \geq h \wedge return(s_t^i, a) > \tau\}$
8:     Compute $lc(s_t^i, a), \forall a \in \mathcal{A}$
9:     $A_{\text{joint-safe}} \leftarrow \{a | lc(s_t^i, a) \geq c\}$
10:    $A_{\text{safe}} \leftarrow A_{\text{hi-rew}} \bigcap A_{\text{joint-safe}}$
11:    if $A_{\text{safe}} = \emptyset$ then
12:        $A_{\text{safe}} \leftarrow \arg\max_a lc(s_t^i, a)$
13:    $a_t^i \leftarrow \arg\max_{a \in A_{\text{safe}}} Obj(s_t^i, a)$
14:    $s_{t+1}^i \sim f(s_t^i, a_t^i)$
15:    Update $\mathcal{GP}_r$ using $r(s_{t+1}^i)$.
16:    Update $Q$ with $\mathcal{GP}_r$ and $f$
17:    Update $\boldsymbol{\epsilon}$, $\mathbf{T_b}$ and $\mathbf{T_o}$
```

V. Theoretical Results

In this section, we discuss the theoretical results of our algorithm. Mainly, we discuss the accuracy of our GP estimate on the reward along with $\beta_r(t)$, on the value function and we discuss the computational complexity of our algorithm.

Reward Estimation Accuracy with Gaussian Processes. The confidence interval of the GP for r depends on $\beta_r^i(t)$, whose tuning has been well studied in [10] for the single-agent SafeMDP algorithm. Their result can be applied to our setting by choosing

$$\beta_r^i(t) = 2B^i + 300\alpha_t^i \log^3(t/\delta^i), \tag{14}$$

where B^i is the bound on the RKHS norm of the function $r(\cdot)$, δ^i is the probability of agent i visiting individually unsafe states, and α_t^i is the maximum mutual information that can be gained about $r(\cdot)$ from t noisy observations. The information capacity α_t^i has a sublinear dependency on t for many commonly used kernels [35]. Assuming $||r||_k^2 \leq B^i$ and the noise w_t is zero-mean conditioned on the history as well as uniformly bounded by η for all $t > 0$, if we choose β_t^i above, then for all $s \in \mathcal{S}$ and $t > 0$, with probability at least $1 - \delta^i$ that $r(s^i) \in C_{rt}(s^i)$, where $C_{rt}(s^i)$ is the estimated confidence bounds of the reward function on state s^i at time step t [10].

Value Function Estimation Accuracy. To be able to accurately estimate the Boltzmann policy, the algorithm must make accurate estimates of the value function. Based on the reward functions' estimation accuracy, we can derive the following two theorems for the accuracy of value functions.

Theorem V.1. *If 1) $\beta_r^i(t)$ follows Eq. (14), 2) the states visited by agent i are also visited by the estimating agent, and 3) Q is in the form of universal state representation, then $Q(s,a) \in C_{Q_t}(s,a)$ with at least probability $1-\delta^i$.*

Proof. [10] proved that with the choice of $\beta_r^i(t)$ above, there is at least probability $1-\delta^i$, $r(s^i) \in C_{r_t}(s^i), \forall s^i \in \mathcal{S}$. The Q-learning algorithm we define is by sampling potential transitions (to s') and perform

$$Q_t(s^i,a^i) = r_t(s') + \gamma \max_{a'} Q_t(s',a') \tag{15}$$

Hence at convergence with a universal representation of Q-function, the Q-function can be written as

$$Q_t(s^i,a^i) = \mathbb{E}[r_t(s')] + \gamma\mathbb{E}[r_t(s'')] + \gamma^2\mathbb{E}[r_t(s''')] + \dots \tag{16}$$

where we use s', s'', s''', $\dots$ denote the future states in the optimal trajectory of taking a^i at s^i. Equation (16) holds for each of Q_t^μ, Q_t^l, and Q_t^u respectively with r_t^μ, r_t^l, and r_t^u. Since $\forall s \in S$, $r_t(s) \in C_{r_t}(s)$ with probability at least $1-\delta^i$; we have $\mathbb{E}[r_t(s')] \in [\mathbb{E}[r_t^l(s')], \mathbb{E}[r_t^u(s')]]$, with probability at least $1-\delta^i$ for any state distribution. Therefore with probability at least $1-\delta^i$,

$$\begin{aligned} Q_t(s^i,a^i) \in & [\mathbb{E}[r_t^l(s')] + \gamma\mathbb{E}[r_t^l(s'')] + \dots, \\ & \mathbb{E}[r_t^u(s')] + \gamma\mathbb{E}[r_t^u(s'')] + \dots] \\ = & [Q_t^l(s^i,a^i), Q_t^u(s^i,a^i)]. \end{aligned} \tag{17}$$

□

Even without the prior knowledge on δ^i, $C_{Q_t}(s^i,a^i)$ covers at least $\beta_r^i(t)$ standard deviations from the mean value, where the probability of $Q(s^i,a^i)$ being bounded by $C_{Q_t}(s^i)$ can be directly found using a standard Z-score table.

