Robotics for People: Perspectives on Interaction, Learning and Safety

Amir Aly, Andrea Bajcsy, Angela Schoellig, David Fridovich-Keil, Filipa Correia, Kim Baraka, Matthew Gombolay, Nakul Gopalan, Ransalu Senanayake, Shreyas Kousik, et al.

Proceedings of the Workshop

Robotics for People (R4P): Perspectives on Interaction, Learning and Safety

in Conjunction with the Robotics: Science and Systems (RSS) Conference (July 15th, 2021)

Amir Aly, Andrea Bajcsy, Angela Schoellig, David Fridovich-Keil, Filipa Correia, Kim Baraka, Matthew Gomboly, Nakul Gopalan, Ransalu Senanayake, Shreyas Kousik, SiQi Zhou, Tesca Fitzgerald

Game-theoretic Model of Trust to Infer Human's Observation Strategy of Robot Behavior

Sailik Sengupta*†
Amazon AI
sailiks@amazon.com

Zahra Zahedi*
Arizona State University
zzahedi@asu.edu

Subbarao Kambhampati
Arizona State University
rao@asu.edu

***Abstract*—We consider scenarios where a worker robot has incentive to deviate from a preferred plan when a human supervisor is not monitoring it. We show that in such scenarios, via human subject evaluation, human supervisors choose sub-optimal observation strategies. To address this, we first consider a game-theoretic framework of trust to formally model such interaction. Then, we leverage this model to infer an optimal supervision strategy that does not need the human to place their trust on the robot. Using a task-planning domain example, we showcase the efficacy of our inferred policies.**

I. Introduction

We consider a multi-agent scenario where a robot R makes and executes a plan and a human supervisor H is held accountable for the robot's behavior. In settings where R can deviate from the supervisor's expectation, a notion of trust becomes a key factor. While it is possible to develop trust in longitudinal setting [1, 15], in one-off interactions (where no trust exists) conventional wisdom often guides the supervisor to spend all their time in monitoring the robot's behavior to ensure it adheres to their expectations. In this work, we challenge the latter belief and by modeling the interaction in a game theoretic framework, show that H can consider resource-efficient monitoring strategies.

There are cases when a robot's expectation may deviate from its supervisor's expectations? First, a robot may have side-goals that do not align with a supervisor's expectation. For example, an autonomous car ride-sharing (or, in general, robot-as-a-)service may have certain expectations from its supervisor (eg. travel on shortest routes) but may need to adhere to passenger's expectation (eg. avoid hilly roads) that are in conflict with one another. Second, the robot may not be fully aware of the human's expectation of itself. In such scenarios, we formally model the inference problem related to the finding a monitoring strategy for the human supervisor.

Specifically, we present a notion of trust that a human supervisor H places on a worker robot R when H chooses to *not observe* R's plan (or its execution) by modeling the interaction in a game-theoretic framework of trust motivated by [10]. To capture the aforementioned scenarios, we assume the robot is unaware of the human's model of itself M_H^R, but has knowledge about all the possible models $\mathcal{M}_H^R$ (or constraints) the human may have; hence, $M_H^R \in \mathcal{M}_H^R$, M_H^R is not known to R, $\mathcal{M}_H^R$ is. We leverage the game theoretic framework to devise a probabilistic observation strategy for H that ensures (1) R does not deviate away from executing a plan that respects all constraints and in turn, (2) H's saves valuable resources such an monitoring time, effort, etc.

While we propose a novel type of assistance that can assist H on when to supervise R to ensure expected behavior, we explore if such assistance is required by performing human studies. We show that without assistance, humans are either too risk-averse (monitors R most of the time to ensure that R adheres to their expectation) or too risk-taking (minimizes their observation time even if the R deviates from expectation). In either case, assumptions made in earlier works [8, 2], where humans are expected to monitor the robot all the time fails to hold. Thus, it makes sense to analyse the supervision scenario and propose methods to suggest optimal monitoring strategies. Furthermore, answers to subjective questions show that participants prefer such automated assistance.

II. Related Work

Our supervision scenario is situated in a specturm of fully-cooperative settings to fully-adversarial ones. In fully-cooperative settings, researchers argue that the robot should only consider plans that adhere to the human's expectation; then these plans are said to be explicable [16], legible [2], adhering to social norms [7]. The assumption that robots sole-objective is to cater to a single human's expectation (the supervisor) may not be true in our case, and the supervisor's monitoring time may be costly. While some works suggest introducing impreciseness in specification on the human's expectation [3] as a solution, other consider robot producing explanations [14] to soothe the human; neither can guarantee behavior produced by R adheres to human's expectation. Other methods where the supervisor communicates implicit constraints [5], or their preferences [6] may not work in our scenario, as a two-way channel is necessary for the robot to identify conflicting constraints, communicate back to the supervisor and convince H the rational behind their behavior. In fully-adversarial settings, related work seek to find monitoring strategies to catch a perpetrator in physical and cyber defense scenarios [13, 11, 12] by framing the interaction in a game-theoretic manner. While our modeling shares similarities, existing works do not consider a

∗Equal contribution
†Work done while at Arizona State University.

Presented at the RSS Workshop on Robotics for People– Perspectives on Interaction, Learning, and Safety, 2021.

	$O_{P,\neg E}$	$O_{\neg P,E}$	NO-OB
π_{pr}	$\color{blue}{-C_P^H(\pi_{pr})-I_P^H(\pi_{pr})}$, $-C_P^R(\pi_{pr})-C_{\tilde{E}}^R(\pi_{pr})-C_{\tilde{G}}^R$	$\color{blue}{-C_E^H(\tilde{\pi}_{pr})-I_E^H(\hat{\pi}_{pr})}$, $-C_P^R(\pi_{pr})-C_E^R(\tilde{\pi}_{pr})-C_{\tilde{G}}^R$	$\color{blue}{-V_I^H(\pi_{pr})}$, $-C_P^R(\pi_{pr})-C_E^R(\pi_{pr})$
π_s	$\color{blue}{-C_P^H(\pi_s)\overbrace{-I_P^H(\pi_s)}^{0}}$, $-C_P^R(\pi_s)-C_E^R(\pi_s)$	$\color{blue}{-C_E^H(\pi_s)\overbrace{-I_E^H(\hat{\pi}_H)}^{0}}$, $-C_P^R(\pi_s)-C_E^R(\pi_s)$	$\color{blue}{\overbrace{-V_I^H(\pi_s)}^{0}}$, $-C_P^R(\pi_s)-C_E^R(\pi_s)$

TABLE I
NORMAL-FORM GAME MATRIX FOR MODELING THE ROBOT-MONITORING SCENARIO. R (H) IS THE ROW (COLUMN) PLAYER.

cooperative aspect between the players of the game. This makes it difficult to use their framework for longitudinal interaction where repeated interaction can build a sense of trust between H and R. While we do not explore this aspect explicitly, our framework keeps this at its core for future extension.

III. GAME THEORETIC FORMULATION

We formulate a two-player general-sum game between the human supervisor H and the robot R. In this section, we explain the components of this game shown in Table I.

A. Player Actions

R, the row-player, has two pure strategies– plans π_{pr} and π_s– plans that are probably risky (does not adhere to all constraints) and safe (one that does). H, the column player, has three strategies– (1) to only observe the plan made by the robot $O_{P,\neg E}$ and decide whether to let it execute (or not), (2) to only observe the execution $O_{\neg P,E}$ and stop R from executing at any point, and (3) not to monitor (or observe) the robot at all (NO-OB). We make two inherent assumptions in this formulation– (1) the robot cannot switch from a plan (or a policy) it commits to in the planning phase during execution phase and (2) the human only stops the robot from executing the plan if they believe that the robot's plan does not achieve the goal G while satisfying their constraints.

B. Utilities

The values on the top, highlighted in blue, indicates the H's utilities and the ones at the bottom represent R's utilities.

R's Utility Values: The utilities are defined in terms of:

$C_P^R(\pi)$	Cost of making a plan π.
$C_E^R(\pi)$	Cost to robot for executing plan π.
$C_{\tilde{G}}^R$	Penalty of not achieving the goal G.

We denote partial plans as $\hat{\pi}_{pr}$; partial plans arise when H aborts execution of a probably risky plan. Note that depending on where the human stops the robot, the cost for the partial plans can be different. $C_{\tilde{G}}^R$ that represents the cost of not achieving the goal. The robustness $r(\in (0,1])$ of a plan represents the fraction of models in $\mathcal{M}_H^R$ where the plan π_{pr} is executable and can be obtained via model counting approaches [9]. As a particular M_H^R is sampled from $\mathcal{M}_H^R$, we model $C_{\tilde{G}}^R$ as a random variable drawn from the Bernoulli distribution s.t. it represents a non-zero penalty if the plan is not robust in a sampled human model ($Pr = 1-r$) or zero otherwise.

The cost incurred by the R when H chooses to observe the plan π_{pr} (before execution) is $C_{\tilde{E}}^R(\pi_{pr}) = \begin{cases} C_E^R(\pi_{pr}) & \text{if } C_{\tilde{G}}^R = 0 \\ 0 & o.w. \end{cases}$. If H chooses to monitor the execution directly, R's utility is $C_E^i(\tilde{\pi}_{pr}) = \begin{cases} C_E^i(\pi_{pr}) & \text{if } C_{\tilde{G}}^i = 0 \quad i \in \{R,H\} \\ C_E^i(\hat{\pi}_{pr}) & o.w. \end{cases}$.For execution costs, it is natural to assume $C_E^R(\pi_{pr}) \leq C_E^R(\pi_s)$, i.e cost of executing the plan that satisfies all constraints is greater than executing a plan that satisfies a sub-set of constraints. Similarly, for planning, coming up with π_{pr} will be easy if the value of r is small while coming up with the plan π_s will take considerably longer. Hence, we also assume $C_P^R(\pi_{pr}) \leq C_P^R(\pi_s)$.

H's Utility Values: The utilities are defined in terms of:

$C_P^H(\pi)$	Cost of observing a plan π.
$C_E^H(\pi)$	Cost of observing the robot's execution of π.
$V_I^H(\pi)$	Cost incurred by the supervisor when R violates a constrain due to lapse in H's monitoring
$I_P^H(\pi)$	Inconvenience to H if R presents a plan π that H cannot allow R to execute (note $I_P^H(\pi_s) = 0$).
$I_E^H(\pi)$	Inconvenience to H if R is stopped from executing π(note $I_E^H(\pi_s) = 0$).

When R proposed π_{pr}, it is only executable in a sub-set of models in $\mathcal{M}_H^R$. As this sub-set may not contain the human's actual model M_H^R, we need to factor in this uncertainty into $V_I^H(\pi)$, $I_P^H(\pi)$ and $I_E^H(\pi)$. We leverage the robustness value r and the Bernoulli distribution for this purpose. We assume R violating a constraint due to lapse in H's monitoring has the highest penalty for (the supervisor) H; thus,

$$V_I^H(\pi_{pr}) > C_P^H(\pi_{pr}) + I_P^H(\pi_{pr}) \quad (1)$$
$$V_I^H(\pi_{pr}) > C_E^H(\tilde{\pi}_{pr}) + I_E^H(\hat{\pi}_{pr}) \quad (2)$$

We also assume that (1) $C_E^H(\pi) > C_P^H(\pi)$ (the cost of observing and the a plan is less than observing the execution of a plan) and (2) $I_E^H(\hat{\pi}_{pr}) > I_P^H(\pi_{pr})$ (same assumption for the inconvenience caused).

IV. GAME-THEORETIC NOTION OF TRUST

In our game, the amount of trust placed in R increases as the H selects $O_{P,\neg E} < O_{\neg P,E} <$NO-OB. When H selects

NO-OB, it exposes itself to a vulnerability– R executes π_{pr} resulting in the high negative reward, V_I^H, for H. On the other hand, if H chooses ($O_{P,\neg E}$), H has the least amount of risk– even before R can execute, the plan is verified by H. There exists a trade-off due to this notion of trust– monitoring depletes H's resources (time, concentration etc.), but if R cannot be fully trusted, H needs to monitor costs to ensure R adheres to constraints.

The No-Trust Scenario: In this setting, H should never play an action that exposes them to a risk of a high negative utility. If a pure-strategy Nash Equilibrium exists, the players should consider it as neither can deviate to get a better utility [10]. Given we consider a Bayesian game where the rewards represent random variable, the expected utility values need to satisfy the following inequalities for a pure-strategy Nash Equilibrium to exists,

$$(1-r)V_I^H(\pi_{pr}) < C_P^H(\pi_{pr}) + (1-r)I_P^H(\pi_{pr})$$
$$C_P^R(\pi_{pr}) + (1-r)C_{\tilde{G}}^R + rC_E^R(\pi_{pr}) < C_P^R(\pi_s) + C_E^R(\pi_s) \quad (3)$$

If $r = 1$, we can guarantee that $(\pi_{pr}, NO-OB)$ is the Nash equilibrium. But, to reduce costs, $r << 1$ (otherwise, $\pi_{pr} = \pi_s$), leading to the following proposition.

Proposition 1. *The game defined in Table I has no pure strategy Nash Equilibrium where π_{pr} is not executable in some of the models.*

Absence of Pure Strategy Nash Equilibrium: The absence of a pure-strategy Nash eq. makes it difficult to define a human's best course of action in this no-trust setting [10]. Thus, we devise the notion of a trust boundary.

Consider a human chooses the mixed strategy $\vec{q} = [(1-q_E-q_N), q_E, q_N)]^T$ over the actions $O_{P,\neg E}, O_{\neg P,E}$ and NO-OB respectively. In order to ensure that the robot cannot deviate away from making and executing π_s, we have to ensure that the expected utility (U) for the robot given $\vec{q}$ is greater for π_s than for π_{pr}.

$$\begin{aligned} \mathbb{E}_{\vec{q}}[U(\pi_s)] &> \mathbb{E}_{\vec{q}}[U(\pi_{pr})] \Rightarrow \quad (4)\\ r\ -C_P^R(\pi_s) - C_E^R(\pi_s) &> (-C_P^R(\pi_{pr}) - C_{\tilde{G}}^R - C_{\tilde{E}}^R(\pi_{pr})) \\ &\quad \times(1-q_E-q_N) \\ &\quad +(-C_P^R(\pi_{pr}) - C_E^R(\tilde{\pi}_{pr}) - C_{\tilde{G}}^R) \times q_E \\ &\quad +(-C_P^R(\pi_{pr}) - C_E^R(\pi_{pr})) \times q_N \end{aligned}$$

where $\mathbb{E}_{\vec{q}}[U(\pi)]$ denotes the expected utilities. This inequality is linear w.r.t. the variables q_N and q_E. Thus, in the region on one side of the linear boundary, the robot always executes π_s. Thus, we call this linear boundary the ***trust boundary***.

V. Experimental Setup and Evaluation

The aim of this section is to first describe a task-planning scenario in which we can compute the trust boundary and then, perform human subject studies in a simplified version of this supervision scenario. To do so, we initially describe the robot-delivery domain that we will use throughout the section.

A. Robot Delivery Domain

We used a robot delivery domain [8] in which the robot can collect and deliver parcels (that may not be waterproof) or coffee by picking it from the reception desk and taking it to a particular location. The robot in the *PDDL* domain has the following actions: {*pickup, putdown, stack, unstack, move*}.

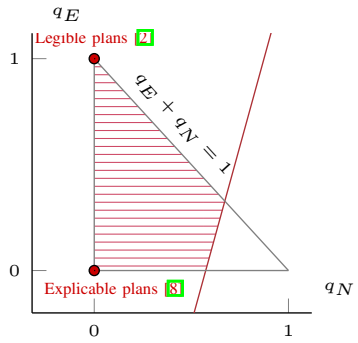


Fig. 1. An observation strategy in the trust region (shaded) ensures that the robot sticks to π_s. In contrast to observation strategies discussed in existing works, one can reduce monitoring costs while ensuring explicable/legible/safe behavior.

Problem Instance: The problem instance in our setting has the initial setting where (1) the robot is standing at a position equidistant to the reception and the kitchen, (2) there is a parcel located at the reception that is intended for the employee, (3) there is brewed coffee in the kitchen that needs to be delivered in a tray to the employee. The goal for the robot is to collect and deliver the coffee and the parcel to the employee.

Robot Plans: There are two plans in which the robot achieves the goal of collecting coffee from the kitchen and parcel from the reception desk and delivers them to an employees' desk. (a) π_s, the robot (1) collects coffee, (2) delivers it to the employee, (3) goes back along the long corridor to collect the parcel from the reception desk and finally (4) delivers it back to the same employee. (b) π_{pr}, the robot collects coffee from the kitchen, (2) collects parcel from the reception desk and puts them on the same tray and finally, (3) delivers both of them to the employee.[1]

B. Computing the Trust Boundary in a Task-Planning Scenario

In order to compute the trust boundary, we calculate the utility values for our game leveraging Table I and the cost incurred by R and H in this robot delivery domain.

In this example, if the robot makes π_{pr}, it will be executable (or safe) as per one of the two observers whose models make up the set $\mathcal{M}_H^R$, because the coffee and parcel taken in the same tray runs the risk of the coffee spilling, thereby ruining the package. Thus, the robustness for π_{pr} is $r = \frac{1}{2} = 0.5$. On the other hand, the plan π_s is executable (and thus, overall safe) in both the models in $\mathcal{M}_H^R$.

We used the Fast Downward planner [4] on the robot delivery domain [8] to find the costs for R and H and to compute them we used a machine with an Intel Xeon CPU (clock speed 3.4 Ghz) and 128GB RAMv. Given the calculated costs, we can define the utility matrix for the players (R, H). [2]

[1] Given the (actual and the human's) domain models and the problem instance, these plans can simply be computed using available open-source software like `Fast-Downward` or web-services like `planning.domains`.

[2] The details of each cost and the hasn't been given due to page limit

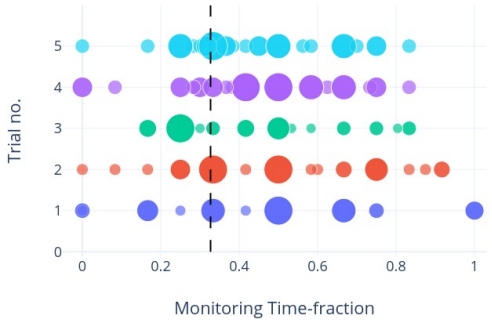


Fig. 2. Participant's monitoring strategies across multiple trials. Trust boundary indicated using the black vertical line.

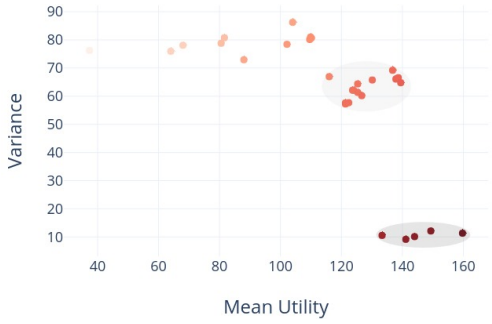


Fig. 3. Average utility and its variance for each of the participants across the five trials.

According to Proposition 1, this game does not have a pure Nash Eq. strategy with probability 0.5. Therefore, we now find the boundary in the space of mixed strategies for second type of H who can choose to adopt which will ensure that the robot always executes π_s. To do so, we use the calculated values plug them into equation 4 and obtain,

$$10 \times q_N - 3 \times q_E - 5.74 \quad < \quad 0 \tag{5}$$

In Figure 1, we plot the trust boundary represented by the lines in Eqn. 5. The three black lines (sides of the larger triangle) represent the feasible region for the human's mixed strategy $\vec{q}$. Monitoring strategy in the shaded region guarantees the robot, being a rational agent, executes π_s. The strategy that optimizes H's monitoring cost and yet ensures the robot adheres to π_s lies on the trust boundary indicated using the red line.

C. Human Studies

The human-subjects study was designed to evaluate whether (1) the human can find a good strategy to cut-down the monitoring time while they ensure the constraints structured manner from the robot and (2) the humans tend to deviate to more split-time strategies where some of the time, originally meant for monitoring, can be used for other tasks. We designed a user-interface to represent the robot-delivery scenario. The participants in the study play the role of a student in a robotics department who are asked to monitor the robot for an hour. In order to make the monitoring action be associated with a cost, we added a second task in which participants could choose to grade exam papers (and get paid for it) instead of just monitoring the robot and this represents the action to not-monitor the robot. For simplicity, we combine the actions to monitor the plan and monitor the execution as a single 'monitor the robot' action. We ask them to give us a time slice for which they would choose a particular action (eg. 30 minutes to monitor the robot and 30 minutes to grade exam papers). We let each participant do five trials and after each trial, the overall utility based on the participant's monitoring strategy and the robot's strategy is reported to them. The robot does not adapt itself to the human's strategy in the previous trial (which intents to preserve the non-repeated nature of our game). We collected data from 32 participants who were all graduate students across various engineering departments at our university.

Aggregate Results – Changes in Monitoring Strategy across Trials: Note that a participant, given the information on the interface, can formulate a simplified version of the game-theoretic model proposed in this paper and find the optimal strategy for monitoring (which is to monitor the robot for 0.327 or 19.62 minutes of an hour and use the remaining time to grade papers). The participants' time slice allocated for monitoring, across the five trials, are shown in Fig. 2. Given that there are only two actions for the participant, the strategy can be represented using a single variable (fraction to monitor the robot) and thus, is plotted along the x-axis. The size of each bubble is proportional to the number of participants who selected a particular strategy. The optimal strategy is shown using a black vertical line (i.e. $x = 0.327$). In the first trial, most users ($n = 18$) choose a risk-averse strategy, i.e. monitored the robot to ensure it performs a safe plan even if it meant losing out on money that could be earned from grading. As the trials progressed, participants started discarding extreme strategies (i.e. only monitor or only grade papers) and started considering strategies closer to the optimal. In Fig 2, note that for the first two trials, the strategies are well spread out in the range $[0, 1]$ where as in the last two trials, the strategies are clustered around the optimal decision boundary, with very few data points below 0.25 and very few above 0.7. This shows humans hardly can find an optimal monitoring strategy when there is no prior interaction with the robot and finding an near optimal monitoring strategy after many trial and error can cause a lot of loss. So, a strategy suggestion is needed to provide an assistant to the human to deal with unsafe robots.

Participant Types: In Figure 3, we plot the average utility of each participant across five trials on the x-axis. The y-axis represents the variance. Highlighted in dark, at the bottom right, are five participants that chose observation probabilities in the trust region but not exactly at the trust boundary, i.e. sub-optimal w.r.t. the optimal trust boundary strategy (at 0.327) that yields a reward of 173.77. After that, they did behave in a greedy fashion to reduce the observation time in the hope to make more money by grading papers and stuck to the good policies they initially discovered. Towards the top-right corner, the set of points circled in light gray, we saw a dense cluster of participants ($= 15$) who obtained a high average utility but tried to tweak their strategies significantly, sometimes observing less and therefore, allowing the robot to choose the riskier plan. which

eventually lead to a large loss in reward. This implies that the human often takes risk and deviates to more split-time strategies since the time meant to monitoring can be used for other tasks.

VI. Conclusions and Future Work

We model the notion of trust that a human supervisor places on a worker robot by modeling this interaction as a Bayesian Game. We show that existing notions of game-theoretic trust break down in our setting when the worker robot cannot be trusted due to the absence of pure strategy Nash Equilibrium. Thus, we introduce a notion of trust boundary that optimizes the supervisor's monitoring cost while ensuring that the robot workers stick to safe plans.

Acknowledgments

This research is supported in part by ONR grants N00014-16-1-2892, N00014-18-1- 2442, N00014-18-1-2840, N00014-9-1-2119, AFOSR grant FA9550-18-1-0067, DARPA SAIL-ON grant W911NF-19- 2-0006, NASA grant NNX17AD06G, and a JP Morgan AI Faculty Research grant.

References

[1] Min Chen, Stefanos Nikolaidis, Harold Soh, David Hsu, and Siddhartha Srinivasa. Planning with trust for human-robot collaboration. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 307–315, 2018.

[2] Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. Legibility and predictability of robot motion. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 301–308. IEEE Press, 2013.

[3] Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. The off-switch game. 2017.

[4] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[5] Emmanuel Johnson and Jonathan Gratch. The impact of implicit information exchange in human-agent negotiations. In *Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents*, pages 1–8, 2020.

[6] Joseph Kim, Christopher Banks, and Julie Shah. Collaborative planning with encoding of users' high-level strategies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[9] Tuan Nguyen, Sarath Sreedharan, and Subbarao Kambhampati. Robust planning with incomplete domain models. *Artificial Intelligence*, 245:134–161, 2017.

[7] Uwe Köckemann, Federico Pecora, and Lars Karlsson. Grandpa hates robots-interaction constraints for planning in inhabited environments. In *AAAI*, pages 2293–2299, 2014.

[8] Anagha Kulkarni, Tathagata Chakraborti, Yantian Zha, Satya Gautam Vadlamudi, Yu Zhang, and Subbarao Kambhampati. Explicable robot planning as minimizing distance from expected behavior. *CoRR, abs/1611.05497*, 2016.

[10] Vidyaraman Sankaranarayanan, Madhusudhanan Chandrasekaran, and Shambhu Upadhyaya. Towards modeling trust based decisions: a game theoretic approach. In *European Symposium on Research in Computer Security*, pages 485–500. Springer, 2007.

[11] Sailik Sengupta, Satya Gautam Vadlamudi, Subbarao Kambhampati, Adam Doupé, Ziming Zhao, Marthony Taguinod, and Gail-Joon Ahn. A game theoretic approach to strategy generation for moving target defense in web applications. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 178–186. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[12] Sailik Sengupta, Ankur Chowdhary, Abdulhakim Sabur, Adel Alshamrani, Dijiang Huang, and Subbarao Kambhampati. A survey of moving target defenses for network security. *IEEE Communications Surveys & Tutorials*, 22 (3):1909–1941, 2020.

[13] Arunesh Sinha, Thanh H Nguyen, Debarun Kar, Matthew Brown, Milind Tambe, and Albert Xin Jiang. From physical security to cybersecurity. *Journal of Cybersecurity*, 1(1):19–35, 2015.

[14] Sarath Sreedharan, Subbarao Kambhampati, et al. Explanations as model reconciliation—a multi-agent perspective. In *2017 AAAI Fall Symposium Series*, 2017.

[15] Anqi Xu and Gregory Dudek. Optimo: Online probabilistic trust inference model for asymmetric human-robot collaborations. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 221–228. IEEE, 2015.

[16] Yu Zhang, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. Plan explicability and predictability for robot task planning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1313–1320. IEEE, 2017.

Leveraging Semantic Scene Graphs and Explainable AI to Explain Robot Failures

Devleena Das
School of Interactive Computing
Georgia Institute of Technology
Atlanta, Georgia Email:ddas41@gatech.edu

Sonia Chernova
School of Interactive Computing
Georgia Institute of Technology
Atlanta, Georgia Email:chernova@gatech.edu

***Abstract*—When interacting in unstructured human environments, occasional robot failures are inevitable. When such failures occur, everyday people, rather than trained technicians, will be the first to respond. The field of explainable AI has sought to make complex-decision making systems more interpretable but most existing techniques target domain experts. On the contrary, in many failure cases, robots will require recovery assistance from *non-expert* users. Existing natural language explanations hand-annotate contextual information from an environment to help everyday people understand robot failures. However, this methodology lacks generalizability and scalability. In our work, we introduce a more generalizable explanation framework that autonomously captures the semantic information in a scene to produce semantically descriptive explanations for everyday users. To generate failure-focused, semantic explanations we leverage both semantic scene graphs to extract spatial relations and object attributes from an environment, as well as pairwise ranking. Our results show that semantically grounded explanations significantly improve everyday users' ability to identify and understand failures than the existing state-of-the-art context-based explanations.**

I. Introduction

Increasingly, robots are becoming deployed in everyday environments – homes, hospitals, and offices – in which the robot's primary users are everyday people rather than trained technicians [12]. Occasional robot failures are inevitable when operating in unstructured human environments, as the robot may be unable to find an object it requires, be unable to reach an object, encounter a planning error, etc. When an error occurs, everyday people in the robot's environment are typically the first to respond, but to effectively assist in failure recovery users must have an understanding of the robot's behavior, decision making, and what went wrong [3].

Research on Explainable AI (XAI) focuses on the development of techniques that increase the transparency and interpretability of complex, black box systems [1]. The vast majority of XAI techniques developed to date have been designed for experts and system developers [1, 8, 9, 6], however XAI systems also have the potential to explain the cause of a system error to everyday users. In particular, recent work has shown that natural language explanations are effective in improving user confidence in an AI system [4], and in improving user assistance in fault recovery [3]. In both of the above works, a contributing factor to the effectiveness of their explanations is the ability to incorporate *situational*, or *environmental* context from the agent's environment. However, these early works lack generalizability and scalability as both techniques require that contextual information from the environment be hand-annotated a priori for explanation generation, preventing the generalization of explanations to new scenarios.

In this work, we introduce a generalizable framework for explaining robot pick errors to non-expert users shown in Figure 1. Specifically, we focus on explaining pick errors that occur amidst a robot's task plan, causing a halt in the robot's task execution. The key innovation of our approach is the use of scene graphs to produce *semantically descriptive explanations* that communicate why a failure to pick up a given object in the scene occurred. First, we adapt a state-of-the art semantic scene graph (SSG), MOTIFNET [13], to autonomously extract both inter-object spatial relations and object attribute information as contextual reasoning for robot failures in any scene. Second, we improve the semantically descriptive explanations producible through scene graphs by utilizing pairwise ranking. We show that pairwise ranking can be utilized to autonomously place attention on parts of a scene graph output that are truly relevant to a given failure. As a result, our framework can produce failure focused, semantically descriptive explanations which improve everyday users' understanding and identification of robot failures.

II. Scene Graph Model

A scene graph, G, describes the semantic information contained in a given image and is represented by a set of nodes, N, and edges, E, [13]. Each $n_i \in N$ is defined by a pair (b_i, o_i) in which b_i represents a detected bounding box, and o_i represents the associated object label. Similar to [2], we also provide each n_i with an object attribute, $a_i \in A$, where A is the set of object attributes. Additionally, each $e_{ij} \in E$ is defined as a predicate label between n_i and n_j. The predicate labels refer to the inter-object relations in a scene (e.g., underneath, inside, close to). Given these definitions, the output of a scene graph is defined by a set of triples $R = \{r_1, r_2 ... r_m\}$ in which each $r_k \in R$ is defined by $< n_i, e_{ij}, n_j >$.

We adapt the state-of-the-art scene graph model MOTIFNET [13][1] to predict spatial relationships and object attributes in a given scene. Figure 2 depicts our model archi-

[1]We utilize the codebase provided by [11] to adapt our MOTIFNET.

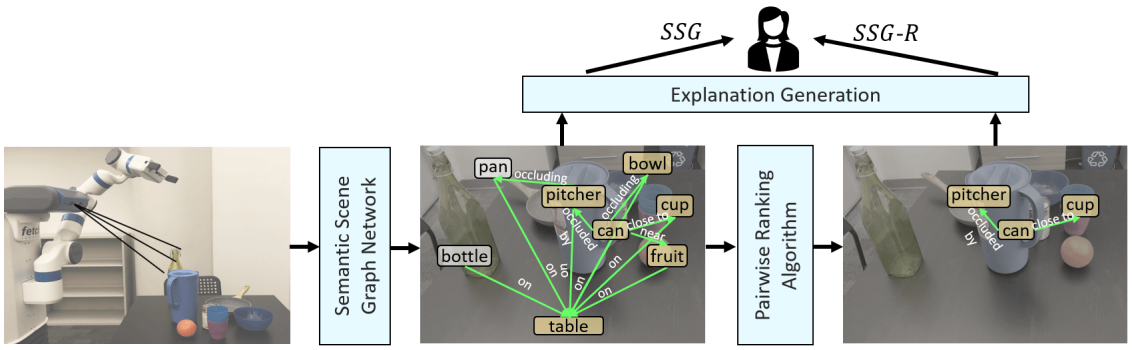


Fig. 1: Semantic explanation framework used to generate SSG explanations and failure-focused, ranked SSG-R explanations. The framework consists of three modules: (1) a scene graph network that autonomously extracts semantic information from a scene, (2) pairwise algorithm that ranks semantic information based on relevancy to a failure scenario, and (3) an explanation generation template that produces both variants of natural language explanations.

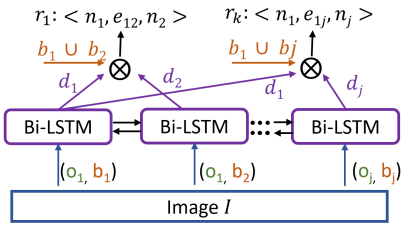


Fig. 2: Adapted MOTIFNET model architecture utilized to evaluate predicate classification.

Fig. 3: Confusion Matrix of our SSG model's performance where the y-axis denote ground truth predicates and the x-axis denote predicted predicates.

tecture. For the purposes of our application, we evaluate our model on predicate classification, a form of SSG evaluation that utilizes both ground truth bounding box regions and object labels to predict predicate edge labels. As shown in Figure 2, ground truth bounding box regions, $\{b_1, b_2..b_j\}$, and object labels, $\{o_1, o_2..o_j\}$, are extracted from an image I and passed into a bi-LSTM network structure with highway connections [10]. To predict a triple r_k, the contextualized states for two objects, d_i and d_j, are utilized in conjunction with the union of corresponding bounding box information, b_i and b_j, to determine the final predicate label e_{ij}.

A. Data Collection

To train our adapted MOTIFNET model, we collected a dataset D consisting of 188 household images from the AI2Thor simulator [7]. Images in D were taken from the perspective of the robot as it fails to pick up a desired object d_{obj} and capture the unstructured, cluttered environment of human households. Our images capture scenes of major receptacles such as kitchen counter tops or dining tables. On average, each image in D includes 13 object with approximately 6 object attributes and 30 inter-object spatial relations.

B. Training & Evaluation

We train our adapted MOTIFNET on ground truth bounding box regions and object labels to predict predicate and attribute labels. We utilize a 66%-17%-17% split in which we use 126 scenes for training, 32 for validation and 32 for evaluation. Our model is trained with 2000 iterations and utilizes a Cross Entropy loss that is optimized using stochastic gradient descent with a learning rate of 0.01 and momentum of 0.9.

The confusion matrix in Figure 3 shows the average performance of our predicate classification. Our adapted MOTIFNET can generalize the predicate labels with 84.9% accuracy. While our model has low false positive labels for most relations, we see that our model has a greater challenge differentiating labels that are semantically similar. For example, "close to" is most erroneously classified as "near". Improvements on the SSG model architecture, as well as additional training data, will likely lead to improvements in the classification accuracies.

III. Generating Explanations from SSGs

Given a failure type f, desired object d_{obj}, an image of the local environment I, and the corresponding scene graph G, our goal is to produce semantically descriptive, natural language explanations that reason about why a robot cannot pick up d_{obj}. Below, we detail how explanation variants SSG and

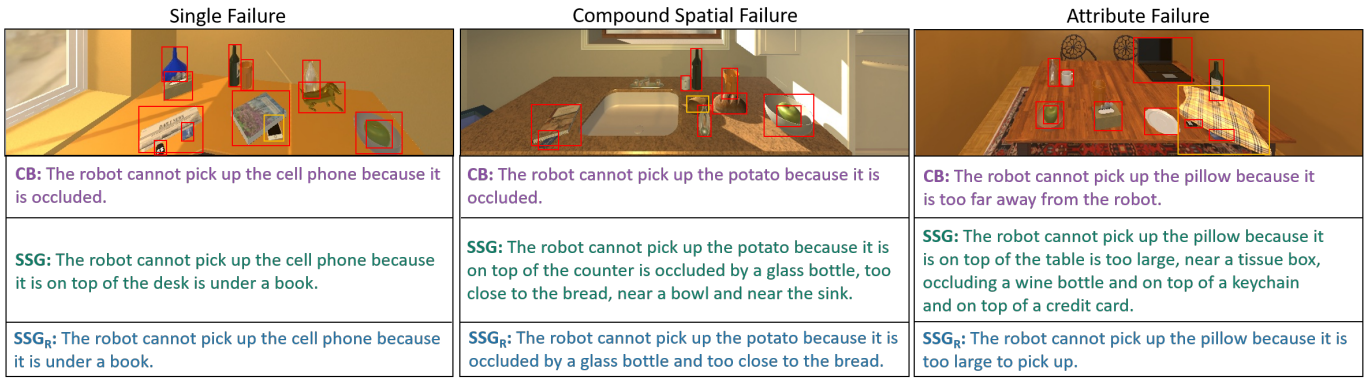


Fig. 4: Sample failure scenarios, where the red boxes indicate the bounding boxes of ground truth objects in the scene, and the yellow box represents d_{obj}. We illustrate model-generated explanations, comparing CB, SSG and SSG-R explanations.

SSG-R are generated in the context of three failure types: single, compound, and attribute failures. Single failures represent failures caused by a single inter-object spatial relationship, compound failures represent failures caused by multiple inter-object spatial relationships, and attribute failures represent failures related to the attribute of d_{obj}. As reference, inter-object spatial relations studied include $\{$*close to, far away, underneath, inside, occluded*$\}$, whereas the list of attribute failures include $\{$*heavy, large, hazardous, slippery, fragile*$\}$.

A. SSG Explanations

To develop SSG explanations, we follow a template-based approach that traverses a scene graph, G, and extracts a subgraph g containing all relations $r_k \in R$ which contain d_{obj} as a node in the triple. We describe g as the elements of the scene that pertain to the object of interest. Specifically, we format the explanation as *The robot could not pick up the* $< d_{obj} >$ *because* $< reasoning >$, where $< reasoning >$ is a list of phrases that enumerates all object relations $r_k \in g$.

In Figure 4 we showcase examples of SSG explanations in the context of our failure types. In every scene, the SSG explanations include *all* relationships associated with d_{obj}. We observe that SSG explanations are semantically richer, and more detailed than their CB counterpart. However, we also observe that SSG explanations include extraneous information that hide the true cause of a failure. A drawback of these SSG explanations is that there is no insight into *which* $r_k \in g$ are relevant in describing the robot's failure.

B. SSG-R Explanations via Pairwise Ranking

To provide only relevant relations as $reasoning$ for a failure, we develop SSG-R explanations. In addition to extracting a subgraph g, we utilize pairwise ranking to autonomously determine the relevancy of each triple $r_k \in g$. Pairwise ranking is used to learn preferences between pairs of entities when multiple available entities exist [5]. In our application, a preference denotes how accurately a relationship describes the true cause(s) of failure(s).

To formulate our pairwise ranking problem we let a pair of relationship triples, r_k and r_m, be defined by feature vectors $f_k = [e_{ij_k}, a_{ik}, a_{jk}]$ and $f_m = [e_{ij_m}, a_{im}, a_{jm}]$. Recall from Section II that eij represents the predicate label between two object nodes n_i and n_j, while a_i and a_j represent the predicted object attributes for n_i and n_j.

