Advances in Urban Resilience and Sustainable City Design
2024 (https://orientreview.com/index.php/aurscd-journal)

Reinforcement Learning-based Approaches for Improving Safety and Trust in Robot-to-Robot and Human-Robot Interaction

Mahmoud Abouelyazid [1]

Abstract

The increasing deployment of robotic systems in various domains has emphasized the need for ensuring safety and trust in robot-to-robot and human-robot interaction. Existing approaches often prioritize performance metrics without explicitly addressing safety constraints or trust-building behaviors. The transfer of learned policies from simulation to real-world environments remains a challenge due to the complexities and uncertainties of real-world scenarios. This study aims to address the gap in the literature by proposing a framework that integrates multiple RL techniques to enhance safety and trust in robotic systems. The proposed framework contains several key components. First, constrained RL approaches, such as Constrained Policy Optimization or Safe Exploration via Constrained Policy Optimization, are employed to incorporate safety constraints into the learning process, ensuring that the learned policies adhere to specified safety requirements. Second, a reward shaping mechanism is designed to include metrics for quantifying safety and trust, enabling the learning of behaviors that prioritize safety and enhance trust in human-robot interaction. Third, active learning techniques are integrated with RL to enable human-in-the-loop learning, allowing the robot to efficiently learn from human feedback and demonstrations. Fourth, domain randomization techniques are applied to improve the sim-to-real transfer of learned policies, making the policies more robust and adaptable to real-world scenarios. Finally, the proposed framework is extended to multi-robot systems by employing multi-agent RL algorithms that enable safe and efficient coordination through learned communication protocols. The study provides a theoretical and algorithmic foundations for the proposed framework, discussing the benefits and challenges of integrating these RL techniques for safety and trust in robotics applications. This research aims to contribute to the advancement of RL techniques for ensuring safety and trust in robotic systems. The proposed framework can be applied as a conceptual foundation for future research and development of more reliable, trustworthy, and human-friendly robotic systems. Although experimental validation for the study is beyond the scope of this study, it highlights the possibility of integrating RL techniques to enhance safety and trust in robot-to-robot and human-robot interaction for further empirical investigations and practical applications.

1. Introduction

Robots are experiencing rapid growth in sophistication and diversity. While autonomous vehicles, drones, and automated vacuum cleaners are widely recognized, there are many lesser-known robots, such as those used in laboratories [1], industrial settings [2], ocean and space exploration, search-and-rescue operations, and even surgery. The advancement of robotics is making various aspects of life easier, cleaner, healthier, and more prosperous [3].

Robots are not a monolithic entity but rather a combination of various technologies that are progressing at different rates. The development of robotics relies on advancements in several key areas, including sensors, control systems, actuators, and materials science. Laser systems with enhanced range and angle resolution are among the sensor technologies that are driving progress in robotics. These advanced sensors enable robots to perceive and interact with their environment more accurately and efficiently [4].

Control systems, such as cloud-based robots and predictive control, are another component of robotics. These systems allow for more responsive

[1] Purdue University

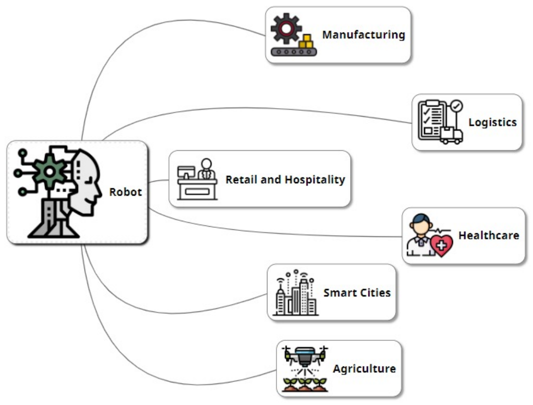

Figure 1: Application of robot in various sectors

robot behavior, enabling them to adapt to changing conditions and make decisions based on real-time data. Actuators, like dexterous grippers, are also essential for enabling robots to physically interact with their surroundings and perform tasks with greater precision and flexibility. Additionally, advancements in materials science are helping robots to become more energy-efficient by allowing them to harvest energy from their environment. As these various technologies continue to evolve and converge, the capabilities of robots will likely expand, leading to new applications and opportunities across a wide range of industries [2].

Safety is a fundamental human need and an indispensable aspect of our daily existence [5]. Consequently, ensuring the physical safety of people should be the top priority in the creation of any robotic system intended for human interaction. Recognizing this, there is already a significant body of research dedicated to studying and enhancing the safety of human-robot interaction (HRI). As we push the boundaries of what robotic assistants can do, we must never lose sight of our responsibility to make them safe and deserving of the trust we place in them. Only by treating safety as an essential design requirement, rather than an optional feature or afterthought, can we hope to create robotic helpers that truly benefit humanity without putting us at risk.

To fully realize the many benefits that autonomous robotic assistants offer, it is necessary to accept them acting in close proximity to people, sharing space, e.g., at work or at home, and even jointly manipulating objects. From personal carrier robots to robotic co-workers in a flexible manufacturing assembly line, the required physical closeness adds a new dimension of risk and potential dangers for users. Traditionally, there was a clear spatial and physical separation between people and robots, as reflected by current standards such as ISO 13849-1 (Safety of machinery – Safety-related parts of control systems). Safety barriers were put in place to prevent robots from making contact with people. The most prominent examples for the safety-by-separation paradigm can be found in conventional car production plants. Currently, robots in an industrial setting need to reduce their operational speed and may even need to fully stop in close proximity of people, as suggested by the technical specification ISO/TS 15066:2016 (Robots and robotic devices – Collaborative robots) and required by the standard ISO 13482:2014 (Robots and robotic devices – Safety requirements for personal care robots).