However, from Eq. (5), we observe the variance of the value function is monotonically increasing during the update. Therefore, we need to bound the confidence intervals of the value function. The next theorem establishes such a bound.

Theorem V.2. $Q_t^u(s,a) - Q_t^l(s,a) \le \frac{2\beta_r(t)\max_{s\in S}\sigma_{r_t}(s)}{1-\gamma}$ *for discounted infinite-horizon MDP, $\forall s \in \mathcal{S}, a \in \mathcal{A}$ and any given $\beta_r(t)$.*

Proof. Using Eq. (16),

$$\begin{aligned} & Q_t^u(s,a) - Q_t^l(s,a) \\ & = (\mathbb{E}[r^u(s')] + \gamma\mathbb{E}[r^u(s'')] + \gamma^2\mathbb{E}[r^u(s''')] + \dots) \\ & \quad - (\mathbb{E}[r^l(s')] + \gamma\mathbb{E}[r^l(s'')] + \gamma^2\mathbb{E}[r^l(s''')] + \dots) \\ & = (\mathbb{E}[r^u(s') - r^l(s')] + \gamma\mathbb{E}[r^u(s'') - r^l(s'')] + \dots \\ & \le 2\beta_r(t)\mathbb{E}[\sigma_{r_t}(s')]) + 2\gamma\beta_r(t)\mathbb{E}[\sigma_{r_t}(s'')] + \dots \\ & = 2\beta_r(t)(\mathbb{E}[\sigma_{r_t}(s')] + \gamma\mathbb{E}[\sigma_{r_t}(s'')] + \dots \\ & \le \frac{2\beta_r(t)\max_s \sigma_{r_t}(s)}{1-\gamma}. \end{aligned} \tag{18}$$

□

Computational Complexity. For the algorithm to have online execution ability, it must be scalable and fast. With the prior knowledge that GP updates are $O(|\mathcal{S}_\mathrm{v}|^3)$ where $\mathcal{S}_v$ is the set of states that are visited by any agent [36], we conclude our algorithm is linear in $|\mathcal{A}|$ and polynomial in $|\mathcal{S}_v|$.

Theorem V.3. *The Multi-agent Safe Q-Learning algorithm has a time complexity of $O(|\mathcal{S}_v|^3 + |\mathcal{A}| + Nt)$ at time step t in a MDP-MA with N agents using classic GPs. Using GP kernel approximation methods described in [42], [43], the time complexity reduces to $O(|\mathcal{S}_v| + |\mathcal{A}| + Nt)$. If we apply sampling methods described in [44], we can further reduce the time complexity to $O(|\mathcal{A}| + Nt)$.*

Proof. With the prior knowledge that GP updates are $O(|\mathcal{S}_v|^3)$ [36], we can derive the time complexity bounds as follows.

There are four major components of this algorithm. GP update for the reward function is on the state space and for all agents, hence complexity $O(|\mathcal{S}_v|^3)$. Choosing the optimal action among the safe actions has a time complexity of $O(|\mathcal{A}|)$. Q-function update with sampling takes $O(1)$. Optimization for ϵ and T_b^i is linear to the number of steps in trajectory, hence complexity $O(Nt)$.