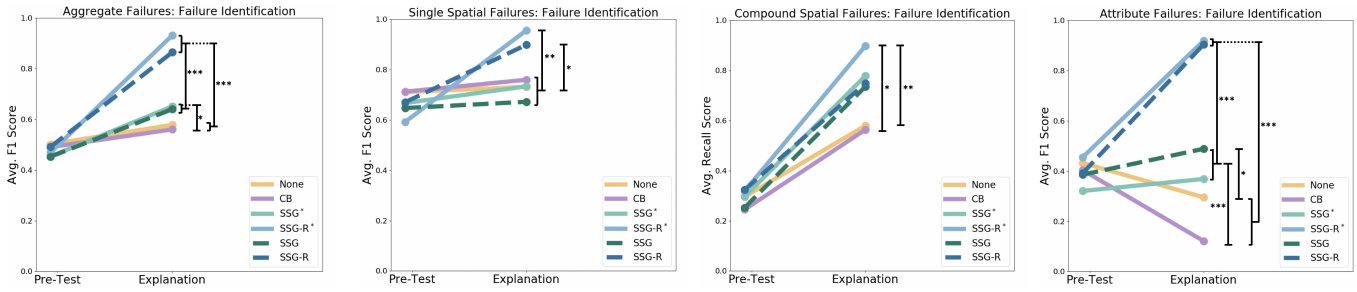
Given our feature vectors, we adapt the pairwise algorithm from Furnkranz et al. [5]. Given a comparison of two relationships, r_k and r_m, we establish three possible preference labels $\{0, 1, 2\}$, where 0 denotes when r_k better describes a failure than r_m, 1 denotes when r_m better describes a failure than r_k, and 2 denotes when r_k and r_m equally describe the cause of a failure. Associated feature vectors f_k and f_m are classified through three binary classifiers (one for each label pair), where the predicted label is one of the preference labels. In our work, we utilize random forest classifiers, trained using cross validation. Based on the predicted label, the rank of one or both relations is incremented. At the end, we order the ranks of all relationships r in a subgraph g.

To develop SSG-R explanations, we utilize the template for SSG explanations but, replace $reasoning$ with the top ranked relationship(s). Figure 4 shows how pairwise ranking eliminates extraneous relationships compared to SSG explanations, while being semantically richer than CB explanations.

IV. Analysis of Semantic Explanations

We evaluate the efficacy of our model-generated SSG and SSG-R explanations in improving users' ability to identify a failure. Furthermore, we observe the effectiveness of our ranked SSG-R explanations in comparison to our unranked SSG explanations. To do so, we conduct a six-way between subjects study, with the following study conditions that differ by the type of explanation participants received:

- *None (Baseline)*: Participant receives no explanation describing the cause of error. As noted by [3], this is the standard in currently deployed robotic systems.
- *CB (Baseline)*: Participant receives a context-based explanation from prior work [3].
- *SSG**: Participant receives a ground truth, unranked, semantically descriptive explanation.
- *SSG-R**: Participant receives a ground truth, ranked, semantically descriptive explanation.



(a) (b) (c) (d)

Fig. 5: Average F1 and Recall score for participants' failure identification across all study conditions. Statistical significance is reported as: * p $<$ 0.05, ** p $<$ 0.01, *** p$<$ 0.001.

- *SSG*: Participant receives a model-generated, unranked semantically descriptive explanation.
- *SSG-R*: Participant receives a model-generated, ranked, semantically descriptive explanation.

A. User Study

Study Design. Similar to Das et al. [3], our user study consisted of two stages, *Pre-Test* and *Explanation*. In both stages, users were presented with images of the environment in which the robot encountered a failure when tasked to pick up d_{obj}. In the *Pre-Test* stage, participants were shown 16 randomly ordered failure scenarios and asked to identify the possible cause(s) of failure. None of the participants were provided explanations in this stage in order to establish their initial level of error understanding. In the *Explanation* stage, participants were shown another 16 different, failure scenarios. However, participants were now provided an explanation depending on their study condition. Similar to the *Pre-Test* stage, participants were tasked to identify the cause(s) of robot failure.

To evaluate participant performance, we measure the difference between each participant's *Pre-Test* and *Explanation* F1 score or Recall score using a metric from Das et al. [3], *Failure Identification (FId)*, which measures a participant's ability to accurately select the correct cause(s) of failure in a scene. Since participants were allowed to select multiple answers for each question, in the case of compound spatial failure types, the quantity of false negatives is a better measure of a participant's performance. Therefore, Recall score is utilized for compound failures and F1 score for all other failure types.

Participants. We recruited 93 participants from Amazon Mechanical Turk. Participants were required to be non-experts in the domain of robotics, thus we removed three participants for scoring a 100% accuracy on the *Pre-Test* stage. The remaining 90 participants, 15 for each study condition, included 53 males and 37 females, all whom were over the age of 18 (M=39.0, SD=10.8). The task took on average 20-30 minutes and participants were compensated $2.50.

B. Quantitative Results

The participants' failure identification (*FId*) scores follow a normal distribution, thus we utilize a one-way ANOVA with a Tukey HSD post-hoc test to evaluate statistical significance between study conditions. In Figure 5, we examine the average F1 score and Recall score for participants' failure identification (*FId*) across the aggregated failure types as well as across each individual failure type. Overall, we see that $SSG\text{-}R^*$ and SSG^* explanations have the highest improvement in *FId* scores in comparison to the other study conditions. This indicates the effectiveness of semantically descriptive explanations in improving participants' understanding of robot failures.

When looking at the *FId* scores for aggregate failures (Figure 5(a)), we see $SSG\text{-}R^*$ and $SSG\text{-}R$ explanations lead to a significant improvement in failure understanding compared to None, CB, and both SSG and SSG^* explanations ($p < 0.001$ for all). Similar trends are observed with single spatial failures (Figure 5(b)), and attribute failures (Figure 5(d)), reiterating the effectiveness of grounding explanations in the semantic information present in a scene. Additionally, for compound spatial failures (Figure 5(c)), we observe that only $SSG\text{-}R^*$ leads to significant improvement in *FId* scores. This highlights the importance of ranked semantic-based explanations in significantly improving participants' failure understanding, as well as highlights an area of improvement for our SSG model in detecting multiple failures in a scene. Furthermore, Figure 5(d) presents the effectiveness of SSG and SSG^* explanations. We observe that SSG and SSG^* lead to significantly improved failure understanding in comparison to None ($p < 0.05$) and CB ($p < 0.001$). However, $SSG\text{-}R$ and $SSG - R^*$ show a significantly higher rate of improvement than SSG and SSG^* ($p < 0.001$).

V. Conclusion & Future Work

In this work we have introduced a generalizable framework that autonomously captures the semantic information in a scene to explain robot pick errors to everyday users. From our results, we demonstrate that ranked, semantically descriptive explanations quantifiably improve everyday users' ability to understand robot failures. In future work, we would like to examine the effects of these semantic explanations in the context of other manipulation and navigation failures. Additionally, future work, in the form of additional data collection and model improvements, is required to further expand the scope of relations needed to explain a wider range of robot failures.

REFERENCES

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[2] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5664–5673, 2019.

[3] Devleena Das, Siddhartha Banerjee, and Sonia Chernova. Explainable ai for robot failures: Generating explanations that improve user assistance in fault recovery. *arXiv preprint arXiv:2101.01625*, 2021.

[4] Upol Ehsan, Pradyumna Tambwekar, Larry Chan, Brent Harrison, and Mark O Riedl. Automated rationale generation: a technique for explainable ai and its effects on human perceptions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 263–274, 2019.

[5] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning and ranking by pairwise comparison. In *Preference learning*, pages 65–82. Springer, 2010.

[6] Krishna Gade, Sahin Cem Geyik, Krishnaram Kenthapadi, Varun Mithal, and Ankur Taly. Explainable ai in industry. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3203–3204, 2019.

[7] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[8] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[9] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700. Springer Nature, 2019.

[10] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *arXiv preprint arXiv:1507.06228*, 2015.

[11] Kaihua Tang. A scene graph generation codebase in pytorch, 2020. https://github.com/KaihuaTang/Scene-Graph-Benchmark.pytorch.

[12] Georgios A Zachiotis, George Andrikopoulos, Randy Gornez, Keisuke Nakamura, and George Nikolakopoulos. A survey on the application trends of home service robotics. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1999–2006. IEEE, 2018.

[13] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5831–5840, 2018.

Human Action Nodes for Behavior Trees

Mohamed Behery
Knowledge-Based Systems Group (KBSG)
RWTH Aachen University, Germany
Email: behery@kbsg.rwth-aachen.de

Minh Trinh
Laboratory for Machine Tools
and Production Engineering (WZL)
RWTH Aachen University, Germany
Email: m.trinh@wzl.rwth-aachen.de

Gerhard Lakemeyer
Knowledge-Based Systems Group (KBSG)
RWTH Aachen University, Germany
Email: gerhard@kbsg.rwth-aachen.de

***Abstract*—Modern industrial scenarios often require robots to share the work space with a human worker. Having a human in the work space increases the complexity of the robot decision making. Especially in cases where the human and the robot have to handle the same work piece. In such cases, the decision process of the robot must include not only safety of the human teammate, but also factors that maintain the throughput of the whole system. Behavior Trees (BTs) have shown great success as a task level control framework in industrial situations due to their reactivity, modularity, and reusability. However, not much research has gone into their application in industrial Human-Robot Collaboration (HRC) contexts. In this work, we introduce an extension to BTs that exploits the modularity and reactivity to allow BT-based robots to collaborate with human workers without compromising safety or performance.**

I. Introduction

Behavior Trees (BTs) have shown great success in the field of game AI design due to their readability, modularity, and reactivity. This led to their transition into the field of robotics where they excelled in many applications [4]. BTs were used in executing autonomous tasks by robots in both full autonomy [5] and supervised control [2, 3] as well as multi-agent systems [1]. However, there is still potential to exploit the modularity and flexibility offered by BTs in Human-Robot Collaboration (HRC) tasks. Supervisory control use-cases that employ BTs such as the robot assisted surgery task discussed in [2] require the human operator to either have access to traditional interfaces with the robot (such as a keyboard and/or touch screen) or to have an augmented reality interface that facilitates the human-robot communication.

While some tasks can allow the use of such input -and output- devices, the use of such communication schemes is sometimes not feasible. For example, when the human team member is occupied by achieving a task of her own, as in the case of human-robot assembly tasks. Since humans and robots have different capabilities (e.g., robotic precision and reliability and human manipulation capabilities) and both are working to achieve a common goal, it is crucial that each teammate knows what the other is doing and when the handover of the work piece should occur.

Current trends show a development towards customized products with many variants as well as a reduction of the time to market. Therefore, an increased flexibility and agility of the production process is necessary. The assembly process, as the last production step, is especially affected by fluctuations in demand due to its market proximity. Humans show great flexibility compared to robots, particularly when it comes to handling non-rigid objects. Robots on the other hand, are more precise and efficient. Robot-human collaboration combines these advantages. Yet, programming of robots is mostly done by robotics experts. Small and medium-sized business often lack this expertise.

In this *ongoing* work we extend BTs to handle HRC tasks, where human and robotic teammates collaborate to achieve a common goal while sharing not only the work space, but also the work piece. To this end, we make use of the modularity and flexibility of BTs to allow the robot to start executing new tasks -on different work pieces- instead of idly waiting for the human to finish theirs.

This paper is organized as follows: Section II describes BTs and the different node types. Next, Section III describes the details of our extension. We conclude the paper in Section IV.

II. Behavior Trees

BTs have been considered a generalization of previous control architectures such as Teleo-Reactive systems and Decision Trees, and have shown superiority to Finite state Machines [4]. As described by Iovino et al. [4], BTs follow a tree structure where nodes can belong to two classes: execution and control flow. Execution starts by activating (*ticking*) the root node, which in turn starts ticking its children. After receiving a tick, a node will become active and return its status $S \in \{\mathcal{S}, \mathcal{R}, \mathcal{F}\}$; either Success, Running, or Failure respectively to its parent. Execution nodes can be *Action* nodes, which execute an action and return $\mathcal{S}$ or $\mathcal{F}$ depending on the action's success, or $\mathcal{R}$ while it executes the action. They can also be *condition* nodes which return $\mathcal{S}$ or $\mathcal{F}$ depending on whether the condition holds and never return $\mathcal{R}$. Control flow nodes are divided into four node types:

- **Sequence:** execute the children in order from left to right. They return $\mathcal{S}$ iff *all* the children succeed, $\mathcal{F}$ iff at least one fails, and $\mathcal{R}$ otherwise.
- **Selector:** They return $\mathcal{S}$ iff *one* child succeeds, $\mathcal{F}$ iff all the children fail, and $\mathcal{R}$ otherwise.
- **Parallel:** execute the children in parallel. A node with N children and m threshold returns $\mathcal{S}$ iff m children succeed and $\mathcal{F}$ iff $N - m + 1$ children fail, and $\mathcal{R}$ otherwise.
- **Decorator:** they can be used to implement custom behavior (e.g., repeat n times) and manipulate the returned

status of their child (e.g., negation).

III. Human Action Nodes

When the robot hands over the work piece to the human, it should be able to start a new task instead of idling. Task assignment is rather straightforward, the problem is resuming the work after the human has finished her sub-task. By augmenting the BT with a reasoning engine, the robot can know how and when to resume its tasks. This work defines a new execution node for BTs called Human Action Node ($\mathcal{H}$). Using this node, we can 1) mark a node as a task for the human, and 2) allow the robot to continue working on that work piece once the human has finished. The $\mathcal{H}$-nodes will return $\mathcal{S}$ after the human successfully ends her tasks and $\mathcal{F}$ if the human failed. They do not need to return $\mathcal{R}$ during execution because they stop the tree once they are reached.

We assume -without loss of generality- that these nodes will be placed under memory-sequence or memory-selector nodes, because assembly tasks do not require repeating the same steps for the same part[1]. Algorithm 1 shows how a sequence node will be changed to handle a child $\mathcal{H}$-Node, Selector and Parallel nodes can be modified analogously.

Algorithm 1: The tick procedure of a Sequence node

```
1  Function tick:
2      for c in children do
3          if type(c) = H then
4              CLIPS.addRule(
5                      idle ∧ (c.effects ∨
                        ¬c.preconditions) → T.tick()
6                  )
7              save(T)
8              abort(T)
9          end
10         childStatus ← c.tick()
11         if childStatus ≠ S then
12             return childStatus
13         end
14     end
15     return S
```

At design time, these nodes are treated as action nodes. They do not have children (since the robot does not need information about the sub-tasks performed by the human). The only difference to traditional action nodes is that the robot needs to know the preconditions and effects of a $\mathcal{H}$-node in order to know when -and how- to proceed working on its tasks at run-time.

When these nodes are reached, the tree stops before ticking them, As seen in Algorithm 1 Line 3 the agent stores the tree -and the associated tree context- in a buffer. The robot will use the action model later to decide whether the task execution has ended (i.e., the sub-tree is ready to resume execution). For this, we will use an expert system, such as 'C' Language Production System (CLIPS) [6]. CLIPS has three building blocks: a knowledge base, a fact list, and an inference engine. Before stopping the tree, we add a new CLIPS rule (seen in Line 4) that ticks the tree whenever the robot is idle (between trees) and either the effects hold or preconditions do not hold. When the node is ticked, it returns $\mathcal{S}$ iff the effects hold and false otherwise.

[1] using traditional control flow nodes will cause all tree nodes to receive ticks every time the root is ticked causing sub-tasks to be repeated

IV. Conclusion

We introduce the $\mathcal{H}$-node, a new type of execution node for BTs. This node stops the tree execution allowing a human teammate to perform a sub-task while the robot picks up another task. After the human has finished, the robot will be able to continue working according to the outcome of the human task. If the human finishes her task successfully, the robot can continue working on it, otherwise the robot can do some error handling (by adding this to a selector node) such as discarding of the work piece. This addition is crucial to allow behavior trees to handle HRC tasks. We plan to implement and test the work presented here on the assembly use case mentioned above and publish the results.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

References

[1] Michele Colledanchise, Alejandro Marzinotto, Dimos V Dimarogonas, and Petter Oegren. The advantages of using behavior trees in mult-robot systems. In *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pages 1–8. VDE, 2016.

[2] Blake Hannaford, Randall Bly, Ian Humphreys, and Mark Whipple. Behavior trees as a representation for medical procedures. *arXiv preprint arXiv:1808.08954*, 2018.

[3] Danying Hu, Yuanzheng Gong, Blake Hannaford, and Eric J. Seibel. Semi-autonomous simulated brain tumor ablation with ravenii surgical robot using behavior tree. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3868–3875, 2015. doi: 10.1109/ICRA.2015.7139738.

[4] Matteo Iovino, Edvards Scukins, Jonathan Styrud, Petter Ögren, and Christian Smith. A survey of behavior trees in robotics and AI. *CoRR*, abs/2005.05842, 2020. URL https://arxiv.org/abs/2005.05842.

[5] Petter Ögren. Increasing modularity of uav control systems using computer game behavior trees. In *AIAA Guidance, Navigation, and Control Conference 2012 :*, 2012. ISBN 978-160086938-9. QC 20130522.

[6] Robert M Wygant. Clips a powerful development and delivery expert system tool. *Computers & industrial engineering*, 17(1-4):546–549, 1989.

Sampling-Based Robust Control of Autonomous Systems with Non-Gaussian Noise

Thom S. Badings, Alessandro Abate, Nils Jansen, David Parker, Hasan A. Poonawala, Marielle Stoelinga

Consider a so-called *reach-avoid problem* for an unmanned aerial vehicle (UAV), where the goal is to reach a desirable region within a given time horizon, while avoiding specific unsafe regions. We model such an autonomous system formally as a *dynamical system*, whose n-dimensional *state* $\boldsymbol{x}_k \in \mathbb{R}^n$ reflects the position and velocity of the UAV at time k, and whose *control inputs* $\boldsymbol{u}_k \in \mathcal{U} \subset \mathbb{R}^p$ reflect choices that may change the state over time [13, 16]. However, factors like turbulence and human intervention cause *uncertainty* in the outcome of control inputs. For example, a human decision maker may overrule control decisions made autonomously by the UAV. If the transition of the state is *linear* in the current state and control input, the state at time $k+1$ is given by

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k + \boldsymbol{q}_k + \boldsymbol{w}_k, \tag{1}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$ are matrices, and $\boldsymbol{q}_k \in \mathbb{R}^n$ is a deterministic disturbance. The uncertainty is modeled by the *process noise* $\boldsymbol{w}_k \in \mathbb{R}^n$, which is an additive random variable that affects the transition of the state. The problem is to compute a *controller*, such that the state progresses safely (i.e., without entering unsafe regions) to its goal [2]. In particular, controllers for autonomous systems that operate in safety-critical settings like the UAV must account for uncertainty.

To ensure analytical tractability of the problem of computing a controller, it is widely assumed that the process noise is Gaussian, for example in linear-quadratic-Gaussian control [1]. However, in many cases, like the UAV under turbulence or with a human decision maker, this assumption yields a poor approximation of the uncertainty [4]. Moreover, distributions may even be unknown, meaning that it is not possible to derive a set-bounded or stochastic representation of the noise. In such cases, it is generally hard or even impossible to derive *hard guarantees* on the probability that a given controller ensures a safe progression of the system's state to the objective.

In this work, we provide *probably approximately correct (PAC)* guarantees on the performance of a controller for the aforementioned reach-avoid problem. We assume an arbitrary or even unknown distribution of the noise. We express the objective of reaching a desirable goal in finite time, while avoiding unsafe regions, as a so-called *reachability property* [3, 8]. As such, we solve the following problem:

> Given a linear dynamical system perturbed by additive noise of unknown distribution, compute a controller under which, with high confidence, the probability to satisfy a reachability property is above a given threshold value.

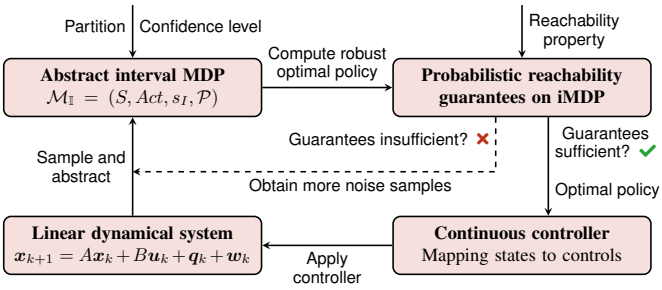


Fig. 1: Our iterative abstraction scheme.

To solve this problem, we propose the iterative abstraction scheme shown in Fig. 1, of which we highlight three key steps:

Abstraction as a Markov decision process

The fundamental concept of our approach is to compute a finite-state abstraction of the dynamical system as a *Markov decision process (MDP)* [18]. We obtain such a model from a *partition* of the continuous state space into a set of convex *regions*. Every discrete state of the MDP represents a single region, and actions correspond to control inputs that induce transitions between the regions, which are stochastic due to the process noise. Efficient methods like value iteration compute *policies* that are guaranteed to reach certain states of the MDP with optimal probability [18]. Mature tool support exists, for instance, via PRISM [14] or Storm [9].

Computing transition probability intervals via sampling

Since the distribution of the noise is unknown, it is not possible to compute the transition probabilities of the abstraction exactly. Instead, we estimate the transition probabilities based on a finite number of *samples* (also called scenarios) of the noise, which may be obtained from a high fidelity (blackbox) simulator or from experiments. However, depending on the number of samples, the true transition probabilities may critically deviate from these estimates. To be *robust* against such deviations, we employ a non-trivial sampling method known as the *scenario approach* (also called scenario optimization), which is a methodology to deal with stochastic convex optimization in a data-driven fashion [6, 7, 10]. Specifically, we compute *upper and lower bounds* on every transition probability $P(s,a)(s')$ from MDP state s to state s' under action a with a desired *confidence level* $\beta \in (0,1)$, which we choose up front. For every transition, we use the scenario approach [10] to compute a probability interval $[\underline{p}, \bar{p}]$ such that

$$\mathbb{P}\Big\{\underline{p} \leq P(s_i, a_l)(s_j) < \bar{p}\Big\} \geq 1 - \beta. \tag{2}$$

These intervals are *PAC*, as they contain the true probabilities with at least this confidence level. To formalize the abstraction, we use so-called *interval MDPs (iMDPs)*, which have transition probabilities as intervals instead of precise values [11].

Iteratively improving the abstraction quality

The tightness of the probability intervals of the iMDP can be controlled through the number of samples. Hence, we propose an *iterative abstraction scheme*, to iteratively improve the probability intervals of the iMDP by using increasing sample sizes. For the resulting iMDP, we compute a robust policy that maximizes the probability to safely reach the goal states. This policy can be derived by a variant of value iteration or convex programming and has to robustly account for all possible probabilities within these intervals [12, 17, 19]. If the reachability probability is above the threshold, we use the policy to compute a controller for the dynamical system. The confidence level β reflects the likelihood that the guarantees for the iMDP carry over to the dynamical system. This means that we compute a controller, for which we guarantee with high confidence that the probability to satisfy the objective is above a known value. If the guarantees are unsatisfactory, we collect additional samples to obtain an abstraction with a reduced level of uncertainty in the transition probabilities.

Solving the UAV motion control problem

We use our method to solve the UAV reach-avoid problem. The problem is to compute a controller that guarantees (with high confidence) that the probability to reach the goal (shown in Fig. 2) within 64 time steps, while avoiding unsafe regions, is above 75%. The 6D state encodes the position and velocity of the UAV, and control inputs reflect actuators that change the velocity components. The effect of turbulence on the state is modelled as process noise, based on a Dryden gust model [5, 15]. The effect of a human operator that overrules a control decision of the autonomous UAV could be modeled similarly. We partition the state space into $42,875$ regions: 7 values per position variable, and 5 values per velocity component.

We implement our method in Python, and tailor the model checker PRISM [14] for iMDPs to compute robust optimal policies. At every iteration, we feed the iMDP obtained for that number of samples to PRISM, which computes the optimal policy associated with the maximum reachability probability. We apply our iterative scheme with up to 6400 noise samples.

Handling abstractions with millions of transitions

The time to compute the states and actions of the initial abstraction is around 40 minutes. Thereafter, every iteration of our scheme takes 4-18 minutes and yields an iMDP with 12-46 million transitions. Run times increase with the partition size, but are almost unaffected by the time horizon of the problem.

Our iMDP abstractions yield safe controllers

As shown in Fig. 3, a reachability probability of at least 75% is guaranteed when using 3200 or more noise samples.

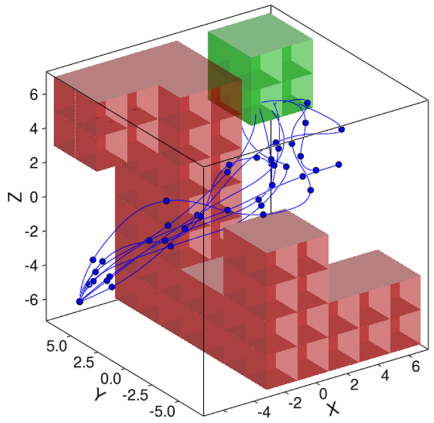


Fig. 2: UAV problem layout (goal in green; obstacles in red), plus 8 trajectories under the optimal (iMDP-based) controller.

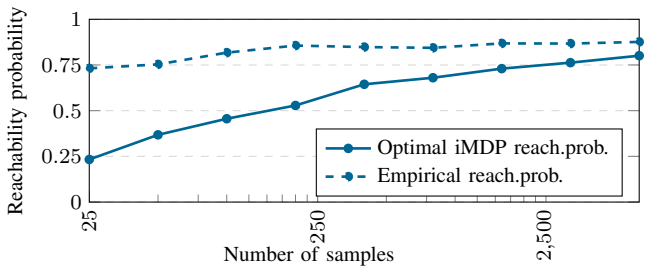


Fig. 3: Probabilistic reachability guarantees on the iMDPs, vs. empirical (simulated) performance on the dynamical system.

To validate this claim, we extract the optimal policy for every number of samples, and use it to compute a controller for the dynamical system. By simulating the system under the derived controller for every sample size $10,000$ times, we obtain the trajectories in Fig. 2 and values shown in Fig. 3. Importantly, we observe that our robust iMDP approach yields guarantees which are a strong lower bound on the actual empirical (simulated) performance of the dynamical system.

More samples means less uncertainty

Our results confirm the intuition that better probabilistic reachability guarantees are obtained when more samples are used to compute the iMDP probability intervals. In particular, the higher the number of samples, the lower the uncertainty in the probability intervals (i.e. smaller difference between the upper and lower bounds of the intervals).

Conclusion

We presented a novel sampling-based method for robust control of autonomous systems with process noise of an unknown distribution. Our experiment shows how our method effectively solves a realistic UAV motion control problem, and provides strong lower bound guarantees on the performance of the computed controller. In the future, we plan to apply our method to other control problems where human decision makers are involved. Moreover, we wish to study adaptive partitioning schemes and partial state observability.

References

[1] Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[2] Karl Johan Åström and Richard M Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[3] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[4] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C. Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE Trans. Robotics*, 26(3):502–517, 2010.

[5] Eivind Bøhn, Erlend M Coates, Signe Moe, and Tor Arne Johansen. Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 523–533. IEEE, 2019.

[6] Giuseppe Carlo Calafiore and Marco C. Campi. The scenario approach to robust control design. *IEEE Trans. Autom. Control.*, 51(5):742–753, 2006.

[7] Marco C. Campi and Simone Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM J. Optim.*, 19(3):1211–1230, 2008.

[8] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986.

[9] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *CAV (2)*, volume 10427 of *Lecture Notes in Computer Science*, pages 592–600. Springer, 2017.

[10] Simone Garatti and MC Campi. Risk and complexity in scenario optimization. *Mathematical Programming*, pages 1–37, 2019.

[11] Robert Givan, Sonia M. Leach, and Thomas L. Dean. Bounded-parameter markov decision processes. *Artif. Intell.*, 122(1-2):71–109, 2000.

[12] Ernst Moritz Hahn, Vahid Hashemi, Holger Hermanns, Morteza Lahijanian, and Andrea Turrini. Multi-objective robust strategy synthesis for interval markov decision processes. In *QEST*, volume 10503 of *Lecture Notes in Computer Science*, pages 207–223. Springer, 2017.

[13] Bohdan T Kulakowski, John F Gardner, and J Lowen Shearer. *Dynamic modeling and control of engineering systems*. Cambridge University Press, 2007.

[14] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[15] D Moorhouse and R Woodcock. US military specification MIL-F-8785C. *US Department of Defense*, 1980.

[16] Jan Willem Polderman and Jan C Willems. Introduction to the mathematical theory of systems and control. *New York*, 434, 1998.

[17] Alberto Puggelli, Wenchao Li, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Polynomial-time verification of PCTL properties of mdps with convex uncertainties. In *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 527–542. Springer, 2013.

[18] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

[19] Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. Robust control of uncertain markov decision processes with temporal logic specifications. In *CDC*, pages 3372–3379. IEEE, 2012.

Dialogue Object Search

Monica Roy*, Kaiyu Zheng*, Jason Liu, Stefanie Tellex
Department of Computer Science, Brown University

***Abstract*—We envision robots that can collaborate and communicate seamlessly with humans. It is necessary for such robots to decide both what to say and how to act, while interacting with humans. To this end, we introduce a new task, *dialogue object search*: A robot is tasked to search for a target object (e.g., fork) in a human environment (e.g., kitchen), while engaging in a "video call" with a remote human who has additional but inexact knowledge about the target's location. That is, the robot conducts speech-based dialogue with the human, while sharing the image from its mounted camera. This task is challenging at multiple levels, from data collection, algorithm and system development, to evaluation. Despite these challenges, we believe such a task blocks the path towards more intelligent and collaborative robots. In this extended abstract, we motivate and introduce the dialogue object search task and analyze examples collected from a pilot study. We then discuss our next steps and conclude with several challenges on which we hope to receive feedback.**

I. Introduction

Humans can act in the physical world (such as walking, looking, or opening a cabinet) while having a conversation with others. As robots enter homes and care centers, we envision them to have such capability as well when collaborating and communicating with humans. To achieve this, robots must decide both what to say and how to act towards a goal. This involves combining task-oriented dialogue systems with decision making under uncertainty for embodied agents. Traditionally, dialogue systems have involved users interacting with a virtual agent for tasks such as technical support [1], personal assistance (e.g., Siri) and booking reservations [2, 3]. While recent works have proposed datasets that combine dialogue and dynamic, embodied decision making [4, 5], the investigated problems over these datasets are limited to prediction tasks that bypass the challenges of evaluating a conversational embodied agent. For example, the Navigation from Dialog History Task [5] asks the agent to predict the next navigation action, given a history of dialogue and past navigation actions. The tourist localization task [4] asks the system to predict a location given a language description.

Our goal is to enable robots to naturally engage in a dialogue with a human while completing a task autonomously. We believe a task that captures the sequential nature of both the dialogue and physical decision making is necessary for in-depth study towards this goal. We choose to focus on object search, a useful and widely-studied problem [6, 7, 8, 9], and introduce a new task: *dialogue object search*. Before providing a detailed description in the next section, we note that we consider speech-based dialogue in this task. From the pilot study (Sec. III), we observed that participants produced language and behavior that are more natural using speech, because text-based dialogue requires users to decide whether to type or act at every step. Although this creates more challenges in scalable data collection and evaluation, we believe that overcoming these challenges is essential towards our goal, and they are our ongoing focus.

*These authors contributed equally to this work.

II. Dialogue Object Search

A robot is tasked to search for a target object in a human environment (e.g., kitchen) while engaging in an audio dialogue with a remote human assistant, who possesses inexact prior knowledge about the target object's location. In our pilot study, this is given in the form of a 2D scatter plot (Fig. 1). The robot has a mounted RGB-D camera, and shares its view with the human assistant. We assume the robot and the human assistant have access to two different sequences of RGB-D images of the scene, which represent their prior experiences of living in that environment. Target objects are excluded from these images. The robot must decide what to say and how to act, in order to efficiently find the target object while naturally interacting and collaborating with the human assistant.

Our inspiration for the above setting comes from the following scenario between two people living together (family or friends). One person is searching for something, such as a document or a key, but not sure where it is. They decide to video call the other home member who is currently out of the house but may have a better idea. They then engage in a dialogue while the first person conducts the search for the target object. We envision that in the future, this could happen between a home assistant robot and a human user.

III. Pilot Study

To investigate the above task, we first attempted to understand how a human would behave if they are in the robot's position. We designed and conducted a pilot study among three pairs of people (authors' lab members) using AI2-THOR [10] as the simulated home environments. In this study, we designate two roles according to the above problem setting. The *Assistant* is the person assisting in the process as the robot searches for a given target object. The *Controller* is the person who is taking on the role of the robot. Due to the pandemic, we used Zoom to record the audio and create transcripts of the dialogue. We implemented a web-based data collection tool where the *Controller* controls the agent in AI2-THOR through the web interface, and the *Assistant* has access to a 2D scatter plot of a subset of objects in the scene (Fig. 1). Each pair of participants are assigned three object search trials in one environment. They have 90 seconds to explore the environment

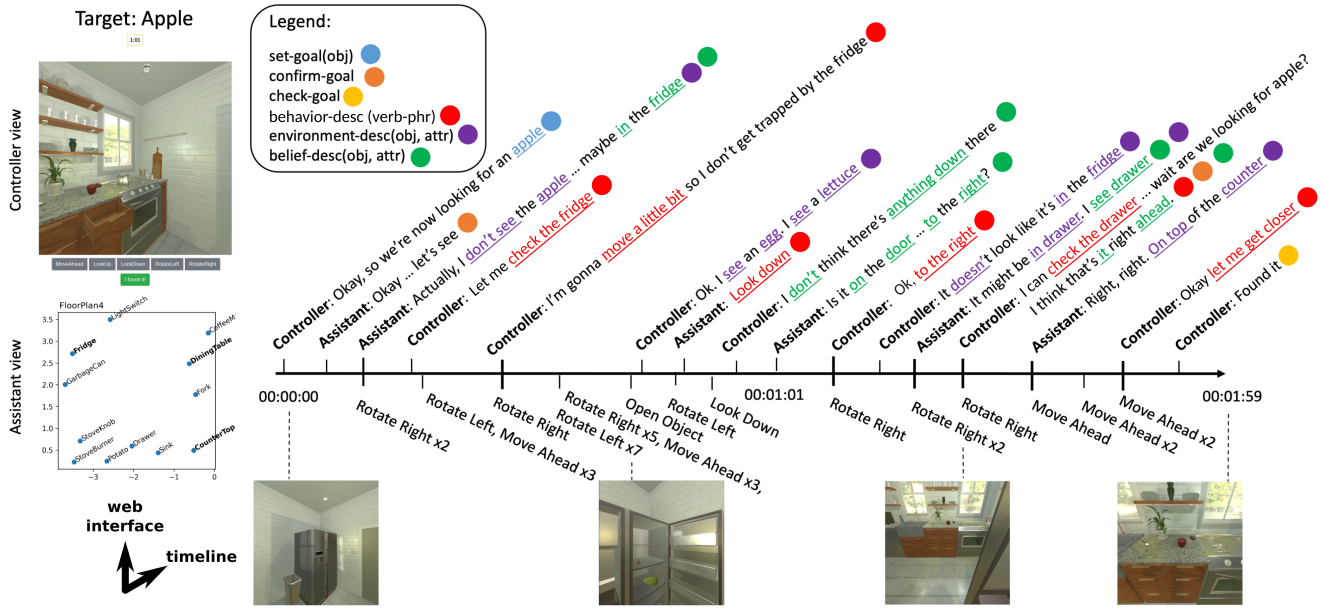


Fig. 1: We conducted a pilot study to understand desirable behavior for the dialogue object search task. Shown here is a screenshot of the web interface (left) and the dialogue and actions organized onto a timeline (right), for an object search trial where the target object is `Apple`. We classified the dialogue utterances into a preliminary set of parameterized intents, indicated by the colors.

(with target objects removed) and 180 seconds to complete each trial. In addition to dialogue audio and transcripts, we collected data about the scene per view, the action executed, and the agent's groundtruth pose as provided by the AI2-THOR framework. We considered a discrete action space of {*MoveAhead*(0.25m), *RotateLeft*(45°), *RotateRight*(45°), *LookUp*(30°), *LookDown*(30°), *Open*, *Close*}.[1].

Despite the small scale of our pilot dataset, we observed some interesting behaviors shared between trials. For example, at the beginning of the object search trials the *Assistant* would specify the target object and the *Controller* would confirm. Additionally, as the task progresses, both roles would describe behaviors, beliefs about the environment and location of objects, and visual observations. We codified these into a set of preliminary intent types; some examples are given in the figure above. Using this pilot dataset, we have started to explore the development of an autonomous agent (*Controller*), both modular and end-to-end that can plan actions for this task.

As mentioned in the introduction, we experimented with both speech-based dialogue and text-based dialogue, using the recording and chat features of Zoom. With speech, participants typically engage in frequent back-and-forth, as the *Controller* controls the agent. Such exchanges involve discussing, for example, the scene and possible target locations. Participants report that when using text, the *Controller* must decide between controlling the agent in AI2-THOR versus typing in the chat. Consequently, they would try to search for the object themselves without interacting with the *Assistant*, who, as a result, finds it difficult to tell if their input is being considered by the *Controller*. This suggests collecting dialogue data through text is unnatural and misaligned with our goal.

IV. Discussion & Next Steps

Though truthful to the task, our pilot data collection procedure is currently not scalable. We plan to implement a system that can be deployed on the crowdsourcing platform Amazon Mechanical Turk (AMT), to pair up Turkers to participate in the task entirely through their web browsers for accessibility. AMT a powerful platform, yet not designed for multi-user tasks. Due to audio communication and running AI2-THOR servers, we face a more difficult situation than Das et al. [15] who had to implemented a live chatbot on AMT. We also need solutions to scalable and accurate transcription of the collected audio as well as intent labling. We seek suggestions for strategies to collect such data at scale. In terms of evaluation, we believe both experiment with simulated assistants and real human assistants are necessary. For the simulated assistant, we are considering an oracle agent that communicates using template-based language. The goal of this simulated agent is to facilitate efficient and repeatable evaluations during algorithm development for the embodied dialogue agent, which could be a long-term effort. Ultimately, the agent should be deployed to perform the task with real human subjects. We plan to consider objective metrics for both object search performance (e.g., success rate and discounted total return[2]) and dialogue quality [17], and, eventually, subjective metrics such as naturalness [18]. We believe finding solutions to scalable speech-based dialogue data collection for embodied tasks and plausible evaluation protocol are daunting, yet unavoidable challenges towards future collaborative robots.

[1]We first experimented with a rotation angle of 90° following [11, 12], but experienced sudden jumps that are unnatural as felt by the participants. Therefore, we switch to 45°, also used by some existing works [13, 14]

[2]Because we consider open/close actions, the SPL metric [16] widely used in the object-goal navigation task is not applicable.

REFERENCES

[1] D. Mouromtsev, L. Kovriguina, Y. Emelyanov, D. Pavlov, and A. Shipilo, "From spoken language to ontology-driven dialogue management," in *International Conference on Text, Speech, and Dialogue*. Springer, 2015, pp. 542–550.

[2] T.-H. Wen, D. Vandyke, N. Mrkšić, M. Gašić, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, and S. Young, "A network-based end-to-end trainable task-oriented dialogue system," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 438–449. [Online]. Available: https://aclanthology.org/E17-1042

[3] W. Wei, Q. Le, A. Dai, and J. Li, "Airdialogue: An environment for goal-oriented dialogue research," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3844–3854.

[4] H. de Vries, K. Shuster, D. Batra, D. Parikh, J. Weston, and D. Kiela, "Talk the walk: Navigating new york city through grounded dialogue," *arXiv preprint arXiv:1807.03367*, 2018.

[5] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer, "Vision-and-dialog navigation," in *Conference on Robot Learning*. PMLR, 2020, pp. 394–406.

[6] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt, "Active visual object search in unknown environments using uncertain semantics," *IEEE Transactions on Robotics (T-RO)*, vol. 29, no. 4, pp. 986–1002, Aug. 2013. [Online]. Available: http://www.pronobis.pro/publications/aydemir2013tro

[7] T. Kollar and N. Roy, "Utilizing object-object and object-scene context when planning to find things," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2168–2173.

