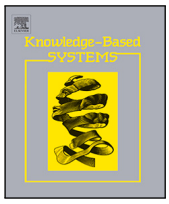
Knowledge-Based Systems 309 (2025) 112870

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



Novel automated interactive reinforcement learning framework with a constraint-based supervisor for procedural tasks

Íñigo Elguea-Aguinaco [a,*], Aitor Aguirre-Ortuzar [b], Unai Izagirre-Aizpitarte [b], Ibai Inziarte-Hidalgo [c], Simon Bøgh [d], Nestor Arana-Arexolaleiba [b]

[a] Research & Development Department, Electrotécnica Alavesa S.L., 1010 Vitoria-Gasteiz, Spain
[b] Intelligent Systems for Industrial Systems Research Group, Electronics and Computer Science Department, University of Mondragon, 20500 Mondragon, Spain
[c] Automation Department, Montajes Mantenimiento y Automatismos Eléctricos Navarra S.L., 31195 Aizoain, Spain
[d] Advanced Robotics & AI Group, Department of Electronic Systems, Aalborg University, 9220 Aalborg East, Denmark

ARTICLE INFO

Keywords:
Automated supervisor
Contact-rich manipulation
Industrial manipulators
Interactive reinforcement learning
Sample efficiency
Procedural tasks

ABSTRACT

Learning to perform procedural motion or manipulation tasks in unstructured or uncertain environments poses significant challenges for intelligent agents. Although reinforcement learning algorithms have demonstrated positive results on simple tasks, the hard-to-engineer reward functions and the impractical amount of trial-and-error iterations these agents require in long-experience streams still present challenges for deployment in industrially relevant environments. In this regard, interactive reinforcement learning has emerged as a promising approach to mitigate these limitations, whereby a human supervisor provides evaluative or corrective feedback to the learning agent during training. However, the requirement of a human-in-the-loop approach throughout the learning process can be impractical for tasks that span several hours. This study aims to overcome this limitation by automating the learning process and substituting human feedback with an artificial supervisor grounded in constraint-based modeling techniques. In contrast to the logical constraints commonly used for conventional reinforcement learning, constraint-based modeling techniques offer enhanced adaptability in terms of conceptualizing and modeling the human knowledge of a task. This modeling capability allows an automated supervisor to acquire a closer approximation to human reasoning by dividing complex tasks into more manageable components and identifying the associated subtask and contextual cues in which the agent is involved. The supervisor then adjusts the evaluative and corrective feedback to suit the specific subtask under consideration. The framework was assessed using three actor-critic agents in a human–robot interaction environment, demonstrating a sample efficiency improvement of 50% and success rates of ≥95% in simulation and 90% in real-world implementation.

1. Introduction

Reinforcement learning (RL) methods have achieved promising results in solving control tasks in unstructured environments [1–3]. However, previous studies have generally focused on approaches based on single simple tasks with limited state–action pairs [4]. These simplifications constrain the implementation of these control methods in high-dimensional real-world settings, such as collaborative robotic manipulation tasks. For instance, although much research focuses on peg-in-hole [5,6] or via-point tasks [7,8], collaborative robot tasks should extend beyond these simplified scenarios to include more complex procedural tasks. In this regard, hierarchical RL (HRL) methodologies have emerged as a way to acquire control policies that are capable of managing tasks characterized by long sequences of experiences. HRL involves learning both high-level and low-level policies to solve tasks hierarchically, which allows the agent to operate at multiple levels of abstraction. However, similar to conventional RL, HRL struggles with intricate reward engineering tailored to each subtask and suffers from low sample efficiency due to exploration [9,10], resulting in prolonged training times.

Rewards can be either sparse [11,12], provided only at the conclusion of an episode, or dense [13,14], given at each time step. Dense rewards offer intermediate feedback to the agent, indicating the effectiveness of the previous action toward the goal. This intermediate feedback plays a key role in defining certain problems, especially when considering long-experience streams. Without such intermediate feedback, learning would be infeasible in problems with large exploration spaces. Nonetheless, no guidelines or methodology have

* Corresponding author.
E-mail address: ielguea@aldakin.com (Í. Elguea-Aguinaco).

https://doi.org/10.1016/j.knosys.2024.112870
Received 5 June 2024; Received in revised form 2 December 2024; Accepted 10 December 2024
Available online 16 December 2024

been established on how to construct dense reward functions, which often involve complex mathematical functions that may lack straightforward interpretability or even affect convergence [15]. In environments with inherent uncertainty, the reward function must capture the contextual nuances required to discern the circumstances under which the agent executes its actions. The same action within a subtask may yield different evaluations depending on the context in which it occurs. Consequently, intricate reward functions may lead to unexpected policy behaviors.