Incorporating all complexity bounds above, the time complexity bound of the algorithm is $O(|\mathcal{S}_v|^3 + |\mathcal{A}| + Nt)$.

With optimizations on GPs in [42], [43] to make linear complexity and sampling in [44] to make constant complexity, the algorithm complexity becomes $O(|\mathcal{S}_v| + |\mathcal{A}| + Nt)$ and $O(|\mathcal{A}| + Nt)$ respectively. □

VI. Experiments

To assess the performance of our method, **Multi Safe Q-Agent**, we performed several simulation experiments using the following baselines:

- **Single Safe MDP Agent:** The algorithm in [10] with the modifications discussed in Section IV-A
- **Naïve Q-Agent:** Q-learning agent that assumes all other agents apply uniformly random policies.
- **Bayesian Q-Agent:** Q-learning agent receives additional negative reward for entering unsafe states and keeps a Bayesian belief on others' policies on discretized domain.
- **Safe Q-Agent:** Q-learning agent that takes all agents' states as its state space and receives additional negative reward for both individual unsafety and joint unsafety.

The GPs employed by the agents use an RBF Kernel with a length scale of 10 and a prior standard deviation of 10 and a White Noise Kernel with a prior standard deviation of 10. The Q-functions are estimated using neural networks with two hidden layers of size 50.

A. Mars Rover Experiment

We downloaded the Mars surface map from `http://www.uahirise.org/PDS/DTM/PSP/ORB_010200_010299`High Resolution Imaging Science Experiment (HiRISE) [45], and selected a square region with an area of 100 m^2, starting at 30.6° latitude and 202.2° longitude. The agents have 4 possible actions: up, down, left and right. Each action moves the agent 1 m with some error that follows independent Gaussian distributions in both axes with means 0 and variances 0.1. When two rovers are too close to each other, they collide. These define joint safety. If an agent takes an action towards outside the boundary, it respawns at the other side of the map so that an agent is always forced to move instead of trying to stay at its original state to avoid collision. The individual safety condition is the altitude: the rover may be not recoverable when its altitude

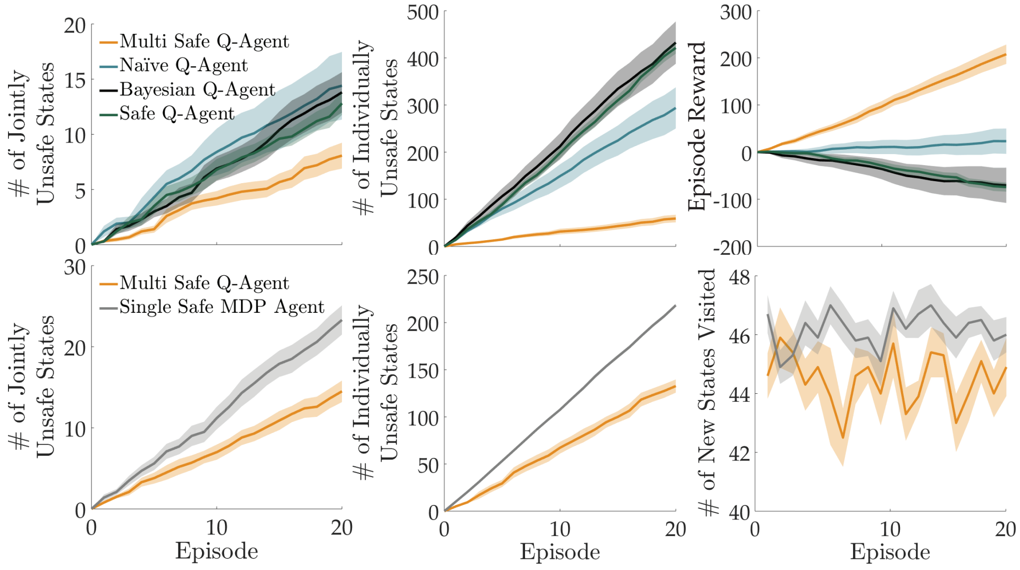

Fig. 2: Experiment Results for Exploitation (top) and Exploration (bottom) with Mars Rovers (mean±s.e.)