The development of new safety strategies that allow for close collaboration between humans and robots while prioritizing user safety is essential. This requires a shift from the traditional safety-by-separation approach to a more nuanced, context-aware safety framework that leverages advanced sensors, intelligent control systems, and adaptive behaviors to ensure safe and efficient human-robot interaction in various settings. The implementation of such strategies will enable the realization of the full potential of autonomous robotic assistants while maintaining a high level of safety for users

2. Robot-to-Robot Interaction

The deployment of interconnected robots, as opposed to individual robots operating independently is enabling effective coordination among the robots. This coordination allows for efficient distribution of tasks, comprehensive spatial coverage, and specialization based on the capabilities of each robot. A growing range of applications, such as logistics, resource distribution, transportation systems, manufacturing, and agriculture, rely on these networked robot systems. The success of these applications hinges on the seamless orchestration of robots across time and space, enabling them to collaborate towards common high-level objectives, prevent conflicts in shared environments, and exchange information for distributed computing purposes. Effective communication and the mutual sharing of state and control information are crucial for facilitating these interactions.

Tasks can be accomplished more efficiently and in a shorter timeframe by utilizing multiple robots concurrently. The use of heterogeneous robots with diverse capabilities allows for cost-effective solutions, as each robot can be assigned specific components of the task that align with its particular strengths. This approach is advantageous when dealing with tasks that are inherently distributed over a wide area, as multiple robots can effectively cover the entire region. The presence of multiple robots capable of performing similar processes provides fault tolerance, ensuring that if any individual robot fails, the others can compensate for the loss and maintain the overall functionality of the system.

Several major architectures have been identified in the field of Multi-Robot Systems (MRS) that significantly impact the robustness [6], reliability, and scalability of the system, In Figure 2. These architectures are determined by the strategy employed to make decisions, manage interactions between robots, and generate group behavior within the team.

The centralized approach is one such architecture, where a single point of control manages the behavior of all the robots in the team. This architecture suffers from the single point of failure problem, which can reduce its reliability. Scalability is also diminished because the central controller must constantly monitor the state of all team members, leading to numerous message exchanges in addition to the control messages sent back to individual robots to control their actions [7].

The hierarchical approach organizes robots in a command and control hierarchy similar to that in the military. A robot controls a group of other robots, and each of those robots, in turn, controls another group of robots. This pattern can continue for several levels down the hierarchy, depending on the size of the network. This approach is highly scalable and can be appropriate for applications with a large number of robots. However, it has reduced reliability due to the considerable vulnerability in handling failures of robots at higher levels in the hierarchy [8].

The decentralized architecture is the most common category for MRS systems. Robots take actions based on their own local view, following certain strategic guidelines and goals for the team. This model is characterized by its robustness and ability to adjust to failures since no centralized control is used. Maintaining synchronization and coherency among the robots can be challenging. Coordinating actions when mission objectives change can also be a complex task.

The hybrid approach combines local decentralized

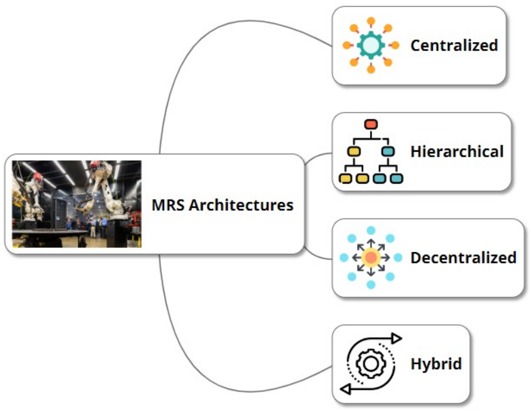

Figure 2: Types of Multi-Robot Systems (MRS) Architecture

control, which provides robustness, with hierarchical control to achieve global synchronization and coordination of actions, goals, and tasks. This hybrid strategy is employed in many MRS systems that require scalability due to the large size of the network, as well as the ability to make quick decisions at the local level to achieve better performance and faster reactions to local events and failures.

3. Human-Robot Interaction

Human-Robot Interaction (HRI) is a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans. Interaction requires communication between robots and humans. Communication between a human and a robot may take several forms, largely influenced by whether the human and the robot are in close proximity to each other or not. Communication and interaction can be separated into two general categories: *Remote interaction* – The human and the robot are not co-located and are separated spatially or even temporally. For example, the Mars Rovers are separated from earth both in space and time. *Proximate interactions* – The humans and the robots are co-located. For example, service robots may be in the same room as humans.

Modalities of Human-Robot Interaction

3.1 Vision Systems in Robots

Visual perception provides the most important information to robots, allowing them to achieve successful interaction with human partners. This information can be used in various tasks, such as navigation, obstacle avoidance, detection, understanding, and manipulation of objects, and assigning meanings to a visual configuration of a scene. The vision has been used for estimating the 3D position and orientation of a user in an environment, estimating distances between a robot and users, tracking human targets and obtaining their poses, and understanding human behavior to contribute to the cohabitation between assistive robots and humans. The vision has also been used in various other applications, such as recognizing patterns and figures in exercises in a teaching assistance context in a high school, detecting and classifying waste material as a child would do, and detecting people entering a building for possible interaction. Moreover, vision has been used for medication sorting, taking into account pill types and numbers, sign recognition in a sign tutoring task with deaf or hard of hearing children, and as part of a platform used for cognitive stimulation in elderly users with mild cognitive impairments.

3.2 Conversational Systems in Robots

Some applications of social robotics involve robots taking vocal commands without generating a vocal reply. However, interactions can be made richer when the robot can engage in conversations. A typical social robot with autonomous conversation ability must have the capacity to acquire sound signals, process them to recognize speech, recognize the whole sequence of words pronounced by the human interlocutor, formulate an appropriate reply, synthesize the sound signal corresponding to the reply, and emit this signal using a loudspeaker. The core component of this ability is the recognition of word sequences and the generation of reply sequences. This can rely on a learning stage where the system acquires the experience of answering word sequences by observing a certain number of conversations that are mainly between humans. Techniques used in this area involve word and character embeddings, and learning through recurrent neural network (RNN) architectures, long short-term memory networks (LSTM), and gated recurrent units (GRU). Not all social robotic systems with conversational capacities have the same levels of complexity, as some use limited vocabularies in their verbal dialogues.