[8] K. Zheng, D. Bayazit, R. Mathew, E. Pavlick, and S. Tellex, "Spatial language understanding for object search in partially observed cityscale environments," in *2021 IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2021.

[9] K. Zheng, Y. Sung, G. Konidaris, and S. Tellex, "Multi-resolution POMDP planning for multi-object search in 3D," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[10] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, 2017.

[11] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "Iqa: Visual question answering in interactive environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4089–4098.

[12] X. Ye and Y. Yang, "Hierarchical and partially observable goal-driven policy learning with goals relational graph," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2021.

[13] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. Mottaghi, "Learning to learn how to learn: Self-adaptive visual navigation using meta-learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6750–6759.

[14] Y. Qiu, A. Pal, and H. I. Christensen, "Learning hierarchical relationships for object-goal navigation," in *2020 Conference on Robot Learning (CoRL)*, 2020.

[15] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra, "Visual dialog," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 326–335.

[16] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "Objectnav revisited: On evaluation of embodied agents navigating to objects," *arXiv preprint arXiv:2006.13171*, 2020.

[17] A. Venkatesh, C. Khatri, A. Ram, F. Guo, R. Gabriel, A. Nagar, R. Prasad, M. Cheng, B. Hedayatnia, A. Metallinou *et al.*, "On evaluating and comparing conversational agents," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[18] V. Hung, M. Elvir, A. Gonzalez, and R. DeMara, "Towards a method for evaluating naturalness in conversational dialog systems," in *2009 IEEE international conference on systems, man and cybernetics*. IEEE, 2009, pp. 1236–1241.

Leveraging Temporal Structure in Safety-Critical Task Specifications for POMDP Planning

Jason Liu, Eric Rosen, Suchen Zheng, Stefanie Tellex, George Konidaris
Computer Science Department, Brown University

***Abstract*—Navigating a partially observable environment while satisfying temporal and spatial constraints is an essential safety feature of many robotic applications. For example, an autonomous drone needs to understand the command "Find the supermarket while avoiding the park" to avoid possible collisions with trees. Previous approaches chose to sacrifice generality for computational efficiency in large state spaces by designing action heuristics that do not apply across different tasks or used a value-iteration-based planner that does not scale well. Our approach automatically extracts structured rewards from linear temporal logic (LTL) task specifications to guide a sampling-based POMDP planner, named LTL-POMCP. We augment a partially observable Markov decision process (POMDP) with an LTL task specification then use LTL-POMCP to solve the resultant composite POMDP. Quantitative results from a classic POMDP domain show that LTL-POMCP can generalize to various LTL task specifications and scale to large state spaces. We then demonstrate the first end-to-end system from temporally-constrained natural language to robot policies in partially observable maps in simulation.**

I. Introduction

Navigating partially observable environments by following natural language commands that specify goals and path constraints is an essential safety feature of robots interacting with humans. For example, in a search and rescue mission, first responders can command an autonomous drone by saying "Search for survivors while avoiding the explosion at location A." We can model this temporally constrained navigation problem as a partially observable Markov decision process (POMDP), whose reward is specified by a linear temporal logic (LTL) expression [13].

Previous work [2] solved this problem with a value-iteration-based planning algorithm and demonstrated its performance in small environments. It could not scale because full-width Bellman backups are intractable in large state and action spaces due to the curse of dimensionality and the curse of history. To overcome these challenges, Silver and Veness [15] proposed POMCP, a sampling-based planner. Instead of estimating the value function via iterative applications of the Bellman equation using an exact model, sampling-based methods use Monte-Carlo simulations to estimate action values from interactions with a generative model of the environment. However for different task specifications, POMCP requires different hand-specified action priors to bias the exploration. These heuristics use observations cached during simulations to decide the best action to take next. Our approach does not need domain experts to design heuristics to solve a task.

This work uses an LTL to specify a safety-critical navigation task and translates the LTL to a deterministic finite automaton (DFA) [14] that encodes terminal and subgoal rewards. The DFA and the environment POMDP are combined to construct an LTL-POMDP problem. To solve it in large domains, we propose LTL-POMCP, a sampling-based planner with an additional term added to its action value estimates to bias the sampling of actions that likely lead to subgoals specified by the DFA. During Monte-Carlo simulations, besides keeping track of action value estimates, visitation counts of states and actions, LTL-POMCP caches all DFA transitions occurred after taking an action in the current history state. It then uses the augmented action value estimates to select the next action.

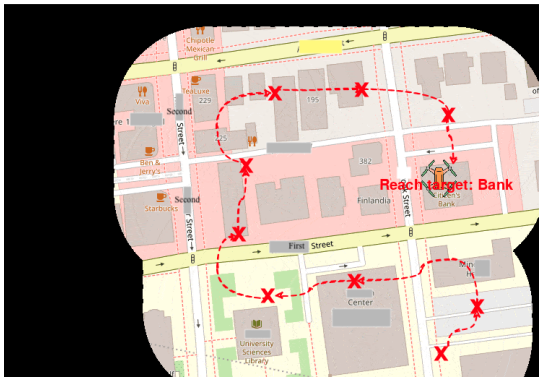


Fig. 1: An illustration of object search in the OpenStreetMap simulator. The natural language command is "Stay off the 2nd Street while looking for a bank." Its corresponding LTL formula is "G(!street2) & F(bank)." The agent has partial observability of the target bank, demonstrated by the fog-of-war effect, and perfect observation of the 2nd Street.

Results from a classic POMDP domain show that LTL-POMCP is more generalizable across task specifications than POMCP with hard-coded heuristics [15] and faster than a planner based on value iteration [2] in solving LTL-POMDPs with large state spaces. We then demonstrate the first end-to-end system from temporally-constrained natural language to robot behavior in partially observed maps [11].

The main contributions of this work are as follows.

- A POMDP formulation, LTL-POMDP, that automatically extracts structured rewards from LTL task specifications.
- A sampling-based POMDP planner, LTL-POMCP, that leverages the structured rewards, generalizes across tasks and scales well.
- An end-to-end system from temporally-constrained nat-

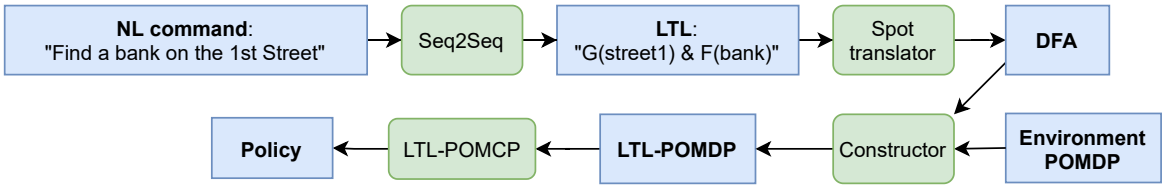


Fig. 2: **End-to-End System for PO-OSM.** Natural language is translated to an LTL then a DFA. The DFA and the environment POMDP are composed to construct an LTL-POMDP, which is solved by the LTL-POMCP online to produce a policy.

ural language to robot behavior in partially observed OpenStreetMap (PO-OSM) domain.

II. Related Work and Background

A large body of research has investigated the usage of LTL task specifications in fully observable domains. [7][9][10][1][12] studied navigation in fully observed environments while enforcing temporal constraints. We consider a more challenging partially observable setting, where an agent must actively plan to gather information. [5] proposed to learn LTL constraints from multi-step demonstrations to facilitate robotic manipulation in fully observed domains. [16] required domain experts to define reward machines that specifies goals and temporal constraints. A reward machine is a fully observable version of LTL-POMDP.

Bouton et al. [2] solved the same LTL-POMDP problem with a value-iteration-based planner for small environments. [3] requires perfect local perception to partition the environment into known and unknown areas and does not maintain a probability distribution over states in the unknown area, thus the planning is only done in a fully observable area to solve an LTL task. Silver and Veness [15] solved large POMDPs with a sampling-based planner and heuristics defined for specific tasks. Our approach extracts subgoal rewards from LTLs to guide the action selection of a sampling-based POMDP planner. This work can be generalize to solve any LTL-POMDP problem with a generative model of the environment and noisy sensors.

Linear Temporal Logic (LTL): We use LTLs [13] to specify tasks because they can represent both goals and temporal constraints.

LTL has the following syntax:

$$\phi := \sigma \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid X\phi \mid F\phi \mid G\phi \mid \phi U \psi, \quad (1)$$

where ϕ and ψ are LTL formulas; $\sigma \in \Sigma$ is an atomic proposition. $\neg, \wedge, \vee$ are logic connectives negation, conjunction and disjunction. LTL extends propositional logic with temporal operators, X (next), F (finally), G (globally or always) and U (until), applying to future time steps. We evaluate the satisfaction of an LTL formula on an infinite sequence $w = w_0 w_1 \ldots$, where $w_i \in 2^{\Sigma}$. $X\phi$ is satisfied by w at step i if ϕ is satisfied at the next step $i+1$. $F\phi$ is true at step i if ϕ holds true at some future time $j \geq i$. $G\phi$ holds if ϕ is true for the entire sequence. $\phi U \psi$ is satisfied if ϕ holds true at least until ψ becomes true, which must happen at the current or a future time. Table I shows examples of LTL formulas in the PO-OSM domain. For example, to satisfy "G(street1) & F(bank)," a robot needs to stay on the 1st Street while looking for a bank.

Deterministic Finite Automaton (DFA): We use the Spot library [6] to translate an LTL formula to an equivalent DFA [4]. A DFA is a 5-tuple $\mathrm{D} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$, where $\mathcal{Q}$ is a finite set of states; Σ is a finite set of atomic propositions; $\delta : \mathcal{Q} \times 2^{\Sigma} \to \mathcal{Q}$ is a deterministic transition function; $q_0 \in Q$ is the initial state; $\mathcal{F} = \mathcal{F}_{\text{success}} + \mathcal{F}_{\text{fail}} \subseteq \mathcal{Q}$ is a set of success and failure terminal states. A run on a finite sequence $w = w_0 w_1 \ldots w_n$ with $w_i \in 2^{\Sigma}$ produces a sequence of states $q_0 q_1 \ldots q_n$ with $q_t \in \mathcal{Q}$, where q_0 is the initial state, $q_n \in \mathcal{F}$ is a final state and $q_{t+1} = \delta(q_t, w_t)$. Table I shows some examples of LTL formulas and their corresponding DFAs.

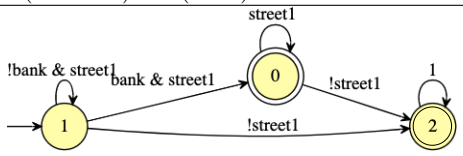
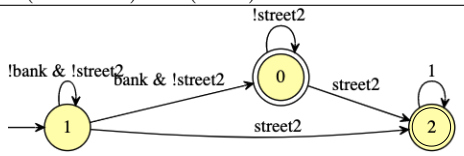
Environment POMDP: We model the environment as a Partially Observable Markov Decision Process (POMDP). A POMDP is defined by a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, \gamma)$. The dynamics $T(s, a, s') = P(s'|s, a)$ and $R(s, a) = E[r|s, a]$ determine the distribution of the next state $s' \in \mathcal{S}$ and the immediate reward after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. In POMDP, states cannot be fully observed. Instead the agent receives an observation $o \in \mathcal{O}$ based on an observation model $O(a, s', o) = P(o|a, s')$. A policy of a POMDP is define by $\pi(h) = a$, where h is a history of actions and observations. Any POMDP has at least one optimal policy π^* that maximizes $V^{\pi}(h) = E_{\pi}[\sum_{t}^{\infty} \gamma^{t-1} r_t | h]$. A belief state is a probability distribution over possible states given the history, $B(s, h) = P(s|h)$, and it is Markovian. We define our environment POMDP to be generative such that given a transition (s, a, s'), we can sample from models T, R and O to get the next state, an immediate reward and an observation. A belief state is represented by a set of particles.

III. Technical Approach

This section provides technical details on how we augment an environment POMDP with a DFA to construct an LTL-POMDP. We will also describe LTL-POMCP, a sampling-based planner that leverages structured rewards provided by the DFA, generalizes across LTL tasks and scales well.

LTL-POMDP: We augment an environment POMDP with a DFA, so states of the resultant LTL-POMDP are Markovian and encoding progressions in the DFA. LTL-POMDP $= (\tilde{\mathcal{S}}, L, \mathcal{A}, \mathcal{O}, \tilde{T}, \tilde{O}, \tilde{R}, \gamma)$, where $\tilde{\mathcal{S}} = \mathcal{S} \times \mathcal{Q}$ is a Cartesian product of environment POMDP and DFA state spaces; $L : \mathcal{S} \to 2^{\Sigma}$ is a labeling function that maps environment POMDP

TABLE I: Examples of LTL formulas and their corresponding DFAs.

Language	Find a bank while staying on the 1st Street.	Avoid the 2nd Street while looking for a bank.
LTL	$G(\texttt{street1})\ \&\ F(\texttt{bank})$	$G(!\texttt{street2})\ \&\ F(\texttt{bank})$
DFA		

states to atomic propositions.

The transition probability of entering LTL-POMDP state $\tilde{s}' = (s', q')$ after taking action a from state $\tilde{s} = (s, q)$ is

$$\tilde{T}(\tilde{s}, a, \tilde{s}') = \begin{cases} T(s, a, s'), & \text{if } q' = \delta(q, L(s')) \\ 0, & \text{otherwise} \end{cases}. \quad (2)$$

As shown in the Table I, an example LTL-POMDP transition can be that an agent takes an action in the environment to reach a bank while avoiding the 2nd Street, which induces the DFA transition from $q = 1$ to the goal $q = 0$.

The observation model of LTL-POMDP is

$$\tilde{O}(\tilde{s}, a, \tilde{s}') = O(s, a, s'). \quad (3)$$

The structured reward function is specified by the LTL progression

$$\tilde{R}(\tilde{s}, a, \tilde{s}') = \begin{cases} r_{\text{goal}}, & \text{if } q' \in \mathcal{F}_{\text{success}} \\ r_{\text{fail}}, & \text{if } q' \in \mathcal{F}_{\text{fail}} \\ r_{\text{subgoal}}, & \text{if } q' \in \mathcal{Q} - \mathcal{F} \wedge q' \neq q \\ r, & \text{otherwise} \end{cases} \quad (4)$$

where $r_{\text{goal}} \gg 0$, $r_{\text{fail}} \ll 0$, $r_{\text{subgoal}} > 0$ and $r < 0$.

LTL-POMCP: LTL-POMDPs model POMDP planning problems with temporally constrained task specifications. We introduce LTL-POMCP, a sampling-based planner that can solve LTL-POMDP problems in large environments.

We adopt the POMCP algorithm [15] with two modifications. In addition to the estimated Q-values $\hat{Q}(ha)$, visitation counts $N(h)$ and $N(ha)$, LTL-POMCP caches the frequencies of the DFA transitions occurred after taking action a in the current history state h during the Monte-Carlo simulation. We then augment the Q-value estimates with an additional third term as follows,

$$Q(ha) = \hat{Q}(ha) + \alpha\sqrt{\frac{\log N(h)}{N(ha)}} + \beta\sqrt{\frac{\log N(e_a)}{N(ha)}}, \quad (5)$$

where e_a represents a DFA transition towards a final goal state in the future trajectory after taking the action a from the current history state h; α and β are coefficients. The LTL-POMCP algorithm leverages high-level subgoals encoded in the DFA and automatically favors the transitions leading to the DFA goal state without explicitly constructing preferred action sets for rollout as in [15]. Equation 5 balances exploring less taken actions and exploiting actions that have led to a DFA transitions and high rewards. The third term in Equation 5 diminishes to 0 asymptotically because logarithm grows slower than linear, and $N(e) \leq N(ha)$.

Translating Natural Language to LTL: The language model first uses a pretrained name entity recognizer (NER) by [8] to replace all landmark names from a natural language command by place holders, then feeds the masked language into a sequence to sequence (Seq2Seq) model with LSTM cells, and finally substitute the landmark names in the output LTL expression. With the help of NER, the Seq2Seq model only needs to memorize LTL templates, not landmark names. Because NER was pretrained on a very large dataset of landmark names, this language model can recognize places unseen in the training set. It takes 118 seconds, 443 data points and 10 epochs to train the Seq2Seq model to achieve 100% accuracy on the test set.

IV. Experiments

The aim of our experiments is to test the hypothesis that the LTL-POMCP planner is more general than POMCP used with hard-coded action heuristics [15] and more scalable than value-iteration-based planners [2] in the RockSample domain. We also show an end-to-end system from temporally-constrained language to navigation policy in partially observed maps.

RockSample: A RockSample problem $RS(n, k)$ has k rocks randomly placed in an $n \times n$ grid, $k + 5$ actions (i.e. 4 move, 1 pick up and k sensing actions), 2 observations of rock types (i.e. good, bad) and deterministic transitions. The initial belief is uniformly distributed over rock types. Sensing accuracy decreases exponentially in the distance to a rock.

To compare generality, we measure the success rate and total reward of solving different tasks on the same RockSample domains using LTL-POMCP and Silver POMCP [15]. The first task requires the robot to pick up a good rock then go to exit area while avoiding bad rocks. The second task requires the robot to pick up a bad rock then exit while avoiding good rocks. We use LTL expressions, "G(! bad) & F(good & F(exit))" and "G(! good) & F(bad & F(exit))" representing both tasks respectively. As shown in Figure 3, because the action heuristics used by Silver POMCP are defined for the first task, it achieves higher success rate and more rewards. But for the second task, Silver POMCP performs even worse than basic POMCP without heuristics because the same action heuristics used to solve the first task direct the agent to pick up unfavorable rocks. LTL-POMCP can sovle both LTL tasks with comparable performance. This shows that LTL-POMCP is generalizable across different LTL task specifications.

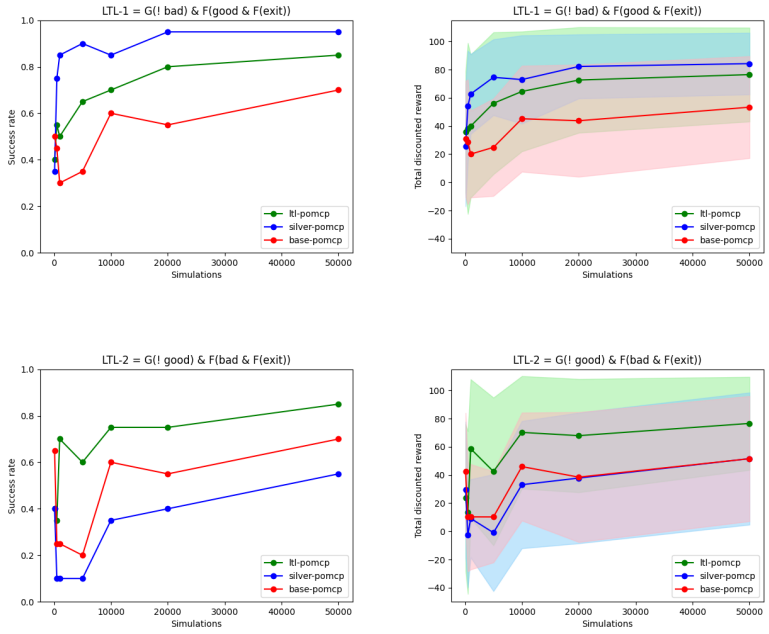


Fig. 3: The top row plots are the success rate and reward vs. the number of simulations for the LTL G(!bad) & F(good & F(exit)). The bottom row plots are for G(!good) & F(bad & F(exit)). Each data point is average over 20 runs.

To compare scalability of LTL-POMCP and SARSOP used in [2], we measure the planning time in large domain $RS(11, 11)$. An value-iteration-based-planner SARSOP provided with sparse LTL rewards [2] cannot produce a policy within 96 hours. LTL-POMCP can solve the problems constantly within 2 hours, and its speed can be further improved by using a compiled language and parallelization.

Partially Observable OpenStreetMap (PO-OSM): In PO-OSM, the locations of major landmarks, e.g. streets, are known, and the locations of small landmarks, e.g. banks, are unknown. It mimics the real world scenarios where some landmarks, like a new construction site or disaster area, are not stored in a map database, and an autonomous agent needs to locate or avoid them by following natural language commands from humans. We map landmarks from the OpenStreetMap database to a 20×20 grid. The state contains the agent's pose and target landmark locations. The initial belief is uniformly distributed over possible landmark locations. The agent can rotate in 4 cardinal directions and move forward. Every step, the agent receives a noisy observation of whether the target landmark is within its fan-shaped sensing range. We use a deterministic transition function and a non-deterministic observation model for computational reasons. They are also realistic assumptions of the drones because existing drones can reliably move around the environment but have less reliable sensors.

The end-to-end system from temporally-constrained natural language can consistently produce navigation policies to complete different tasks as shown in Table II.

TABLE II: Examples of LTL tasks that LTL-POMCP can solve and their corresponding natural language commands in PO-OSM.

Language	LTL
Find a bank.	$F(\texttt{bank})$
Find a store.	$F(\texttt{store})$
Find a cafe.	$F(\texttt{cafe})$
Stay on the 1st Street, and find a bank.	$G(\texttt{st1})\&F(\texttt{bank})$
Find a store while staying on the 2nd Street.	$G(\texttt{st2})\&F(\texttt{store})$
Stay on the 2nd Street while looking for a cafe.	$G(\texttt{st2})\&F(\texttt{cafe})$
Avoid the 2nd Street while looking for a bank.	$G(!\texttt{st2})\&F(\texttt{bank})$
Find a store and avoid the 1st Street.	$G(!\texttt{st1})\&F(\texttt{store})$
Look for a cafe while avoiding the 1st Street.	$G(!\texttt{st1})\&F(\texttt{cafe})$
Fly on the 1st Street until you find a bank.	$\texttt{st1}\mathcal{U}\texttt{bank}$
Be on the 2nd Street until you find a store.	$\texttt{st2}\mathcal{U}\texttt{store}$
Stay on the 2nd Street until you find a cafe.	$\texttt{st2}\mathcal{U}\texttt{cafe}$
Avoid the 2nd Street until you find a bank.	$(!\texttt{st2})\mathcal{U}\texttt{bank}$
Avoid the 1st Street until you find a store.	$(!\texttt{st1})\mathcal{U}\texttt{store}$
Stay off the 1st Street until you find a cafe.	$(!\texttt{st1})\mathcal{U}\texttt{cafe}$

V. Conclusion

We introduced a generalizable planner that can solve LTL-POMDP problems in large environments and demonstrated an end-to-end system from temporally-constrained natural language to robot behavior in the partially observed OpenStreetMap domain.

The automatic extraction of structured rewards from LTLs and the planner are generic, and they are are not limited to solve only navigation tasks. One interesting future work is to apply LTL-POMCP to mobile manipulation tasks specified by LTL.

References

[1] Matthew Berg, Deniz Bayazit, Rebecca Mathew, Ariel Rotter-Aboyoun, Ellie Pavlick, and Stefanie Tellex. Grounding Language to Landmarks in Arbitrary Outdoor Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[2] Maxime Bouton, Jana Tumova, and Mykel J. Kochenderfer. Point-based methods for model checking in partially observable Markov decision processes. 2020. doi: https://doi.org/10.1609/aaai.v34i06.6563.

[3] Christopher Bradley, Adam Pacheck, Gregory Stein, Sebastian Castro, Hadas Kress-Gazit, and Nicholas Roy. Learning and planning for temporally extended tasks in unknown environments. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[4] J Richard Büchi. On a decision method in restricted second order arithmetic. In *The collected works of J. Richard Büchi*, pages 425–435. Springer, 1990.

[5] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Explaining multi-stage tasks by learning temporal logic formulas from suboptimal demonstrations. In *Proceedings of Robotics: Science and Systems (RSS) XVI*, 2020.

[6] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, October 2016. doi: 10.1007/978-3-319-46520-3_8.

[7] Georgios E Fainekos, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for mobile robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2020–2025. IEEE, 2005.

[8] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL https://doi.org/10.5281/zenodo.1212303.

[9] Michael L Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via gltl. *arXiv preprint arXiv:1704.04341*, 2017.

[10] Yoonseon Oh, Roma Patel, Thao Nguyen, Baichuan Huang, Ellie Pavlick, and Stefanie Tellex. Planning with State Abstractions for Non-Markovian Task Specifications. In *Proceedings of Robotics: Science and Systems*, Freiburg, Germany, June 2019.

[11] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[12] Roma Patel, Ellie Pavlick, and Stefanie Tellex. Grounding language to non-markovian tasks with no supervision of task specifications. In *Proceedings of Robotics: Science and Systems*, June 2020.

[13] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. IEEE.

[14] Michael O Rabin and Dana Scott. Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125, 1959.

[15] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[16] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 2112–2121, 2018.

Safe Human-Interactive Control via Shielding

Jeevana Priya Inala[1], Yecheng Jason Ma[2], Osbert Bastani[2], Xin Zhang[3], Armando Solar-Lezama[1]

***Abstract*—Ensuring safety for human-interactive robotics is important due to the potential for human injury. The key challenge is defining safety in a way that accounts for the complex range of human behaviors. We propose an approach for ensuring safety based on *backup actions* we believe the human always considers taking to avoid an accident—e.g., braking to avoid rear-ending the robot. Given a set of backup actions, our approach guarantees safety as long as the human takes the appropriate backup actions when necessary to ensure safety. We evaluate our approach on real humans interacting with a simulated robot.[1]**

I. Introduction

Robots are increasingly operating in environments where they must interact with humans, such as collaborative grasping [1, 2] and autonomous driving [3, 4, 5, 6]. Ensuring safety for such robots is paramount due to the potential to inflict harm on humans [7]. These challenges are particularly salient in settings such as autonomous driving, where robots and humans may have disjoint or conflicting goals—e.g., a self-driving car making an unprotected left turn [5].

The key challenge is how to define safety for human-interactive robots. Modeling the human as an adversary is one approach, but is typically prohibitively conservative. Another approach is to learn a model to predict human actions [8, 9, 10], and ensure safety with respect to this model. However, different humans may exhibit very different behaviors—e.g., people in different regions may drive differently. If a human behavior is not exhibited in the data used to train the model, then the model may not account for it. Another alternative, called *responsibility sensitive safety (RSS)* [11], is to manually specify the range of acceptable robot actions in various scenarios. In this approach, the designer of the robot controller is responsible for ensuring that acceptable actions only include safe actions. However, manually defining acceptable robot actions for all possible scenarios is challenging, especially for robots operating in open-world environments.

We propose a novel approach for ensuring safety in human-interactive robotics systems based on the following key ideas:

- **Bounding human behavior via backup actions:** The controller designer specifies *backup actions* that they believe the human always considers taking to avoid an accident (e.g., braking while steering in some direction). We assume the human may take any action in general, take these actions when necessary to ensure safety.
- **Ensuring safety:** We use *abstract interpretation* [12] to overapproximate the reachable set of the system for the above model of human behavior, and then ensure safety with respect to this overapproximation.

[1]Appendix and video are available at: https://osf.io/atbp4/?view_only=865b3796b5454594996bc5173ce75504

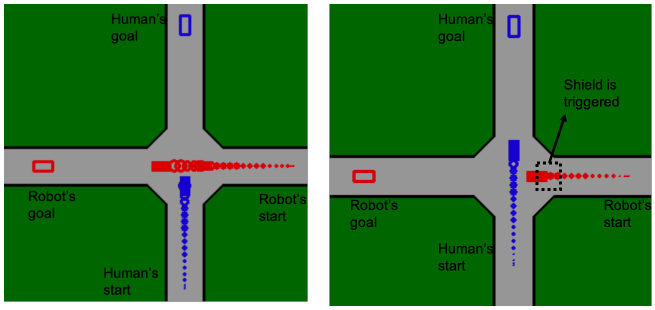


Fig. 1. Trajectories showing a robot (red) and a human (blue) interacting at an intersection (for 25 timesteps). Left: The robot passes before the human, leveraging the fact that a responsible human would slightly brake to allow the robot to cross safely. Right: Human arrives at the intersection first; the robot triggers the shield to brake and allow the human to cross first.

First, our notion of backup actions captures the idea that we reasonably believe the human will take a limited range of evasive maneuvers to avoid an accident—e.g., if the robot gradually slows to a stop, then we may expect the human to slow down to avoid rear-ending it. If the robot is on a highway, coming to a stop is more dangerous; in this case, we might conservatively restrict to the case where the robot pulls over to the shoulder before coming to a stop. Specifying backup actions provides a way to define safety; we refer to such a safety constraint as *safety modulo fault*. In particular, to instantiate our framework, the controller designer provides:

- **Robot backup action:** An action that the human anticipates the robot may take to ensure safety (e.g., to brake without changing directions), chosen based on intuitions based on traffic rules and common sense.
- **Human backup action set:** A set of actions that includes *at least one* action the human considers taking to ensure safety (e.g., braking while steering in some direction), chosen based on intuitions about human behavior.

Next, we propose an algorithm for ensuring safety modulo fault. We build on *model predictive shielding* [13, 14], which takes an arbitrary controller designed to reach the goal, but then overrides it if needed to ensure safety. In particular, our algorithm, called *MPS modulo fault*, uses on-the-fly verification based on abstract interpretation to determine whether the goal-reaching controller is safe; if so, it uses the given controller, but otherwise, it switches to a safe backup controller. Figure 1 shows how our algorithm ensures safety while interacting with a human driver without being overly cautious.

We empirically evaluate our approach on a real human interacting with a simulated robot via keyboard. We demonstrate that our algorithm enables the robot to avoid accidents, even when combined with a naïve controller that ignores the human.

II. Preliminaries

a) Human-robot system: We consider a robot R and a human H. As in prior work [5], we assume they act in alternation, which is reasonable if the time steps are small. For a state x_t where R is acting, we have

$$x'_t = f_R(x_t, u_{R,t}) \qquad \text{and} \qquad x_{t+1} = f_H(x'_t, u_{H,t}),$$

where $\mathcal{X} \subseteq \mathbb{R}^{n_X}$ is the state space, $\mathcal{U}_A \subseteq \mathbb{R}^{n_{U,A}}$ are the actions for $A \in \{R, H\}$, and $f_A : \mathcal{X} \times \mathcal{U}_A \to \mathcal{X}$ are the dynamics for A. Given initial state $x_0 \in \mathcal{X}_0 \subseteq \mathcal{X}$ where R is acting, and actions $\vec{u}_A = (u_{A,0}, u_{A,1}, ...) \in \mathcal{U}_A^\infty$ for each $A \in \{R, H\}$, we define the trajectory $\zeta_R(x_0, \vec{u}_R, \vec{u}_H) = (x_0, x'_0, x_1, ...) \in \mathcal{X}^\infty$, where $x'_t = f_R(x_t, u_{R,t})$ and $x_{t+1} = f_H(x'_t, u_{H,t})$. Similarly, given initial state $x'_0 \in \mathcal{X}$ where H is acting, we define $\zeta_H(x'_0, \vec{u}_H, \vec{u}_R) = (x'_0, x_0, x'_1, ...)$, where $x_t = f_H(x'_t, u_{H,t})$ and $x'_{t+1} = f_R(x_t, u_{R,t})$. We can replace each $\vec{u}_A$ by a policy $\pi_A : \mathcal{X} \to \mathcal{U}_A$. Our goal is to ensure the system stays in a given safe region $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$.

Definition II.1. A trajectory $\zeta = (x_0, x'_0, x_1, ...)$ (or $\zeta = (x'_0, x_0, x'_1, ...)$) is *safe* if $x_t, x'_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \mathbb{N}$.

III. Safety Modulo Fault

Here, we formalize our assumptions and safety notion.

a) Human objective: We assume the human acts according to a maximin objective, where the "min" is the worst-case over actions the human anticipates the robot may take, and the "max" is over the human's own actions. That is, the human plans optimally while conservatively accounting for actions they anticipate the robot might take. In particular, at state x', the human takes an action $\pi_H(x') = u^*_{H,0}$ such that

$$\vec{u}^*_H \in \arg\max_{\vec{u}_H \in \mathcal{U}_H^\infty} J_H(x', \vec{u}_H), \tag{1}$$

where the $\arg\max$ denotes the set of all optimal values, and

$$J_H(\vec{u}_H) = \min_{\vec{u}_R \in \hat{\mathcal{U}}_R^\infty} J_H(\zeta_H(x', \vec{u}_H, \vec{u}_R))$$

$$J_H((x'_0, x_0, x_1, ...)) = \sum_{t=0}^{\infty} \gamma^t r_H(x'_t, u_{H,t}, x_t),$$

where $\hat{\mathcal{U}}_R \subseteq \mathbb{R}^{n_{U,R}}$ is the set of actions the human anticipates the robot may take, $r_H : \mathcal{X} \times \mathcal{U}^H \times \mathcal{X} \to \mathbb{R} \cup \{-\infty\}$ is the human reward function, and $\gamma \in (0, 1)$ is a discount factor.

The key challenge for the robot to plan safely is that it does not know the human reward function r_H, the human action set $\mathcal{U}_H$ or the human-anticipated robot action set $\hat{\mathcal{U}}_R$. Assuming we know these values exactly is implausible. Instead, we assume access to minimal knowledge about each of these objects, which we formalize in the next section.

b) Assumption on human objective: First, we assume that the human reward for reaching an unsafe state is $-\infty$.

Assumption III.1. For any $x', x \in \mathcal{X}$ and $u_H \in \mathcal{U}_H$, we have $r_H(x', u_H, x) = -\infty$ if $x' \notin \mathcal{X}_{\text{safe}}$ or $x \notin \mathcal{X}_{\text{safe}}$.

That is, the human driver always acts to avoid an accident. Other than Assumption III.1, r_H can be arbitrary.

With this assumption, there are two reasons accidents may happen: (i) there was a safe action sequence $\vec{u}_H \in \mathcal{U}_H^\infty$ that the human driver failed to take, or (ii) if the robot takes an action $u_R \notin \hat{\mathcal{U}}_R$ that the human driver failed to anticipate. Thus, we can always conservatively take $\hat{\mathcal{U}}_R$ to be smaller than it actually is. Conversely, we can always take $\mathcal{U}_H^0$ to be larger than it actually is. Thus, we make minimal assumptions about what actions are contained in $\hat{\mathcal{U}}_R$ and $\mathcal{U}_H$.

First, we make the following assumption on the set of actions $\hat{\mathcal{U}}_R$ that the human anticipates the robot may take:

Assumption III.2. We are given a *robot backup action* $u_R^0 \in \mathcal{U}_R$ that is anticipated by the human—i.e., $u_R^0 \in \hat{\mathcal{U}}_R$.

That is, the human always accounts for the possibility that the robot might take action u_R^0. For example, we might assume that u_R^0 is gradually braking and coming to a stop.

Next, we make the following assumption about the human:

Assumption III.3. We are given a *human backup action set* $\mathcal{U}_H^0 \subseteq \mathcal{U}_H$ such that if $\max_{\vec{u}_H \in \mathcal{U}_H^\infty} J_H(\vec{u}_H) = -\infty$, then the human takes an action $\pi_H(x') \in \mathcal{U}_H^0$.

That is, if the human is unable to guarantee safety (i.e., their objective value is $-\infty$), then they take *some* action in $\mathcal{U}_H^0$. For example, $\mathcal{U}_H^0$ may contain all actions where the human driver decelerates by at least some rate; this choice allows them to slow down more quickly or steer in any direction.

c) Problem formulation: Our goal is to ensure that the robot acts in a way that ensures safety for an infinite horizon for any human that satisfies our assumptions.

Definition III.4. A robot policy $\pi_R : \mathcal{X} \to \mathcal{U}_R$ is *safe modulo fault* for initial states $\mathcal{X}_0 \subseteq \mathcal{X}$ if for any human policy π_H satisfying Assumptions III.1, III.2, & III.3, and any $x_0 \in \mathcal{X}_0$, the trajectory $\zeta_R(x_0, \pi_R, \pi_H) \in \mathcal{X}^\infty$ is safe.

That is, π_R that ensures safety as long as the human acts in a way that satisfies our assumptions. Our goal is to design a policy π_R that is safe modulo fault.

Finally, we cannot guarantee safety starting from an arbitrary state x_0. For instance, if the robot is about to crash into a wall, no action can ensure safety. We assume that the initial states $\mathcal{X}_0$ are ones where we can guarantee safety.

Definition III.5. A *safe equilibrium state* $x \in \mathcal{X}$ satisfies (i) $x \in \mathcal{X}_{\text{safe}}$, and (ii) $x = f(x, u_R^0, u_H)$ for all $u_H \in \mathcal{U}_H^0$.

We denote the set of safe equilibrium states by $\mathcal{X}_{\text{eq}}$. At a state $x \in \mathcal{X}_{\text{eq}}$, the robot and human can together ensure safety for an infinite horizon by taking actions u_R^0 and u_H for any $u_H \in \mathcal{U}_H^0$. In our driving example, $\mathcal{X}_{\text{eq}}$ contains states where both agents are at rest (i.e., their velocity is zero).

Assumption III.6. We have $\mathcal{X}_0 \subseteq \mathcal{X}_{\text{eq}}$.

In other words, the system starts at a safe equilibrium state where we can ensure safety for an infinite horizon.

Algorithm 1 Model predictive shielding modulo fault.

procedure $\pi_R(x)$
  $x' \leftarrow f_R(x, \hat{\pi}_R(x))$
  return if IsRec(x') **then** $\hat{\pi}_R(x)$ **else** u_R^0 **end if**
end procedure
procedure IsRec(x')
  $X'_0 \leftarrow \{x'\}$
  for $t \in \{0, ..., k-1\}$ **do**
    if $X'_t \not\subseteq \mathcal{X}_{\text{safe}}$ **then return false end if**
    $U_{R,t} \leftarrow$ **if** $t = 0$ **then** $\{u_R\}$ **else** $\{u_R^0\}$ **end if**
    $U_{H,t} \leftarrow \mathcal{U}_H^0$
    $X'_{t+1} \leftarrow F(X_t, U_{R,t}, U_{H,t})$
  end for
  return if $X_k \subseteq \mathcal{X}_{\text{eq}}$ **then true else false end if**
end procedure

IV. Model Predictive Shielding Modulo Fault

We describe our algorithm for constructing a robot controller $\pi_R : \mathcal{X} \to \mathcal{U}_R$ that is safe modulo fault. Our approach is based on *model predictive shielding (MPS)* [13, 14], which converts an arbitrary controller $\hat{\pi}_R : \mathcal{X} \to \mathcal{U}_R$ into a controller π_R that uses $\hat{\pi}_R$ but overrides it when it cannot ensure it is safe. The challenge is checking whether it is safe to use $\hat{\pi}_R$. The idea is to maintain the invariant that the current state is *recoverable*—i.e., that there is some sequence of actions each agent can take that safely brings the system to a stop.

Definition IV.1. Given $k \in \mathbb{N}$, a state $x' \in \mathcal{X}$ is *recoverable* (denoted $x' \in \mathcal{X}_{\text{rec}}$) if for $\vec{u}_R = (u_R^0, u_R^0, ...) \in \mathcal{U}_R^\infty$ and any $\vec{u}_H \in (\mathcal{U}_H^0)^\infty$, $\zeta_H(x', \vec{u}_H, \vec{u}_R) = (x'_0, x_0, x'_1, ...)$ satisfies (i) $x'_t, x_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \{0, ..., k\}$, and (ii) $x_k \in \mathcal{X}_{\text{eq}}$.