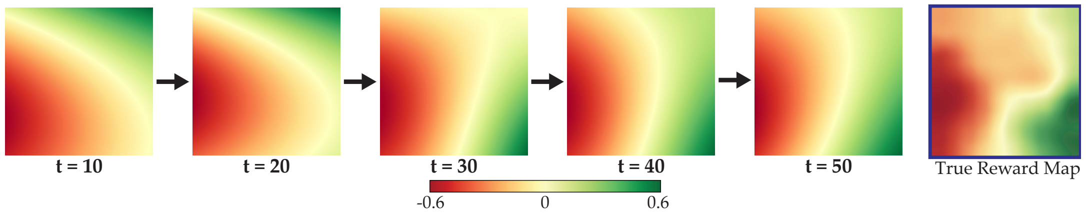

Fig. 3: Learned reward values by the Multi Safe Q-Agent when exploring, and the true reward map are shown.

is too low.

We simulate two objectives: Exploitation and exploration. In each, there exist 19 ϵ-greedy Q-learning agents with randomly chosen ϵ values. They do not try to avoid unsafe states. The experiment is set up in this way so that probability of collisions in the environment is high without careful navigation. We test how each agent performs in such hostile environment with its own objective. Agents adopt $h = -0.5$, $\tau = 1$, $c = 0.7$ and collision distance threshold of 0.1 meters.

We run each experiment 10 times with random initial states (constrained to individually and jointly safe states). Each experiment is run with 20 episodes of 50 time steps. We use Naïve Q-Agent, Bayesian Q-Agent and Safe Q-Agent as baselines for the exploitation setting because they optimize episode reward without a specific goal of optimizing for exploration. We use Single Safe MDP Agent as baseline for exploration because it optimizes only for state exploration.

Exploitation. In this experiment, we use altitude as the agents' reward. The results are shown in Fig. 2(top). Multi Safe Q-Agent significantly outperforms other candidates in both safety and episode reward (we exclude the additional negative reward for Bayesian Q-Agent and Safe Q-Agent for fairness).

Bayesian Q-Agent and Safe Q-Agent are safer than the Naïve Q-Agent, but since their rewards are corrupted by the additional penalty, they did not properly learn how to maximize reward. Naïve Q-Agent's episode reward is also low due to the limit in the actions it can choose. Since it assumes random policy for other agents, it unnecessarily eliminates many trajectories that would possibly lead to higher episode reward. On the other hand, Multi Safe Q-Agent outperforms the baselines by getting higher and higher rewards after learning from previous episodes.

Exploration. In this experiment, Multi Safe Q-Agent chooses the actions to visit the most uncertain state (the state with highest variance by the GP) that satisfies the safety constraints in the algorithm. We compare its performance against Single Safe MDP Agent, which is already designed for exploration. We show our agent's estimate of the altitude map in Fig. 3. Quantitative results are in Fig. 2(bottom). Multi Safe Q-Agent significantly outperforms in safety, but is marginally worse in terms of the number of new states visited. This is reasonable, because achieving higher safety usually means a more constrained set of actions, which then harms the exploration. Hence, we conclude it is much safer and performs exploration comparably well.

B. Quadcopter Collaborative Experiment

In this experiment, we aim to evaluate agents' capability of safe collaboration[1]. Two quadcopters are initialized in the domain to ship an item to the destination together. The item would be lost if the quadcopters are too far apart from each other. The setup is shown in Fig. 4.

We discretize the action space of the quadcopters such that each has 6 possible actions: up, down, forward, backward,

[1]A video of the experiment is at `http://youtu.be/l76glwgF67k`.