3.3 Expressions and Gestures

A social robot requires more capacities to increase engagement and realism in the interaction with a human, aside from the ability to process and generate sequences of words. This can be done through speech-accompanying gestures and facial expressions. Facial expression has an important role in communication between humans because it is rich in information, together with gestures and sound. There are six main emotions associated with distinctive facial expressions. Different examples can be found for applications of gestures and expressions in social robotics. Gestures have been combined with verbal dialogue and screen display for health data acquisition in hospitals with Pepper. A robot with the ability to display facial expressions was used in studies related to storytelling robots, focusing on the roles of emotional facial display, contextual head movements, and voice acting.

4. Literature Gap

While significant progress has been made in applying reinforcement learning (RL) to robotics, there is still a need for further research on ensuring safety and trust in robot-to-robot and human-robot interaction. Existing RL approaches often focus on optimizing performance metrics without explicitly addressing safety constraints or trust-building behaviors. Moreover, the transfer of learned policies from simulation to real-world environments remains a challenge, as simulations may not capture all the complexities and uncertainties of real-world scenarios. There is a gap in the literature regarding the development of RL techniques that can effectively incorporate safety and trust metrics, enable human-in-the-loop learning, and ensure robust sim-to-real transfer [9], [10], [11], [12].

5. Contribution of the Study

This study aims to address the identified literature gap by proposing a framework that integrates various RL techniques to improve safety and trust in robot-to-robot and human-robot interaction. The main contributions of this study are as follows: 1. Proposing a constrained RL approach that incorporates safety constraints into the learning process, ensuring that the learned policies adhere to specified safety requirements. 2. Designing a reward shaping mechanism that includes metrics for quantifying safety and trust, enabling the learning of behaviors that prioritize safety and enhance trust in human-robot interaction. 3. Integrating active learning techniques with RL to enable human-in-the-loop learning, allowing the robot to efficiently learn from human feedback and demonstrations. 4. Applying domain randomization techniques to improve the sim-to-real transfer of learned policies, making the policies more robust and adaptable to real-world scenarios. 5. Extending the proposed framework to multi-robot systems by employing multi-agent RL algorithms that enable safe and efficient coordination through learned communication protocols. 6. Providing a theoretical foundation and algorithmic insights for the proposed framework, discussing the potential benefits and challenges of integrating these RL techniques for safety and trust in robotics applications. The proposed framework can be used for future research and development of more reliable, trustworthy, and human-friendly robotic systems. While the study does not include experimental validation, it provides a conceptual foundation and highlights the potential of integrating RL techniques to enhance safety and trust in robot-to-robot and human-robot interaction.

6. Reinforcement learning (RL) to improve safety and trust in robot-to-robot and human-robot interaction

6.1 Safe exploration using constrained RL

Constrained Markov Decision Processes:

We denote by $P(S)$ the set of probability distributions on a set S. A Markov Decision Process is defined as a tuple (S, A, T, R, s_0), where S is a set of states, A is a set of actions, $T : S \times A \to P(S)$ is a transition probability function, $R : S \times A \to [0, R_{max}]$ is a reward function, and s_0 is the initial state. For simplicity, we assume a deterministic reward function and initial state, but our results can be generalized. A Constrained Markov Decision Process (CMDP) is an MDP with additional constraints that must be satisfied, thereby restricting the set of allowable policies for the agent. Formally, a CMDP is defined as a tuple $(S, A, T, R, s_0, C, C_{max})$, where $C : S \to [0, C_{max}]$ is the cost function and $C_{max} \in \mathbb{R}_{\geq 0}$ is the maximum allowed cumulative cost. Note that the cost is only a function of states rather than state-action pairs [13]. We consider a finite time horizon H after which the episode terminates. The set of feasible policies that satisfy the CMDP is a subset of stationary policies: $\Pi_C := \pi : S \to P(A) | \mathbb{E}[\sum_{t=0}^{H} C(s_t) | s_0, \pi] \leq C_{max}$

The expected sum of rewards following a policy π from an initial state s is given by the value function $V^\pi(s) = \mathbb{E}[\sum_{t=0}^{H} R(s_t, a_t) | \pi, s_0 = s]$. Similarly, the expected sum of costs is given by the cost value function $V_C^\pi(s) = \mathbb{E}[\sum_{t=0}^{H} C(s_t) | \pi, s_0 = s]$. The optimization problem in the CMDP is to find the feasible policy that maximizes expected returns from the initial state s_0, i.e., $\pi^* C = \arg\max \pi \in \Pi_C V^\pi(s_0)$ An important point to note about CMDPs is that the cost function depends on immediate states, but the constraint is cumulative and depends on the entire trajectory, making the optimization problem much more challenging [14].

In the case of MDPs, where a model of the environment is not known or is not easily obtained, it is still possible for the agent to find the optimal policy using Temporal Difference (TD) methods. In general, these methods update the estimates of the value functions via bootstraps of previous estimates on sampled transitions. In the on-policy setting, we alternate between estimating the state-action value function Q^π for a given π and updating the policy to be greedy with respect to the value function.

Constrained Markov Decision Processes (CMDPs) provide a powerful framework for formulating Reinforcement Learning (RL) problems with safety constraints. Through incorporating constraints on the expected cumulative cost of unsafe actions, CMDPs enable the development of algorithms that ensure the robot explores its environment while satisfying safety requirements.

In the context of CMDPs, the objective is to find a policy $\pi^* C$ that maximizes the expected return while ensuring that the expected cumulative cost, $V_C^\pi(s_0)$, remains below a specified threshold $Cmax$. This constrained optimization problem can be formally expressed as: $\pi^* C = \arg\max \pi \in \Pi_C V^\pi(s_0)$, subject to $V_C^\pi(s_0) \leq C_{max}$ where Π_C is the set of feasible policies that satisfy the cost constraint [15].