Now, our MPS modulo fault algorithm for computing π_R is shown in Algorithm 1. Here, IsRec checks whether $x' = f_R(x, \hat{\pi}_R(x))$ is recoverable. If so, π_R returns $\hat{\pi}_R(x)$; otherwise, it returns u_R^0. To check recoverability, IsRec overapproximates the reachable set of states after t steps as a set $X_t \subseteq \mathcal{X}$. It assumes given a *dynamics overapproximation* $F : 2^{\mathcal{X}} \times 2^{\mathcal{U}_R} \times 2^{\mathcal{U}_H} \to 2^{\mathcal{X}}$ mapping sets of states $X \subseteq \mathcal{X}$, sets of robot actions $U_R \subseteq \mathcal{U}_R$, and sets human action $U_H \subseteq \mathcal{U}_H$ to sets of states $F(X, U_R, U_H) \subseteq 2^{\mathcal{X}}$, that satisfies

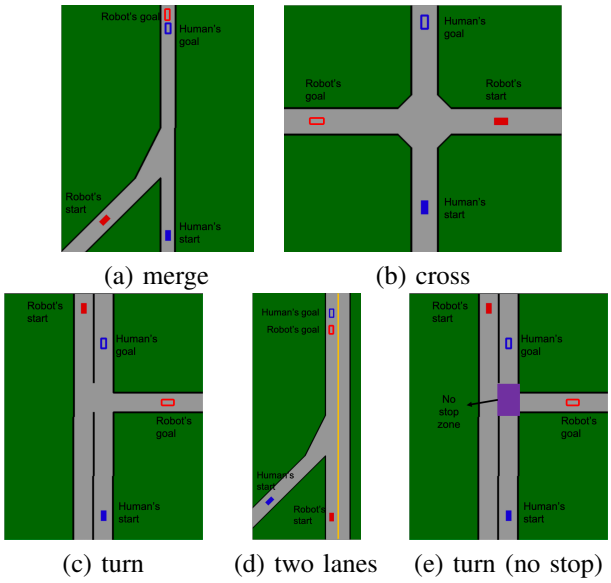
$$f(x, u_R, u_H) \in F(X, U_R, U_H) \tag{2}$$

for all $x \in X$, $u_R \in U_R$, and $u_H \in U_H$. Then, IsRec checks whether (i) safety holds for every state $x_t \in X_t$ (i.e., $X_t \subseteq \mathcal{X}_{\text{safe}}$), and (ii) every state $x_k \in X_k$ is a safe equilibrium state (i.e., $X_k \subseteq \mathcal{X}_{\text{eq}}$). If both hold, then x is recoverable. We have the following (see Appendix A for a proof):

Theorem IV.2. *Assuming (2) holds, then our policy π_R is safe modulo fault (i.e., it satisfies Definition III.4).*

V. Evaluation

We have implemented our approach in a simulation for three robotics tasks. We consider an aggressive robot controller with and without the shield as well as a cross entropy method controller (CEM) designed to avoid humans. Also, we consider real humans interacting with the simulation via keyboard.



(a) merge (b) cross (c) turn (d) two lanes (e) turn (no stop)

Fig. 2. Visualizations of the different tasks along with the initial positions and the goals for the robot and the human. The red box is the robot and the blue box is the human.

We focus on understanding whether our approach can ensure safety in aggressive driving scenarios; our approach can easily be tailored to drive more conservatively, which would further improve safety (but may reduce performance). We focus on settings where the human and the robot must compete to reach their goals. We tune the parameters of our MPS modulo fault algorithm (i.e., the robot backup action u_R^0 and the human backup action set $\mathcal{U}_H^0$) to be as aggressive as possible while still ensuring safety on the simulated humans. Furthermore, for our experiments with real-world humans, we strongly encourage them to try and reach their goal before the robot, albeit keeping safety as the top priority. Then, our results are designed to answer the following questions:

- Can MPS modulo fault can be used to ensure safety?
- Can MPS modulo fault outperform a handcrafted MPC based on CEM in terms of performance?

A. Experimental Setup

a) Robotics tasks: We consider three non-cooperative tasks (see Figure 2): (i) "merge": there are two lanes that merge—i.e., the robot is coming in from one lane and the humans from another; both their goals are to navigate the merge and reach their goal, (ii) "cross": both agents are moving towards an intersection from different directions—i.e., the robot is moving horizontally and the human is moving vertically; both their goals are to get to the other side of the intersection, (iii) "turn": an unprotected left turn—i.e., the human is driving without turning and the robot needs to make a left turn that crosses the human path.

b) Safety property: We assume the robot and human are each a rectangle; then, the safety property is that the the robot and human rectangles should not intersect.

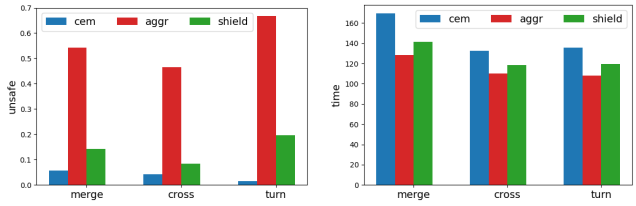


Fig. 3. Results with real humans, for the aggressive controller (red), the CEM MPC (blue), and our shielded aggressive policy (green). Left: Fraction of unsafe runs. Right: Time the robot takes to reach its goal in seconds.

c) Robot dynamics: The robot dynamics are the ones in our running example—i.e., its state is (x, y, v, θ), where (x, y) is position, v is velocity, and θ is orientation, and its actions are (a, ϕ), where a is acceleration and ϕ is steering angle.

d) Humans: We consider real human users interacting with the simulation via keyboard. They control the human using the up/down arrows to control acceleration and the left/right arrows to control steering angle. We asked the human users to prioritize safety first, but to drive aggressively to try and reach their goal before the robot. We also considered simulated humans, including multiple humans; see Appendix B.

e) Controllers: We consider three controllers for the robot: (i) an aggressive controller, (ii) a handcrafted MPC based on the cross-entropy method (CEM) designed to ensure safety without a shield, and (iii) our MPS modulo fault algorithm used in conjunction with the aggressive controller. We give details in Appendix B.

B. Experimental Results

We describe our experimental results, based on 18 users.

a) MPS modulo fault ensures safety for real humans: Next, we had real human users interact with our simulated robot via keyboard input. we show both the fraction of unsafe runs (left), and the time taken by the robot to reach the goal (right), including the aggressive controller (red), the MPC based on CEM (blue), and our shielded aggressive controller (green). As can be seen, for the aggressive controller, the robot gets to its goal the fastest, but is frequently unsafe. The MPC based on CEM is significantly safer; in this case, it is somewhat safer than our shielded aggressive controller. On the other hand, our shield controller reaches its goal significantly faster than the MPC, while being almost as safe as the MPC CEM. As described above, we set the shield parameters aggressively based on the simulated humans to ensure it could reach its goal; in practice, we could further improve safety by setting these parameters more conservatively and by tuning them to the real human driver data.

b) Alternative robot backup actions: A key feature of our approach is that we can flexibly design the robot backup action to ensure safety. To demonstrate this flexibility, we design an alternative backup action that pulls the robot over to the shoulder of a highway. Note that this backup policy is time varying—i.e., the robot steering depends on the current state. We test this backup policy with simulated humans on the task in Figure 2 (d), where there are two lanes on the highway and an on-ramp that merges onto the highway. The human is on the on-ramp and the robot is on the highway. To avoid collisions, the robot can pull over to the right-most lane. Figure 4 shows the fraction of the unsafe runs (leftmost), and the time the robot takes to reach its goal (second from left) for all three controllers—aggressive (red), the MPC based on CEM (blue), and our shielded aggressive controller with the pull over backup policy (green). Our shielded controller is always safe and is significantly faster than the MPC.

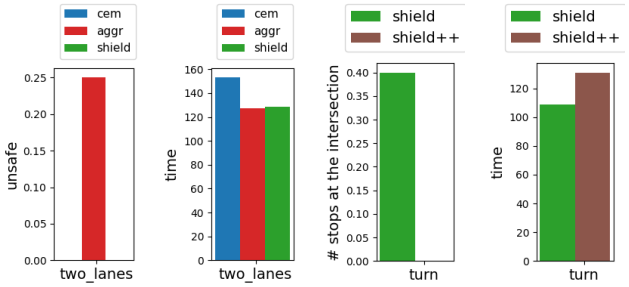


Fig. 4. Results for alternative robot backup actions with simulated humans. For the "pull over" backup action, we show the fraction of unsafe runs (leftmost) and the time the robot takes to reach its goal in seconds (second from the left), for the aggressive controller (red), the CEM MPC (blue), and our shielded aggressive controller (green). For the "no-stop zone" backup action, we show the number of stops in the intersection (second from the right) and the time the robot takes to reach its goal in seconds (rightmost), for the original (green, "shield") and the new (brown, "shield++") shielded controllers; both controllers are always safe.

We also design a robot backup action that avoids stopping in the middle of an intersection and blocking it, which is often illegal. To this end, we modify the turn task to include a no-stop zone (shown in Figure 2 (e)) where the robot is prohibited from stopping. In this zone, the robot backup action does not come to a stop immediately; instead, it drives through the zone until it crosses the intersection, and only brakes once it has fully cleared the intersection. The results for this experiment using simulated humans are shown in Figure 4 (right). We compare the original shielded controller ("shield") that may stop in the intersection with the new one that adheres to the no-stop zone in Figure 2 (e) ("shield++"). In this case, both the controllers were always safe; instead, we show the fraction of runs where the robot stops in the intersection (second from the right), and the time the robot takes to reach its goal (rightmost). The new shielded controller takes slightly longer to reach the goal, but never stops in the intersection.

VI. Conclusion

We have proposed an approach for ensuring safety in human-interactive robotics systems. We define a notion of safety that models human behavior by specifying their backup behaviors, and propose our MPS modulo fault algorithm for ensuring our safety with respect to this model. Finally, we validate our approach on both real and simulated humans.

References

[1] K. Strabala, M. K. Lee, A. Dragan, J. Forlizzi, S. S. Srinivasa, M. Cakmak, and V. Micelli, "Toward seam-

less human-robot handovers," *Journal of Human-Robot Interaction*, vol. 2, no. 1, pp. 112–132, 2013.

[2] A. D. Dragan, K. C. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in *8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2013, pp. 301–308.

[3] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[4] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 163–168.

[5] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions." in *Robotics: Science and Systems*, vol. 2. Ann Arbor, MI, USA, 2016.

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

[7] K. Eder, C. Harper, and U. Leonards, "Towards the safety of human-in-the-loop robotics: Challenges and opportunities for safety assurance of robotic co-workers'," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2014, pp. 660–665.

[8] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[9] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin, and A. D. Dragan, "Probabilistically safe robot planning with confidence-based human predictions," in *RSS*, 2018.

[10] D. Sadigh, S. S. Sastry, S. A. Seshia, and U. Berkeley, "Verifying robustness of human-aware autonomous cars," *IFAC-PapersOnLine*, vol. 51, no. 34, pp. 131–138, 2019.

[11] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[12] P. Cousot, "Abstract interpretation," in *In POPL*. Citeseer, 1977.

[13] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 7130–7135.

[14] S. Li and O. Bastani, "Robust model predictive shielding for safe reinforcement learning with stochastic dynamics," in *ICRA*, 2019.

[15] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

Simultaneously Learning Human Preferences and Environment Dynamics Parameters

Adam Allevato
The University of Texas at Austin
Austin, Texas, USA
Email: allevato@utexas.edu

Elaine Schaertl Short
Tufts University
Medford, MA, USA

Andrea L. Thomaz
The University of Texas at Austin
Austin, Texas, USA

***Abstract*—Learning and respecting human preferences is an important step towards creating robots that can effectively operate in human-centric environments and social contexts. We consider the specific problem of learning preferences from a human while also learning unknown dynamics parameters in order to optimize a robot controller. We develop a novel Bayesian Optimization technique that uses two linked models to estimate the human preferences and environment parameters as it alternates between collecting preference input and testing different parameter hypotheses. Our approach uses a simulator to collect off-policy human input and a novel preference-weighted acquisition function to guide exploration in the presence of both model/dynamics uncertainty and preference uncertainty. Our method violates the true human preferences up to 75% less often than existing parameter optimization algorithms while learning a simulated controller with similar performance. We also present preliminary results on a real robotics task, enabling a robot to learn about and avoid an undetectable hazard using human input during controller optimization in a new environment.**

I. Introduction

A robot learning a new task in an unfamiliar environment may need to learn from human feedback in order to develop an appropriate control methodology for that task, even before it has adequate knowledge of its environment's dynamics. Much existing human-robot interaction research focuses on learning either the human preferences or how to complete a task, but not both simultaneously. However real human-robot interaction scenarios (such as collaborative manufacturing and shared autonomy) have uncertainty in both domains.

This paper introduces Preference-based, Uncertainty-aware Model Adaptation (PUMA), a new interactive machine learning algorithm that uses Bayesian Optimization [5] to learn unknown human preferences while also optimizing a control scheme to account for unknown environment dynamics. PUMA allows learning from human feedback while a robot's knowledge of environment dynamics are still incomplete by 1) considering both types of uncertainty using two linked Gaussian processes, 2) alternating between sampling human feedback and exploring the environment, and 3) using a simulator as a predictive model for off-policy feedback collection.

In experiments, PUMA has significantly fewer preference violations during learning even with a single human feedback sample per iteration, and allows optimizing a simulated controller with 35-75% fewer violations compared to existing methods. We also validate the approach on a real-world robotics task, learning the dynamics of a new environment while avoiding an undetectable hazard using human input.

II. Related Work

Researchers have developed numerous techniques for learning from simple human feedback on robot actions [15, 19, 11, 7] . Our work also guides learning with human input, but rather than using on-policy feedback collection on robot state-action pairs, we use a simulator to collect feedback on hypothetical (off-policy) states. This is similar to two existing works [24, 22], but our robot also has to learn about its environment on the fly. These existing approaches are also model-free interactive machine learning techniques, whereas PUMA is model-based.

PUMA uses a probabilistic formulation, instead of a constraint-based approach often used in safe reinforcement learning (RL) research [9]. Probabilistic preferences allow compatibility with sparse, arbitrary, or noisy human feedback. Other works have explored probabilistic safety in learning [21, 14], including SafeOpt [3], which we compare with PUMA in this paper. In other recent works that combine human feedback with exploration strategies [12, 18, 4, 20], the environment parameters are well-characterized and fixed beforehand, rather than being learned, or the queries do not use data-efficient active learning.

III. Background: Bayesian Optimization

Bayesian optimization [5] is a core component of PUMA. It is a technique to optimize an expensive-to-evaluate *objective function* $f(x)$ efficiently, and has been used for many simulated and real-world robotics optimization tasks [10, 23], including tuning cart-pole balancing controllers [2].

Bayesian optimization maintains a hypothesis, $P(f)$, of the objective function, as well as its uncertainty. Given $P(f)$ and an evidence dataset $\mathcal{D} = \{(x_0, f(x_0)), \dots (x_n, f(x_n))\}$, the hypothesis is updated at each step of the optimization via Bayes' rule: $P(f \mid \mathcal{D}) \propto P(\mathcal{D} \mid f)P(f)$. In this paper, the evidence is based on a hypothetical state of a robot system and the scalar-valued human preference input associated with that state. A Gaussian Process (GP) is commonly used to approximate $f(x)$ probabilistically. We use the notation $\text{GP}_f(x)$ to represent the GP hypothesis of $f(x)$, and the notation

FITGP($\mathcal{D}$) to denote any method for fitting a GP to evidence (ex: Rasmussen [17]).

To determine x, the next point sample from the objective function, Bayesian optimization maximizes an *acquisition function*, $a(x)$: $x = \arg\max_{x'} a(x' \mid \mathcal{D})$, which is designed to be fast to evaluate. One of the most common acquisition functions is the *expected improvement*, or EI [16].

IV. Problem Statement

Our formal goal is to find a controller that minimizes a cost function $J(x_t, u(x_t))$ for a dynamical system over multiple timesteps, where $x_t \in \mathcal{X}$ is the state at time t and $u(x_t)$ is the resulting control input. The control inputs u and some bounded set of system parameters $\theta \in \Theta$ determine transitions: $x_{t+1} = f_\theta(x_t, u(x_t))$. We make the assumption that for some true and proposed sets of system parameters θ^* and θ, we can develop an appropriate control law $u_\theta(x)$. This is reasonable for many systems—we can use existing, well-understood methods to generate a controller via techniques such as linear-quadratic-Gaussian control (for our purposes, the controller need not be optimal). We define a *total cost function* $\mathcal{J}(\theta)$ over a τ-step horizon as $\mathcal{J}(\theta) = \sum_{t=0}^{\tau} J(x_t, u_\theta(x_t))$.

We seek a controller which minimizes this total cost function. This problem reduces to finding a θ that maximizes $-\mathcal{J}(\theta)$. (θ may not equal θ^* because of model mismatch or unidentifiable sets of parameters.)

We also want to avoid violating human preferences, represented by a *preference function* $h(x)$ that maps from states to scores in the range [0,1], where 1 is most preferable. $h(x)$ may encode actual or perceived safety or any other criteria and is initially unknown to the robot. Minimizing preference violations is equivalent to maximizing the cumulative value of $h(x)$ over some time horizon. We can sample $h(x)$ by querying a human about the preference of a specific system state, but seek to minimize these queries for data efficiency. Preferences are assumed to be a function of *states*, rather than actions or parameters, allowing the robot to collect preferences without demonstrations or corrections.

We define a *preference score* $g(\theta)$ over a period of time as the fraction of preferred states visited during an episode on the real system while deploying a controller u_θ (Eq. (1)).

$$\begin{aligned} g(\theta) &= \frac{1}{\tau}\sum_{t=1}^{\tau} h(x_t) \\ x_{t+1} &= f_{\theta^*}(x_t, u_\theta(x_t)) \end{aligned} \tag{1}$$

A preference score of 1 implies no preference violation; a score of 0 means that all states in a trajectory fully violated the preferences. Note that the real system evolves based on the true system parameters θ^* but the controller is based on the current parameter hypothesis θ.

V. Model Adaptation with Preferences

Our approach, Preference-based Uncertainty-aware Model Adaptation (PUMA) (Algorithm 1), uses Bayesian Optimization and two separate Gaussian Processes (GPs) to estimate $h(x)$ and $-\mathcal{J}(\theta)$ and iteratively optimize the two estimates, seeking a better controller while avoiding preference violations.

Algorithm 1 The PUMA algorithm

Input

$\mathcal{X}, \Theta$	State and parameter spaces
N	optimization rounds
M	# of human pref. samples per iteration

$\mathcal{D}_h \leftarrow \{\}, \mathcal{D}_\mathcal{J} \leftarrow \{\}$
for $n \in 1 \dots N$ **do**
  for $m \in 1 \dots M$ **do**
    $x_{n,m} \leftarrow \arg\max_{x \in \mathcal{X}} a_{\text{EI}}(\text{GP}_h, x)$
    $\mathcal{D}_h \leftarrow \mathcal{D}_h \cup \{(x_{n,m}, h(x_{n,m}))\}$
    $\text{GP}_h(x) \leftarrow \text{FITGP}(\mathcal{D}_h)$
  $\hat{\theta} \leftarrow \arg\max_{\theta' \in \Theta} \text{GP}_\mathcal{J}(\theta')$
  $\theta_n \leftarrow \arg\max_{\theta \in \Theta} a_{\text{PEI}}(\hat{\theta}, \theta)$
  $\mathcal{D}_\mathcal{J} \leftarrow \mathcal{D}_\mathcal{J} \cup \{(x_{n,m}, -\mathcal{J}(\theta_n))\}$
  $\text{GP}_\mathcal{J} \leftarrow \text{FITGP}(\mathcal{D}_\mathcal{J})$

Each PUMA iteration begins with M Bayesian optimization steps to improve the estimate of the human preference function $h(x)$. At each step $m = 1 \dots M$, we choose the next *query state* $x_{n,m}$ to sample by maximizing the standard EI acquisition function [16]. We set the exploration hyperparameter ξ to 1.0 to encourage state exploration. Then, using our simulator, we render the system at the query state $x_{n,m}$ and collect a preference value $h(x_{n,m})$ from a human evaluator. The preference feedback $(x_{n,m}, h(x_{n,m}))$ enters the preference dataset $\mathcal{D}_h$, which the algorithm uses to update $h(x)$.

After M rounds of preference learning, PUMA performs one round of Bayesian optimization on $\text{GP}_\mathcal{J}(\theta)$, the estimate of the total negative cost $-\mathcal{J}(\theta)$. To avoid violating human preferences while selecting θ_n, the next parameter value to sample, we develop a new acquisition function, the *preference-weighted expected improvement* (PEI). PEI combines the standard EI acquisition function with a preference term, which gives the estimated preference score that will result from testing a new set of system parameters.

Because the preferences $h(x)$ are defined over states, not parameters, we must *predict* the states that will occur for a parameter hypothesis. To predict, PUMA uses the *maximum a posteriori* MAP parameter estimate, $\hat{\theta} = \arg\max_{\theta' \in \Theta} \text{GP}_\mathcal{J}(\theta')$. The algorithm then estimates the preference score given by Eq. (1) for different hypothetical parameter values $\theta \in \Theta; \theta \neq \hat{\theta}$. This estimate, $\tilde{g}(\hat{\theta}, \theta)$ and shown in Eq. (2), is based on a state rollout of τ timesteps using the current preference estimate GP_h, the estimated underlying dynamics model $\hat{\theta}$, and the associated controller, $u_\theta(x)$.

$$\begin{aligned} \tilde{g}(\hat{\theta}, \theta) &= \frac{1}{\tau}\sum_{t=1}^{\tau} \text{GP}_h(\tilde{x}_t) \\ \tilde{x}_{t+1} &= f_{\hat{\theta}}(\tilde{x}_t, u_\theta(\tilde{x}_t)) \end{aligned} \tag{2}$$

A higher predicted score implies that a controller is more

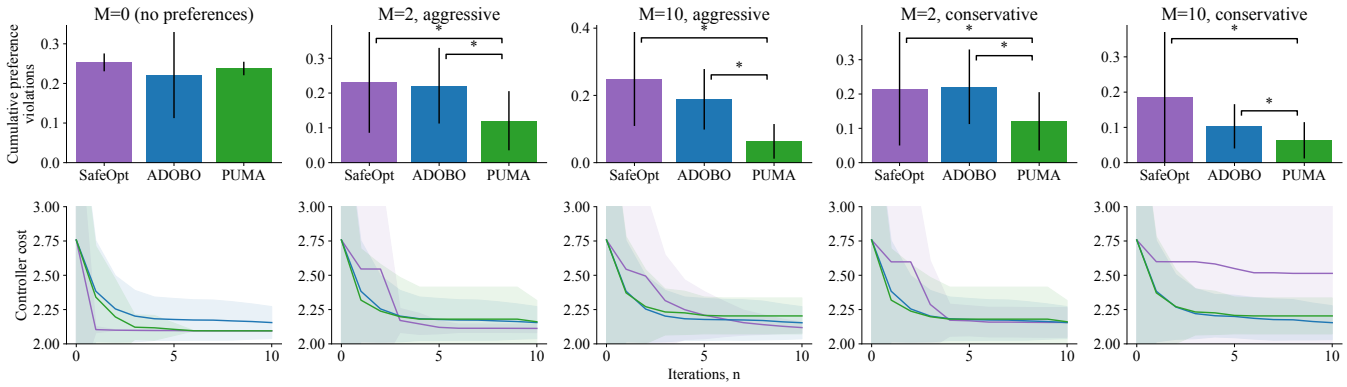
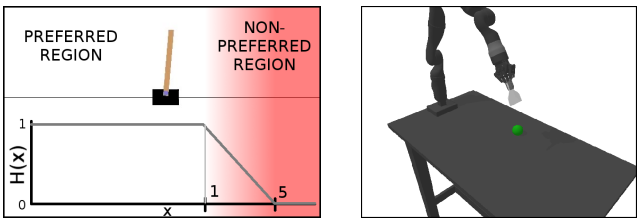


Figure 1: Experiment 1 results comparing PUMA (our method) to baselines. In all plots, lower is better. The top row shows the final error after n=10 rounds of tuning averaged over 40 runs of each algorithm. The bottom row shows the corresponding controller cost evolution (arbitrary units). See the text for details on the different cases. * denotes significance ($p < 0.01$).



(a)

(b)

Figure 2: (a): The simulator, preferred region, and human preference function for Experiment 1. (b): The simulator used for Experiment 2.

preferred *given the current best estimate of the model parameters*. The PEI is then found by adding the predicted preference score to the standard EI acquisition function value: $a_{\text{PEI}}(\hat{\theta}, \theta) = a_{\text{EI}}(\theta) + \gamma \tilde{g}(\hat{\theta}, \theta)$. PEI also provides γ as a scaling factor to control how PEI balances between preferred vs. potentially improved parameters. θ_n, the final choice for the next parameter to use for controller evaluation, is then calculated as $\theta_n = \arg\max_\theta a_{\text{PEI}}(\hat{\theta}, \theta)$.

VI. Experiments

We compare PUMA to two baselines in our experiments. The first baseline is Dynamics Optimization via Bayesian Optimization (aDOBO) [2], which directly optimizes controller parameters using Bayesian Optimization. We include preferences by adding a weighted adjustment term: $\mathcal{J}_{aDOBO}(\theta) = \mathcal{J}_{controller}(\theta) - \zeta g'(\theta)$. The second baseline is SafeOpt [3], a parameter optimization technique which avoids violating arbitrary safety functions. We use the human preferences as one of SafeOpt's safety functions, and vary the lower bound on the admissible preference score to balance preference and controller optimization. We calculate $g'(\theta)$ for the baselines by sampling Eq. (1) M times at regularly-spaced intervals throughout the episode.For all methods, we used a GP radial basis function kernel with a length scale of 0.1, variance of 0.2, and $\alpha = 1\text{e}{-5}$. For PUMA, we set the parameter $\gamma = 10$ for all experiments, chosen empirically from the range $[0.1, 1000]$.

A. Experiment 1: Cartpole Optimization

In our first experiment, we deploy PUMA on a simulated cartpole system (Fig. 2a) based on the open-source implementation provided in OpenAI Gym [6], where the length of the pendulum is unknown. The goal is to generate a controller that keeps the pendulum upright and minimizes control effort, as given by $J(x, u) = \sum_{t=0}^{\tau} x_t^T Q x_t + u_t^T R u_t$. In this experiment, we simulate the human preferences using an oracle, $H(x) = \text{clamp}(1.25 - x/4, 0, 1)$ (see Fig. 2a). This represents a human's preference to avoid nearing the end of the cartpole rail or other unknown hazard.

Optimization occurs in a series of episodes, between which the cart position and pole angle are set to 0 and $\pi/8$ radians respectively. For small angles, we can calculate a (linear) control law to keep the pendulum upright, $u(\boldsymbol{x}) = -K\boldsymbol{x}$, where K is the linear quadratic regulator (LQR) solution [13] to the cost function. As is common practice, we set the Q and R matrices to $\text{diag}(0.1, 1, 100, 1)$ and $[0.1]$ to emphasize keeping the pendulum upright. The pendulum length was set to $0.5\,\text{m}$. The exploration bounds were $\Theta = [0.1, 0.9]$ and $\mathcal{X} = [-5\,\text{m}, 5\,\text{m}]$. In each run, the initial model hypothesis was sampled uniformly from the parameter space: $\theta_0 \sim \text{U}(\Theta)$.

We report both the cumulative preference violation, as well as the controller cost at each iteration. We measure these metrics for each algorithm for $M = 0$ (no preferences), $M = 2$ and $M = 10$. We also test an *aggressive* case and *conservative* case. For the aggressive case, The SafeOpt preference *fmin*$= 0.8$ and the aDOBO preference weight $\alpha = 10$; for conservative, SafeOpt *fmin*$= 0.9$ and aDOBO $\alpha = 100$. PUMA is unchanged across cases. We conducted 40 independent runs of each algorithm.

The results in Fig. 1 show that in the $M = 0$ case, the algorithms behave similarly, but with feedback ($M > 0$), PUMA incurred significantly less ($p < 0.01$) cumulative preference violation than the baselines with near-equal controller

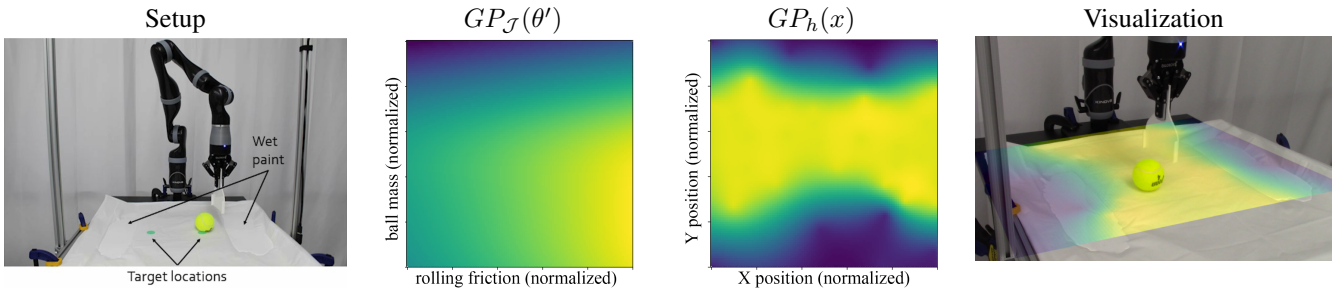
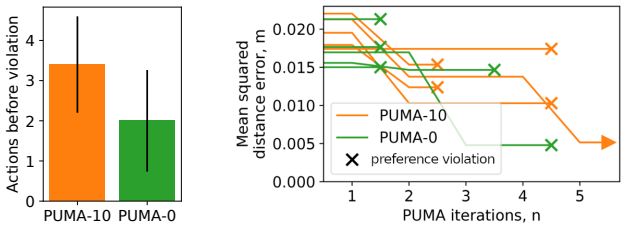


Figure 3: Visualization of the setup and learned GP estimates for Experiment 2.



(a) (b)

Figure 4: Results on ball manipulation task (Experiment 2).

performance. SafeOpt was highly sensitive to the initial θ guess and it had the highest variance in all experiments. The aggressive SafeOpt case found a marginally better controller, but had high preference violation. In the conservative case and with more preference information ($M = 10$), the algorithm had less violations, but often failed to find a good controller. aDOBO developed similar controllers to PUMA, but had significantly more preference violations. The largest PUMA-to-baseline difference was in the $M = 10$ aggressive case, where PUMA reduced the amount of preference violation by 74.6% compared to SafeOpt, and the smallest difference was a 38.5% reduction from aDOBO to PUMA in the $M = 10$ conservative case.

B. Experiment 2: Manipulation Task

Next, we deployed PUMA on a real robot (Fig. 3) to jointly learn the mass and rolling friction (μ_{roll}) coefficient of a previously unseen ball while learning to roll the ball between two points in a robot's workspace. The robot must keep the ball between two paint spills that are undetectable by the robot's vision system during the learning process. We compared PUMA-0 (no preference input) and PUMA-10 in this experiment.

The environment state was the (x, y) position of the ball on the table, and the robot's controller pushed the ball in a straight line towards the goal point, with trajectory length proportional to the estimated rolling friction of the object. The cost function $\mathcal{J}(\boldsymbol{\theta})$ for each rollout was the mean squared error between the ball's final position and the target. We used a PyBullet[1] environment (see Fig. 2b) as $f_{\hat{\theta}}$ for generating rollouts and rendering images to use for eliciting human preferences. One of the authors provided the human feedback manually for this experiment, giving a binary score ($H(x) \in \{0, 1\}$) for whether or not the rendered ball position shown on a computer screen appeared to be clear of the hazard. Collecting 10 preferences at each iteration took approximately 30 seconds.

The robot detected the ball position with an overhead camera. The exploration spaces were $[0.01\,\text{kg}, 0.15\,\text{kg}]$ and $[1\text{e}{-4}, 1\text{e}{-2}]$ for mass and rolling friction respectively, and the state space was a $0.4\,\text{m}$ x $0.4\,\text{m}$ workspace area, with the robot at one edge. We conducted 5 trials for each algorithm with up to 5 iterations in each trial, stopping early if the ball was pushed into the paint spills. We recorded the number of robot actions before a preference violation as well as the error between the goal and the final ball position after each action.

Fig. 4a shows that PUMA-10 completed more actions (3.4 $\pm$ 1.2) before violating the preferences compared to PUMA-0 (2.0 $\pm$ 1.3), although the sample size for this experiment was too small for statistical significance. PUMA-10 also achieved a slightly lower tracking error than PUMA-0 on average (0.012 $\pm$ 0.004 vs. 0.015 $\pm$ 0.005). Fig. 4b provides more detail on each trial, showing that 60% of PUMA-0 trials violated preferences on the first exploration action, and no trial completed all 5 actions. In contrast, all PUMA-10 trials completed the first iteration and one trial completed all actions without preference violations.

VII. Discussion

By modeling human preferences separately from environment uncertainty, but linking the two models using the PEI, PUMA allows information sharing and ensures that the two optimizations do not proceed completely independently. In comparison, the baseline algorithms take preference feedback and exploration actions, but apply both directly to a single underlying model.

Using a simulator *inside the learning loop* is becoming an increasingly common way to give a robot physical intuition [8, 1]. Our work extends this idea, using the simulation to augment a robot's prediction skills and allow fast and informative human-robot communication. In the future, we plan to conduct additional robot experiments to continue to validate PUMA in realistic human-robot interaction scenarios.

[1] https://pybullet.org

References

[1] Adam David Allevato, Elaine Schaertl Short, Mitch Pryor, and Andrea L Thomaz. Iterative residual tuning for system identification and sim-to-real robot learning. *Autonomous Robots*, pages 1–16, 2020.

[2] Somil Bansal, Roberto Calandra, Ted Xiao, Sergey Levine, and Claire J Tomlin. Goal-driven dynamics learning via Bayesian optimization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5168–5173. IEEE, 2017.

[3] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *arXiv preprint arXiv:1602.04450*, 2016.

[4] Erdem Biyik and Dorsa Sadigh. Batch Active Preference-Based Learning of Reward Functions. In *Conference on Robot Learning*, pages 519–528, 2018.

[5] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *CoRR*, 6 2016. URL http://arxiv.org/abs/1606.01540.

[7] Carlos Celemin and Javier Ruiz-del Solar. COACH: learning continuous actions from corrective advice communicated by humans. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 581–586. IEEE, 2015.

[8] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience. *CoRR*, 10 2018. URL http://arxiv.org/abs/1810.05687.

[9] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[10] Javier González, Zhenwen Dai, Andreas Damianou, and Neil D Lawrence. Preferential bayesian optimization. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1282–1291. JMLR. org, 2017.

[11] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L. Isbell, and Andrea L. Thomaz. Policy Shaping: Integrating Human Feedback with Reinforcement Learning. In C J C Burges, L Bottou, M Welling, Z Ghahramani, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 2625–2633. Curran Associates, Inc., 2013.

[12] Michael Herman, Tobias Gindele, Jörg Wagner, Felix Schmitt, and Wolfram Burgard. Inverse reinforcement learning with simultaneous estimation of rewards and dynamics. In *Artificial Intelligence and Statistics*, pages 102–110, 2016.

[13] Rudolf Emil Kalman. Contributions to the theory of optimal control. *Boletin de la Sociedad Matematica Mexicana*, 5(2):102–119, 1960.

[14] Johannes Kirschner, Mojmir Mutny, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and Safe Bayesian Optimization in High Dimensions via One-Dimensional Subspaces. In *International Conference on Machine Learning*, pages 3429–3438, 2019.

[15] W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The TAMER framework. In *Proceedings of the 5th Intl. Conf. on Knowledge Capture*, pages 9–16. ACM, 2009.

[16] Daniel James Lizotte. *Practical bayesian optimization*. University of Alberta, 2008.

[17] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[18] Yanan Sui, Vincent Zhuang, Joel W Burdick, and Yisong Yue. Stagewise Safe Bayesian Optimization with Gaussian Processes. *Proceedings of Machine Learning Research*, 80:4781–4789, 2018.

[19] Ana C Tenorio-Gonzalez, Eduardo F Morales, and Luis Villaseñor-Pineda. Dynamic reward shaping: training a robot by voice. In *Ibero-American conference on artificial intelligence*, pages 483–492. Springer, 2010.

[20] Sanjay Thakur, Herke van Hoof, Juan Camilo Gamboa Higuera, Doina Precup, and David Meger. Uncertainty Aware Learning from Demonstrations in Multiple Contexts using Bayesian Neural Networks. *arXiv preprint arXiv:1903.05697*, 2019.

[21] Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. Safe Exploration and Optimization of Constrained MDPs using Gaussian Processes. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[22] Christian Wirth and Johannes Furnkranz. On Learning From Game Annotations. *IEEE Transactions on Computational Intelligence and AI in Games*, 7:304–316, 2015.

[23] S Zhu, D Surovik, K E Bekris, and A Boularias. Closing the Reality Gap of Robotic Simulators through Task-oriented Bayesian Optimization. *Journal of Machine Learning Research*, 2019.

[24] Matt Zucker, J Andrew Bagnell, Christopher G Atkeson, and James Kuffner. An optimization approach to rough terrain locomotion. In *2010 IEEE International Conference on Robotics and Automation*, pages 3589–3595. IEEE, 2010.

Towards Interactively Improving Human Users' Understanding of Robot Behavior

Peizhu P. Qian and Vaibhav V. Unhelkar
Department of Computer Science
Rice University, Houston, Texas USA
{pqian, vaibhav.unhelkar}@rice.edu

***Abstract*—Humans increasingly use embodied AI systems (such as service robots and autonomous vehicles) in a variety of complex domains. Despite this impressive trend, however, users often have little understanding of behavior of robots – leading to misplaced trust and unintended consequences during robot use. Human-interpretable instructions regarding robot behavior provide a way to improve user understanding and alleviate aforementioned concerns. In this short article, we briefly discuss challenges of arriving at these instructions and formalize the problem computationally. In our discussion, we highlight the essential role of *interactivity* in improving user's understanding and discuss mechanisms to leverage it to generate user-specific instructions.**

I. Introduction

Robots provide services to a broad array of end-users in homes, offices, and beyond. They collaborate with people on complex tasks such as disaster response, manufacturing, and transportation. While these robots typically behave according to preprogrammed policies or plans, their behavior may not be intuitive to end-users. For example, a fire fighter working with a rescue robot might wonder how the robot plans its route to rescue victims: will it *go through* uneven terrain if that provides the shortest path, or will it *circumvent* said terrain to avoid the risk of falling? Knowing the answers to questions such as these is essential for human end-users to truly realize the benefits of robots, ensure safety, and avoid unintended side effects [8, 14].

Robot behaviors can be designed or learned using a variety of methods (such as hand-crafted policies, reinforcement learning, or planning algorithms), which might not prioritize interpretability. Irrespective of the method used to design robot behavior, however, humans can make informative inferences about robot behavior provided they have a faithful understanding (mental model) of the robot [6, 12]. While humans naturally create mental models of entities they interact with, this process might be slow and inaccurate [4, 10]. To ensure safe use of robots, thus, the onus of establishing accurate mental models regarding robots lies on us – robotics researchers and designers. Towards this call to action, our vision is to create an **AI Teacher that can assist end-users in establishing accurate understanding of robots that they work with.**

Recognizing the need for teaching users about robots, multiple formative approaches have been proposed in recent years [9, 16, 17, 18]. Existing work has primarily focused on *first* computing human-interpretable instructions (typically, explanations or examples) of robot behavior and *then* communicating the precomputed instructions to end-users. These precomputed instructions help users acquire models of robots. However, in absence of interactivity, the instructions may not be tailored to specific end-users.