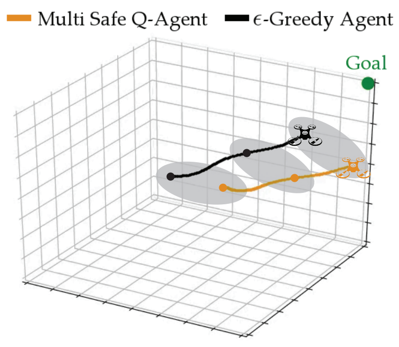

Fig. 4: Quadcopter Experiment Setup

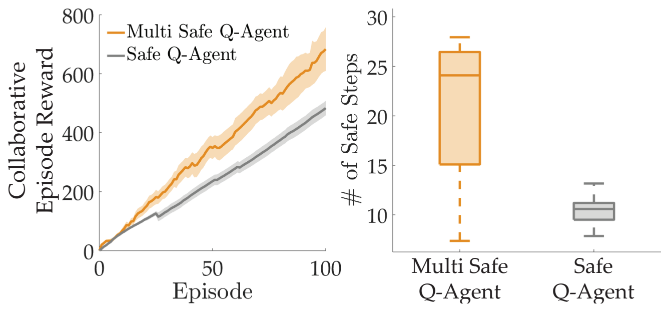

Fig. 5: Quadcopter Experiment Results (mean±s.e. for the line plots). The box plot shows the distribution of average number of safe steps across 10 experiment runs before termination due to unsafety. Each experiment's number of safe steps is averaged over 100 episodes.

left and right, each of which would move the agent 0.1 unit. When the agents move, their actions have independent Gaussian errors in all three axes with means 0 and variances 0.1. We discretize the time such that when an action is chosen, a PID controller navigates the quadcopter to the destination before the next time step. Quadcopters start at $(0.5, 0, 0)$ and $(-0.5, 0, 0)$. The destination is at $(2, 2, 2)$. The maximum safe distance between the agents is 3. Reward is the normalized negation of the Euclidean distance to the destination. $h = -8.0, \tau = 1, c = 0.7$. This task is extremely challenging as the quadcopters are uncoordinated and do not have any information about each other. We run 10 independent experiments in this domain. Each experiment consists of 100 episodes. Each episode terminates when any safety constraint is violated or when the number of time steps reaches 100. We compare our method with Safe Q-Agent, as it is the only baseline that specifically accounts for both individual and joint safeties. Each experiment is initialized with one of the agents of interest, and an ϵ-greedy Q-learning agent with $\epsilon = 0.1$.

Figure 5 shows the total episode reward of the agents and also the number of safe steps before the task is failed. While both agents collect higher rewards by learning from previous episodes, Multi Safe Q-Agent clearly outperforms the Safe Q-Agent in both total reward and safety.

VII. Discussion

Summary. In this paper, we presented a decentralized planning algorithm that enables agents to avoid stepping into *individually* or *jointly* unsafe states. We use a GP-based approach to estimate safety and uncertainty. Our algorithm assumes very little prior knowledge on other agents and learns their policies through observations. We showed the algorithm has polynomial time complexity in the number of visited states, available actions, and agents. We also gave a performance guarantee on the estimated Q-functions of other agents. We empirically demonstrate our algorithm in collaborative Mars rover and quadcopter experiments. Our results suggest our algorithm outperforms all other baselines in terms of safety and also has the best performance in the exploitation setting. We would like to emphasize that although there is extensive work in the area of decentralized planning and control, most previous multi-agent safe learning work focuses either on precomputed policies or is not decentralized [46], [47]. Hence, a fair comparison would be only with algorithms related to our problem such as SafeMDP [10].

We would also like to note that our work extends to human-robot collaboration tasks where each human can be modeled in the same decentralized way. For example, in underwater robotics [48], robots assist scuba divers to complete the tasks where they share the same reward and can use the diver's vital failure, e.g. running out of oxygen, as the joint unsafety. Another example is that surgical robots can share the reward with surgeons and use the patient's condition to define joint unsafety. This method also has the potential to go beyond robotics and can be applicable in other multi-agent AI settings such as the game of Dota and League of Legends, where a team of agents would share a common reward and the joint unsafety would be the home being attacked by the opponents.

Limitations and Future Work. There are a few limitations that we plan to address as part of future work. The algorithm only avoids immediate jointly unsafe states, but does not plan on other agents' potential trajectories. One can easily imagine a scenario where the joint unsafe state is guaranteed to occur multiple steps ahead. A potential opportunity here is to have a multi-step trajectory roll out and select the safest option. Another strong assumption we make is that reward function is shared across agents. This assumption allows for adequate online learning and quick reaction from our agent. However, in some of the real-world collaborative tasks, this assumption may not hold. Two potential solutions are to directly learn other agents policies or rewards. Third, one could improve the modeling of other agents' policies by using different learning algorithms rather than GPs. Finally, an empirical validation of our algorithm on real-world experiments is required.