Algorithms like Constrained Policy Optimization (CPO) and Safe Exploration via Constrained Policy Optimization (SCPO) have been developed to tackle this constrained optimization problem efficiently. These algorithms optimize the policy within the constrained solution space, guaranteeing that the learned behavior adheres to the specified safety constraints. CPO is an actor-critic method that extends the Trust Region Policy Optimization (TRPO) algorithm to handle safety constraints [16]. It approximates the constrained optimization problem using a surrogate objective and a surrogate constraint, which are based on the first-order approximations of the value function and the cost value function, respectively.

SCPO, on the other hand, builds upon CPO and incorporates a safe exploration mechanism. It introduces a notion of a safe set, which is a subset of the state space where the agent can safely explore without violating the constraints. SCPO maintains an estimate of the safe set and adapts the exploration strategy accordingly. Inside the estimated safe set, the agent explores using a standard RL algorithm like Proximal Policy Optimization (PPO) [16]. Outside the safe set, the agent follows a safe backup policy that minimizes the risk of constraint violation. Both CPO and SCPO have theoretical guarantees on the satisfaction of safety constraints during training. They ensure that the expected cumulative cost of the learned policy remains below the specified threshold, C_{max}, with high probability. This is achieved by carefully balancing the exploration-exploitation trade-off and adjusting the policy updates based on the estimated safety constraints. CMDPs provide a principled way to incorporate safety constraints into the RL framework. Algorithms like CPO and SCPO leverage the CMDP formulation to optimize policies that maximize expected returns while strictly adhering to the specified safety constraints. By employing these algorithms, robots can explore their environment and learn effective behaviors while ensuring that they operate within the defined safety boundaries.

Algorithm 1 Safe Policy Iteration Algorithm

```
Input: $\pi_0, C_{\text{max}}, \alpha_\pi, \alpha_V, \alpha_Q, \gamma, \beta$
Output: Optimal safe policy $\pi^*$
Initialize $V_x \leftarrow 0, V_u \leftarrow 0, Q_x \leftarrow 0, Q_u \leftarrow 0, S \leftarrow \{s_0\}$
for iteration $k = 0, 1, 2, ...$ do
    • Collect trajectories using $\pi_k$
    • Update value functions using collected data
    • Solve constrained optimization problem to update policy: $\pi' \leftarrow \arg\max_\pi \mathbb{E}[\sum_t r(s_t, a_t)|\pi]$, subject to $\mathbb{E}[\sum_t c(s_t)|\pi] \leq C_{\text{max}}$
    • Update policy: $\pi_{k+1} \leftarrow (1 - \alpha_\pi) \cdot \pi_k + \alpha_\pi \cdot \pi'$
    • Expand safe set: $S' \leftarrow \{s \in S | V_u(s) \leq \beta \cdot C_{\text{max}}\}$
    • for $s \in S'$ do
        – for $a \in A$ do
            * $s' \leftarrow$ env.step$(s, a)$
            * if $V_u(s') \leq \beta \cdot C_{\text{max}}$ then $S \leftarrow S \cup \{s'\}$
end for
Return optimal safe policy $\pi^*$ (last policy $\pi_k$)
```

6.2 Reward shaping with safety and trust metrics

A reward design problem is defined as a tuple $P = \langle r^*, \tilde{M}, \tilde{R}, \pi(\cdot|\tilde{r}, \tilde{M})\rangle$, where: • r^* is the true reward function. • $\tilde{M}$ is a world model. • $\tilde{R}$ is a set of proxy reward functions. • $\pi(\cdot|\tilde{r}, \tilde{M})$ is an agent model that defines a distribution on trajectories given a (proxy) reward function and a world model. The designer believes that an agent, represented by the policy $\pi(\cdot|\tilde{r}, \tilde{M})$, will be deployed in $\tilde{M}$. She must specify a proxy reward function $\tilde{r} \in \tilde{R}$ for the agent. Her goal is to specify $\tilde{r}$ so that $\pi(\cdot|\tilde{r}, \tilde{M})$ obtains high reward according to the true reward function r. We let $\tilde{w}$ represent weights for the proxy reward function and w represent weights for the true reward function. In this work, our motivation is that system designers are fallible, so we should not expect that they perfectly solve the reward design problem. Instead, we consider the case where the system designer is approximately optimal at solving a known RDP, which is distinct from the MDP that the robot currently finds itself in. By inverting the reward design process to infer (a distribution on) the true reward function r^*, the robot can understand where its reward evaluations have high variance and plan to avoid those states. We refer to this inference problem as the inverse reward design problem: Definition 3. (Inverse Reward Design) The inverse reward design (IRD) [17] problem is defined by a tuple $\langle R, \tilde{M}, \tilde{R}, \pi(\cdot|\tilde{r}, \tilde{M}), \tilde{r}\rangle$, where: • R is a space of possible reward functions. • $\langle -, \tilde{M}, \tilde{R}, \pi(\cdot|\tilde{r}, \tilde{M})\rangle$ partially specifies an RDP P, with an unobserved reward function $r^* \in R$. • $\tilde{r} \in \tilde{R}$ is the observed proxy reward that is an (approximate) solution to P. In solving an IRD problem, the goal is to recover r. We will explore Bayesian approaches to IRD, so we will assume a prior distribution on r and infer a posterior distribution on r given $\tilde{r}$: $P(r|\tilde{r}, \tilde{M})$.