To address the needs and preferences of diverse stakeholders that interact with robots, we posit that interactivity between the AI Teacher and users will be critical. Hence, our ongoing work explores the design of human-in-the-loop algorithms to generate instructions and improve user understanding of robot behavior. In this short article, we briefly discuss the challenges of generating human-interpretable instructions using a robot's policy and formulate the problem of designing an interactive AI Teacher.

II. Research Challenges

Explaining robot behavior presents several research challenges. Here, we focus on the computational challenges, which primarily arise due to the need to

- model and assess how users model robots; and
- effectively select user-specific instructions.

We touch upon on these challenges next, which span across disciplines including human-robot interaction (HRI), psychology, education, and cognitive science.

A. Estimating Users' Perception, Understanding, and Belief

Theory of Mind (ToM) refers to humans' ability to interpret and predict behavior of other humans by attributing mental states (e.g., belief, desire, intent) to oneself or others [2, 3]. Recent work in HRI indicates that humans also attribute ToM to robots that they observe or interact with [6, 12]. Consequently, this characteristic of human mind has been utilized to develop methods that generate explanations of robot behavior [9].

To generate instructions tailored to specific users, it is essential for the AI Teacher to estimate human users' mental model of robots. A reliable estimate of the mental model is required to evaluate utility

of different instructions. However, estimating users' mental models are challenging as they depend on latent and user-specific features (such as prior knowledge, experience, and attention during the teaching process). For instance, in our ongoing research, we observe that participants interacting with our AI Teacher arrive at mental models of robots differently based on their prior experience; a participant with background in economics adopted a utilitarian perspective, while a participant who loves solving puzzles treated the robot modeling process as a mathematical puzzle.

Computational techniques (such as Bayesian inference) offer a mechanism to estimate human's mental models, but may require prohibitive amounts of data to generate user-specific estimates [1, 7]. Interactivity during the teaching process can augment these techniques and help ease the inference of mental models. For instance, user evaluations can help the AI Teacher estimate mental models both before and during the teaching process.

B. Designing Human-Centered Pedagogical Methods

In addition to the mental model estimates, algorithms are needed to decide the type, content, and sequence of the most-effective instructions. As effectiveness of instructions depends on a user's prior knowledge and learning style, resolution of this challenge can also benefit from an interactive and user-centered approach.

Researchers in pedagogy have examined a variety of teaching strategies in the human-to-human teaching setting [11, 13, 15]. One categorization of these strategies is based on the use of direct or indirect instructions. Teaching strategies utilizing direct instructions are teacher-centered, involve clear teaching objectives, and consistent classroom organizations. In contrast, strategies involving indirect instructions are student-centered and encourage independent learning. Human-to-human teaching, in practice, typically involves a combination of the two perspectives.

Similarly, an AI Teacher explaining robot behavior to users (i.e., its students) can utilize a combination of the two perspectives. However, existing approaches to explaining robot behavior are primarily teacher-centered, where the teacher decides the appropriate instructions using domain knowledge (e.g., [18]) and by modeling the user's mental model (e.g., [9]). An interactive AI Teacher can help achieve a hybrid approach, where based on the needs and preferences of the user some explanations are teacher-centric and others user-centric. In the remainder of the paper, we summarize our ongoing effort to realize such an interactive AI Teacher.

III. Towards an Interactive AI Teacher

To discuss the design of the AI Teacher (depicted in Fig. 1), we begin by mathematically formulating the problem of *interactively improving a user's understanding of a robot's behavior*. We assume that the robot behavior can

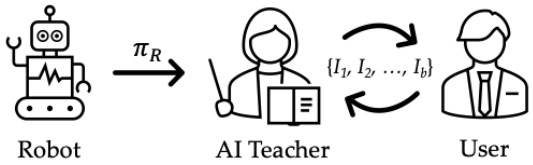


Fig. 1. Schematic of the interactive AI Teacher.

be summarized by its (potentially, stochastic) policy π_R, which maps robot states s to actions a. The AI Teacher has complete knowledge of the robot policy and aims to maximize user knowledge K regarding robot policy using a budget of b instructions $I_{1:b}$.

Instruction Set. In general, the instructions can be disseminated using a variety of modalities (such as natural language and augmented reality). In our initial formulation, however, we limit instructions to either context-specific examples or user assessments of robot behavior, i.e., (s, a)-tuples. The choice of instruction set enables the design of domain-agnostic techniques for the AI Teacher, provided the robot policy is known.

Modeling Interactivity. Our formulation includes two avenues for interactivity: user assessments during the teaching process and user-centered instructions. User assessments allow the AI Teacher to interactively assess a user's understanding (mental model) during the teaching process. Further, in our framework the user can additionally design a subset of the instructions, by requesting context-specific examples of robot behavior (i.e., requesting knowledge a for a given s). Inclusion of this degree of freedom enables our approach to be as interactive as desired by the user.

Representing User Knowledge. We capture user's knowledge of the robot using the variable K, which describes the ratio of robot states s for which the user knows the robot's policy $\pi_R(s)$ to the total number of states. Formally, the objective of the AI Teacher is to generate a sequence of b instructions given policy π_R:

$$f(\pi_R) = \underset{I_{1:b}}{\text{argmax}}\ Pr(K = 1 | I_{1:b}, \pi_R). \tag{1}$$

We highlight that the AI Teacher needs to select not only the instructions but also their sequence, as some instructions might be more effective initially. A subset of the b instructions may be selected by the user, making the decision-making further challenging.

Solution Overview. To tackle this problem, our design of the AI Teacher includes an approach to estimate the user's mental model, a planning algorithm to select salient instructions, and an interactive interface for teaching. We mathematically represent the user's mental model based on Griffiths' probabilistic model of cognition [5]. The effect of instructions on the evolution of mental model is captured using Bayesian theory of mind [1].

Given this representation, we pose the instruction-generation problem as a planning problem and use a modified Monte Carlo Tree Search (MCTS) algorithm to select salient examples of robot behavior. The search algorithm selects the instructions and sequences them by evaluating their effect on a user's knowledge of a robot's policy. Finally, our AI Teacher includes an interactive user interface that allows a user to self-explore a robot's behavior. To evaluate our approach and the effect of interactivity, we are actively conducting a user study to explaining robot behavior in two simulated domains, inspired by recycling and search-and-rescue.

References

[1] Chris Baker, Rebecca Saxe, and Joshua Tenenbaum. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[2] S. Baron-Cohen. Precursors to a theory of mind: Understanding attention in others. *Whiten, Andrew (ed.), Natural theories of mind*, pages 233–251, 1991.

[3] G. S. Becker. *The economic approach to human behavior*. University of Chicago press, 1976.

[4] Ruth Byrne and P.N. Johnson-Laird. 'If' and the problems of conditional reasoning. *Trends in cognitive sciences*, 13:282–7, 07 2009. doi: 10.1016/j.tics.2009.04.003.

[5] Thomas Griffiths, Nick Chater, Charles Kemp, Amy Perfors, and Joshua Tenenbaum. Probabilistic models of cognition: Exploring representations and inductive biases. *Trends in cognitive sciences*, 14: 357–64, 08 2010.

[6] Thomas Hellström and Suna Bensch. Understandable robots. *Paladyn, Journal of Behavioral Robotics*, 9:110–123, 07 2018. doi: 10.1515/pjbr-2018-0009.

[7] Laura M Hiatt, Cody Narber, Esube Bekele, Sangeet S Khemlani, and J Gregory Trafton. Human modeling for human–robot collaboration. *The International Journal of Robotics Research*, 36(5-7): 580–596, 2017.

[8] Ayanna Howard and Jason Borenstein. The ugly truth about ourselves and our robot creations: the problem of bias and social inequity. *Science and engineering ethics*, 24(5):1521–1536, 2018.

[9] Sandy H. Huang, David Held, Pieter Abbeel, and Anca D. Dragan. Enabling robots to communicate their objectives. *Autonomous Robots*, 43(2), February 2019.

[10] Philip Johnson-Laird. Mental models and human reasoning. *Proceedings of the National Academy of Sciences of the United States of America*, 107:18243–50, 10 2010. doi: 10.1073/pnas.1012933107.

[11] Lynn Julien-Schultz, Nancy Maynes, and Cilla Dunn. Managing direct and indirect instruction: A visual model to support lesson planning in pre-service programs. *The International Journal of Learning: Annual Review*, 17:125–140, 2010.

[12] Sau lai Lee, Ivy Yee man Lau, S. Kiesler, and Chi-Yue Chiu. Human mental models of humanoid robots. In *2005 IEEE International Conference on Robotics and Automation*, pages 2762–2772, 2005.

[13] Kevin A. Nguyen, Jenefer Husman, M. A. Trujillo Borrego, Prateek Shekhar, Michael J. Prince, Matt DeMonbrun, Cynthia J. Finelli, Charles Henderson, and Cindy K. Waters. Students' expectations, types of instruction, and instructor strategies predicting student response to active learning. *International Journal of Engineering Education*, 33:2–18, 2017.

[14] Raja Parasuraman and Victor Riley. Humans and automation: Use, misuse, disuse, abuse. *Human factors*, 39(2):230–253, 1997.

[15] Tiia Ruutmann and Hants Kipper. Teaching strategies for direct and indirect instruction in teaching engineering. In *2011 14th International Conference on Interactive Collaborative Learning*, pages 107–114, 2011.

[16] Roykrong Sukkerd, Reid Simmons, and David Garlan. Tradeoff-focused contrastive explanation for mdp planning. *arXiv preprint arXiv:2004.12960*, 2020.

[17] Aaquib Tabrez, Shivendra Agrawal, and Bradley Hayes. Explanation-based reward coaching to improve human performance via reinforcement learning. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 249 – 257. IEEE, 2019.

[18] Yusen Zhan, Anestis Fachantidis, Ioannis Vlahavas, and Matthew E. Taylor. Agents teaching humans in reinforcement learning tasks. In *International Conference on Autonomous Agents and Multiagent Systems*, 2014.

Constrained Feedforward Neural Network Training via Reachability Analysis

Long Kiu Chung*, Adam Dai*, Derek Knowles, Shreyas Kousik, and Grace X. Gao

***Abstract*— Neural networks have recently become popular for a wide variety of uses, but have seen limited application in safety-critical domains such as robotics near and around humans. This is because it remains an open challenge to train a neural network to obey safety constraints. Most existing safety-related methods only seek to verify that already-trained networks obey constraints, requiring alternating training and verification. Instead, this work proposes a *constrained* method to simultaneously train and verify a feedforward neural network with rectified linear unit (ReLU) nonlinearities. Constraints are enforced by computing the network's output-space reachable set and ensuring that it does not intersect with unsafe sets; training is achieved by formulating a novel collision-check loss function between the reachable set and unsafe portions of the output space. The reachable and unsafe sets are represented by constrained zonotopes, a convex polytope representation that enables differentiable collision checking. The proposed method is demonstrated successfully on a network with one nonlinearity layer and ≈ 50 parameters.**

I. Introduction

Neural networks are a popular method for approximating nonlinear functions, with increasing applications in the field of human-robot interactions. For example, the kinematics of many elder-care robots [1], [2], rehabilitation robots [3], [4], industrial robot manipulators [5], and automated driving systems [6], [7] are controlled by neural networks. Thus, verifying the *safety* of the neural networks in these systems, before deployment near humans, is crucial in avoiding injuries and accidents. However, it remains an active area of research to ensure the output of a neural network satisfies user-specified constraints and requirements. In this short paper, we take preliminary steps towards safety via constrained training by representing constraints as a collision check between the reachable set of a neural network and unsafe sets in its output space.

A. Related Work

Many different solutions have been proposed for the *verification* problem, with set-based reachability analysis being the most common for an uncertain set of inputs [8]. Depending on one's choice of representation, the predicted output is either exact (e.g. star set [6], [9], ImageStar [10]) or an over-approximation (e.g. zonotope [11]) of the actual output set. Reachability is most commonly computed layer-by-layer, though methods have been proposed that speed up verification by, e.g., using an anytime algorithm to return unsafe cells while enumerating polyhedral cells in the input space [12], or recursively partitioning the input set via shadow prices [13].

Verification techniques have several drawbacks. First, they do not provide feedback about constraints during training, so one must alternate training and verification until desired properties have been achieved. Furthermore, verification by over-approximation can often be inconclusive, while exact verification can be expensive to compute.

Several alternative approaches have therefore been proposed. For example, [14] employs a constrained optimization layer to use the output of the network as a potential function for optimization while enforcing constraints. Similarly, [15], [16] adds a constraint violation penalty to the objective loss function and penalizes violation of the constraint. These methods augment their networks with constrained optimization, but are unable to guarantee constraint satisfaction upon convergence of the training. Alternatively, [17] uses a systematic process of small changes to conform a "mostly-correct" network to constraints. However the method only works for networks with a Two-Level Lattice (TLL) architecture, requires an already-trained network, and again does not guarantee a provably safe solution. Finally, [18] attempts to learn the optimal cost-to-go for the Hamilton–Jacobi–Bellman (HJB) equation, while subjected to constraints on the output of the neural network controller. Yet, it does not actually involve any network training and is unable to handle uncertain input sets.

Recently, constrained zonotopes have been introduced as a set-based representation that is closed under linear transformations and can exactly represent any convex polytope [19], [20]. Importantly, these sets are well-suited for reachability analysis due to analytical, efficient methods for computing Minkowski sums, intersections, and collision checks; in particular, collision-checking only requires solving a linear program. We leverage these properties to enable our contributions.

B. Contributions

We propose a method to compute the output of a neural network with rectified linear unit (ReLU) activations given an input set represented as constrained zonotopes. We then enforce performance by training under a differentiable zonotope intersection constraint, which guarantees safety upon convergence. Our method is demonstrated on a small numerical example, and illustrated in Fig. 1.

* indicates equal contribution. All authors are with Stanford University, Stanford, CA. L.K. Chung is with the Department of Mechanical Engineering. A. Dai is with the Department of Electrical Engineering. D. Knowles, S. Kousik, and G.X. Gao are with the Department of Aeronautics and Astronautics. Corresponding author: gracegao@stanford.edu.

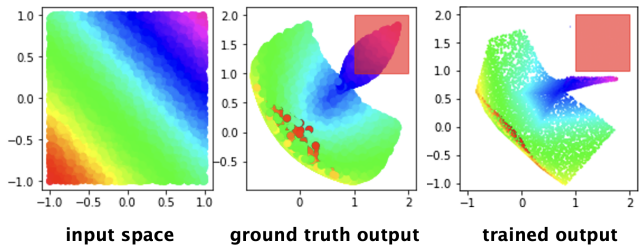


Fig. 1. An example safe training result with the proposed method. The color gradient illustrates corresponding input and output points. With our method, the trained output does not intersect the unsafe set (red box).

II. Preliminaries

We now introduce our notation for neural networks and define constrained zonotopes.

In this work, we consider a fully-connected, ReLU-activated feedforward neural network $\mathbf{n}(\cdot) : X^{(0)} \to \mathbb{R}^{n^{(d)}}$, with output $\mathbf{x}^{(d)} = \mathbf{n}\left(\mathbf{x}^{(0)}\right)$ given an input $\mathbf{x}^{(0)} \in X^{(0)} \subset \mathbb{R}^{n^{(0)}}$. We call $X^{(0)}$ the input set. We denote by $d \in \mathbb{N}$ the *depth* of the network and by $n^{(i)}$ the *width* of the i^{th} layer. For each layer $k = 1, \cdots, d-1$, the hidden state of the neural network is given by

$$\mathbf{x}^{(k)} = \rho\left(\mathcal{L}\left(\mathbf{x}^{(k-1)}, \mathbf{W}^{(k-1)}, \mathbf{w}^{(k-1)}\right)\right), \tag{1}$$

where $\mathbf{W}^{(k-1)} \in \mathbb{R}^{n^{(k-1)} \times n^{(k)}}$, $\mathbf{x}^{(k)}$ and $\mathbf{w}^{(k-1)} \in \mathbb{R}^{n^{(k)}}$, and

$$\mathcal{L}(\mathbf{x}, \mathbf{W}, \mathbf{w}) = \mathbf{W}\mathbf{x} + \mathbf{w}, \tag{2}$$

$$\rho(\mathbf{x}) = \max\{0, \mathbf{x}\}, \tag{3}$$

where $\mathcal{L}(\cdot)$ is a linear layer operation, and $\rho(\cdot)$ is the ReLU nonlinearity with the max taken elementwise. We do not apply the ReLU activation for the final output layer:

$$\mathbf{x}^{(d)} = \mathcal{L}\left(\mathbf{x}^{(d)}, \mathbf{W}^{(d)}, \mathbf{w}^{(d)}\right). \tag{4}$$

The *reachable set* of the neural network is

$$X^{(d)} = \mathbf{n}\left(X^{(0)}\right) \subset \mathbb{R}^{n^{(d)}}. \tag{5}$$

We represent the reachable set as a union of constrained zonotopes. A *constrained zonotope* $\mathcal{CZ}(\mathbf{c}, \mathbf{G}, \mathbf{A}, \mathbf{b}) \subset \mathbb{R}^n$ is a set parameterized by a center $\mathbf{c} \in \mathbb{R}^n$, generator matrix $\mathbf{G} \in \mathbb{R}^{n \times n_{\text{gen}}}$, linear constraints $\mathbf{A} \in \mathbb{R}^{n_{\text{con}} \times n_{\text{gen}}}$, $\mathbf{b} \in \mathbb{R}^{n_{\text{con}}}$, and coefficients $\mathbf{z} \in \mathbb{R}^{n_{\text{gen}}}$ as follows:

$$\mathcal{CZ}(\mathbf{c}, \mathbf{G}, \mathbf{A}, \mathbf{b}) = \{\mathbf{c} + \mathbf{G}\mathbf{z} \mid \|\mathbf{z}\|_\infty \leq 1,\ \mathbf{A}\mathbf{z} = \mathbf{b}\}. \tag{6}$$

Importantly, the intersection of constrained zonotopes is also a constrained zonotope [19, Proposition 1]. Let $Z_1 = \mathcal{CZ}(\mathbf{c}_1, \mathbf{G}_1, \mathbf{A}_1, \mathbf{b}_1)$ and $Z_2 = \mathcal{CZ}(\mathbf{c}_2, \mathbf{G}_2, \mathbf{A}_2, \mathbf{b}_2)$. Then $Z_1 \cap Z_2$ is given by

$$Z_1 \cap Z_2 = \mathcal{CZ}\left(\mathbf{c}_1, [\mathbf{G}_1, \mathbf{0}], \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \\ \mathbf{G}_1 & -\mathbf{G}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{c}_2 - \mathbf{c}_1 \end{bmatrix}\right). \tag{7}$$

We leverage this property to evaluate constraints on the forward reachable set of our neural network.

III. Method

In this section, we first explain how to pass a constrained zonotope exactly through a ReLU nonlinearity; that is, we compute the reachable set of a ReLU activation given a constrained zonotope as the input. We then discuss how to train a neural network using the reachable set to enforce constraints. Finally, we explain how to compute the gradient of the constraint for backpropagation.

Before proceeding, we briefly mention that we can pass an input constrained zonotope $Z = \mathcal{CZ}(\mathbf{c}, \mathbf{G}, \mathbf{A}, \mathbf{b})$ through a linear layer as

$$\mathcal{L}(Z, \mathbf{W}, \mathbf{w}) = \mathcal{CZ}(\mathbf{W}\mathbf{c} + \mathbf{w}, \mathbf{W}\mathbf{G}, \mathbf{A}, \mathbf{b}). \tag{8}$$

This follows from the definition in (6).

A. Constrained Zonotope ReLU Activation

Proposition 1. *The ReLU activation of a constrained zonotope $Z \subset \mathbb{R}^n$ is:*

$$\rho(Z) = \bigcup_{i=1}^{2^n} \mathcal{CZ}\left(\mathbf{c}^{(i)}, \mathbf{G}^{(i)}, \mathbf{A}^{(i)}, \mathbf{b}^{(i)}\right), \tag{9}$$

where each output constrained zonotopes is given by:

$$\mathbf{G}^{(i)} = [\operatorname{diag}(\mathbf{u}_i)\,\mathbf{G},\ \mathbf{0}_{n \times n}], \tag{10a}$$

$$\mathbf{A}^{(i)} = \begin{bmatrix} \mathbf{A} & \mathbf{0}_{n_{\text{con}} \times n} \\ \operatorname{diag}(\mathbf{1}_{n \times 1} - 2\mathbf{u}_i)\,\mathbf{G} & \operatorname{diag}\left(\mathbf{d}^{(i)}\right) \end{bmatrix}, \tag{10b}$$

$$\mathbf{c}^{(i)} = \operatorname{diag}(\mathbf{u}_i)\,\mathbf{c}, \tag{10c}$$

$$\mathbf{b}^{(i)} = \begin{bmatrix} \mathbf{b} \\ -\operatorname{diag}(\mathbf{1}_{n \times 1} - 2\mathbf{u}_i)\,\mathbf{c} - \mathbf{d}^{(i)} \end{bmatrix}, \text{and} \tag{10d}$$

$$\mathbf{d}^{(i)} = \tfrac{1}{2}\left(\mathbf{G}_+ \mathbf{1}_{n_{\text{gen}} \times 1} - \operatorname{diag}(\mathbf{1}_{n \times 1} - 2\mathbf{u}_i)\,\mathbf{c}\right), \tag{10e}$$

where $\mathbf{G}_+ \in \mathbb{R}^{n \times n_{\text{gen}}}$ is a matrix containing the elementwise absolute value of $\mathbf{G}$ and $\mathbf{u}_i \in \mathbb{R}^{n \times 1}$ is the i^{th} combination of the 2^n possible n-tuples defined over the set $\{0, 1\}^n$.

Proof. The formulation in (10) follows from treating the max operation applied to all negative elements of the input zonotope as a sequence of two operations. *First*, we intersect the input constrained zonotope with the halfspace defined by the vector $\mathbf{u}_i$ in the codomain of $\rho(\cdot)$; this is why the linear operator $\operatorname{diag}(\mathbf{u}_i)$ is applied to each $\mathbf{G}^{(i)}$ and $\mathbf{c}^{(i)}$, as given by the analytical intersection of a constrained zonotope with a halfspace [20, Eq. 10]. *Second*, we zero out the dimension corresponding to that halfspace/unit vector (i.e., project all negative points to zero). Since the max is taken elementwise, there are 2^n possible intersection/zeroings when considering each dimension as either activated or not. □

Proposition 1 is illustrated in Fig. 2.

Per Proposition 1, passing a constrained zonotope through a ReLU nonlinearity produces a set of 2^n constrained zonotopes. A similar phenomenon is found in ReLU activations of other set representations [10], with exponential growth in the computational time and memory required as a function of layer width and number of layers. To mitigate this growth, empty constrained zonotopes can be pruned after each activation, hence our next discussion.

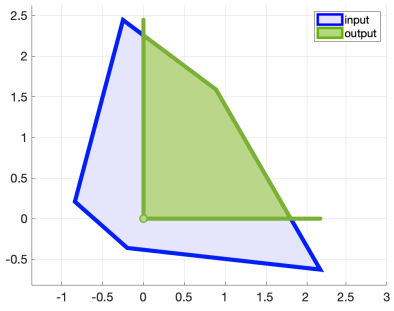


Fig. 2. An illustration of passing a constrained zonotope (blue) through a 2-D ReLU nonlinearity, resulting in the green output set, which is the union of 4 constrained zonotopes.

B. Constrained Zonotope Emptiness Check

To check if $Z = \mathcal{CZ}(\mathbf{c},\mathbf{G},\mathbf{A},\mathbf{b})$ is empty, we solve a linear program (LP) [19, Proposition 2]:

$$\min_{\mathbf{z},v} \{v \mid \mathbf{Az} = \mathbf{b} \text{ and } \|\mathbf{z}\|_\infty \leq v\}. \tag{11}$$

Then, Z is empty if and only if $v > 1$. Importantly, by construction, as long as there exist feasible $\mathbf{z}$ (for which $\mathbf{Az} = \mathbf{b}$), then (III-B) is *always* feasible. Since the intersection of constrained zonotopes is also a constrained zonotope as in (7), we can use this emptiness check to enforce collision-avoidance (i.e., non-intersection) constraints. This is the basis of our constrained training method.

C. Constrained Neural Network Training

The main goal of this paper is constrained neural network training. For robotics in particular, as future work, our goal is to train a robust controller. In this work, we consider an unsafe output set which could represent, e.g., actuator limits or obstacles in a robot's workspace (in which case the output of the neural network is passed through a robot's dynamics).

1) Generic Formulation: Consider an input set $X^{(0)} \subset \mathbb{R}^{n^{(0)}}$ represented by a constrained zonotope, an unsafe set $X_{\text{unsf}} \subset \mathbb{R}^{n^{(d)}}$, a training dataset $(\mathbf{x}_j^{(0)}, \mathbf{y}_j)$, $j = 1,\cdots,m$, $\mathbf{y}_j \in \mathbb{R}^{n^{(d)}}$ of training examples $\mathbf{x}_j^{(0)}$ and labels $\mathbf{y}_j$, and an objective loss function $\ell_{\text{obj}}\left(\mathbf{x}_1^{(0)},\cdots,\mathbf{x}_m^{(0)},\mathbf{y}_1,\cdots,\mathbf{y}_m\right)$. Let $\mathcal{W} = \{\mathbf{W}^{(0)},\cdots,\mathbf{W}^{(d)}\}$ be the collection of all of the neural network weights and $\mathcal{B} = \{\mathbf{w}^{(0)},\cdots,\mathbf{w}^{(d)}\}$ all the biases. We formulate the training problem as:

$$\min_{\mathcal{W},\mathcal{B}} \quad \ell_{\text{obj}}\left(\mathbf{x}_1^{(0)},\cdots,\mathbf{x}_m^{(0)},\mathbf{y}_1,\cdots,\mathbf{y}_m\right), \tag{12a}$$

$$\text{s.t.} \quad X^{(d)} \cap X_{\text{unsf}} = \emptyset, \tag{12b}$$

where $X^{(d)}$ is the reachable set as in (5). We write the loss as a function of all of the input/output data (as opposed to batching the data) for ease of presentation.

2) Set and Constraint Representations: We represent the input set and unsafe set as constrained zonotopes, $X^{(0)} = Z^{(0)}$ and $X_{\text{unsf}} = Z_{\text{unsf}}$. Similarly, it follows from Proposition 1 that the output set $X^{(d)}$ can be exactly represented as a union of constrained zonotopes:

$$X^{(d)} = \bigcup_{i=1}^{n_{\text{out}}} Z_i^{(d)}, \tag{13}$$

where n_{out} depends on the layer widths and network depth.

Recall that $Z_i^{(d)} \cap Z_{\text{unsf}}$ is a constrained zonotope as in (7). So, to compute the constraint loss, we evaluate $X^{(d)} \cap X_{\text{unsf}}$ by solving (III-B) for each constrained zonotope $Z_i^{(d)} \cap Z_{\text{unsf}}$ with $i = 1,\cdots$. Then, denoting v_i^* as the output of (III-B) for each $Z_i^{(d)} \cap Z_{\text{unsf}}$, we represent the constraint $X^{(d)} \cap X_{\text{unsf}} = \emptyset$ as a function $\ell_{\text{con}}(\cdot)$ for which

$$\ell_{\text{con}}\left(Z_i^{(d)}, Z_{\text{unsf}}\right) = 1 - v_i^*, \tag{14}$$

which is negative when feasible as is standard in constrained optimization [21]. Using (14), we ensure the neural network obeys constraints by checking $\ell_{\text{con}}\left(Z_i^{(d)}, Z_{\text{unsf}}\right) < 0$ for each $i = 1,\cdots,n_{\text{out}}$.

D. Differentiating the Collision Check Loss

To train using backpropagation, we must differentiate the constraint loss $\ell_{\text{con}}(\cdot)$. This means we must compute the gradient of (III-B) with respect to the problem parameters $\mathbf{A}$ and $\mathbf{b}$, which are defined by the centers, generators, and constraints of the output constrained zonotope set. To do so, we leverage techniques from [22], which can be applied because (III-B) is always feasible.

Consider the Lagrangian of (III-B):

$$J(\mathbf{z},v,\mathbf{m},\mathbf{n}) = q^\top(\mathbf{z},v) + \mathbf{A}^\top\mathbf{n} + \mathbf{m}^\top(\mathbf{G}(\mathbf{z},v) - \mathbf{g}), \tag{15}$$

where $\mathbf{m}$ is the dual variable for the inequality constraint and $\mathbf{n}$ is the dual variable for the equality constraint. For any optimizer $(\mathbf{z}^*,v^*,\mathbf{m}^*,\mathbf{n}^*)$, the optimality conditions are

$$\mathbf{q}^\top(\mathbf{z}^*,v^*) + \mathbf{A}^\top\mathbf{n}^* + \mathbf{G}^\top\mathbf{m}^* = 0, \tag{16a}$$

$$\mathbf{A}(\mathbf{z}^*,v^*)^* - \mathbf{b} = 0, \text{ and} \tag{16b}$$

$$\text{diag}(\mathbf{m}^*)\,\mathbf{G}(\mathbf{z}^*,v^*)^* = 0, \tag{16c}$$

where we have used the fact that $\mathbf{g} = \mathbf{0}$. Taking the differential (denoted by d) of (16), we get

$$\begin{bmatrix} \mathbf{0} & \mathbf{G}^\top & \mathbf{A}^\top \\ \text{diag}(\mathbf{m}^*) & \text{diag}(\mathbf{G}(\mathbf{z}^*,v^*)^*) & \mathbf{0} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} d(\mathbf{z}^*,v^*) \\ d\mathbf{m} \\ d\mathbf{n} \end{bmatrix} = \begin{bmatrix} -d\mathbf{q} - d\mathbf{A}^\top\mathbf{n}^* \\ -\text{diag}(\mathbf{m}^*)\,d\mathbf{G}(\mathbf{z}^*,v^*) \\ -d\mathbf{A}(\mathbf{z}^*,v^*)^* + d\mathbf{b} \end{bmatrix}, \tag{17}$$

We can then solve (17) for the Jacobian of v^* with respect to any entry of the zonotope centers or generators by setting the right-hand side appropriately (see [22] for details). That is, we can now differentiate (14) with respect to the elements of $\mathbf{c}_1$, $\mathbf{G}_1$, $\mathbf{c}_2$, or $\mathbf{G}_2$. In practice, we differentiate (III-B) automatically using the `cvxpylayers` library [23].

IV. Numerical Example

We test our method by training a 2-layer feedforward ReLU network with input dimension 2, hidden layer size of 10, and output dimension of 2. We chose this network with only one ReLU nonlinearity layer, as recent results have shown that a shallow ReLU network performs similarly to a deep ReLU network with the same amount of neurons [24]. However, note that our method (in particular Proposition 1) does generalize to deeper networks. We pose this preliminary example as a first effort towards this novel style of training.

Problem Setup. We seek to approximate the function

$$f(x_1, x_2) = \begin{bmatrix} x_1^2 + \sin x_2 \\ x_2^2 + \sin x_1 \end{bmatrix}, \tag{18}$$

with $X^{(0)} = \mathcal{CZ}(\mathbf{0}_{2\times 1}, \mathbf{I}_2, \emptyset, \emptyset)$. We create an unsafe set in the output space as

$$X_{\text{unsf}} = \mathcal{CZ}\left(\begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix}, 0.5\mathbf{I}_2, \emptyset, \emptyset \right). \tag{19}$$

The training dataset was generated as $n_{\text{data}} = 10^4$ random input/output pairs $(\mathbf{x}_j^{(0)}, \mathbf{y}_j)$ with $\mathbf{y}_j = f(\mathbf{x}_j^{(0)})$ by sampling uniformly in $X^{(0)}$. The unsafe set X_{unsf} and each $\mathbf{x}_j^{(0)}$ and $\mathbf{y}_j$ are plotted in Figs. 3 and 4. The objective loss is

$$\ell_{\text{obj}}(\cdot) = \frac{1}{n_{\text{data}}} \sum_{j=1}^{n_{\text{data}}} \left\| \mathbf{n}\left(\mathbf{x}_j^{(0)}\right) - \mathbf{y}_j \right\|_2^2 \tag{20}$$

where we use $\cdot$ for concision in place of the training data.

Implementation. We implemented our method[1] in PyTorch [25] with `optim.SGD` as our optimizer on a desktop computer with 6 cores, 32 GB RAM, and an RTX 2060 GPU.

We trained the network for 10^3 iterations with and without the constraints enforced. To enforce hard constraints as in (12), in each iteration, we compute the objective loss function across the entire dataset, then backpropagate the objective gradient; then, we compute the constraint loss as in (14) for all active constraints, then backpropagate. For future work we will apply more sophisticated constrained optimization techniques (e.g., an active set method) [21, Ch. 15].

With our current naïve implementation, constrained training took approximately 5 hours, whereas the unconstrained training took 0.5 s. Our method is slower due to the need to compute an exponentially-growing number of constrained zonotopes as in Proposition 1. However, we notice that the GPU utilization is only 1-5% (the reachability propagation is not fully parallelized), indicating significant room for increased parallelization and speed.

Results and Discussion. Results for unconstrained and constrained training are shown in Fig. 3 and Fig. 4. Our proposed method avoids the unsafe set. Note the output constrained zonotopes (computed for both networks) contain the colored output points, verifying our exact set representation in Proposition 1.

Table I shows results for unconstrained and constrained training; importantly, our method obeys the constraints. As

[1]Our code is available online: https://github.com/Stanford-NavLab/constrained-nn-training

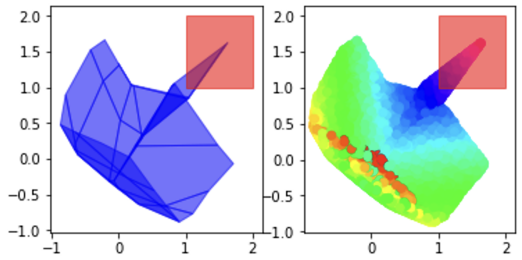


Fig. 3. Unconstrained training, with $X^{(d)}$ plotted on the left and each $\mathbf{n}\left(\mathbf{x}_j^{(0)}\right)$ plotted on the right. The output approximates the function well but does not avoid the unsafe space.

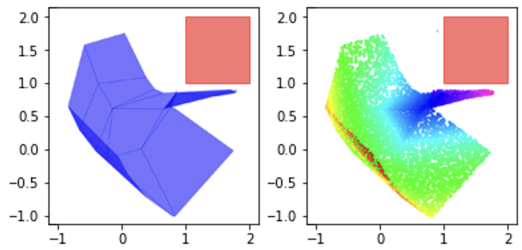


Fig. 4. Constrained training. The output approximates the function while avoiding the unsafe space.

expected for nonlinear constrained optimization, the network converged to a local minimum while obeying the constraints. The key challenge is that the constrained training is several orders of magnitude slower than unconstrained training. We plan to address in future work by increased parallelization, by pruning of our reachable sets [6], and by using anytime verification techniques [12].

	Unconstrained	Constrained
final objective loss	**0.0039**	0.0127
final constraint loss	0.0575	**0.0000**

TABLE I

V. Conclusion and Future Work

This work proposes a constrained training method for feedforward ReLU neural networks. We demonstrated the method successfully on a small example of nonlinear function approximation. Given the ability to enforce output constraints, the technique can potentially be applied to offline training for safety-critical neural networks.

Our current implementation has several drawbacks to be addressed in future work. First, the method suffers an exponential blowup of constrained zonotopes through a ReLU. We hope to improve the forward pass step by using techniques such as [12] instead of layer-by-layer evaluation to compute the output set, and by conservatively estimating the reachable set similar to [13]. We also plan to apply the method on larger networks, such as for autonomous driving in [6] or the ACAS Xu network [26]–[28] for aircraft collision avoidance. In general, our goal is to train robust controllers where the output of a neural network must obey actuator limits and obstacle avoidance (for which the network output is passed through dynamics).

REFERENCES

[1] G. Xiong, J. Gong, T. Zhuang, T. Zhao, D. Liu, and X. Chen, "Development of assistant robot with standing-up devices for paraplegic patients and elderly people," in *2007 IEEE/ICME International Conference on Complex Medical Engineering*, IEEE, 2007, pp. 62–67.

[2] B. Ko, H.-J. Choi, C. Hong, J.-H. Kim, O. C. Kwon, and C. D. Yoo, "Neural network-based autonomous navigation for a homecare mobile robot," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, IEEE, 2017, pp. 403–406.

[3] G. Xu and A. Song, "Adaptive impedance control based on dynamic recurrent fuzzy neural network for upper-limb rehabilitation robot," in *2009 IEEE International Conference on Control and Automation*, IEEE, 2009, pp. 1376–1381.

[4] S. Hussain, S. Q. Xie, and P. K. Jamwal, "Adaptive impedance control of a robotic orthosis for gait rehabilitation," *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 1025–1034, 2013.

[5] E. Gribovskaya, A. Kheddar, and A. Billard, "Motion learning and adaptive impedance for robot control during physical interaction with humans," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 4326–4332.

[6] H.-D. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *International Conference on Computer Aided Verification*, Springer, 2020, pp. 3–17.

[7] L. Shengbo, G. Yang, H. Lian, G. Hongbo, D. Jingliang, L. Shuang, W. Yu, C. Bo, L. Keqiang, R. Wei, *et al.*, "Key technique of deep neural network and its applications in autonomous driving," *Journal of Automotive Safety and Energy*, vol. 10, no. 2, p. 119, 2019.

[8] C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, and M. J. Kochenderfer, "Algorithms for verifying deep neural networks," *arXiv preprint arXiv:1903.06758*, 2019.

[9] H.-D. Tran, D. M. Lopez, P. Musau, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Star-based reachability analysis of deep neural networks," in *International Symposium on Formal Methods*, Springer, 2019, pp. 670–686.

[10] H.-D. Tran, S. Bak, W. Xiang, and T. T. Johnson, "Verification of deep convolutional neural networks using imagestars," in *International Conference on Computer Aided Verification*, Springer, 2020, pp. 18–42.

[11] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Ph.D. dissertation, Technische Universität München, 2010.

[12] J. A. Vincent and M. Schwager, "Reachable Polyhedral Marching (RPM): A Safety Verification Algorithm for Robotic Systems with Deep Neural Network Components," *arXiv preprint arXiv:2011.11609*, 2020.

[13] V. Rubies-Royo, R. Calandra, D. M. Stipanovic, and C. Tomlin, "Fast neural network verification via shadow prices," *arXiv preprint arXiv:1902.07247*, 2019.

[14] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[15] R. Stewart and S. Ermon, "Label-free supervision of neural networks with physics and domain knowledge," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[16] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Broeck, "A semantic loss function for deep learning with symbolic knowledge," in *International Conference on Machine Learning*, PMLR, 2018, pp. 5502–5511.

[17] U. S. Cruz, J. Ferlez, and Y. Shoukry, "Safe-by-Repair: A Convex Optimization Approach for Repairing Unsafe Two-Level Lattice Neural Network Controllers," *arXiv preprint arXiv:2104.02788*, 2021.

[18] L. Markolf and O. Stursberg, "Polytopic Input Constraints in Learning-Based Optimal Control Using Neural Networks," *arXiv preprint arXiv:2105.03376*, 2021.

[19] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, "Constrained zonotopes: A new tool for set-based estimation and fault detection," *Automatica*, vol. 69, pp. 126–136, 2016.

[20] V. Raghuraman and J. P. Koeln, "Set operations and order reductions for constrained zonotopes," *arXiv preprint arXiv:2009.06039*, 2020.