In contrast, RL is characterized by lengthy training times [16], particularly in tasks with continuous streams of experience. To expedite learning and avoid starting from scratch, a growing trend is to provide prior knowledge to the agent, often through demonstrations [17,18]. These demonstrations can be recorded using human movements, kinesthetic guidance, or telemanipulation [19]. However, collecting sufficient demonstrations can be time-consuming and costly. In addition, the efficacy of these approaches heavily relies on the geometry of the provided trajectories, potentially limiting the generalizability of the agent [20].

In contrast to human demonstrations, the interactive RL paradigm presents an alternative [21]. In this framework, an artificial or human trainer is included in the RL agent's learning process. The trainer leverages task expertise to provide evaluative and corrective feedback and captures the desired behavior of the agent. This is particularly helpful in scenarios where defining the reward signal is challenging. For instance, Knox and Stone [22] introduced the TAMER framework, which allows task experts without programming expertise to evaluate the RL agent's actions as either "good" or "bad" in the Tetris and mountain car example. In addition, the trainers' guidance enhances the sample efficiency of the agent by providing helpful hints. In another study [23], the human supervisor not only provided feedback in the form of rewards but also provided guidance messages, which enhanced both the sample efficiency and the performance of the RL agent in a computer game. By combining guidance with independent exploration, the potential for generalization can be preserved with minimal disruption [24].

Nevertheless, currently, all interactive RL approaches applied to continuous problems involve a human trainer. In scenarios characterized by extensive action spaces, tasks that are divisible into subtasks, including human-in-the-loop, can be time-consuming and labor-intensive processes. Moreover, the efficacy of the learning process is significantly contingent upon the human trainer's expertise and potential biases. Therefore, the provision of ill-suited or misplaced evaluative and corrective feedback carries the risk of resulting in suboptimal policy outcomes.

Diverging from existing interactive RL frameworks that rely on neural network-based artificial trainers, such as other RL policies [25], which may face challenges in variable or ill-defined settings with continuous state actions pairs or human-in-the-loop trainers [26], this study proposes the utilization of an automated supervisor grounded on constraint-based modeling techniques. These techniques include qualitative temporal constraints based on Allen's interval algebra [27], punctual qualitative temporal constraints, quantitative temporal constraints, constraints over specific properties or trajectories, and logical constraints. Unlike conventional RL frameworks that commonly employ logical constraints, constraint-based modeling techniques offer enhanced flexibility in terms of capturing human knowledge in complex procedural tasks. By leveraging such techniques and integrating them within a system that emulates human cognition, an automated supervisor akin to human reasoning can be constructed, facilitating the formalization of human behavioral patterns and understanding in an interpretable manner. Consequently, the role of the human supervisor is replaced with an artificial counterpart capable of providing evaluative and corrective feedback based on the agent's decision-making process, even under uncertain circumstances, as a human would. To implement this supervisor, we leverage and adapt an existing framework designed to mimic human cognitive processes across three key stages: observing the environment, interpreting ongoing activities, and assessing their correctness within the given context, specifically via evaluative and corrective feedback. The framework is then tested in a human–robot interaction (HRI) disassembly environment, where the robot is required to perform a series of subtasks based on the interaction mode between the human and the robot. The results demonstrate that incorporating an artificial supervisor into the apprenticeship loop enhances sample efficiency and performance of different RL agents by approximately 50%.

The primary contributions of this study are as follows:

- an automated interactive RL framework that uses an artificial supervisor that mimics human cognition and featuring constraint-based modeling techniques to define and provide evaluative and corrective feedback;
- abstraction of task complexity into manageable procedural components and contexts and;
- conceptualization and modeling of human knowledge, thus eliminating the need for human intervention in domains with large and continuous action spaces.