Designing a suitable reward function is crucial for ensuring that a reinforcement learning (RL) agent learns behaviors that are not only optimal for the task at hand but also safe and trustworthy. In the context of human-robot interaction (HRI), the reward function should incorporate metrics that quantify safety and trust, encouraging the robot to prioritize these aspects while learning its policy. One approach to incorporating safety into the reward function is to include penalties for unsafe actions. For example, if the robot is navigating in a shared environment with humans, the reward function can assign a large negative reward for collisions or near-misses with humans or objects. This penalty can be proportional to the severity of the collision or the proximity to the human, encouraging the robot to maintain a safe distance and avoid dangerous situations. Similarly, if the robot is assisting humans in a task that requires physical interaction, the reward function can penalize actions that apply excessive force or cause discomfort to the human. In addition to safety, trust is a key factor in HRI. To enhance trust, the reward function can include rewards for behaviors that are transparent, predictable, and aligned with human expectations. For instance, the robot can be rewarded for providing clear and timely communication about its intentions, current state, and any potential risks or uncertainties. This can be achieved by incorporating metrics that measure the quality and frequency of the robot's communication, such as the information gain or the human's understanding of the robot's actions. Moreover, the robot can be rewarded for executing movements that are smooth, legible, and predictable, as sudden or erratic motions can be unsettling and diminish trust. To ensure that the learned behavior aligns with human preferences and values, techniques like Inverse Reward Design (IRD) can be employed. IRD is a framework that learns the reward function from human demonstrations, rather than relying on a manually specified reward signal. In this approach, the human expert demonstrates the desired behavior in various scenarios, and the algorithm infers the underlying reward function that explains the observed demonstrations. By learning from human examples, IRD can capture the implicit priorities and trade-offs that humans make when balancing safety, trust, and task performance. Formally, IRD can be formulated as an inverse reinforcement learning (IRL) problem. Given a set of human demonstrations $\mathcal{D} = (x_0, a_0), (x_1, a_1), \ldots, (x_T, a_T)$, where x_t and a_t denote the state and action at time step t, the objective is to find a reward function $R_\theta(x, a)$ parameterized by θ that maximizes the likelihood of the demonstrations:

$$\max_\theta \prod_{(x,a)\in\mathcal{D}} P(a|x; R_\theta)$$

where $P(a|x; R_\theta)$ is the probability of taking action a in state x under the optimal policy derived from the reward function R_θ. Once the reward function is learned, it can be used to train an RL agent that exhibits safe and trustworthy behavior. The agent's policy can be optimized using standard RL algorithms, such as Q-learning or policy gradient methods, with the learned reward function guiding the exploration and exploitation process. designing a reward function that incorporates safety and trust metrics, along with employing techniques like Inverse Reward Design, can help ensure that the learned behavior of an RL agent aligns with human preferences and values. By prioritizing safety and trust in the learning process, robots can be trained to operate in a manner that is not only efficient but also acceptable and comfortable for humans in shared environments.

Algorithm 2 Interactive Learning with Safety and Trust Monitoring

```
for each evaluation episode:
  • $x \leftarrow x_0$
  • for $t = 0$ to $T - 1$ do:
      – $a \leftarrow \arg\max_a \pi(a|x)$
      – $x' \leftarrow$ Execute($a$)
      – MonitorSafetyTrust($x, a$)
      – $x \leftarrow x'$
  • $feedback \leftarrow$ GetHumanFeedback()
if not SafeAndTrusted($feedback$) then
  • $R_\theta \leftarrow$ UpdateReward($R_\theta, feedback$)
  • $R_\theta \leftarrow$ InverseRewardDesign($D, R_\theta, X, A, \alpha, \beta$, num_iter)
  • $\pi \leftarrow$ InitializePolicy()
```

6.3 Human-in-the-loop learning with active learning

Active Preference Learning (APL) Active Preference Learning (APL) is a subfield of machine learning that focuses on actively querying users to learn their preferences or reward functions [18]. The goal is to efficiently gather informative feedback and converge towards the true preferences with minimal interactions. Mathematically, let $\mathcal{X}$ be the set of possible outcomes or items, and $\mathcal{R}$ be the set of possible reward functions mapping from $\mathcal{X}$ to real numbers. APL maintains a belief distribution $p(r)$ over the reward functions and updates it based on user feedback using Bayesian inference. At each iteration t, the learner selects a query q_t to elicit feedback f_t from the user, which is then used to update the belief distribution: $p(r|f_1, \ldots, f_t) \propto p(f_t|r) \cdot p(r|f_1, \ldots, f_{t-1})$. The query selection strategy aims to maximize information gain, such as through uncertainty sampling or expected improvement. APL employs various feedback mechanisms, including pairwise comparisons, where the user provides relative preferences between pairs of outcomes, and ratings or scores assigned to individual outcomes. The learned reward function can be used to make recommendations, optimize decision-making, or guide interactive systems. Technical challenges in APL include handling noisy and inconsistent feedback, scaling to large outcome spaces, and ensuring the safety and robustness of the learned preferences. Researchers have proposed different approaches to address these challenges, such as using probabilistic models, regularization techniques, and active learning algorithms that balance exploration and exploitation. APL has found applications in diverse domains, including personalized recommendations, robotic decision-making, and interactive design optimization. ***Active Reward Learning (ARL)*** Active Reward Learning is a machine learning paradigm that aims to learn a reward function through active interaction with an environment or user [19]. Unlike traditional reinforcement learning, where the reward function is assumed to be known, ARL seeks to estimate the underlying reward function by actively querying the user or environment for feedback. Mathematically, let $\mathcal{S}$ be the set of states, $\mathcal{A}$ be the set of actions, and $\mathcal{R}$ be the set of possible reward functions mapping from state-action pairs to real numbers. The goal of ARL is to learn the true reward function $r^* \in \mathcal{R}$ that accurately captures the desired behavior or preferences. ARL maintains a belief distribution $p(r)$ over the reward functions and updates it based on the collected feedback using Bayesian inference. At each step t, the learner selects an action a_t based on its current belief and the observed state s_t, and receives feedback f_t from the user or environment. The feedback is used to update the belief distribution: $p(r|f_1, \ldots, f_t) \propto p(f_t|r, s_t, a_t) \cdot p(r|f_1, \ldots, f_{t-1})$. The action selection strategy balances exploration and exploitation to gather informative feedback efficiently. ARL employs various techniques to learn the reward function based on the collected feedback. One approach is Inverse Reinforcement Learning (IRL), where the goal is to infer the reward function that explains the observed behavior or demonstrations. IRL methods, such as maximum entropy IRL or Bayesian IRL, estimate the reward function by finding the one that maximizes the likelihood of the observed data under a model of rational behavior. Another approach is Preference-based Reinforcement Learning, where the learner directly learns a policy that maximizes the user's preferences without explicitly estimating the reward function. This can be achieved through methods like preference-based policy search or preference-based value iteration. ARL faces technical challenges, such as dealing with noisy and inconsistent feedback, handling large state and action spaces, and ensuring the safety and stability of the learned reward function. Researchers have proposed various techniques to address these challenges, including robust learning algorithms, active query selection strategies, and safe exploration methods. ARL has been applied in domains such as robotics, game AI, and personalized assistants, enabling systems to adapt and align their behavior with user preferences or specified objectives. Active learning techniques can be seamlessly integrated with Reinforcement Learning (RL) to facilitate human-in-the-loop learning, enabling robots to actively seek human feedback or demonstrations in situations of uncertainty.