[21] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[22] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*, PMLR, 2017, pp. 136–145.

[23] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable convex optimization layers," *arXiv preprint arXiv:1910.12430*, 2019.

[24] B. Hanin and D. Rolnick, "Deep relu networks have surprisingly few activation patterns," *arXiv preprint arXiv:1906.00904*, 2019.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[26] M. J. Kochenderfer and J. Chryssanthacopoulos, "Robust airborne collision avoidance through dynamic programming," *Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371*, vol. 130, 2011.

[27] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next-generation airborne collision avoidance system," Massachusetts Institute of Technology-Lincoln Laboratory Lexington United States, Tech. Rep., 2012.

[28] M. J. Kochenderfer, C. Amato, G. Chowdhary, J. P. How, H. J. D. Reynolds, J. R. Thornton, P. A. Torres-Carrasquillo, N. K. Ure, and J. Vian, "Optimized airborne collision avoidance," *Decision Making under Uncertainty*, p. 251, 2015.

Reactive Synthesis for Human-aware Robotic Manipulation using Regret Games

Karan Muvvala and Morteza Lahijanian
University of Colorado Boulder
Email: karan.muvvala@colorado.edu, morteza.lahijanian@colordo.edu

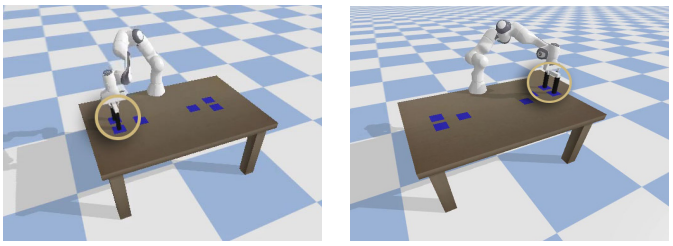
***Abstract*—This paper proposes a regret-based reactive synthesis framework for robotic manipulators with complex tasks operating in the presence of a human. The framework uses temporal logic for task specification and generates strategies that are guaranteed to complete the task while exploring possible cooperation with the human. The method is based on a two-player game formulation and uses the notion of *regret* to reason about the players' actions. This approach relaxes the conservative assumption that the human is always adversarial and allows exploration for collaboration. We discuss the efficacy of this framework through case studies.**

I. Introduction

From factories to households, robots are rapidly leaving behind their robot-centric environments and entering our society. To be successful in our human-centric world, they must develop the ability to interact with dynamic environments. This includes performing complex tasks in presence of humans, who may interfere with the task, and seeking collaboration when possible. Achieving such capabilities is challenging, especially in the robotic manipulation domain, where tasks are complex and crucially require interaction with humans. This work addresses some of these challenges by providing a reactive synthesis framework that guarantees task completion and at the same time enables efficient collaboration.

As an example, consider the scenario in Fig. 1, where the robot is tasked with building an arch either on the left or the right side of the table with the black boxes as the supports and the white box on top. In this workspace, there is a human (not shown), who can reach and manipulate the boxes placed on the right side but not the ones on the left. To operate in the region on the left, the robot has to spend more energy than the one closer to the human. To solve this problem, classical planning methods that compute a fixed sequence of actions are not sufficient. Instead, the robot needs a plan (strategy) that chooses motions in reaction to the actions of the human.

Previous work [4, 5, 6] addresses this problem through a *reactive synthesis* approach [7, 8, 10]. Their framework uses *Linear Temporal Logic over finite traces* (LTLf) [1] for task specification and views the interaction between the human and the robot as a game. The framework first constructs a discrete abstraction of the continuous planning problem in the form of a turn-based two-player game and then synthesizes a winning strategy on this abstraction for the robot to achieve the task by assuming the human to be purely adversarial. While this approach guarantees task completion, the adversarial assumption is conservative and eliminates the possibility of collaboration with the human, which may lead to higher energy spending by the robot. In the scenario in Fig. 1, the strategy obtained by this framework leads the robot to build an arch away from the human and spending more energy as shown in Fig. 1a.



(a) Adversarial Behavior (b) Probabilistic Behavior

Fig. 1: Arch building: Adversarial vs Probabilistic human

Recent work [9] relaxes this assumption by modelling the human as a probabilistic agent. This leads to an abstraction in the form of a Markov Decision Process (MDP) and the objective reduces to synthesize an optimal policy that maximizes the probability of satisfying the specification on this MDP. This approach optimizes the robot's actions according to the expected behavior of the human instead of assuming the human to be adversarial. In the scenario in Fig. 1, the policy obtained by this method leads the robot to build the arch near the human (as shown in Fig. 1b) if the expected behavior of the human is to be cooperative; otherwise, the robot builds the arch away from the human. While cooperation can be achieved using this probabilistic framework, it requires prior knowledge on the human, which is generally hard to obtain. Also, the method fails to capture the human as a strategic player.

In this work, we propose a framework that relaxes the assumption that the human is purely adversarial while also capturing the human's objective as a strategic agent without requiring *a priori* knowledge. Similar to [5], we model the interaction as a two-player game, but we use the notion of regret to reason about the human's actions. Intuitively, regret can be defined as a measure of how good an action is. Thus, the objective for the robot is to choose an action (strategy) that *minimizes its regret*.

II. Problem Formulation and Approach

Problem Formulation: Similar to [5, 6], we use a discrete abstraction of a robotic manipulator and a human operating in a shared workspace as a two-player game. This *game abstraction of a manipulation domain* is a tuple a tuple

$\mathcal{G} = (V, v_0, A_s, A_e, F, \delta, \Pi, L)$ where $V = V_s \cup V_e$ is the set of states partitioned into robot states V_s and human states V_e, A_s and A_e are the sets of finite actions for the robot and human, respectively, $F : A_s \to \mathbb{R}_{\geq 0}$ is the cost function that maps each robot action to the energy required for that action, $\delta : V \times (A_s \cup A_e) \to V$ is the transition function, Π is a set of task related atomic propositions, and $L : V \to 2^{\Pi}$ is a labeling function that indicates the property of each state relative to the task. To evolve over this game, each player picks a strategy, which chooses the next action given the sequence of states taken so far. We denote by $\sigma : V^* \cdot V_s \to A_s$ and $\tau : V^* \cdot V_e \to A_e$ the strategies associated with the robot and the human respectively.

We employ LTLf (see [1] for syntax and semantics) to express the task of the robot. It is an expressive language that combines Boolean connectives with temporal operators, allowing the expression of complex tasks such as the one in Fig. 1. An LTLf formula is defined over Π and can be translated to a *deterministic finite automaton* (DFA) with alphabet 2^{Π} using existing tools [2].

Problem: Given abstraction $\mathcal{G}$, LTLf task specification ϕ, and a user-defined energy budget $\mathcal{B}$, compute a strategy σ^* for the robot that not only guarantees completion of task ϕ but also *explores possible cooperation* with the human while keeping the total energy consumption less than or equal to $\mathcal{B}$.

Approach: Our proposed approach first converts the LTLf formula ϕ to a DFA A_ϕ and then composes it with the two-player game $\mathcal{G}$ to construct the DFA game $\mathcal{P} = A_\phi \times \mathcal{G}$ (see [5, 6] for details). The DFA A_ϕ only reasons over the task ϕ, and the abstraction $\mathcal{G}$ captures the constraints of the physical world. By composing the two structures, we obtain a new game $\mathcal{P} = (S, S_f, s_0, A_s, A_e, F, \delta_{\mathcal{P}})$ that captures all the possible ways in which ϕ can be accomplished for the manipulation domain captured by $\mathcal{G}$. Here, S is the set of states, which represent the current configuration of the physical world (V) as well as how much of the task ϕ has been accomplished, $s_0 \in S$ is the initial state, $S_f \subseteq S$ is the set of accepting states that correspond to the task being successfully accomplished, $\delta_{\mathcal{P}}$ is the transition function, and A_s, A_e, and F are inherited from $\mathcal{G}$. Thus, the objective of our problem reduces to synthesizing a strategy σ^* for the robot on $\mathcal{P}$ that assures *reachability* of S_f under *all* strategies of the human. We call σ^* a *winning strategy* on $\mathcal{P}$ if, not only it guarantees reaching a state in S_f, but also ensures that the robot does not spend more than $\mathcal{B}$ amount of energy under all human strategies.

A winning strategy σ^* can be computed by assuming the human is always adversarial [5], which is a conservative approach as discussed in Sec. I. To relax this assumption, we use the notion of regret as a quality measure. Informally, regret can be defined as the difference between what the robot actually spends and what it could have spent if it had known how the human would react beforehand. Mathematically,

$$\mathrm{reg}^{\sigma,\tau}(s) := \mathrm{Val}^s(\sigma, \tau) - \min_{\sigma'} \mathrm{Val}^s(\sigma', \tau), \tag{1}$$

where $\mathrm{Val}^s(\sigma, \tau)$ is the total energy (cumulative payoff) associated with the run (path) of $\mathcal{P}$ induced by strategies σ and τ starting from state s. We use *best-responses* - $\min_{\sigma'} \mathrm{Val}^s(\sigma', \tau)$ as yardstick to compare the quality of each robot action for a fixed response τ by the human. To also ensure satisfaction of ϕ, we specifically design Val so that it returns infinity if the induced run does not end in S_f. Hence, our goal is now to compute a regret minimizing strategy

$$\sigma^* = \arg\min_{\sigma}(\max_{\tau} \mathrm{reg}^{\sigma,\tau}(s_0)). \tag{2}$$

To synthesize winning strategy σ^*, we employ a two-step process based on [3]. First, we unroll $\mathcal{P}$ to construct a graph of utility G^u, which in essence captures all the possible scenarios that can be accomplished by the human and the robot. We then augment the nodes in G^u with the the total energy spent by the robot in executing that strategy σ for a fixed human strategy τ. We then compute the best-response for each path induced by σ and τ to construct the graph of best-responses G^{br}. Finally, we compute the regret $\mathrm{reg}^{\sigma,\tau}(s_0)$ associated with each path and employ a value iteration algorithm to compute the optimal regret minimizing winning strategy σ^*. This algorithm is pseudo-polynomial in the size of the graph $\mathcal{P}$ and largest energy cost in $\mathcal{P}$.

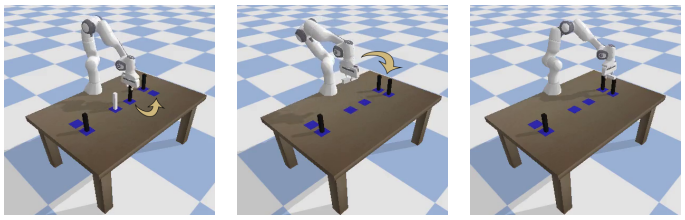


Fig. 2: The regret minimizing strategy for the robot is to give the human a chance to be cooperative.

III. Case Study

We illustrate the efficacy of our framework on the arch building example in Fig. 1. Under the regret-minimizing strategy, the robot starts to build the arch near the human as shown in Fig. 2. If the human does not intervene, the robot completes the arch by spending a small amount of energy. However, if the human does intervene adversarially, based on the energy budget provided by the user, the robot gives the human more opportunities to be collaborative. As soon as it reaches close to the energy budget and the human still refuses to cooperate, it switches to a conservative behavior and builds the arch away from the human. In all our case studies, which we could not present here due to space constraints, we observe this cooperation-seeking behavior. Thus, using regret, we can model the human as a strategic agent with its own objectives that may not be necessarily adversarial, allowing robot to explore cooperation.

IV. Conclusion

In this work, we propose a regret-based reactive synthesis framework to synthesize a regret minimizing strategy for a robotic manipulator operating in presence of a human. Our approach relaxes the adversarial assumption and allows the robot to explore possible collaboration with the human while spending no more than $\mathcal{B}$ amount of energy.

References

[1] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 854–860. AAAI Press, 2013. ISBN 9781577356332.

[2] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, October 2016. doi: 10.1007/978-3-319-46520-3_8.

[3] Emmanuel Filiot, Tristan Le Gall, and Jean-François Raskin. Iterated regret minimization in game graphs. In *International Symposium on Mathematical Foundations of Computer Science*, pages 342–354. Springer, 2010.

[4] Keliang He, Morteza Lahijanian, Lydia E Kavraki, and Moshe Y Vardi. Towards manipulation planning with temporal logic specifications. In *IEEE international conference on robotics and automation*, pages 346–352. IEEE, 2015.

[5] Keliang He, Morteza Lahijanian, Lydia E Kavraki, and Moshe Y Vardi. Reactive synthesis for finite tasks under resource constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5326–5332. IEEE, 2017.

[6] Keliang He, Morteza Lahijanian, Lydia E. Kavraki, and Moshe Y. Vardi. Automated abstraction of manipulation domains for cost-based reactive synthesis. *IEEE Robotics and Automation Letters*, 4(2):285–292, 2019. doi: 10.1109/LRA.2018.2889191.

[7] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6): 1370–1381, 2009. doi: 10.1109/TRO.2009.2030225.

[8] Cristian Ioan Vasile and Calin Belta. Reactive sampling-based temporal logic path planning. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4310–4315, 2014. doi: 10.1109/ICRA.2014.6907486.

[9] Andrew M. Wells, Zachary Kingston, Morteza Lahijanian, Lydia E Kavraki, and Moshe Y. Vardi. Finite-horizon synthesis for probabilistic manipulation domains. In *IEEE International Conference on Robotics and Automation*. IEEE, 2021. (to appear).

[10] Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. Efficient reactive controller synthesis for a fragment of linear temporal logic. In *2013 IEEE International Conference on Robotics and Automation*, pages 5033–5040, 2013. doi: 10.1109/ICRA.2013.6631296.

DULA: A Differentiable Ergonomics Model for Postural Optimization in Physical HRI

Amir Yazdani*, Roya Sabbagh Novin*, Andrew Merryweather*, Tucker Hermans*†
*University of Utah Robotics Center, Salt Lake City, UT, †NVIDIA, Seattle, WA
Email: amir.yazdani@utah.edu

***Abstract*—Ergonomics and human comfort are essential concerns in physical human-robot interaction applications. Defining an accurate and easy-to-use ergonomic assessment model stands as an important step in providing feedback for postural correction to improve operator health and comfort. In order to enable efficient computation, previously proposed automated ergonomic assessment and correction tools make approximations or simplifications to gold-standard assessment tools used by ergonomists in practice. In order to retain assessment quality, while improving computational considerations, we introduce DULA, a differentiable and continuous ergonomics model learned to replicate the popular and scientifically validated RULA assessment. We show that DULA provides assessment comparable to RULA while providing computational benefits. We highlight DULA's strength in a demonstration of gradient-based postural optimization for a simulated teleoperation task.**

I. Introduction

Autonomous Postural Optimization has received substantial attention in research with the new technologies around humans such as collaborative robots [18], smart personal trainers [4], and VR systems [8]. In these systems, the interacting agent should consider human comfort and ergonomics in its behaviour and motion planning [24]. For example, in a physical human-robot interaction (pHRI) application such as co-manipulation, one of the objectives in motion planning of the collaborative robot must be satisfying ergonomic safety for the human.

Developing a model of human comfort lies at the heart of effective postural optimization. pHRI researchers have proposed several computational models for assessing ergonomics and human comfort in terms of peripersonal space [3], muscle fatigue [17], and joint overloading [13]. In contrast ergonomists have provided simpler models which are easier for human experts to calculate by hand and as such are more common in practice. These models include the NASA TLX [6], RULA [15], REBA [7], strain index [22] and ACGIH TLV [11]. Importantly these models are supported by extensive human subject studies that validate their effectiveness on reducing ergonomic risk factors [9, 12].

Among all risk assessment tools, RULA and REBA depend most on human posture and provide quantitative scores, making them good choices for postural optimization applications. However, the discrete scores and the presence of plateaus in RULA and REBA [1] create challenges when using them in gradient-based postural optimization. Based on our experience, using the risk assessment models directly in gradient-free optimization is time-expensive and the plateaus often prevent progress toward the global optimal solution in postural optimization. Thus, researchers in pHRI often use approximations of ergonomic assessment models in gradient-based postural optimizations; quadratic approximations [20, 1, 2] being the standard approach in the literature. However, these approximations deviate far from the scientifically validated assessments, causing doubt that they can reliably provide the same level of ergonomic benefit.

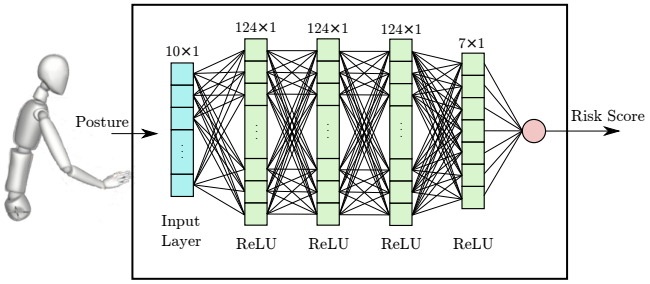


Fig. 1: The structure of the DULA neural network.

To overcome these issues, we introduce the *Differentiable Upper Limb Assessment* (DULA), a differentiable and continuous risk assessment model that is learned using a neural network to replicate the popular RULA survey tool (Fig. 1). Instead of discrete scores from 1 to 7, DULA reports the risk score as a continuous real number from 1 to 7. Furthermore, it provides the gradient of the risk with respect to each joint enabling efficient use in optimization. We compare the prediction of risk scores of DULA with RULA and provide an open-source package demonstrating how to use DULA in a gradient-based postural optimization in a simulated teleoperation.

II. Related Work

Although human postural optimization has received significant attention in pHRI, only a few studies [20, 19] investigate postural improvement in teleoperation. Table I summarizes the relevant literature for postural optimization in both areas.

Researchers have examined ergonomics and postural optimization in three different types of physical human-robot interaction scenarios: (1) *Assistive Holding* (e.g. [14, 21, 3]), (2) *object handover* (e.g. [1, 2, 17]), and (3) *Co-Manipulation* (e.g. [23, 18, 17, 13]). In the first two tasks the postural optimization provides optimized posture for the human and joint configurations for the robot while maintaining contact with the object at the interaction interface. For the case of co-

Ergonomics Model		Analytical Models	Learned Models	Risk Assessment Tools	
Approximation Method				Quadratic	No Approximation
Optimization	gradient-based	■ Peternel 2018 [18] ■ ■ Peternel 2017 [17] □ Chen 2018 [3] ◆ Peternel 2020 [19] ■ Kim 2017 [13]		◇ Rahal 2020 [20] ■ Busch 2017 [1] ■ Busch 2018 [2]	
	gradient-free		□ Marin 2018 [14]		■ van der Spaa 2020 [23]
Non-optimization					□ Shafti et al. [21]
Legend	□ physical HRI: Assistive Holding, ■ physical HRI: Handover, ■ physical HRI: Co-Manipulation ◆ Teleop: Goal-Constrained with Repositioning Postural Correction, ◇ Teleop: Goal-Constrained with Online Postural Correction				

TABLE I: State of the art of postural optimization in pHRI and teleoperation.

manipulation, the postural optimization outputs a trajectory of optimal postures for the human and an optimal joint-space trajectory for the robot to perform the co-manipulation task.

Ergonomic assessment models provide the primary cost in postural optimization objectives. pHRI researchers have proposed many computational models to assess ergonomics and human comfort of users including peripersonal space [3], muscle fatigue [17], and joint overloading [13]. To make the optimization simpler, Marin et al. suggested the idea of a contextual ergonomics model which is a set of Gaussian process models including joint angles, moments, reaction, load and muscle activation, trained with the musculoskeletal simulation task contexts [14]. Using Gaussian process models enables search in a 2D latent space while their cost function is defined in the high-dimensional musculoskeletal space.

Some literature use approximations of the ergonomics risk assessment tools. Busch et al. proposed a differentiable surrogate of the REBA score by fitting a weighted combination of quadratic functions plus a constant for the task payload [2]. Rahal et al. suggested a quadratic approximation for RULA which is the summation of the quadratic norm of the deviation from the human neutral posture for shoulder, elbow and wrist joints angles [20]. Their approximation conceptually agrees with the qualitative idea behind RULA in which the risk score goes higher when the human deviates more from the neutral posture. Van der Spaa et al. provide the only study of directly using risk assessment tools in derivative-free postural optimization [23]. They add the REBA score to the transition cost function in an A* optimization for task and motion planning of a robot in a co-manipulation task.

III. Differentiable Human Ergonomics Model

To build a differentiable RULA, we developed a dataset of 7.5 million upper body postures of a human model consisting of 3 joints in the torso and 7 joints in the arm. We additionally define task parameters based on the RULA worksheet—the frequency of the arm and body motions; type and maximum load on the arm and body; neck angle; and whether any legs, feet, or arms are supported. As the human range of motion is pose dependant, to ensure the validity of the postures, we used the learned pose-dependant model of posture validity provided by [10]. We developed a script for automatic RULA assessment based on the posture and tasks parameters and verified it with several ergonomists. We used this script to label the posture dataset. Since postures with labels 1, 2, 6, and 7 are not frequent in the full range of human motion, we

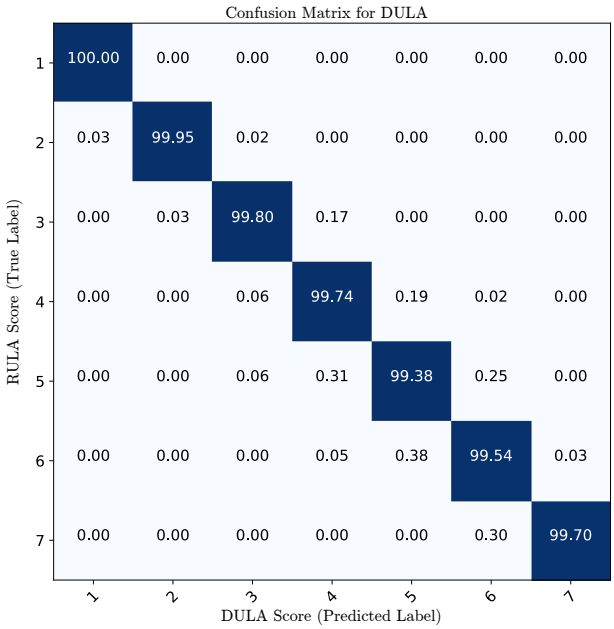


Fig. 2: Confusion matrix (accuracy %) for DULA vs RULA.

balanced the dataset by forcing the data generation scripts to generate enough data points with those labels. We split the dataset into 80% training and 20% testing sets.

Moreover, to learn a continuous and differentiable function for RULA, we designed a fully-connected regression-based neural network. While RULA provides discrete integer scores from 1 to 7, we choose to predict continuous labels. If we had instead performed multi-class classification based on the discrete labels the resulting model would be less useful in optimization as there is no natural choice of smooth objective to minimize or constrain ergonomic cost. This would negate our desire for a computationally useful model. Hence, we perform regression to the multi-class labels.

The structure of the neural network is shown in Fig. 1. It includes 4 hidden layers with ReLU activation function. We found that a network with 124 units for the first three layers and 7 units for the last hidden layer worked best. We train the network using the standard mean squared error loss function for 2000 epochs using a learning rate of 0.001. We used 5-fold cross-validation to find the optimal network parameters. This results in a model with 99.73% accuracy. We round our continuous output to the nearest integer when reporting accuracy. Figure 2 shows the confusion matrix for the learned DULA model. The lowest diagonal element is 99.38% which

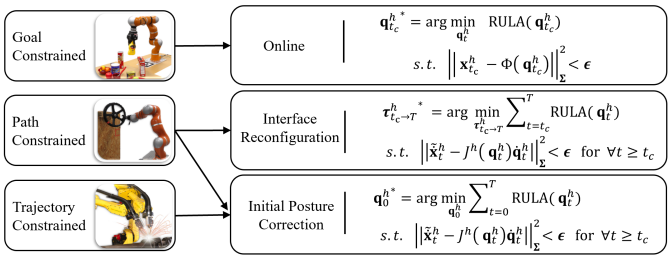


Fig. 3: Postural optimization categories in teleoperation.

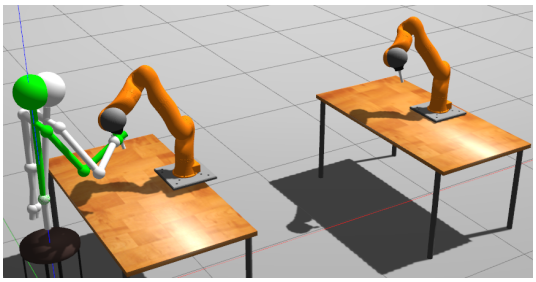


Fig. 4: Teleoperation simulation environment in ROS. The green skeleton visualizes the suggested optimal posture and the white skeleton shows the simulated human.

shows the high accuracy of the learned model across all ranges.

IV. Postural Optimization Using DULA

We use the learned DULA model in a gradient-based postural optimization for a simple teleoperation scenario in which a human interacts with a leader robot to remotely control a follower robot. As the human performs the task, the intelligent teleoperation system estimates the human posture using e.g. marker-based or markerless posture estimation, or directly from the leader robot using the method proposed in [25]. Then, it performs risk assessment to obtain the ergonomics risk score using the standard RULA model. The postural optimization algorithm uses DULA to find the optimal posture that results in the same hand pose at the interaction point between the human and the leader robot, while having the minimum risk of injuries and then provides the user with the online optimal postural correction to move towards while completing the task.

The human operator can correct the posture while performing the task without pausing. This approach is beneficial for goal-constrained teleoperation tasks such as pick-and-place in which the goal position for placing the object is defined, but, the path toward the goal point is not constrained.

We define online postural optimization for teleoperation as:

$$\begin{aligned} \mathbf{q}_t^* = \underset{\mathbf{q}_t}{\text{argmin}} \quad & \text{DULA}(\mathbf{q}_t) \\ \text{s.t.} \quad & ||\mathbf{x}_t - \Phi(\mathbf{q}_t)||_\Sigma^2 < \epsilon \\ & \mathbf{q}_t \in \text{Range of Motion} \end{aligned} \tag{1}$$

where $\mathbf{q}_t^*$ and $\mathbf{q}_t$ are optimal posture and posture at time t, respectively, $\mathbf{x}_t$ is the observed pose of the hand measured by the leader robot, Φ is the forward kinematics of the human, and Σ is the weight vector for position and orientation elements.

It is important to note that the optimal posture $\mathbf{q}_t^*$ from Eq. (1) for each time step is then suggested to the human to move towards. The human can refuse or accept, and try to apply it as much as possible while completing the task.

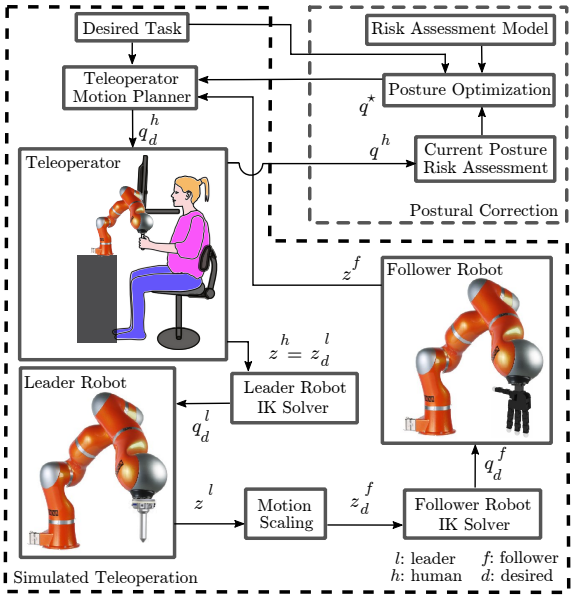


Fig. 5: Overview of teleoperation simulation.

In addition to goal-constrained teleoperation, we also defined and formulated the use of DULA in postural optimization for path-constrained teleoperation (e.g. turning a valve where the path to follow is constrained based on the diameter of the valve) and trajectory-constrained teleoperation (e.g. arc welding where the operator should follow a velocity profile in addition to the welding path). Figure 3 summarizes our formulation for different types of teleoperation tasks and their corresponding postural optimization approaches. We focus on online postural correction in this paper; analyzing the other problem formulations is ongoing work.

We used a sequential quadratic programming (SQP) [16] solver from SciPy, bounded on the range of motion, to solve the nonlinear optimization in Eq. (1), and calculated DULA gradients using automatic differentiation in PyTorch.

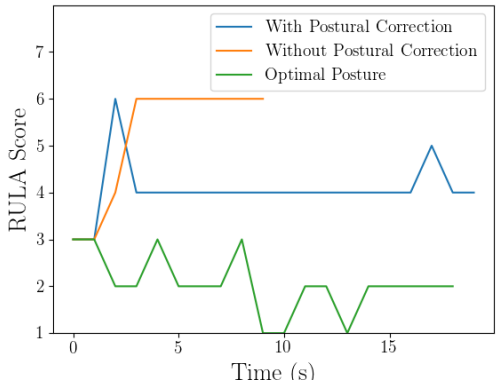
V. Simulated Teleoperation Environment

We developed an open-source simulator for postural correction in teleoperation using ROS. It includes a human seated on a stool, and two 7-DOF KUKA LWR-4 robots as leader and follower robots as shown in Fig. 4. We model our simulated human operator to behave like a human in two ways: (1) physically controlling the teleoperation task and (2) accepting or rejecting the recommended postural corrections.

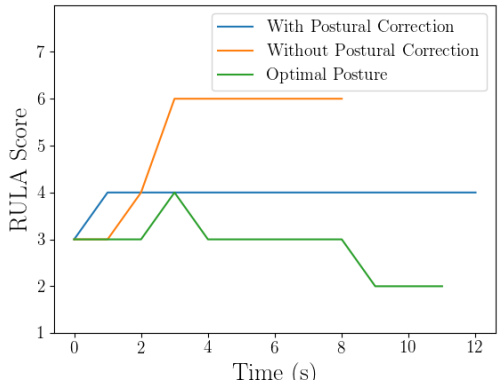
We model this as an optimal motion planning framework with re-planning that finds a human joint trajectory that controls the follower robot for the desired task while moving toward the optimal ergonomic posture:

$$\tau_{t\to H}^{h\,*} = \underset{\tau_{t\to H}^h}{\arg\min} \sum_{t=t}^{H} ||\mathbf{x}_g^f - \mathbf{x}_t^f||_\Sigma^2 + \alpha||\mathbf{q}_t^h - \mathbf{q}^{h*}_t||_2^2 \tag{2}$$

where $\tau_{t\to H}^f$ is the trajectory of human posture from time t to the time of the horizon H, $\mathbf{x}_g^f$ is the goal pose of the follower robot, and ${}^f\mathbf{x}_t$ is the pose of the follower robot at time t.



A. Gradient-free postural optimization



B. Gradient-based postural optimization

Fig. 6: Comparison of gradient-free and gradient-based postural correction on a teleoperation task. Lower scores are better.

$\mathbf{q}_t$ and $\mathbf{q}_t^*$ are the current posture and optimal posture of the human from the postural optimization at time t, respectively. Here, $0 \leq \alpha \leq 1$ is a scalar number that models the postural correction acceptance and the effort of the human operator towards applying the postural correction. Details of the motion planning for the human and the robots are presented in Fig. 5. We note that this model is likely a simplification of true human teleoperation behavior. We do not advocate for its use over human subject studies. Instead, we propose it as a useful tool for systematically exploring new algorithms in human safety assessment and improvement.

As a comparison to DULA, we directly use the standard RULA in a gradient-free postural optimization using the cross-entropy method [5]. Figure 6(A) shows the postural correction using this approach in a simple teleoperation task. It presents the RULA scores for calculated optimal posture (green), human posture without postural correction (orange), and the human posture after applying the postural correction (blue) according to the human control and acceptance model during the task. The plot shows that the risk is reduced after the simulated human applies the suggested optimal postural correction. Also the task completion time increases without a significant decrease in risk.

Figure 6(B) shows the postural correction for the same task using gradient-based optimization with DULA. We can see that gradient-based approach provides a smoother motion with respect to the RULA risk score and avoids going through postures with risk higher than 4. It also results in shorter task completion time than the gradient-free approach.

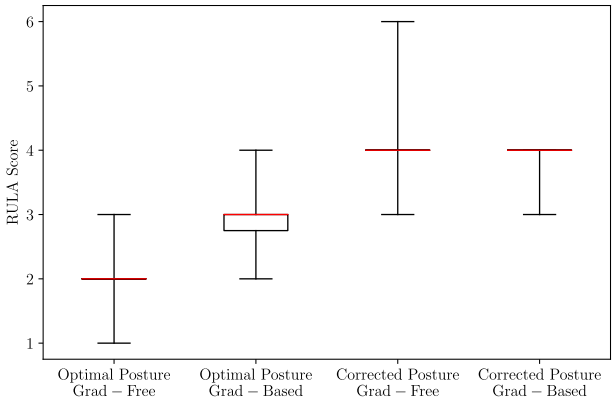


Fig. 7: comparing the optimal posture and corrected posture between gradient-free optimization using RULA and gradient-based optimization using DULA. Lower scores are better. An iteration of the gradient-free method takes 2.7 minutes, where the gradient-based method takes only tenths of a second.

Figure 7 provides more information on comparing gradient-free and gradient-based postural optimization. It shows that the median risk scores of target optimal postures from the gradient-free approach is lower. However, the postures of the simulated human are more comfortable after applying the optimal posture calculated from the gradient-based optimization. We believe it is due to the smoother optimal postures that has been suggested to the simulated human from the gradient-based method. Moreover, the gradient-based approach is much faster. Each iteration of the gradient-free approach using 10000 samples takes 2.7 minutes to solve, while the gradient-based approach takes only tenths of a second.

VI. Conclusion and Future Directions

We introduced DULA, a differentiable and continuous ergonomics model to assess human upper body posture. DULA learned to replicate the non-differentiable RULA using a neural network. The proposed model is 99.73% accurate and computationally designed for effecient use in postural optimization for pHRI and other related applications. We also introduced a framework for postural optimization in teleportation using DULA and presented a demo task. The results reveal postural optimization using DULA lowers the risk score for goal-constrained teleop. The trained DULA model, data, algorithm, and demo are available as an open-source package[1].

There are several directions for future work. As a fist step, we plan to follow the same procedure for REBA a whole body assessment tool. Additionally, we intend to conduct a human subject study to evaluate our postural optimization and correction approach. We wish to compare different means of feedback for the posture correction to the human operator including visual, auditory, and haptic feedback.

VII. Aknowledment

This work is supported by DARPA under grant N66001-19-2-4035.

[1] https://sites.google.com/view/differentiable-ergonomics

References

[1] Baptiste Busch, Guilherme Maeda, Yoan Mollard, Marie Demangeat, and Manuel Lopes. Postural optimization for an ergonomic human-robot interaction. In *Intl. Conf. on Intelligent Robots and Systems*, pages 2778–2785, 2017.

[2] Baptiste Busch, Marc Toussaint, and Manuel Lopes. Planning ergonomic sequences of actions in human-robot interaction. In *Intl. Conf. on Robotics and Automation*, pages 1916–1923, 2018.

[3] Lipeng Chen, Luis FC Figueredo, and Mehmet R Dogar. Planning for muscular and peripersonal-space comfort during human-robot forceful collaboration. In *IEEE-RAS Intl. Conf. on Humanoid Robotics*, pages 1–8, 2018.

[4] Steven Chen and Richard R Yang. Pose trainer: correcting exercise posture using pose estimation. *arXiv preprint arXiv:2006.11718*, 2020.

[5] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[6] Sandra G Hart and Lowell E Staveland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183, 1988.

[7] Sue Hignett and Lynn McAtamney. Rapid entire body assessment (REBA). *Applied ergonomics*, 31(2):201–205, 2000.

[8] Thuong N Hoang, Martin Reinoso, Frank Vetere, and Egemen Tanin. Onebody: remote posture guidance system using first person view in virtual environment. In *Proceedings of the 9th Nordic Conf. on Human-Computer Interaction*, pages 1–10, 2016.

[9] Syazwan Aizat Ismail, Shamsul Bahri Mohd Tamrin, Mohd Rafee Baharudin, Mohamad Azhar Mohd Noor, Muhamad Hanafiah Juni, Juliana Jalaludin, and Zailina Hashim. Evaluation of two ergonomics intervention programs in reducing ergonomic risk factors of musculoskeletal disorder among school children. *Res J Med Sci*, 4(1):1–10, 2010.

[10] Yifeng Jiang and C Karen Liu. Data-driven approach to simulating realistic human joint constraints. In *Intl. Conf. on Robotics and Automation*, pages 1098–1103, 2018.

[11] Jay M Kapellusch, Arun Garg, Kurt T Hegmann, Matthew S Thiese, and Elizabeth J Malloy. The Strain Index and ACGIH TLV for HAL: risk of trigger digit in the WISTAH prospective cohort. *Human factors*, 56(1): 98–111, 2014.

[12] Zahra Khodabakhshi, Seyed Amin Saadatmand, Mehrdad Anbarian, and Rashid Heydari Moghadam. An ergonomic assessment of musculoskeletal disorders risk among the computer users by rula technique and effects of an eight-week corrective exercises program on reduction of musculoskeletal pain. *Iranian Journal of Ergonomics*, 2(3):44–56, 2014.

[13] Wansoo Kim, Jinoh Lee, Luka Peternel, Nikos Tsagarakis, and Arash Ajoudani. Anticipatory robot assistance for the prevention of human static joint overloading in human–robot collaboration. *IEEE Robotics and Automation Letters*, 3(1):68–75, 2017.

[14] Antonio Gonzales Marin, Mohammad S Shourijeh, Pavel E Galibarov, Michael Damsgaard, Lars Fritzsch, and Freek Stulp. Optimizing contextual ergonomics models in human-robot interaction. In *Intl. Conf. on Intelligent Robots and Systems*, pages 1–9, 2018.

[15] L. McAtamney and E. N. Corlett. RULA: a survey method for the investigation of work-related upper limb disorders. *Applied ergonomics*, 24(2):91–99, 1993.

[16] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer, 2006.

[17] L. Peternel, W. Kim, J. Babič, and A. Ajoudani. Towards ergonomic control of human-robot co-manipulation and handover. In *IEEE-RAS Intl. Conf. on Humanoid Robotics*, pages 55–60, 2017.

[18] L. Peternel, N. Tsagarakis, D. Caldwell, and A. Ajoudani. Robot adaptation to human physical fatigue in human-robot co-manipulation. *Autonomous Robots*, pages 1–11, 2018.

[19] Luka Peternel, Cheng Fang, Marco Laghi, Antonio Bicchi, Nikos Tsagarakis, and Arash Ajoudani. Human arm posture optimisation in bilateral teleoperation through interface reconfiguration. In *IEEE RAS/EMBS Intl. Conf. for Biomedical Robotics and Biomechatronics*, 2020.

[20] Rahaf Rahal, Giulia Matarese, Marco Gabiccini, Alessio Artoni, Domenico Prattichizzo, Paolo Robuffo Giordano, and Claudio Pacchierotti. Caring about the human operator: haptic shared control for enhanced user comfort in robotic telemanipulation. *IEEE Tran. on Haptics*, 13 (1):197–203, 2020.

[21] Ali Shafti, Ahmad Ataka, B Urbistondo Lazpita, Ali Shiva, Helge A Wurdemann, and Kaspar Althoefer. Real-time robot-assisted ergonomics. In *Intl. Conf. on Robotics and Automation*, pages 1975–1981, 2019.

[22] J Steven Moore and Arun Garg. The strain index: a proposed method to analyze jobs for risk of distal upper extremity disorders. *American Industrial Hygiene Association Journal*, 56(5):443–458, 1995.