Conclusion. Despite these limitations, we are encouraged to see our algorithm demonstrating safe behavior in collaboration with other agents with very little prior knowledge of their policies. We also look forward to exploring applications of our algorithm beyond collaborative navigation to other multi-agent partially observable settings such as in manipulation or in human-robot interaction.

Acknowledgments

Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We acknowledge funding by FLI grant RFP2-000 and the DARPA Assured Autonomy program.

References

[1] Y. Cao, W. Yu, W. Ren, and G. Chen, “An overview of recent progress in the study of distributed multi-agent coordination,” *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2013.

[2] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, “Hierarchical game-theoretic planning for autonomous vehicles,” in *International Conference on Robotics and Automation (ICRA)*, 2019.

[3] A. Viseras, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, and L. Merino, “Decentralized multi-agent exploration with online-learning of gaussian processes,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 4222–4229.

[4] R. Bähnemann, D. Schindler, M. Kamel, R. Siegwart, and J. Nieto, “A decentralized multi-agent unmanned aerial system to search, pick up, and relocate objects,” in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, IEEE, 2017, pp. 123–128.

[5] Z. Wang, S. Singh, M. Pavone, and M. Schwager, “Cooperative object transport in 3d with multiple quadrotors using no peer communication,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1064–1071.

[6] C. Amato, G. Konidaris, S. Omidshafiei, A.-a. Agha-mohammadi, J. P. How, and L. P. Kaelbling, “Probabilistic planning for decentralized multi-robot systems,” in *2015 AAAI Fall Symposium Series*, 2015.

[7] C. Amato, G. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling, “Planning for decentralized control of multiple robots under uncertainty,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 1241–1248.

[8] S. Omidshafiei, A.-A. Agha-Mohammadi, C. Amato, and J. P. How, “Decentralized control of partially observable markov decision processes using belief space macro-actions,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 5962–5969.

[9] C. Amato, G. Chowdhary, A. Geramifard, N. K. Ure, and M. J. Kochenderfer, “Decentralized control of partially observable markov decision processes,” in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, IEEE, 2013, pp. 2398–2405.

[10] M. Turchetta, F. Berkenkamp, and A. Krause, “Safe exploration in finite markov decision processes with gaussian processes,” in *Advances in Neural Information Processing Systems*, 2016, pp. 4312–4320.

[11] A. Wachi, Y. Sui, Y. Yue, and M. Ono, “Safe exploration and optimization of constrained mdps using gaussian processes,” in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[12] F. Berkenkamp, A. P. Schoellig, and A. Krause, “Safe controller optimization for quadrotors with gaussian processes,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, 2016, pp. 491–496.

[13] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, “Safe exploration for optimization with gaussian processes,” in *International Conference on Machine Learning*, 2015, pp. 997–1005.

[14] E. Bıyık, J. Margoliash, S. R. Alimo, and D. Sadigh, “Efficient and safe exploration in deterministic markov decision processes with unknown transition models,” in *Proceedings of the American Control Conference (ACC)*, 2019.

[15] 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,” *arXiv preprint arXiv:1905.00532*, 2019.

[16] D. Fridovich-Keil, J. F. Fisac, and C. J. Tomlin, “Safely probabilistically complete real-time planning and exploration in unknown environments,” in *International Conference on Robotics and Automation (ICRA)*, 2019.

[17] A. Basu, T. Bhattacharyya, and V. S. Borkar, “A learning algorithm for risk-sensitive cost,” *Mathematics of operations research*, vol. 33, no. 4, pp. 880–898, 2008.

[18] P. Geibel and F. Wysotzki, “Risk-sensitive reinforcement learning applied to control under constraints,” *Journal of Artificial Intelligence Research*, vol. 24, pp. 81–108, 2005.

[19] T. M. Moldovan and P. Abbeel, “Safe exploration in markov decision processes,” in *Proceedings of the 29th International Conference on Machine Learning*, Omnipress, 2012, pp. 1451–1458.