In this scenario, let consider a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the transition function, $\mathcal{R}$ is the reward function, and γ is the discount factor. The goal is to learn an optimal policy π^* that maximizes the expected cumulative reward. When integrating active learning with RL, the robot maintains a belief distribution $p(\theta)$ over the parameters θ of the reward function $\mathcal{R}\theta$ or the optimal policy $\pi\theta$. At each timestep t, the robot observes the current state s_t and selects an action a_t based on its current belief. However, in situations of high uncertainty, the robot can query the human for feedback or a demonstration. APL and ARL provide frameworks for efficiently querying the human and updating the belief distribution based on the received feedback. In APL, the robot presents the human with a set of candidate actions or trajectories and asks for their preference. The human's feedback is then used to update the belief distribution using Bayesian inference: $p(\theta|D_{t+1}) \propto p(D_{t+1}|\theta) \cdot p(\theta|D_t)$ where D_t represents the accumulated feedback up to timestep t, and D_{t+1} includes the new feedback. Similarly, in ARL, the robot queries the human for a reward signal or a demonstration in states where it is uncertain about the optimal action. The obtained feedback is used to update the belief distribution over the reward function parameters: $p(\theta|D_{t+1}) \propto p(D_{t+1}|\theta) \cdot p(\theta|D_t)$ Bayesian optimization can be employed to model the uncertainty in the human feedback and guide the query selection process. The robot maintains a Gaussian Process (GP) model of the expected feedback quality for each state-action pair. The GP model provides a mean estimate and a variance estimate for the feedback quality. The robot can select queries that maximize an acquisition function, such as Upper Confidence Bound (UCB) or Expected Improvement (EI), which balances exploration and exploitation. For example, using UCB, the robot selects the state-action pair (s_t, a_t) that maximizes: $UCB(s_t, a_t) = \mu(s_t, a_t) + \kappa \cdot \sigma(s_t, a_t)$ where $\mu(s_t, a_t)$ and $\sigma(s_t, a_t)$ are the mean and standard deviation of the GP model for the state-action pair (s_t, a_t), and κ is a hyperparameter that controls the exploration-exploitation trade-off.

Integrating active learning techniques with RL enables robots to leverage human expertise efficiently and adapt their behavior based on human preferences and demonstrations. This human-in-the-loop learning paradigm enhances the safety, interpretability, and trustworthiness of robot learning, making it suitable for real-world applications where human oversight and collaboration are essential

6.4 Sim-to-real transfer with domain randomization

Domain Adversarial Neural Networks (DANN) is a deep learning approach that aims to learn domain-invariant features for improved domain adaptation [20]. The architecture of DANN consists of three main components: a feature extractor $G_f(x)$, a label predictor $G_y(f)$, and a domain classifier $G_d(f)$. The feature extractor is trained to generate a feature representation f that is both discriminative for the main learning task and invariant to the domain shift. This is achieved by simultaneously optimizing two objectives: minimizing the loss of the label predictor $\mathcal{L}_y(G_y(G_f(x)), y)$ and maximizing the loss of the domain classifier $\mathcal{L}_d(G_d(G_f(x)), d)$. The domain classifier attempts to distinguish between the source and target domains based on the extracted features, while the feature extractor tries to confuse the domain classifier by generating domain-invariant features. The overall objective function of DANN is expressed as: $\mathcal{L}(G_f, G_y, G_d) = \mathcal{L}_y(G_y(G_f(x)), y) - \lambda\mathcal{L}_d(G_d(G_f(x)), d)$, where λ is a hyperparameter that controls the trade-off between the two objectives. By training the network in an adversarial manner, DANN effectively aligns the feature distributions of the source and target domains, enabling better generalization to the target domain.

Randomized Adversarial Imitation Learning (RAIL) combines ideas from imitation learning and adversarial training to learn robust policies from expert demonstrations [21]. The main objective of RAIL is to learn a policy $\pi_\theta(a|s)$ that can accurately imitate the expert's behavior while being resilient to variations in the environment dynamics. The framework introduces a randomized adversary $\rho_\phi(\tau)$ that perturbs the trajectories τ generated by the learned policy. The adversary is trained to maximize the discrepancy between the state-action distribution induced by the perturbed trajectories and the expert's distribution, denoted as $D_{JS}(\rho_\phi(\tau)||\pi_E)$, where D_{JS} is the Jensen-Shannon divergence. Simultaneously, the policy is trained to minimize this discrepancy, effectively learning to generate trajectories that closely match the expert's distribution even under adversarial perturbations. The training objective for RAIL can