[23] Linda van der Spaa, Michael Gienger, Tamas Bates, and Jens Kober. Predicting and Optimizing Ergonomics in Physical Human-Robot Cooperation Tasks. In *Intl. Conf. on Robotics and Automation*, pages 1799–1805. IEEE, 2020.

[24] Amir Yazdani and Roya Sabbagh Novin. Posture estimation and optimization in ergonomically intelligent teleoperation systems. In *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, pages 604–606, 2021.

[25] Amir Yazdani, Roya Sabbagh Novin, Andrew Merryweather, and Tucker Hermans. Is The Leader Robot an Adequate Sensor for Posture Estimation and Ergonomic Assessment of A Human Teleoperator? In *IEEE Intl. Conf. on Automation Science and Engineering*, 2021.

Information-Theoretic Based Target Search with Multiple Agents

Minkyu Kim Department of Mechanical Engineering
The University of Texas at Austin
Austin, Texas 78705
Email: steveminq@utexas.edu

Ryan Gupta Department of Aerospace Engineering
Austin, Texas 78705
Email: ryan.gupta@utexas.edu

Luis Sentis Department of Aerospace Engineering
Austin, Texas 78705
Email: lsentis@austin.utexas.edu

***Abstract*—This paper proposes an online path planning and motion generation algorithm for heterogeneous robot teams performing target search in a real-world environment. Path selection for each robot is optimized using an information-theoretic formulation and is computed sequentially for each agent. First, we generate candidate trajectories sampled from both global waypoints derived from vertical cell decomposition and local frontier points. From this set, we choose the path with maximum information gain. We demonstrate that the hierarchical sequential decision-making structure provided by the algorithm is scalable to multiple agents in a simulation setup. We also validate our framework in a real-world apartment setting using a two robot team comprised of the Unitree A1 quadruped and the Toyota HSR mobile manipulator searching for a person. The agents leverage an efficient leader-follower communication structure where only critical information is shared.**

I. Introduction

There has been tremendous attention for search behaviors using single and multiple agent systems. These studies can be classified into various categories, such as offline([1; 2; 3; 4]) and online ([5; 6; 7; 8; 9]) coverage path planning, exploration and mapping ([10; 11; 12; 13; 14; 15; 16; 17]), and search itself ([18; 19; 20; 21; 22; 23; 24]). With improvements to robot mobility, sensing and computing power there have been many successful applications ([8; 25; 26; 27; 28; 29; 30]) in real-world settings like moon exploration, search and rescue, and unmanned surveillance with teams of robots made up of UAVs, legged systems and mobile robots. Several of these groups have ongoing work focusing on using multiple agents to improve the performance of their system.

This study attempts to solve the problem of target search with a single robot or multiple robots within a given search boundary in real-time fashion. The contribution of this paper is an online search algorithm which can be scaled to n heterogeneous agents for performing a probabilistically optimal search in real-world environments. Because the target search problem can be formulated in varying ways, we first discuss a set of assumptions for which we are solving. Such conditions are prior knowledge of the search region, the presence of a target prediction model, and characteristics of the target (i.e., the target being static or dynamic). Our online planning algorithm accounts for changes to the environment and uses sensor observations to perform a probabilistically optimal continuous search of the region.

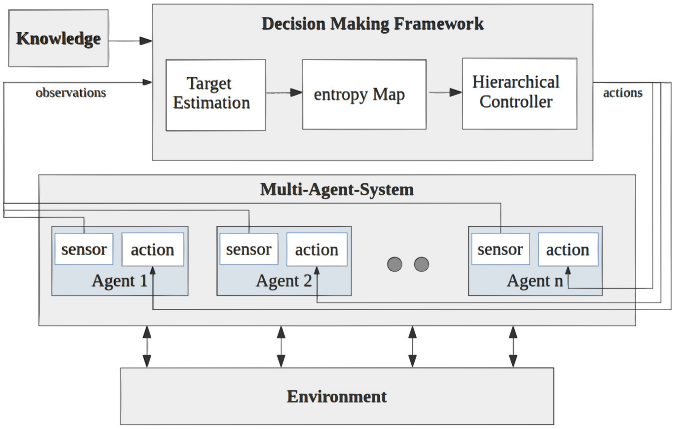


Fig. 1. The proposed architecture for Multi-Agent-System

This work proposes the use of multiple agents to perform the search task defined above for settings in which robot speed and sensor coverage are heterogeneous. The most significant challenge for multi-agent systems in real-world cooperation tasks such as search is robust, quick and efficient communication. A key characteristic of a successful design is minimizing information shared between the agents to maintain efficiency and reduce computational time. This is imperative to ensure that agents receive action commands in real-time.

II. Methods

A. Problem Definition

We define a cooperative multi-agent target search problem with a hierarchical knowledge structure. This problem aims to find a control policy (sensing action) for multi-agent system (MAS) search in a known area for a target with known probability distribution in the predefined search region. Each agent's control policy is supposed to find control inputs that maximize target information (or reduce target uncertainty).

Assuming a heterogeneous robots setup, we denote the index of the agent in each equation.

B. Target Estimation

We use Bayesian Inference to recursively estimate target state, x, through sequential observations, ys. Bayesian inference is a commonly used framework used to estimate a target state in a probabilistic manner. This inference model aims to predict the posterior distribution of target position at time k, namely, $p(x_k)$. Bayesian filtering includes a prediction step and a correction step using incoming sensing information. Assuming that the prior distribution $p(x_{k-1})$ is available at time $k-1$, the prediction step attempts to estimate $P(x_k|y_{1:k-1}^{1:n})$ – where n is the number of agents – from previous observations as follows.

$$p(x_k|y_{1:k-1}^{1:n}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1}^{1:n})dx_{k-1}, \quad (1)$$

where $p(x_k|x_{k-1})$ is the target's motion model based on a first order Markov process. Then, when the measurement $y_k^{1:n}$ is available, the estimated state can be updated as

$$p(x_k|y_k^{1:n}) = \frac{p(y_k^{1:n}|x_k)p(x_k|y_{1:k-1}^{1:n})}{p(y_k^{1:n}|y_{1:k-1}^{1:n})} \quad (2)$$

where $p(y_k^{1:n}|y_{1:k-1}^{1:n}) = \int p(y_k^{1:n}|x_k)p(x_k|y_{k-1}^{1:n})dx_k$ and $p(y_k^{i:n}|x_k)$ is a sensing model for multi agent system, which can also be decomposed to each agent's sensing model $p(y^i|x)$. For the correction stage, the measurement of all agents are used to modify the prior estimate, leading to the target belief. If a static target is assumed the target motion model can be described as $p(x_k|x_{k-1}) = \mathcal{N}(x_{k-1}; x_k, \Sigma)$, only containing a noise term with the previous target state. Instead if a dynamic target is assumed to have some constant velocity, we can represent the target model as $p(x_k|x_{k-1}) = \mathcal{N}(x_{k-1}; x_k + V\Delta, \Sigma)$.

C. Hierarchical Bayesian Model

We propose a leader-follow hierarchy setting in which we assume that the first agent is the leader and the second is the sub-leader, consisting of sequentially lower-class followers. Assuming that decisions (desired paths) can be mapped into the expected sensing outputs, within the proposed hierarchy structure, our decision-making process can be generalized to n-multi agent systems, estimating target state x_t with expected observations of all agents, $[y_{t:t+h}^1, y_{t:t+h}^2, \cdots, y_{t:t+h}^n]$ with time horizon h. To be more specific, given the target state probability $p(x)$ and the expected sensing outputs $\hat{y}$ from the agents, the belief state variable $\hat{\pi}_t^i = p(x|\hat{Y}^i)$ can be written as

$$\begin{aligned} \hat{\pi}_t^1 &= p(x_t|y_{t:t+h}^1) & s_t^1 &= \gamma(\hat{\pi}_t^1) \\ \hat{\pi}_t^2 &= p(x_t|y_{t:t+h}^1, y_{t:t+h}^2) & s_t^2 &= \gamma(\hat{\pi}_t^2) \\ \vdots &= \quad \vdots \quad \vdots & \vdots &\quad \vdots \\ \hat{\pi}_t^n &= p(x_t|y_{t:t+h}^1, \cdots, y_{t:t+h}^n) & s_t^n &= \gamma(\hat{\pi}_t^n) \end{aligned}, \quad (3)$$

where $\hat{Y}^i$ denotes the set of expected outputs up to the i-th agent. In this way, the expected belief state can be updated sequentially based on the decisions of each agent, and can be used as known information for the $i+1$ agent. Here, $\gamma(*)$ is a decision making function given a target belief state whose output is desired path for each agent.

Algorithm 1 Multi-Agent Search()

Input: Y, M, C
(Robot poses, Entropy Map, Sensing Capabilities (speed, coverage))
Output: $S^* = \{s_1^*, s_2^*, \cdots, s_N^*\}$ (A set of paths)

```
w_g ← SampleWaypoints()   ▷ Vertical Cell Decomposition
w_l ← GetFrontiers()
for n ← 1 to N do                                   ▷ for each agent
    s_g ← SampleGlobalPaths(y_i, w_g)               ▷ A*
    s_l ← SampleLocalPaths(y_i, w_l)                ▷ A*
    S_i = {s_g, s_l}
    Ŝ_i = reparameterize(S_i, C_i)                  ▷ speed and coverage
    for k ← 1 to |S_i| do
        IG(s_k) = CalculateIG(s_k, S*_{1:n-1})      ▷ Equation (7)
                                  ▷ Use S*_{1:n-1} from higher hierarchy
        U(s_k) = IG(s_k) − c(s_k)
    end for
    s* = s_k ← arg max U(s_k)                       ▷ Get the best path
    S*.append(s*)
end for
return S*
```

D. Information-Theoretic Objective Path planning

Our strategy is to maximize the information gain regarding the target belief state in a greedy fashion. Given each robot's expected information gains and travel costs, we obtain paths for all agents that maximize the overall utility.

$$\max_{\gamma^{1:n}} \mathbb{E}[\tilde{U}(x_t|y^1, y^2, \cdots, y^n)] \quad (4)$$

where $\mathbb{E}[\tilde{U}]$ refer to the expected value of the utility function given the target belief and the sensing outputs. This utility function can be calculated based on the path candidates for each agent in order to select the best path (or sequence of control inputs). Given a target belief at time t, obtaining the desired path for each agent allows MAS to gather information as quickly as possible. Our method aims to find the optimal path using a sampling-based optimization problem over a time horizon with an information-theoretic utility function. The utility function is described as

$$\mathbb{E}[\tilde{U}(x_t|y^1, y^2, \cdots, y^n)] = \sum_i^n (IG(\mathrm{s}_i) - c(\mathrm{s}_i)) \quad (5)$$

1) Path Selection: To obtain the best path for each agent, we use a sampling-based optimization approach. The proposed strategy is to create global and local candidate paths. The goal points of paths can be sampled from waypoints within a search map boundary, sampled based on the entropy map M, which considers the target estimation model being updated by local measurements. Precisely, the global candidate points are sampled from a vertical cell decomposition of sub-regions

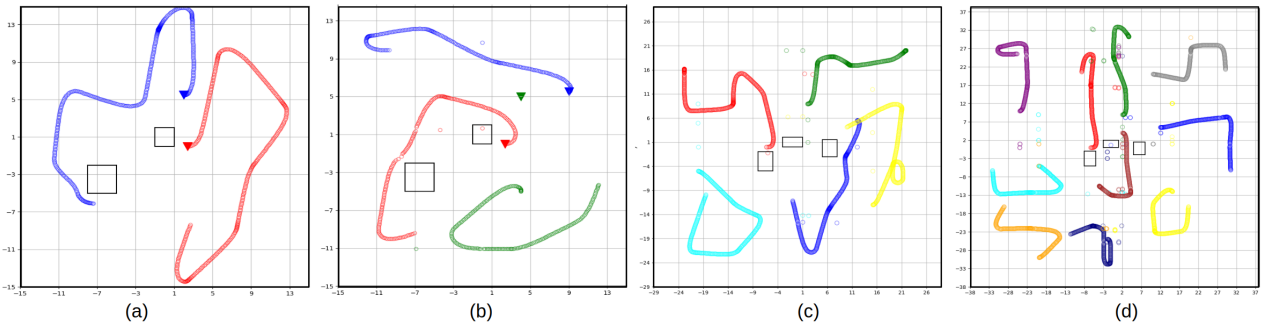


Fig. 2. Simulation results using increasing number of agents. (a) 2 agents in a 13x13 grid. (b) 3 agents in a 13x13 grid. (c) 5 agents searching a 25x25 grid space. (d) 10 agents searching a 35x35 grid space.

whose occupancy grid type is unknown, $M_{[m=0.5]}$. Cells with the value 0.5 are unknown while cells that are occupied or unoccupied are given 1.0 and 0.0, respectively. Local goal points are sampled from the cluster of frontiers (31), which is defined as the boundary between known (occupied or free) and unknown areas, which are potentially informative. The frontiers can be obtained from the current entropy map.

$$p(x) = \text{uniform over } M_{[m=0.5]} \tag{6}$$

Given the next viewpoint candidates and the current robot position, we use an A* planner to generate an obstacle-free path, s, for each agent. We note that any other planner like RRT or PRM could be used in its place. The set of paths to each global target point and a local target point are of different lengths and therefore must be re-parameterized. The reason for this re-parameterization is that the optimization problem should consider path length based on the robot's speed and the time horizon. In order to consider the difference in the speed of robots, the interval between the sampling points along the generated path is set accordingly. For example, if the path length $|\mathrm{s}|$ is 10 and $ds = 2$, we can sample five points along the trajectory. In order to avoid overlap between the computed FOV area, we sparsely sample points along a path to calculate the expected amount of information. Based on the number of sampled points along a path, we compute the Information Gain, denoted $IG(s)$, by the following equation:

$$\begin{aligned} IG(s) &\approx \sum_{i=1}^{N_\mathrm{s}} H(FOV(\mathrm{s}^i)) \\ &= \sum_{i=1}^{N_\mathrm{s}} \left[\sum_{j=1}^{N_c} (p(m_{i,j}) log(p(m_{i,j})) + (1 - p(m_{i,j})) log(1 - p(m_{i,j}))) \right] \end{aligned} \tag{7}$$

, where s^i denotes the i-th sampled point and $p(m)$ is the occupancy probability, while N_s and N_c are the number of sampling points and the number of cells in the FOV given the sampled points, respectively. Thus, The $IG(\mathrm{s})$ is calculated by summing over the FOV regions defined by sampled points through the trajectory. When calculating the information gain, the paths of the robots do not overlap as much as possible by not calculating the information gain corresponding to the path

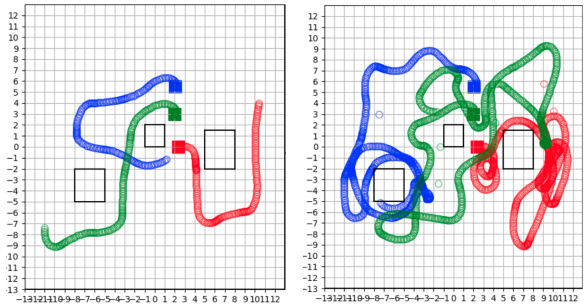


Fig. 3. Simulation results with different conditions. (a) One time search (static target) (b) Continuous search (dynamic target).

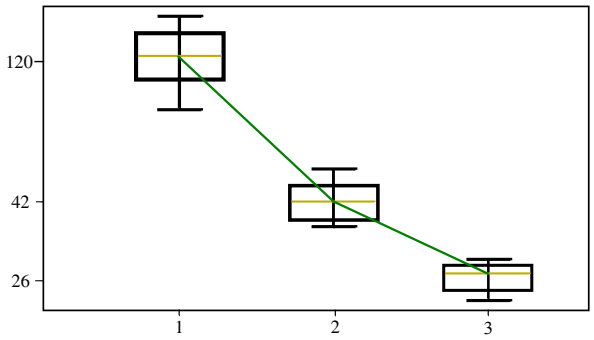


Fig. 4. Average search time with varying number of agents.

selected from the agent in the upper hierarchy. The proposed search algorithm is described in the Algorithm 1.

III. Results

A. Simulation Results

In this section we present python-based simulation results of our proposed approach for a multi-agent search behavior and demonstrate the scalability of the algorithm. It is assumed that the simulation environment (search region) and all the static obstacles have a rectangular shape and obstacles are not known in advance. Each agent is equipped with a simulated ray sensor, which has a square field of view with limited range, $F(x,y)$, determined by the 2D position of the agent.

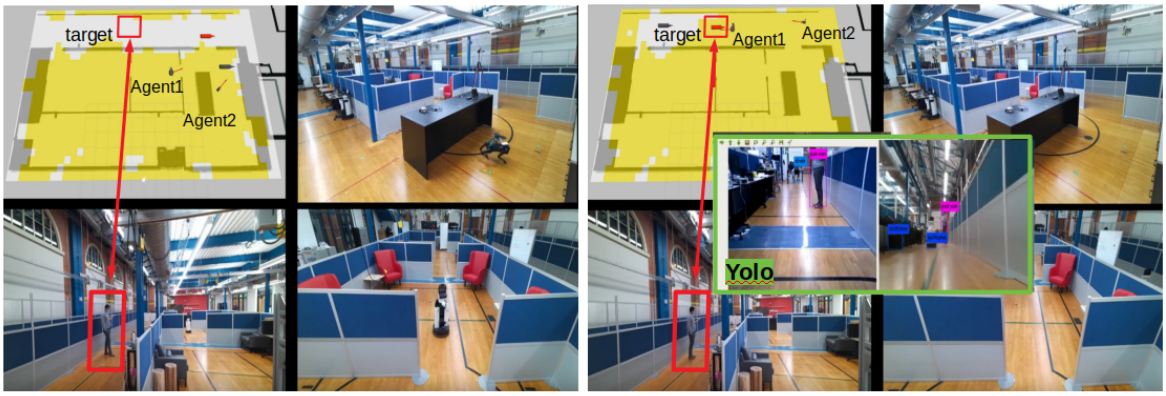


Fig. 5. (a) Experimental validation for a two robot search in the Anna Hiss Gymnasium apartment area. In the top left figure, the yellow regions are those which have been explored, while the white areas are regions of uncertainty. The target location is described as a red box and is unknown to the robots. The red and black markers represent the next waypoints for each agent. (b) The completion of the experimental validation with both agents converging to the target of interest. The object recognition is performed using the well-known YOLO algorithm to detect people (shown in the green box).

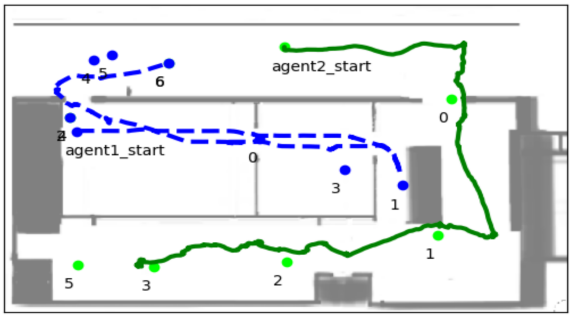


Fig. 6. Trajectories for each agent in the AHG apartment setup with different sets of initial conditions

Given the resolution of the map, the entire environment can be decomposed into square-type grid cells. To achieve robust collision avoidance, we use a dynamic window approach to generate the control input to navigate towards goal points.

1) Single Search vs Continuous Search: In order to test the search performance, simulations were conducted under various initial conditions. Depending on how the search map is updated (time-varying condition), we can implement one time search (similar to the exploration and mapping problem) and a continuous search. As shown in Fig 3, in the case of three agents, we can prove that the agents were able to effectively search for the target over the search region.

2) n-Agent Case: To validate the scalability of the proposed method, we test the exploration with n agents. As the number of agents increases, we enlarge the search space. Fig. 2 demonstrates the trajectories of each agent in the case number of agents equals to 2,3,5, and 10. Each color represents the trajectory of a different agent. These results demonstrate that our algorithm is scalable and can be extended to a general multi-agent system and still perform in real-time.

3) Search Time: The search time varies depending on the initial condition or the dimension of the search space. Therefore, for accurate comparison, given a fixed size of the search map(13x13), fixed maximum moving speed (1m/s), the search time was compared. Fig. 4 shows the average search time (the entropy reduction rate) for three cases. By adding an additional agent, the time to completion is reduced by more than half.

B. Experimental Results

We used the Unitree A1 quadruped and the Toyota HSR mobile manipulation robot for experimentation of multiple agent target search. The A1 is equipped with a Velodyne VLP-16 3D Lidar and a RealSense D435 camera. On-board computing is performed in an Intel NUC Mini PC, which communicates with the low-level control systems. The HSR is equipped with a Hokuyo 2D Lidar, a RGB-D camera, and an on-board Jetson TK1 GPU. We assume that both robots have perfect localization, although in practice we provide it via Episodic non-Markov Localization (32). The use of such different systems demonstrates that our algorithm is well suited to perform with a heterogeneous team, each with varying motion models.

Experiments were performed in the Anna Hiss Gymnasium apartment at the University of Texas. Fig. 5 shows two different moments in the search. The search map is 20(m) x 10(m) and the maximum velocity of each agent are 0.3(m/s) and 1.0(m/s) for the HSR and A1, respectively. In the top left of Fig. 5 (a) and (b) is the global entropy map. Regions in yellow indicate that they have been explored (known to be free or occupied), while regions in white have high uncertainty. In the lower-left of each figure (a) and (b), the target is found in the hallway along the left side of the apartment setup. The red and black markers indicate the next waypoints for each agent, which is determined by using the search server. In (b), the object detection feed is shown as the two agents approach the target of interest. Finally, Fig. 6 shows resulting trajectories for the two agents for one of experimental trials. The video demo can be found at https://youtu.be/7WMqG7EiUVY.

IV. Conclusion

This paper addresses online search for a heterogeneous multi-agent system. We employ an information-theoretic utility function and sampling-based optimization to obtain each agent's path. A hierarchical decision-making structure allows us to reduce computational burden and perform the search in real-time. Simulation results show that our proposed algorithm proves its scalability and that it can be extended to the general case of multiple agents. We further validate this algorithm by implementing it in a real-world environment. Overall results validate the effectiveness and robustness of the proposed method.

Acknowledgments

The authors would like to thank the members of the Human Centered Robotics Laboratory at The University of Texas at Austin for their great help and support. This work was supported by the Army Futures Command and the Office of Naval Research, ONR Grant #N000141512507.

References

[1] E. I. Grotli and T. A. Johansen, "Path planning for uavs under communication constraints using splat! and milp," pp. 265–282, 2012.

[2] T. I. F. Erik J. Forsmo, Esten I. Grotli and T. A. Johansen, "Optimal search mission with unmanned aerial vehicles using mixed integer linear programming," 2013.

[3] N. Agmon, N. Hazon, and G. A. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1698–1703.

[4] J. H. Richard Bormann, Florian Jordan and M. Hagele, "Indoor coverage path planning: Survey, implementation, analysis," 2018.

[5] J. Song and S. Gupta, "Epsilon*: An online coverage path planning algorithm," pp. 526–533, 2018.

[6] ——, "Care: Cooperative autonomy for resilience and efficiency of robot teams for complete coverage of unknown environments under robot failure," 2019.

[7] A. Khan, I. Noreen, H. Ryu, N. L. Doh, and Z. Habib, "Online complete coverage path planning using two-way proximity search," pp. 229–240, 2017.

[8] X. Kan, H. Teng, and K. Kayrdis, "Online exploration and coverage planning in unknown obstacle-cluttered environments," 2020.

[9] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara, "Bsa: a complete coverage algorithm," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 2040–2044.

[10] F. Amigoni and V. Caglioti, "An information-based exploration strategy for environment mapping with mobile robots," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 684–699, 2010.

[11] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger, "Fast frontier-based information-driven autonomous exploration with an mav," *arXiv preprint arXiv:2002.04440*, 2020.

[12] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar, "Information-theoretic planning with trajectory optimization for dense 3d mapping." in *Robotics: Science and Systems*, vol. 11, 2015.

[13] B. Charrow, S. Liu, V. Kumar, and N. Michael, "Information-theoretic mapping using cauchy-schwarz quadratic mutual information," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4791–4798.

[14] M. Kontitsis, E. A. Theodorou, and E. Todorov, "Multi-robot active slam with relative entropy optimization," in *2013 American Control Conference*. IEEE, 2013, pp. 2757–2764.

[15] K. Krinkin, A. Filatov, and A. Filatov, "Modern multi-agent slam approaches survey," in *Proceedings of the XXth Conference of Open Innovations Association FRUCT*, vol. 776, 2017, pp. 617–623.

[16] S. Lee, H. Kim, and B. Lee, "An efficient rescue system with online multi-agent slam framework," *Sensors*, vol. 20, no. 1, p. 235, 2020.

[17] A. Filatov and K. Krinkin, "A simplistic approach for lightweight multi-agent slam algorithm," *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, vol. 11, no. 3, pp. 67–83, 2020.

[18] Y. Ye and J. K. Tsotsos, "Sensor planning for 3d object search," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 145–168, 1999.

[19] K. Shubina and J. K. Tsotsos, "Visual search for an object in a 3d environment using a mobile robot," *Computer Vision and Image Understanding*, vol. 114, no. 5, pp. 535–547, 2010.

[20] M. Göbelbecker, A. Aydemir, A. Pronobis, K. Sjöö, and P. Jensfelt, "A planning approach to active visual search in large environments." in *Automated Action Planning for Autonomous Mobile Robots*, 2011.

[21] A. Rasouli and J. K. Tsotsos, "Sensor planning for 3d visual search with task constraints," in *2016 13th Conference on Computer and Robot Vision (CRV)*. IEEE, 2016, pp. 37–44.

[22] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt, "Active visual object search in unknown environments using uncertain semantics," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 986–1002, 2013.

[23] A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, and P. Jensfelt, "Search in the real world: Active visual object search based on spatial relations," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2818–2824.

[24] S. Zhang and M. Sridharan, "Active visual sensing and collaboration on mobile robots using hierarchical POMDPs," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous

Agents and Multiagent Systems, 2012, pp. 181–188.

[25] P. Schmuck and M. Chli, "Multi-uav collaborative monocular slam," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3863–3870.

[26] T.-m. Wang, Y.-c. Zhang, J.-h. Liang, Y. Chen, and C.-l. Wang, "Multi-uav collaborative system with a feature fast matching algorithm," *Frontiers of Information Technology & Electronic Engineering*, pp. 1–18, 2020.

[27] J. Scherer, S. Yahyanejad, S. Hayat, E. Yanmaz, T. Andre, A. Khan, V. Vukadinovic, C. Bettstetter, H. Hellwagner, and B. Rinner, "An autonomous multi-uav system for search and rescue," in *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, 2015, pp. 33–38.

[28] B. Woosley, P. Dasgupta, J. G. Rogers, and J. Twigg, "Multi-robot information driven path planning under communication constraints," *Autonomous Robots*, vol. 44, no. 5, pp. 721–737, 2020.

[29] C. D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Günther, M. Tranzatto, P. Fankhauser, and M. Hutter, "Advances in real-world applications for legged robots," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1311–1326, 2018.

[30] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *Journal of Field Robotics*, vol. 37, no. 8, pp. 1363–1388, 2020.

[31] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*. IEEE, 1997, pp. 146–151.

[32] J. Biswas and M. M. Veloso, "Episodic non-markov localization," *Robotics and Autonomous Systems*, vol. 87, pp. 162–176, 2017.

Robust Policies for Uncertain POMDPs

Marnix Suilen*, Murat Cubuktepe†, Nils Jansen*, Sebastian Junges‡, Ahmadreza Marandi§ and Ufuk Topcu†
*Department of Software Science, Radboud University, The Netherlands
†Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, USA
‡Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA
§Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, The Netherlands

Many real-world problems exhibit a combination of non-deterministic choice, uncertain outcomes, and partial information. As a concrete example, consider the problem of *aircraft collision avoidance* [14], in which a computer system advises a pilot on what actions to take to avoid a collision with an intruding aircraft.
Here, the non-deterministic choice is given by all the actions the pilot can take, the uncertain outcomes are, among others, the pilot's response time to a given advice and the behavior of the intruder, and there is only partial information about the intruder's position.

Partially observable Markov decision processes (POMDPs) are the standard class of models to reason about such decision-making problems under partial information [13]. The likelihood of uncertain events, such as human response, message loss, or adversary behavior, enters a POMDP in the form of a concrete probability. Yet, such probabilities are often derived from (finite, historic) data and are thus an estimate at best. POMDPs, however, *require* the probabilities to be exact.

Uncertain POMDPs (uPOMDPs) overcome this requirement by introducing so-called uncertainty sets [9, 21]. These sets allow, for instance, to account for incomplete data, sensor imprecisions, or human behavior in the decision-making. Instead of an exact probability, the transition or observation functions of an uPOMDP now map to such an uncertainty set, for example intervals of probabilities or likelihood functions. In particular, uncertainty sets are able to faithfully capture *epistemic* uncertainty, which can be reduced by collecting more data. Standard POMDPs cannot deal with such uncertainty.

Given the wide range of important applications of POMDPs, such as the aircraft collision avoidance problem, spacecraft motion planning [10, 12], robotics [17, 20], humanitarian relief aid [5], treatment of heart disease [11], ecology [7, 16], and many more [6]; accounting for such uncertainties may result in more reliable and safer decision-making.

We consider *reachability* and *expected reward* specifications. These optimize or constrain (by some bound) the probability or expected reward of reaching a given set of target states. Such specifications can be defined on a finite, indefinite, or infinite horizon. For infinite horizons, this problem is undecidable [15]. For finite horizons it is PSPACE-complete [18].

The goal for a (u)POMDP and a specification is to resolve the non-determinism in such a way that the specification is satisfied while taking the partial information, in the form of observations, into account. This is done by computing an *observation-based policy*, which maps observations to choices. If the policy only considers the current observation, it is *memoryless*, and if it considers a finite sequence of successive observations, it is a *finite-memory* policy. For uPOMDPs, a policy is *robust* if it satisfies the specification under the *worst-case* instance of the uncertainty in the model. The aim of this work is to compute robust finite-memory policies provably satisfying specifications for uPOMDPs.

Problem Formulation

The *robust policy computation problem* is defined as follows:

> Given a uPOMDP and a reachability or expected reward specification, compute a robust memoryless or finite-memory policy that satisfies the specification.

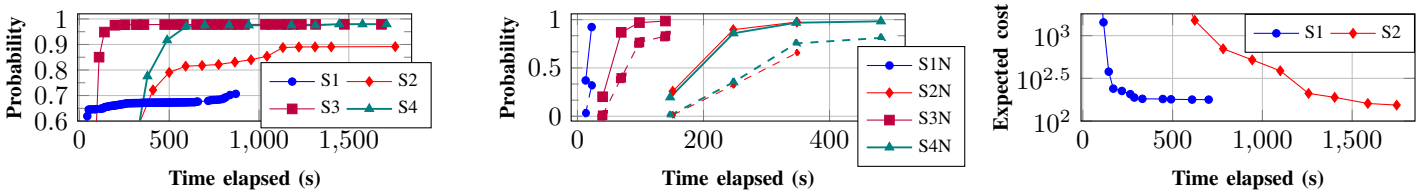
We define a uPOMDP as a tuple $(S, s_I, A, \mathbb{I}, \mathcal{P}, Z, O, R)$, with S the finite set of states, $s_I \in S$ the initial state, A the finite set of actions, $\mathbb{I}$ a set of probability intervals, and Z the finite set of observations. The *uncertain transition function* is defined as $\mathcal{P}\colon S \times A \times S \to \mathbb{I}$ and the *uncertain observation function* is given by $O\colon S \times Z \to \mathbb{I}$. Finally, $R\colon S \times A \to \mathbb{R}_{\geq 0}$ is the (standard) reward function.

We consider reachability and expected reward specifications of the following form. For a set of target states T, the goal is to compute a policy such that the probability (expected reward) of reaching T is less than (greater than) some λ, or optimized.

A *randomized finite-memory observation-based policy* is a function $\sigma\colon (Z \times A)^* \times Z \to Dist(A)$. It maps a finite sequence of observations and actions to a distribution over actions. A policy successfully resolves the decision-making in a (u)POMDP if the policy applied to the model satisfies the constraint, which can be checked via linear programming or value iteration [1, 3, 19]. This is also called *robust verification*.

Solution Outline

We first formulate a semi-infinite nonconvex optimization problem with a finite number of variables and an infinite number of constraints that precisely captures the robust policy computation problem. The number of constraints is infinite because we have an infinite number of possible distributions given by the uncertain transition function. Optimization problems like the one we have are NP-hard and in practice intractable to solve [4]. To overcome this, we apply the following iterative approach. Full details on each step can be found in [9].

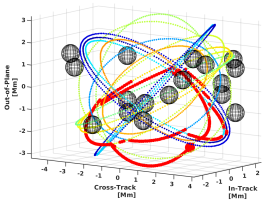


(a) Obtained probability of avoiding close encounters between the spacecraft and other objects in the orbit.

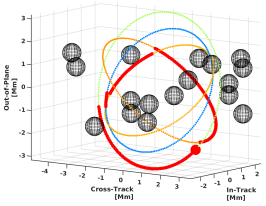
(b) The performance of the policies obtained from the nominal model applied to the uncertain model (dashed lines).

(c) The obtained expected cost of successfully finishing an orbit.

Fig. 1: Computational effort versus the performance of the different policies for the spacecraft motion planning case study.



(a) Satellite example with memoryless policy.



(b) Satellite example with five-memory-node policy.

Fig. 2: Case study on spacecraft motion planning. The red line shows the spacecraft trajectory based on a computed policy, other lines are the orbits used. The policy without memory (a) needs to switch orbit more often than the policy with five memory nodes (b), and thus consumes more fuel.

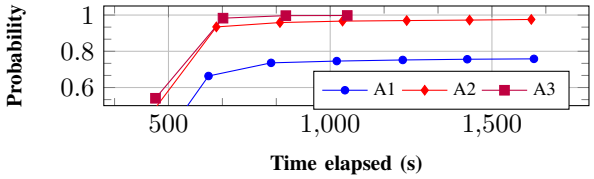


Fig. 3: Probability of successfully avoiding collisions in the aircraft collision avoidance case study.

Dualize: Via dualization we derive a *finite* nonconvex optimization problem that is only polynomially larger than the original optimization problem, while still accounting for all the uncertainty from the uPOMDP [2].

Linearize and solve: We linearize the finite nonconvex optimization problem into a finite linear program (LP), which can be solved in polynomial time [8]. The linearization is done around a previous solution, hence an iterative approach. In the first iteration we linearize around a randomly generated point.

Robust verification: We apply the computed policy to the uPOMDP and use robust verification to check if it satisfies the specification. If the result is positive, we have found a robust policy and we terminate the procedure. If not, we use the solution from the LP we just solved as new initial values for the next iteration.

Iterate: We continue iterating until we have found a robust policy satisfying the specification or have converged to a *local* optimum. Globally optimal solutions, however, cannot be guaranteed due to the linearization step.

Experimental results

We highlight some of our results on the spacecraft motion planning and aircraft collision avoidance problems.

Spacecraft motion planning: In this model, a satellite has to switch between orbits while avoiding collisions with other objects. We consider two instances of this model, one (S1) with 36 048 states and 65 263 transitions, and one (S3) with 108 000 states and 195 573 transitions. We extend both models to finite-memory policies. Model (S2) is the same as (S1), but now with five memory nodes, enlarging the model to 349 480 states and 698 960 transitions. Model (S4) is the same as (S3), but now with two memory nodes, resulting in 342 750 states and 665 073 transitions. In all of these, the uncertain probability of successfully switching orbit is given by the interval $[0.50, 0.95]$. Figures 1a and 1c show the convergence towards local optima for the various instances of the model.

Aircraft collision avoidance: In Figure 3 we maximize the probability of avoiding a collision with an intruder aircraft. All three instances of the model have 476 009 states and 866 889 transitions, and the uncertain pilot response is given by the interval $[0.7, 0.9]$. The three instances differ in the level of uncertainty on the behavior of the intruder aircraft. A1 uses intervals $[0.2, 0.8]$, A2 has $[0.3, 0.7]$, and A3 $[0.4, 0.6]$. As seen in the figure, more uncertainty leads to a lower maximum, while it does not affect the computation time.

Robust policies are more robust: Figure 1b compares robust policies with non-robust policies. We compute a policy for a standard POMDP and apply this policy to the uPOMDP. Robust verification on this model then gives a worst-case performance of this policy under the uncertainty. We see that the computed policy performs significantly worse on the uncertain model for all four instances.

Finite-memory yields better decisions: Finally, in Figure 2 we compare the trajectories of computed policies for (S1) and (S2). Both models yield policies that avoid a collision, but the finite-memory policy requires fewer orbit switches, resulting in a lower fuel consumption.

Conclusion and Future Work

We presented a novel way to compute robust finite-memory policies for uncertain POMDPs. Future work will apply learning methods to derive uncertainty sets from data.

References

[1] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[2] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*, volume 28 of *Princeton Series in Applied Mathematics*. Princeton University Press, 2009.

[3] Michael Benedikt, Rastislav Lenhardt, and James Worrell. LTL model checking of interval markov chains. In *TACAS*, volume 7795 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2013.

[4] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014.

[5] Raissa Zurli Bittencourt Bravo, Adriana Leiras, and Fernando Luiz Cyrino Oliveira. The use of uav s in humanitarian relief: An application of pomdp-based methodology for finding victims. *Production and Operations Management*, 28(2):421–440, 2019.

[6] Anthony R Cassandra. A survey of pomdp applications. In *Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes*, volume 1724, 1998.

[7] Iadine Chadès, Eve McDonald-Madden, Michael A McCarthy, Brendan Wintle, Matthew Linkie, and Hugh P Possingham. When to stop managing or surveying cryptic threatened species. *Proceedings of the National Academy of Sciences*, 105(37):13936–13940, 2008.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[9] Murat Cubuktepe, Nils Jansen, Sebastian Junges, Ahmadreza Marandi, Marnix Suilen, and Ufuk Topcu. Robust finite-state controllers for uncertain pomdps. In *AAAI*. AAAI Press, 2021 (to appear).

[10] Gregory R Frey, Christopher D Petersen, Frederick A Leve, Ilya V Kolmanovsky, and Anouck R Girard. Constrained spacecraft relative motion planning exploiting periodic natural motion trajectories and invariance. *Journal of Guidance, Control, and Dynamics*, 40(12):3100–3115, 2017.

[11] Milos Hauskrecht and Hamish Fraser. Planning treatment of ischemic heart disease with partially observable markov decision processes. *Artificial Intelligence in Medicine*, 18(3):221–244, 2000.

[12] Kerianne L Hobbs and Eric M Feron. A taxonomy for aerospace collision avoidance with implications for automation in space traffic management. In *AIAA Scitech 2020 Forum*, page 0877, 2020.

[13] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2): 99–134, 1998.

[14] Mykel J Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

[15] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *AAAI/IAAI*, pages 541–548. AAAI Press / The MIT Press, 1999.

[16] Marc Mangel and Colin Whitcomb Clark. *Dynamic Modeling in Behavioral Ecology*. Princeton University Press, 2019.

[17] Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Pomdps for robotic tasks with mixed observability. In *Robotics: Science and Systems*. The MIT Press, 2009.

[18] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of markov decision processes. *Math. Oper. Res.*, 12(3):441–450, 1987.

[19] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Model-checking markov chains in the presence of uncertainties. In *TACAS*, volume 3920 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2006.