[20] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, “A general safety framework for learning-based control in uncertain robotic systems,” *IEEE Transactions on Automatic Control*, 2018.

[21] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, “Safe model-based reinforcement learning with stability guarantees,” in *Advances in neural information processing systems*, 2017, pp. 908–918.

[22] J. García and F. Fernández, “A comprehensive survey on safe reinforcement learning,” *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[23] F. L. Da Silva, M. E. Taylor, and A. H. R. Costa, “Autonomously reusing knowledge in multiagent reinforcement learning.,” in *IJCAI*, 2018, pp. 5487–5493.

[24] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[25] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[26] C. Zhang and V. Lesser, "Coordinating multi-agent reinforcement learning with limited communication," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1101–1108.

[27] H. M. Le, Y. Yue, P. Carr, and P. Lucey, "Coordinated multi-agent imitation learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 1995–2003.

[28] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Proceedings of the 18th international conference on hybrid systems: computation and control*, ACM, 2015, pp. 11–20.

[29] S. Devin and R. Alami, "An implemented theory of mind to improve human-robot shared plans execution," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, 2016, pp. 319–326.

[30] T. Hellström and S. Bensch, "Understandable robots-what, why, and how," *Paladyn, Journal of Behavioral Robotics*, vol. 9, no. 1, pp. 110–123, 2018.

[31] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 454–460.

[32] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, IEEE, 2016, pp. 66–73.

[33] J. Song, H. Ren, D. Sadigh, and S. Ermon, "Multi-agent generative adversarial imitation learning," 2018.

[34] S. Ghosal, A. Roy, *et al.*, "Posterior consistency of gaussian process prior for nonparametric binary regression," *The Annals of Statistics*, vol. 34, no. 5, pp. 2413–2429, 2006.

[35] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, Omnipress, 2010, pp. 1015–1022.

[36] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*, Springer, 2004, pp. 63–71.

[37] A. D. Tijsma, M. M. Drugan, and M. A. Wiering, "Comparing exploration strategies for q-learning in random stochastic mazes," in *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, IEEE, 2016, pp. 1–8.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[39] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[40] R. Agrawal, "Sample mean based index policies by o (log n) regret for the multi-armed bandit problem," *Advances in Applied Probability*, vol. 27, no. 4, pp. 1054–1078, 1995.

[41] A. G. Barto, S. J. Bradtke, and S. P. Singh, *Real-time learning and control using asynchronous dynamic programming*. University of Massachusetts at Amherst, Department of Computer and Information Science, 1991.

[42] S. Dasgupta, K. Sricharan, and A. Srivastava, "Finite rank deep kernel learning," in *Third workshop on Bayesian Deep Learning, NeurIPS*, 2018.

[43] A. Banerjee, D. B. Dunson, and S. T. Tokdar, "Efficient gaussian process regression for large datasets," *Biometrika*, vol. 100, no. 1, pp. 75–89, 2012.

[44] A. Jain, T. Nghiem, M. Morari, and R. Mangharam, "Learning and control using gaussian processes," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, IEEE, 2018, pp. 140–149.

[45] A. S. McEwen, E. M. Eliason, J. W. Bergstrom, N. T. Bridges, C. J. Hansen, W. A. Delamere, J. A. Grant, V. C. Gulick, K. E. Herkenhoff, L. Keszthelyi, *et al.*, "Mars reconnaissance orbiter's high resolution imaging science experiment (hirise)," *Journal of Geophysical Research: Planets*, vol. 112, no. E5, 2007.

[46] K. D. Julian, M. J. Kochenderfer, and M. P. Owen, "Deep neural network compression for aircraft collision avoidance systems," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 3, pp. 598–608, 2018.

[47] C. Vrohidis, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Safe decentralized and reconfigurable multi-agent control with guaranteed convergence," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 267–272.

[48] O. Khatib, X. Yeh, G. Brantner, B. Soe, B. Kim, S. Ganguly, H. Stuart, S. Wang, M. Cutkosky, A. Edsinger, *et al.*, "Ocean one: A robotic avatar for oceanic discovery," *IEEE Robotics & Automation Magazine*, vol. 23, no. 4, pp. 20–29, 2016.