Algorithm 3 Interactive Learning with Human Feedback

```
Input: Initial belief distribution $p(\theta)$, Gaussian Process (GP) model, hyperparameter $\kappa$
1. Initialize:
  • Initialize belief distribution $p(\theta)$ over parameters $\theta$ of $R_\theta$ or $\pi_\theta$.
  • Initialize GP model for uncertainty in human feedback.
  • Set hyperparameters, e.g., exploration-exploitation trade-off parameter $\kappa$.
2. For each episode:
  • Initialize starting state $s_0$.
  • For each timestep $t$:
    – Observe current state $s_t$.
    – If uncertainty in $(s_t, a_t)$ is high:
      * Select query $(s_t, a_t)$ maximizing acquisition function, e.g., UCB: $UCB(s_t, a_t) = \mu(s_t, a_t) + \kappa \cdot \sigma(s_t, a_t)$.
      * Query human for feedback or demonstration for $(s_t, a_t)$.
      * Receive human feedback $D_{t+1}$.
      * Update belief distribution $p(\theta)$ using Bayesian inference: $p(\theta|D_{t+1}) \propto p(D_{t+1}|\theta) \cdot p(\theta|D_t)$.
      * Update GP model with new feedback.
    – Else:
      * Select action $a_t$ based on current belief distribution $p(\theta)$ and policy $\pi_\theta$.
    – Execute selected action $a_t$, observe next state $s_{t+1}$ and reward $r_{t+1}$.
    – Update policy $\pi_\theta$ and belief distribution $p(\theta)$ based on observed transition and reward.
3. Repeat step 2 until convergence or maximum episodes reached.
4. Return learned policy $\pi_\theta$ and updated belief distribution $p(\theta)$.
```

be expressed as a minimax optimization problem: $\min_\theta \max_\phi \mathbb{E}\tau \sim \rho\phi(\tau)[D_{JS}(\rho_\phi(\tau)||\pi_E)] - \lambda\mathcal{H}(\rho_\phi)$, where $\mathcal{H}(\rho_\phi)$ is an entropy regularization term that encourages diversity in the adversarial perturbations, and λ is a hyperparameter controlling the strength of the regularization. By exposing the learned policy to a wide range of perturbed environments during training, RAIL enhances its robustness and generalization capabilities, making it more suitable for real-world deployment.

Domain randomization is a technique for bridging the sim-to-real gap in reinforcement learning (RL) by training policies in a variety of simulated environments with randomized properties [22]. The goal is to learn a policy that is robust and transferable to the real world by exposing it to a wide range of variations during training. This can be achieved by randomizing various aspects of the simulation, such as:

- 1. Lighting conditions: Varying the intensity, direction, and color of the lighting in the simulated environment helps the policy learn to handle different illumination scenarios.
- 2. Object properties: Randomizing the shape, size, mass, friction, and texture of objects in the environment allows the policy to generalize to a broader range of object interactions.
- 3. Sensor noise: Introducing random noise to sensor readings, such as camera images or depth sensors, makes the policy more resilient to imperfect sensor data in the real world.

To further enhance the sim-to-real transfer, advanced techniques like Domain Adversarial Neural Networks (DANN) can be employed. DANN aims to learn domain-invariant features by simultaneously training a feature extractor, a label predictor, and a domain classifier. The feature extractor is trained to generate representations that are both informative for the task at hand and invariant to the domain shift between simulation and reality. This is achieved by minimizing the label predictor's loss while maximizing the domain classifier's loss, effectively aligning the feature distributions across domains. Another approach is Randomized Adversarial Imitation Learning (RAIL), which combines imitation learning with adversarial training. In RAIL, an adversary is introduced to perturb the trajectories generated by the learned policy in simulation. The adversary aims to maximize the discrepancy between the state-action distribution induced by the perturbed trajectories and the expert's distribution. Simultaneously, the policy is trained to minimize this discrepancy, learning to generate trajectories that closely match the expert's behavior even under adversarial perturbations. This process enhances the policy's robustness and generalization capabilities.

7. Conclusions

Because robotic assistants become increasingly sophisticated and capable, it is crucial that we prioritize safety and trustworthiness as core design principles. The very qualities that make these machines useful - their intelligence and power - also give them the potential to cause harm if not developed with the utmost care and consideration for the well-being of the humans they will interact with.

Reinforcement learning (RL) can be applied to improve safety and trust in robot-to-robot and human-robot interaction. Constrained Markov Decision Processes (CMDPs) formulate the RL problem with safety constraints, ensuring robots explore while satisfying safety requirements. The reward function can incorporate metrics that quantify safety and trust, with techniques like Inverse Reward Design (IRD) learning from human demonstrations. Active learning enables human-in-the-loop learning, where robots query humans for feedback in uncertain situations.

Domain randomization techniques bridge the gap between simulation and reality, making learned policies more robust and transferable. Multi-agent RL algorithms enable safe and efficient coordination in multi-robot systems. Bayesian RL quantifies uncertainty in learned policies, while Model-Based RL learns a model of the environment for planning and decision-making. Implementing these RL techniques requires careful consideration of computational resources, real-time constraints, and safety guarantees. Research focuses on developing scalable and efficient algorithms, integrating formal verification methods, and establishing standardized benchmarks for evaluating safety and trust in robot interaction scenarios.

The proposed framework in this study lacks experimental validation, which is a significant limitation. While the study focuses on the theoretical and algorithmic foundations of integrating multiple RL techniques for enhancing safety and trust in robotic systems, empirical investigations are necessary to assess the practical effectiveness of the framework. Without experimental validation, the feasibility and

Algorithm 4 Adversarial Domain Adaptation for Robotic Learning (ADARL)

```
Input: Policy network $\pi_\theta$, feature extractor $G_f$, label predictor $G_y$, domain classifier $G_d$ (for DANN), adversary $\rho_\phi$ (for RAIL)
1. Initialize:
    • Initialize policy network $\pi_\theta$, feature extractor $G_f$, label predictor $G_y$, domain classifier $G_d$ (for DANN), and adversary $\rho_\phi$ (for RAIL).
2. For each iteration:
    • a. Sample batch of source domain (simulated) experiences: $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^{N}$.
    • b. Apply domain randomization to simulated experiences by randomizing lighting conditions, object properties, and sensor noise.
    • c. If using DANN:
        – i. Extract features $f_i = G_f(s_i)$ for each state $s_i$ in the batch.
        – ii. Compute label prediction loss $L_y(G_y(f_i), a_i)$ and domain classification loss $L_d(G_d(f_i), d_i)$, where $d_i$ is the domain label (0 for simulation, 1 for real).
        – iii. Update parameters of $G_f$, $G_y$, and $G_d$ using gradient descent to minimize $L_y$ and maximize $L_d$.
    • d. If using RAIL:
        – i. Generate perturbed trajectories $\tau_i$ using adversary $\rho_\phi$.
        – ii. Compute discrepancy loss between perturbed trajectories and expert trajectories: $L_{\text{RAIL}} = D_{\text{JS}}(\rho_\phi(\tau_i)||\pi_E) - \lambda \cdot H(\rho_\phi)$, where $D_{\text{JS}}$ is Jensen-Shannon divergence, $H(\rho_\phi)$ is entropy of adversary, and $\lambda$ is hyperparameter.
        – iii. Update parameters of policy $\pi_\theta$ to minimize $L_{\text{RAIL}}$ and parameters of adversary $\rho_\phi$ to maximize $L_{\text{RAIL}}$.
    • e. Update policy $\pi_\theta$ using chosen reinforcement learning algorithm (e.g., PPO, SAC) with randomized simulated experiences.
3. Repeat step 2 until convergence or for specified number of iterations.
4. Transfer learned policy $\pi_\theta$ to real-world environment and fine-tune if necessary.
```