[20] Matthijs T. J. Spaan and Nikos A. Vlassis. A point-based POMDP algorithm for robot planning. In *ICRA*, pages 2399–2404. IEEE, 2004.

[21] Marnix Suilen, Nils Jansen, Murat Cubuktepe, and Ufuk Topcu. Robust policy synthesis for uncertain pomdps via convex optimization. In *IJCAI*, pages 4113–4120. ijcai.org, 2020.

Towards Explainable Multi-robot Motion Planning

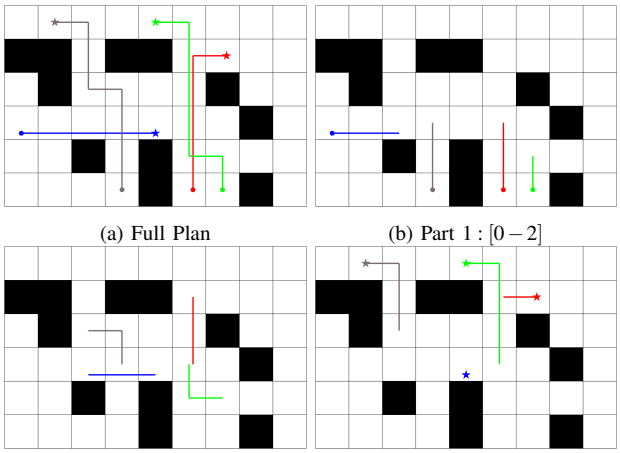
Justin Kottinger*, Shaull Almagor†, and Morteza Lahijanian*
*Aerospace Engineering Sciences, University of Colorado, Boulder, USA
Email: {justin.kottinger, morteza.lahijanian}@colorado.edu
†Computer Science, Technion, Haifa, Israel
Email: shaull@cs.technion.ac.il

I. Introduction

Multi-agent Motion Planning (MMP) is a fundamental problem in robotics and artificial intelligence (AI) where the goal is to determine trajectories for multiple agents to their respective goal regions such that, every vehicle safely completes their plan when all plans are executed simultaneously. Applications include warehouse robots, rendezvous and proximity operations, autonomous cars, etc. There are many works both in the discrete domain (e.g., [13, 12]) and continuous domain (e.g., [4, 3, 16, 11]) that solve the MMP problem. Yet, one limitation of those methods is their inability to explain their plans to human users [15]. For this reason, in safety-critical applications, such as air-traffic control, MMP is not employed since a human cannot verify them. The focus of this study is to develop *explainable* MMP algorithms.

Significant effort has been dedicated to explainable AI and machine learning. For example, [2] gives explanations by analyzing alternative plans with some user-defined properties. Work [6] explains plans via a minimal set of differences between the actual and the proposed plan. Rather, work [8] explains algorithms through visualization. Our work is similar in that we base explanations on the simplicity of visual human verification. Specifically, work [5, 14] showed recognizing line intersections occurs very early in the cognitive process (namely in the primary visual cortex). Thus, MMP can be explained using a collection of *non-intersecting* path segments. An example of such a decomposition is shown in Fig. 1a-1d.

This paper presents our progress on explainable MMP. Previous work [1], defined an *explanation* to be the separation of an MMP solution into a set of disjoint path segments and provided examples of its use for MMP problems over a discrete graph. The paper showed that as the number of disjoint segments required to explain the path decreases, the ease of explainability increases and proved finding optimal explanations for existing plans takes polynomial time, whereas generating plans for explainability is, at best, **NP-Complete**. We are interested in realistic robotic systems in continuous space with the goal of designing computationally-tractable MMP algorithms that generate sound and easily explainable plans. This problem is challenging because the dimensionality of the state space grows exponentially based on the number of agents, and kinodynamical constraints complicates planning. We propose two approaches to this problem: centralized tree-based search and abstraction-based decentralized graph search.



(a) Full Plan (b) Part 1 : $[0-2]$ (c) Part 2 : $[2-4]$ (d) Part 3 : $[4-7]$

Fig. 1: An explanation for four agents in a 9×6 grid. The circles and stars mark the initial and goal regions, respectively.

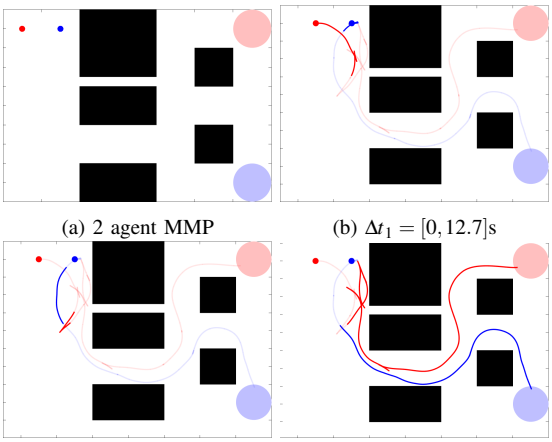
II. Problem Statement

We consider $k \in \mathbb{N}$ robotic agents in a shared workspace $W \subseteq \mathbb{R}^2$ containing a finite set of obstacles $O \subset W$. Each agent $i \in \{1,2,\dots,k\}$ is constrained to the following dynamics: $\dot{\mathbf{x}}_i = f_i(\mathbf{x}_i,\mathbf{u}_i), \quad \mathbf{x}_i \in X_i, \quad \mathbf{u}_i \in U_i$, where X_i and U_i are the i^{th} agent's state and input spaces, respectively, and $f_i : X_i \times U_i \to X_i$ is an integrable, possibly nonlinear, function. A *trajectory segment* $\mathbf{x}_i^{t_1:t_2}$ for agent i is formed by integrating f_i for a given control input and non-zero time interval. Given $m \in \mathbb{N}$ non-zero time-intervals, a *trajectory* $T_i = \{\mathbf{x}_i^{t_0:t_1}, \mathbf{x}_i^{t_1:t_2}, \dots, \mathbf{x}_i^{t_{m-1}:t_m}\}$, is a set of m trajectory segments that take agent i from an initial state $x_{i,0}$ to a desired goal region $X_i^G \subset X_i$.

Note the state space of a robotic agent is generally of higher dimension than the workspace. Thus, we define $\text{PROJ}_W^{X_i} : X_i \to W$ as a function to project trajectory segment $\mathbf{x}_i^{t_1:t_2}$ onto workspace W. Two trajectory segments $\mathbf{x}_i^{t_1:t_2}$ and $\mathbf{x}_j^{t_1:t_2}$ are *disjoint* if $\text{PROJ}_W^{X_i}(\mathbf{x}_i^{t_1:t_2}(t)) \neq \text{PROJ}_W^{X_j}(\mathbf{x}_j^{t_1:t_2}(t')) \quad \forall t,t' \in [t_1,t_2]$.

We now present the following *explainable* MMP problem.

Problem 1 (Explainable MMP): Given k robotic agents with continuous dynamics, initial states $\mathbf{x}_{1,0},\dots,\mathbf{x}_{k,0}$, goal regions $X_1^G,\dots,X_k^G$, and a bound $r \in \mathbb{N}$ on the number of segments, find a controller $\mathbf{u}_i : [t_0,t_f] \to U_i$ for every agent $i \in \{1,\dots,k\}$ and time points $t_1,\dots,t_m$, where $m \leq r$ and $t_0 < t_1 < \dots < t_{m-1} < t_m = t_f$, such that the obtained trajectory T_i takes agent i safely from $\mathbf{x}_{i,0}$ to $\mathbf{x}_i(t_f) \in X_i^G$, and the set of trajectories $T = \{T_1,\dots,T_k\}$ is segment-disjoint.



(a) 2 agent MMP (b) $\Delta t_1 = [0, 12.7]$s

(c) $\Delta t_1 = [12.7, 20.6]$s (d) $\Delta t_1 = [20.6, 81.5]$s

Fig. 2: An explanation using MAPS-RRT

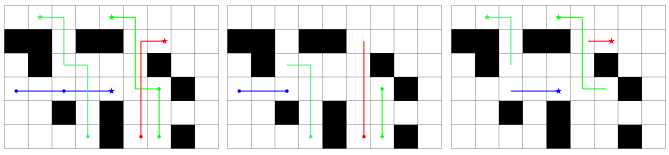
III. Approach

We present two algorithms to solve Problem 1. We start with a recently proposed meta-planner known as multi-agent plan segmenting X (MAPS-X), where X can be any centralized tree-based motion planner. It computes satisfactory motion plans based on a user defined explanation bound but suffers from state space blow-up due to the centralized nature of X. Next, we propose a decentralized and scalable explainable planner building on *conflict based search* (CBS) [10].

Centralized Explainable MMP: Recall that centralized sampling-based tree planners work by combining each agent into a single meta-agent, and then growing a dynamically feasible tree in the composed state space through repeated sampling and propagation procedures. They output a valid trajectory T for the meta-agent and individual trajectories are extracted. Out-of-the-box centralized sampling based tree planners are ill-equipped for solving Problem 1 due to their inability to control the explainability of the plan.

Recently developed MAPS-X [7] gives such planners the ability to overcome these shortcomings. As planner X grows the tree, each node is given a cost that is equivalent to the number of disjoint segments required to explain the plan up to that node. The node is only added if it can satisfactorily be explained ($m \leq r$). Plans become easier to explain as r decreases. The result is a trajectory T that solves Problem 1. Because only satisfiable nodes are added to tree, MAPS-X guarantees the number of disjoint segments that exists in its solution is less than or equal to r. Furthermore, segmentation information is already embedded in the solution, and no further computation is needed. We refer the reader to [7] for details.

Decentralized Explainable MAPF: We now propose an on-going decentralized approach that mitigates state space explosion. As a first attempt, we reduce the problem to a graph G. This is done by abstracting the workspace W into a finite set of discrete vertices V and assuming control laws that guarantee the realization of edges E with a fixed time duration in the continuous domain. The solution is a valid plan $P = \{P_1, \ldots, P_k\}$, where P_i is the plan for agent i, such that each P_i safely directs agent i from initial vertex $v_{i,0}$ to goal region $v_{i,G}$ and P can be decomposed into $m \leq r$ *vertex-disjoint* segments.



(a) Full Plan (b) Part 1 : $[0-4]$ (c) Part 2 : $[4-9]$

Fig. 3: An explanation via exp-CBS where $r = 2$.

Work [1] solves this problem using centralized graph search. We propose a more-scalable approach by extending a well-known decentralized MAPF planner known as conflict-based search (CBS) [10] to plan with explanations. CBS works by individually planning for each agent in a k agent system using graph search (e.g. A^*). It then checks the plans for conflicts (e.g. collisions). If any conflicts occur, CBS resolves them by adding constraints in the form of time-dependent obstacles and re-plans for each conflicting agent to resolve such constraints. This process repeats until a solution is found or until the search is exhausted.

We extend CBS to enable planning for explainability. We call the planner explainable-CBS (exp-CBS). Each time an agent must re-plan, our A^* search algorithm calculates the segmentation cost of each node as it plans. The result of the low-level search is a plan for agent i that minimizes segmentation given a plan for all other agents. The conflict search and resolution phases operate identical to that found in CBS. The result is a decentralized means of determining a satisfactory plan P consisting of $m \leq r$ vertex-disjoint segments.

IV. Case Studies

We present results of both approaches here. We begin with a case study in the continuous domain using MAPS-X (see [7] for more examples). We use MAPS-X with RRT [9] as planner X. We consider two agents with second-order car dynamics operating in the environment shown in Fig. 2a. The small dots represent $\mathbf{x}_{i,0}$ while the larger circles represent X_i^G. The planner found a solution in 30 seconds, which can be explained in 3 images as shown in Fig. 2b- 2d.

We now use our proposed exp-CBS planner for the problem of four agents in Fig. 1a. By setting $r = 3$, a solution was calculated in 0.05 seconds with segmentations 1b-1d. By lowering the explanation bound to $r = 2$, we receive a solution that is much easier to explain (Fig. 3b-3c) at the cost of a higher computation time of 0.3 seconds. This case study shows planning difficulty increases as segmentation bounds decrease.

V. Conclusion

This work proposes to leverage humans' cognitive process to explain MMP solutions using vertex-disjoint segments within a 2D workspace. The continuous MMP problem exploits the framework developed in [7], resulting in a centralized planner. Alternatively, our abstraction-based exp-CBS planner is decentralized and can produce plans with satisfactory explanation schemes for systems with higher number of agents than that is obtainable using centralized methods.

References

[1] Shaull Almagor and Morteza Lahijanian. Explainable multi agent path finding. In *To appear in Int'l Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2020.

[2] Rebecca Eifler, Michael Cashmore, Hoffmann Jorg, Danielle Magazzeni, and Marcel Steinmetz. Explaining the space of plans through plan-property dependencies. *Proceedings of the 2nd Workshop on Explainable Planning (XAIP)*, 2019.

[3] Mokhtar Gharbi, Juan Cortés, and Thierry Siméon. Roadmap composition for multi-arm systems path planning. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2471–2476. IEEE, 2009.

[4] Fabien Gravot and Rachid Alami. A method for handling multiple roadmaps and its use for complex manipulation planning. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 3, pages 2914–2919. IEEE, 2003.

[5] David H. Hubel and Torsten N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3), 1959. doi: 10.1113/jphysiol.1959.sp006308.

[6] Subbarao Kambhampati. Synthesizing explainable behavior for human-ai collaboration. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, page 1–2, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450363099.

[7] Justin Kottinger, Shaull Almagor, and Morteza Lahijanian. Maps-x: Explainable multi-robot motion planning via segmentation. *arXiv preprint arXiv:2010.16106*, 2020.

[8] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking clever hans predictors and assessing what machines really learn. *Nature Communications*, 10(1), Mar 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-08987-4. URL http://dx.doi.org/10.1038/s41467-019-08987-4.

[9] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[10] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2014.11.006. URL https://www.sciencedirect.com/science/article/pii/S0004370214001386.

[11] Rahul Shome, Kiril Solovey, Andrew Dobson, Dan Halperin, and Kostas E Bekris. drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning. *Autonomous Robots*, 44(3):443–467, 2020.

[12] Trevor Scott Standley. Finding optimal solutions to cooperative pathfinding problems. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[13] Roni Stern, Nathan R. Sturtevant, Dor Atzmon, Thayne Walker, Jiaoyang Li, Liron Cohen, Hang Ma, T. K. Satish Kumar, Ariel Felner, and Sven Koenig. Multi-agent pathfinding: Definitions, variants, and benchmarks. *Symposium on Combinatorial Search (SoCS)*, pages 151–158, 2019.

[14] Shiming Tang, Tai Sing Lee, Ming Li, Yimeng Zhang, Yue Xu, Fang Liu, Benjamin Teo, and Hongfei Jiang. Complex pattern selectivity in macaque primary visual cortex revealed by large-scale two-photon imaging. 2018.

[15] Matt Turek. Explainable artificial intelligence, 2018. URL https://www.darpa.mil/program/explainable-artificial-intelligence.

[16] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.

Safe Control with Neural Network Dynamic Models

Tianhao Wei
Robotics Institute
Carnegie Mellon University
Email: twei2@andrew.cmu.edu

Changliu Liu
Robotics Institute
Carnegie Mellon University
Email: cliu6@andrew.cmu.edu

***Abstract*—Safety is critical in autonomous robotic systems. A safe control law is a control law that ensures forward invariance of a safe set (a subset in the state space). It has been extensively studied regarding how to derive a safe control law with a control-affine analytical dynamic model. However, in complex environments and tasks, it is challenging and time-consuming to obtain a principled analytical model of the system. In these situations, data-driven learning is extensively used and the learned models are encoded in neural networks. How to formally derive a safe control law with Neural Network Dynamic Models (NNDM) remains unclear due to the lack of computationally tractable methods to deal with these black-box functions. In this work, we propose MIND-SIS (Mixed Integer for Neural Dynamics with Safety Index Synthesis), the first method to derive safe control laws for NNDM. The method includes two parts: 1) SIS: an algorithm for the offline synthesis of the safety index (also called as barrier function), which uses evolutionary methods and 2) MIND: an algorithm for online computation of the safe control signal, which solves a constrained optimization using a computationally efficient encoding of neural networks. It has been theoretically proved that MIND-SIS guarantees forward invariance and finite convergence. And it has been numerically validated that MIND-SIS achieves safe and optimal control of NNDM. From our experiments, the optimality gap is less than 10^{-8}, and the safety constraint violation is 0.**

I. Introduction

Safety is a major concern of autonomous robotic systems, which can usually be posed as constraint satisfaction problems, such as collision avoidance for vehicles and speed limitation of robot arms. Robot safety depends on the correct functioning of all system components, such as accurate perception, safe motion planning, and safe control. Safe control, as the last defense of system safety, has been widely studied in the context of dynamical systems [10, 2]. A safe control law is a control law that ensures the forward invariance of a subset inside the safety constraint. That means, once the state entered that subset, it will never leave. If we have a control-affine analytical dynamic model of the system, there are many methods to derive the corresponding safe control laws [15, 6, 1]. However, constructing such an analytical dynamic model for complex systems can be difficult, time-consuming, and sometimes impossible [11]. Recent works adopt data-driven approaches to learn these dynamic models, and most of the learned models are encoded in neural networks.

Examples of these neural network dynamic models (NNDM) include virtual world models in video games or dynamic models of a complicated integrated robot, etc [9, 4]. Although NNDMs can greatly alleviate human efforts in modeling, they are less interpretable than analytical models. It is more challenging to derive control laws, especially safe control laws, for these NNDMs than for analytical models.

This paper focuses on safe tracking tasks with NNDMs, which is formulated as a constrained optimization that minimizes the state tracking error given the safety constraint and the NNDM constraint. Even without the safety constraint, the tracking control with NNDMs is already challenging. Existing control-theoretical methods usually use model inverse [14] to compute the desired control inputs to track a state trajectory. Since NNDMs are complex and highly nonlinear, there is no computationally efficient method to compute its model inverse. A widely used method to control NNDMs in practice is the shooting method, which randomly generates many candidate controls and rolls out corresponding future states, then chooses the control leads to the closest state to the reference state. But the shooting method is both incomplete and sub-optimal. That means, it is not guaranteed to find a solution when the problem is feasible, and even if it finds a solution, the solution may not be the best one that optimizes the control objective. Moreover, the safety constraint adds another layer of difficulty to the problem. The robot should not only select an action that satisfies the safety constraint now but also ensure that its current action will not end up in any future state that no action is safe. This property is called *persistent feasibility*. To ensure persistent feasibility, we need to compute the control invariant set inside the original safety constraint and constrain the robot motion in this more restrictive control invariant set. For an analytical model, we can manually craft this control invariant set to meet the requirement [6]. But this becomes difficult for NNDM due to its poor interpretability.

In this work, we address these challenges by introducing an integrated method MIND-SIS to handle both the offline synthesis of the control invariant set and the online computation of the constrained optimization with NNDM constraints as shown in fig. 1. Inspired from neural network verification algorithms [8, 13], we use mixed integer programming (MIP) to encode the NNDMs in the constraint, which makes the constrained optimization much easier to solve. Most importantly, the MIP method is complete and guarantees optimality, *i.e.* it can always find the optimal control that minimizes the control objective under constraints. To synthesize the control invariant set, we first parameterize a safety index whose zero sub-level set should be the control invariant set and then use evolutionary algorithms to learn the parameters of the safety

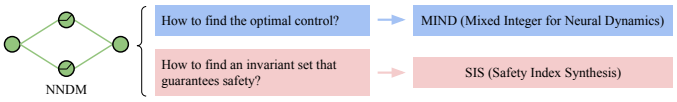


Fig. 1: Overview of MIND-SIS.

index. By substituting the original safety constraint with the new constraint from the learned safety index, the resulting control inputs from the constrained optimization will ensure forward invariance inside the safety constraint.

II. Formulation

Dynamic model: Consider a dynamical system with m_x state and m_u controls.

$$\dot{\mathbf{x}}_k = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \dot{\mathbf{x}}_k dt \tag{1}$$

where k is the time step, $\mathbf{x}_k \in X \subset \mathbb{R}^{m_x}$ is the state vector, $\mathbf{u}_k \in U \subset \mathbb{R}^{m_u}$ is the control vector, $\dot{\mathbf{x}}_k \in \mathbb{R}^{m_x}$ is the time derivative of state vector, and $\mathbf{f} : \mathbb{R}^{m_x} \mapsto \mathbb{R}^{m_u}$ is the dynamic model. We assume the legal state set X and control set U are both defined by linear constraints, which covers most cases in practice. In the NNDM case, the dynamic model $\mathbf{f}$ is encoded by a n-layer feedforward neural network.

Safety constraint: We consider the safety specification as a requirement that the system state should be constrained in a connected and closed set $\mathcal{X} \subseteq \mathbb{R}^{m_x}$ which is called the safe set. $\mathcal{X}$ should be a zero-sublevel set of a safety index function $\phi_0 : X \mapsto \mathbb{R}$, *i.e.*

$$\mathcal{X} = \{\mathbf{x} \mid \phi_0(\mathbf{x}) \leq 0\}. \tag{2}$$

There can be many different definitions of ϕ_0 for a given $\mathcal{X}$. Ideally, a safe control law should guarantee forward invariance, *i.e.* $\phi_0(\mathbf{x}_k) \leq 0 \implies \phi_0(\mathbf{x}_{k+1}) \leq 0$. And when the state is unsafe, a safe control law should drive the system back to the safe set $\mathcal{X}$. One way is to make $\mathcal{X}$ the region of attraction (ROA) of the whole space [1]: $\phi_0(\mathbf{x}_k) > 0 \implies \dot{\phi}_0(\mathbf{x}_k) \leq -\gamma$ where $\gamma > 0$. These two conditions form the safety constraint in the discrete time:

$$\phi_0(\mathbf{x}_{k+1}) \leq 0 \quad \text{or} \quad \dot{\phi}_0(\mathbf{x}_k) \leq -\gamma, \tag{3}$$

However, not all $\mathbf{x}_k \in \mathcal{X}$ has a safe control $\mathbf{u}_k \in U$ that satisfies eq. (3). We define the collection of these inevitable states as $\mathcal{X}_d$:

$$\mathcal{X}_d(\phi_0) = \{\mathbf{x} \mid \phi_0(\mathbf{x}_{k+1}) > 0 \text{ and } \dot{\phi}_0(\mathbf{x}_k) > -\gamma, \forall \mathbf{u} \in U\}. \tag{4}$$

The set $\mathcal{X}_d$ can be nonempty when the relative degree from ϕ_0 to $\mathbf{u}$ is greater than one or when the control inputs are bounded. In these cases, $\mathcal{X}$ may not be forward invariant or globally attractive. To address this problem, we want to find a subset $\mathcal{X}_s \subseteq \mathcal{X}$ and choose a control law to make the subset forward invariant and have finite time convergence. This subset is called a control invariant set.

When we use analytical models, we can manually craft a $\mathcal{X}_s$ by designing a safety index ϕ that minimizes the size of $\mathcal{X}_d(\phi)$ [6]. However, it is challenging to manually design $\mathcal{X}_s$ for NNDM due to the poor interpretability of neural networks.

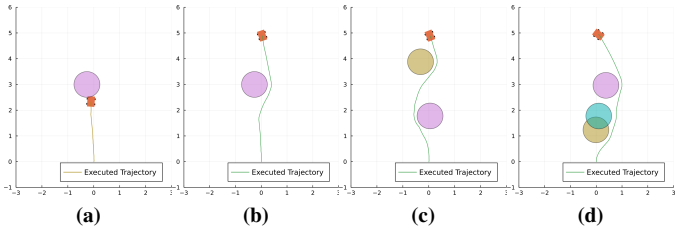


Fig. 2: MIND-SIS in collision avoidance. (a) shows the trajectory with ϕ_0. The vehicle collides with the obstacle because of infeasibility (too late to break). (b) (c) (d) show the trajectories with a synthesized safety index ϕ in different scenarios. The synthesized safety index is general enough so that it can be directly applied to scenarios with multiple obstacles without any modification.

The safe tracking problem: This paper considers the following constrained optimization for safe tracking:

$$\begin{aligned} \min_{\mathbf{u}_k, \mathbf{x}_{k+1}} \ & \|\mathbf{x}_{k+1} - \mathbf{x}^r_{k+1}\|_p \\ s.t. \ & \mathbf{x}_{k+1} = \mathbf{x}_k + f(\mathbf{x}_k, \mathbf{u}_k)dt, \quad \mathbf{u}_k \in U \\ & \phi_0(\mathbf{x}_{k+1}) \leq 0 \text{ or } \dot{\phi}_0(\mathbf{x}_k) \leq -\gamma \end{aligned} \tag{5}$$

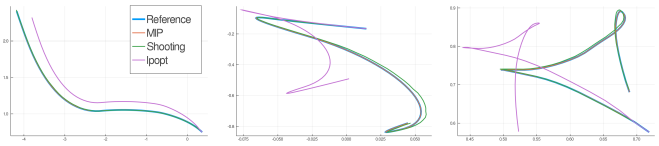
where $\|\cdot\|_p$ can be either $\ell 1$-norm or $\ell 2$-norm. This formulation is essentially a one-step model predictive control (MPC). The extension to multi-step MPC is straightforward, which we leave for future work. At a given step k, (5) is a nonlinear programming problem. However, existing nonlinear solvers have poor performance on neural networks (which will be shown in section IV). That is because neural networks are piece-wise linear, whose second-order derivatives are not informative. New techniques are needed to solve this problem efficiently.

III. Method

In this section, we discuss how to efficiently solve the constrained optimization (5) and ensure it is persistently feasible. First, we introduce MIND, a way to find the optimal control by encoding NNDM constraints as mixed integer constraints. Then we present SIS, a method to find the control invariant set by learning a new safety index ϕ that minimizes the size of $\mathcal{X}_d(\phi)$ in (4). Finally, we present the reformulated problem.

A. MIND: Encode NNDM constraints

To overcome the complexity of NNDM constraints, we first add all hidden nodes $z_{i,j}$ and $\hat{z}_{i,j}$ in the neural network as decision variables and turn (5) into an equivalent form. Nevertheless, the nonlinear non-smooth constraints introduced by the ReLU activation $z_{i,j} = \max\{\hat{z}_{i,j}, 0\}$ is still challenging to handle. Inspired by MIPVerify [13], we use mixed integer formulation to rewrite these constraints. With this encoding, the constrained optimization is converted into a MIP, which can be efficiently solved by existing solvers. But the safety constraint may be violated during execution due to infeasibility. An example is shown in fig. 2. Therefore, we introduce SIS.



(a) **(b)** **(c)**

Fig. 3: Illustration of trajectory tracking with NNDM using different optimization methods: MIND (our method), shooting method with sample size of 100, and Ipopt. (a-c) show 3 randomly generated trajectories. MIND has the smallest tracking error in all three cases.

B. SIS: Guarantee feasibility

The goal of safety index synthesis is to to find a subset $\mathcal{X}_s \subseteq \mathcal{X}$ and choose a control law to make the subset forward invariant and finite time convergent. We do this by synthesizing a safety index ϕ that minimizes the size of $\mathcal{X}_d(\phi)$. According to [6], we assume $\phi(\mathbf{x}) = \phi_0^*(\mathbf{x}) + \sum p_i(\alpha_i, \mathbf{x}) + \beta$, where $\phi_0^*(\mathbf{x})$ defines the same sublevel set as ϕ_0, $p_i(\alpha_i, \mathbf{x})$ is a higher order term of ϕ_0 that is parameterized by α_i, and β is a constant.

We use an evolutionary algorithm CMA-ES [3] to learn the parameters α_i, as well as γ from (3). To evaluate whether a set of parameters minimizes the number of inevitable states, we uniformly sample from the state space and check whether the safe control set for each state sample is empty. We prove that as long as the sampling is dense enough, we can guarantee the learned safety index is feasible for an arbitrary state.

C. MIND-SIS: Safe control with NNDM

Once the safety index is synthesized, we substitute ϕ_0 with ϕ to guarantee the feasibility. To address the nonlinearity in the safety constraint (3), we approximate it with first order Taylor expansion at the current state $\mathbf{x}_k$:

$$\phi(\mathbf{x}_{k+1}) = \phi(\mathbf{x}_k) + \nabla_{\mathbf{x}_k}\phi \cdot \dot{\mathbf{x}}_k dt + o(\|\dot{\mathbf{x}}_k dt\|), \tag{6}$$

where $\lim_{dt\to 0} o(\|\dot{\mathbf{x}}_k dt\|) = 0$. Then (5) is transformed into the following mixed integer problem:

$$\begin{aligned}
&\min_{\mathbf{u}_k, \mathbf{x}_{k+1}, \mathbf{z}_i, \delta_{i,j} \in \{0,1\}} \|\mathbf{x}_{k+1} - \mathbf{x}_{k+1}^r\|_p \\
s.t.\ &\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{z}_n dt, \mathbf{z}_0 = [\mathbf{x}_k, \mathbf{u}_k], \quad \mathbf{u}_k \in U, \\
&z_{i,j} \geq \hat{z}_{i,j}, z_{i,j} \geq 0, z_{i,j} \leq \hat{z}_{i,j} - \hat{\ell}_{i,j}(1 - \delta_{i,j}), \\
&z_{i,j} \leq \hat{u}_{i,j}\delta_{i,j}, \hat{z}_{i,j} = \mathbf{w}_{i,j}\mathbf{z}_{i-1} + b_{i,j}, \\
&\forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, k_i\} \\
&\nabla_{\mathbf{x}_k}\phi \cdot \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \leq -\min\{\frac{\phi(\mathbf{x}_k)}{dt}, \gamma\},
\end{aligned} \tag{7}$$

where $\delta_{i,j}$ are decision variables introduced by MIP. Depending on the norm $\|\cdot\|_p$, eq. (7) is either a Mixed Integer Linear Programming or Quadratic Programming, which both can be solved by existing solvers such as GLPK, CPLEX, and Gurobi.

IV. Experiment

A. Experiment set up

The evaluation is designed to answer the following questions:

1) How does our method (by solving (7)) compare to the shooting method and regular nonlinear solvers on the original problem (5) in terms of optimality and computational efficiency?
2) Does the safety index synthesis improve persistent feasibility?
3) Does our method ensure safety in terms of forward invariance and finite-time convergence?

We evaluate our method on a system with vehicle neural network models. We learn 3 different NNDMs to show the generalizability of our method. The three NNDMs are: I. 3-layer with 50 hidden neurons per layer. II. 3-layer with 100 hidden neurons per layer. III. 4-layer with 50 hidden neurons per layer.

After the training, we use the NNDM to directly control the agent to avoid model mismatch in our evaluation. The mismatch between the learned and ground truth dynamics can be caused by simplification, system noises, insufficient training, etc. When there is model mismatch, the safe control computed using the given model may be unsafe for the ground truth dynamics. Our evaluation only aims to show that the proposed method can synthesize provably safe controls efficiently under the NNDM assuming there is no model mismatch. As future work, we will extend our work to robust safe control, which is able to guarantee safety under model mismatch [7, 12].

To answer the questions we raised in the beginning, we design the following two tasks: trajectory tracking without and with safety constraints.

B. Trajectory tracking

In this task, we randomly generate 500 reference trajectory waypoints that are dynamically feasible for each NNDM. We compare our method with shooting methods with different sampling sizes and Ipopt (short for Interior Point OPTimizer), a popular nonlinear solver. We use CPLEX to solve the MIND-SIS formulation. This experiment is done on a computer with AMD® Ryzen threadripper 3960x 24-core processor, 128 GB memory. Some results are shown in fig. 3.

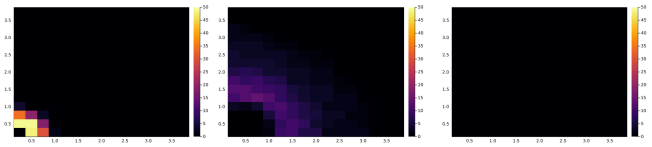
As shown in table I, MIND achieves an average tracking error less than 10^{-8}. We can consider MIND finds the optimal solution, which is a significant improvement comparing to other methods. To achieve the same tracking error as MIND, the sampling size and computation time of the shooting method will be unacceptably large. Ipopt ends quickly because it gets stuck at local optima soon after the beginning. And iteratively warm starting Ipopt with its own solution does not help it get out of local optima. This suggests existing nonlinear solvers do not work well with neural network constraints.

C. Trajectory tracking under safety constraints

This task considers two safety constraints corresponding to different scenarios: collision avoidance and safe following. When infeasibility happens, the agent reuses the control of the last time step.

Method	NNDM 1			NNDM 2			NNDM 3		
	Mean	Std	Time (s)	Mean	Std	Time (s)	Mean	Std	Time (s)
MIND	$< 10^{-8}$	$< 10^{-7}$	36.4	$< 10^{-8}$	$< 10^{-7}$	83.8	$< 10^{-8}$	$< 10^{-7}$	123.5
Shooting-10^3	0.129	0.080	2.1	0.128	0.080	10.0	0.128	0.080	2.9
Shooting-10^4	0.041	0.026	20.9	0.041	0.026	100.2	0.041	0.026	28.3
Shooting-10^5	0.012	0.007	208.4	0.012	0.007	1006.3	0.012	0.007	282.2
Ipopt	1.871	0.626	3.2	1.852	0.619	4.0	1.865	0.623	3.3
Ipopt-iterative	1.871	0.626	13.3	1.852	0.619	16.3	1.865	0.623	13.5

TABLE I: Average tracking error and computation time for different methods in the trajectory tracking task (without safety constraint). The table shows mean and standard deviation of the average tracking error. The number after "Shooting" denotes the sampling size. "Ipopt-iterative" represents iteratively solving the problem by warm starting Ipopt with its own solution. Our method can always find the optimal solution, therefore achieves almost zero tracking error. The actual trajectories are illustrated in fig. 3.



(a) **(b)** **(c)**

Fig. 4: Inevitable states distribution. An obstacle is located at $(0, 0)$. Each grid in the graph corresponds to a (p_x, p_y) location. We sample 100 states at each location. The color denote how many states at this location are inevitable states. (a) shows the original safety index ϕ_0 has a hard boundary, which means when the agent is near the obstacle, no matter with what state, it can barely find a feasible control. (b) shows a hand constructed safety index ϕ^h only can not find a feasible control for part of the states around the obstacle. (c) shows the learned index ϕ has no inevitable state, thus can always find a safe control.

1) Collision avoidance: Collision avoidance is one of the most common requirement in real-world applications. Such as in vehicle driving, and robot operation [5]. The collision avoidance constraint is usually defined as $\phi_0(\mathbf{x}) = d_{min} - d(\mathbf{x}) < 0$, where d_{min} is the acceptable minimal distance between the agent and the obstacle, $d(\mathbf{x})$ is the relative distance from the agent to the obstacle. But this constraint usually can not guarantee persistent feasibility. Therefore, we synthesize a safety index $\phi(\mathbf{x})$ that guarantees persistent feasibility. We design the safety constraint to be of the form:

$$\phi(\mathbf{x}) = d_{min}^{\alpha_1} - d(\mathbf{x})^{\alpha_1} - \alpha_2 \dot{d}(\mathbf{x}) + \beta,$$

where $\dot{d}(\mathbf{x})$ is the relative velocity from the agent to the obstacle, α_1, α_2 and β are parameters to learn. This form guarantees forward invariance and finite time convergence as shown in [6], To demonstrate the effect of the synthesized safety index. We visualize the behavior of the agent with ϕ_0 and ϕ in fig. 2. The figure also shows that the synthesized safety index can be directly applied to unseen multi-obstacle scenarios without any change.

To learn a safety index that minimizes the size of the inevitable state set $\mathcal{X}_d$, we use CMA-ES to optimize the parameters. For each set of parameters. We uniformly sample 40000 states to compute the size of the $\mathcal{X}_d$. fig. 4 shows the distribution of inevitable states of the original safety index ϕ_0, a synthesized safety index ϕ^h with hand designed parameters, and a synthesized safety index ϕ^l with learned parameters. The learned safety index ϕ^l has no inevitable states, thus can always find a safe control.

We evaluate these safety indices on 100 randomly generated collision avoidance tasks. The agent has to track a trajectory in each task while avoiding collision (keep $\phi_0 \leq 0$). We consider a task succeeds if there is no collision or inevitable states on the trajectory. The evaluation results are shown in table II. We can see that the learned safety index achieves 0% ϕ_0-violation rate and 0% infeasible rate.

Metric	Collision avoidance			Safe following		
	ϕ_0	ϕ^h	ϕ^l	ϕ_0	ϕ^h	ϕ^l
Success rate	0%	72%	100%	0%	82%	100%
ϕ_0-violation rate	100%	0%	0%	100%	0%	0%
Infeasible rate	100%	28%	0%	100%	18%	0%

TABLE II: Performance comparison of the original safety index ϕ_0, a synthesized safety index ϕ^h with manually tuned parameters, and a synthesized safety index ϕ^l with learned parameters on 100 randomly generated tasks. We consider one trial successful if there is no safety violation or infeasible state. We can see that ϕ_0 always violates the constraints due to infeasibility (due to our choice of the initial state). ϕ^h guarantees the safety but fail to find controls for some states due to infeasibility. But the learned index ϕ^l can always find a control and guarantees safety.

2) Safe following: The safe following constraint appears in the case that an agent is following a target while keeping a safe distance to the target. Such as in adaptive cruise control, nap-of-the-earth flying, etc. The safety index can be defined as $\phi_0(\mathbf{x}) = (d(\mathbf{x}) - d_{min})(d(\mathbf{x}) - d_{max})$. We design the safety constraint to be of the form:

$$\begin{aligned}\phi(\mathbf{x}) =& (d(\mathbf{x}) - d_{min})^{\alpha_1}(d(\mathbf{x}) - d_{max})^{\alpha_1} \\ &+ \alpha_2[d(\mathbf{x})\dot{d}(\mathbf{x}) + \dot{d}(\mathbf{x})(d_{min} + d_{max})] + \beta\end{aligned} \tag{8}$$

Where α_1, α_2, β are parameters to learn, The learning process and the evaluation is similar to the collision avoidance case. The evaluation results are shown in table II. The learned index achieves 0% ϕ_0-violation rate and 0% infeasible rate.

V. Conclusion

In this work, we propose MIND-SIS, the first method to derive safe control law for NNDM. MIND finds the optimal control for NNDM, and SIS synthesizes a safety index that guarantees forward invariance and finite time convergence. We provide theoretical guarantees that our method can achieve optimality and ensure feasibility.

References

[1] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[2] Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[3] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[4] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

[5] Hsien-Chung Lin, Changliu Liu, Yongxiang Fan, and Masayoshi Tomizuka. Real-time collision avoidance algorithm on industrial manipulators. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1294–1299. IEEE, 2017.

[6] Changliu Liu and Masayoshi Tomizuka. Control in a safe set: Addressing safety in human-robot interactions. In *ASME 2014 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2014.

[7] Changliu Liu and Masayoshi Tomizuka. Safe exploration: Addressing various uncertainty levels in human robot interactions. In *2015 American Control Conference (ACC)*, pages 465–470. IEEE, 2015.

[8] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, Mykel J Kochenderfer, et al. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4, 2020.

[9] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[10] Mitio Nagumo. Über die lage der integralkurven gewöhnlicher differentialgleichungen. *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series*, 24: 551–559, 1942.

[11] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[12] Charles Noren and Changliu Liu. Safe adaptation in confined environments using energy functions. *arXiv preprint arXiv:1912.09095*, 2019.

[13] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

[14] Deepak Tolani, Ambarish Goswami, and Norman I Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, 62(5):353–388, 2000.

[15] Tianhao Wei and Changliu Liu. Safe control algorithms using energy functions: A uni ed framework, benchmark, and new directions. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 238–243. IEEE, 2019.