performance of the proposed approach remain uncertain, limiting its immediate applicability to real-world scenarios.

Moreover, the study does not provide a comprehensive analysis of the computational complexity and scalability of the proposed framework. As robotic systems become increasingly complex and large-scale, implementing the integrated RL techniques may pose computational challenges. The study does not address the potential limitations in terms of computational resources, memory requirements, and training time, which are crucial considerations for the practical deployment of the framework in real-world applications.

Prioritizing safety and trust in robotic systems is undoubtedly important, but it may come at the cost of reduced performance or efficiency in certain situations. The study does not provide a thorough analysis of how the proposed framework balances these competing objectives and the potential implications of prioritizing safety and trust over performance metrics. This lack of analysis limits the understanding of the framework's applicability in scenarios where performance is a critical factor.

The proposed framework heavily relies on the successful transfer of learned policies from simulation to real-world environments. Although the study acknowledges the challenges associated with sim-to-real transfer, it does not provide a comprehensive solution to address them. The complexities and uncertainties of real-world scenarios can significantly impact the performance and effectiveness of the learned policies. The study does not offer a robust methodology for ensuring a transfer of the learned behaviors from simulation to the real world.

References

[1] J. Rosen, B. Hannaford, and R. M. Satava, *Surgical robotics: systems applications and visions*. Springer Science & Business Media, 2011.

[2] J. N. Pires, *Industrial robots programming: building applications for the factories of the future*. Springer Science & Business Media, 2007.

[3] G. A. Bekey, *Autonomous robots: from biological inspiration to implementation and control*. MIT press, 2005.

[4] R. A. Faust, *Robotics in surgery: history, current and future applications*. Nova Publishers, 2007.

[5] A. K. Saxena, M. Hassan, J. M. R. Salazar, M. R. Amin, V. García, and P. P. Mishra, "Cultural intelligence and linguistic diversity in artificial intelligent systems: A framework," *International Journal of Responsible Artificial Intelligence*, vol. 13, no. 9, pp. 38–50, 2023.

[6] R. N. Darmanin and M. K. Bugeja, "A review on multi-robot systems categorised by application domain," in *2017 25th mediterranean conference on control and automation (MED)*, IEEE, 2017, pp. 701–706.

[7] R. De Nicola, L. Di Stefano, and O. Inverso, "Toward formal models and languages for verifiable multi-robot systems," *Frontiers in Robotics and AI*, vol. 5, p. 94, 2018.

[8] E. Tuci, M. H. Alkilabi, and O. Akanyeti, "Cooperative object transport in multi-robot systems: A review of the state-of-the-art," *Frontiers in Robotics and AI*, vol. 5, p. 59, 2018.

[9] C. Liu and M. Tomizuka, "Designing the robot behavior for safe human–robot interactions," *Trends in Control and Decision-Making for Human–Robot Collaboration Systems*, pp. 241–270, 2017.

[10] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. Gonzalez-Sarabia, C. Torre-Ferrero, and J. Perez-Oria, "Working together: A review on safe human-robot collaboration in industrial environments," *Ieee Access*, vol. 5, pp. 26 754–26 773, 2017.

[11] B. D. Argall, "Autonomy in rehabilitation robotics: An intersection," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 441–463, 2018.

[12] L. Pérez, Í. Rodríguez, N. Rodríguez, R. Usamentiaga, and D. F. García, "Robot guidance using machine vision techniques in industrial environments: A comparative review," *Sensors*, vol. 16, no. 3, p. 335, 2016.

[13] D. J. White, "Mean, variance, and probabilistic criteria in finite markov decision processes: A review," *Journal of Optimization Theory and Applications*, vol. 56, pp. 1–29, 1988.

[14] C. Amato, G. Chowdhary, A. Geramifard, N. K. Üre, and M. J. Kochenderfer, "Decentralized control of partially observable markov decision processes," in *52nd IEEE Conference on Decision and Control*, IEEE, 2013, pp. 2398–2405.

[15] W. S. Lovejoy, "A survey of algorithmic methods for partially observed markov decision processes," *Annals of Operations Research*, vol. 28, no. 1, pp. 47–65, 1991.

[16] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[17] M. Carroll, "Overview of current ai alignment approaches," 2018.

[18] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, G. Pezzulo, *et al.*, "Active inference and learning," *Neuroscience & Biobehavioral Reviews*, vol. 68, pp. 862–879, 2016.

[19] J. P. O'Doherty, J. Cockburn, and W. M. Pauli, "Learning, reward, and decision making," *Annual review of psychology*, vol. 68, pp. 73–100, 2017.

[20] J. H. Ryu, M. Irfan, A. Reyaz, *et al.*, "A review on sensor network issues and robotics," *Journal of Sensors*, vol. 2015, 2015.

[21] M. Shin and J. Kim, "Randomized adversarial imitation learning for autonomous driving," *arXiv preprint arXiv:1905.05637*, 2019.

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