The remainder of this paper is organized as follows. Section 2 briefly overviews of related works. In Section 3, a theoretical background on conventional and interactive RL and the proposed framework is outlined. Section 4 provides a comprehensive explanation of the framework's implementation and describes the supervisor's functions in the agent's learning process. Section 5 describes the evaluation task and presents the results obtained from both simulation and reality. Section 6 summarizes the main directions for future research and provides points for discussion and reflection. Finally, Section 7 concludes with a summary of the contributions.

2. Related works

Interactive RL is an approach that tailors an external trainer to adapt key components, such as exploration strategies, of the underlying RL algorithm, thereby enhancing its performance and reducing training times for a given task [28]. An external trainer can be an artificial or a human agent.

When using artificial agents as trainers, RL agents that have previously learned a control policy are typically employed. In such cases, the trainer typically provides corrective feedback, meaning that the supervising agent can intervene by suggesting alternative actions to the learning agent's proposed actions. However, according to Cruz et al. [29], for an RL agent to effectively serve as a trainer, its policy should have a low standard deviation, indicating a more reliable distribution of knowledge across states. Achieving such stability in high-dimensional continuous tasks with complex reward functions is challenging. Consequently, most studies involving RL agents as trainers have focused on tasks with discrete action spaces [30].

As a result, interactive RL commonly employs a human-in-the-loop approach, allowing the trainer to provide both evaluative and corrective feedback based on elapsed time, state importance, or error correction [31].

Evaluative feedback treats human input as a form of reward. However, unlike conventional RL, evaluative feedback accounts for pre-existing human knowledge or models and adapts them to a particular context. These rewards can be provided directly by humans or acquired indirectly. For example, Ritschel and André [32] used human appraisal as input to provide rewards based on gesture and posture. In contrast, other studies have explored biological drives. McDuff and Kapoor [33] used the human's volumetric blood change in a driving simulator, and Akinola et al. [34] and Kim et al. [35] used the error-related potential in robotic navigation and HRI tasks, respectively. However, acquiring such signals can be time-consuming, and their reliability

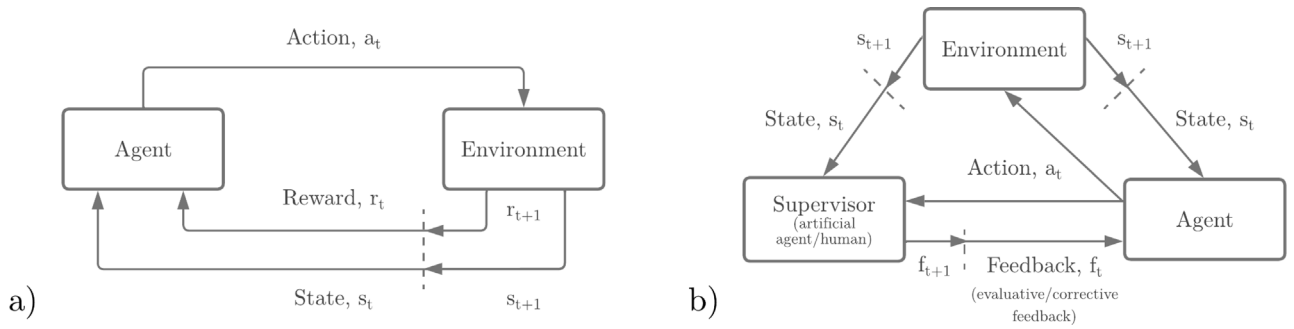


Fig. 1. (a) RL scheme. (b) Interactive RL scheme.

can be affected by external factors, such as background noise. Therefore, approaches involving a human-in-the-loop trainer often emphasize corrective feedback.

Akkaladevi et al. [36] employed a graphical interface to allow the human trainer to select actions the robot should perform. However, the authors discretized the actions based on specific criteria to facilitate the process, thereby limiting the robot's performance in a complex HRI assembly environment. To address this limitation, other studies have proposed approaches that allow the human trainers to provide corrective feedback within continuous action spaces [37]. Celemin et al. [38] developed a framework that enabled the human trainer to provide discrete trend corrections to an agent's continuous actions during the learning process. In a more recent study, Chisari et al. [39] proposed a framework that facilitates both evaluative and corrective feedback provision in various manipulation tasks. The human trainer provided evaluative feedback using numerical values assigned via a button-based interface. Moreover, if the robot executed an erroneous trajectory, the human trainer could correct its path through teleoperation.

However, these methods rely on constant human supervision during the learning process of the RL agent. This requirement can become tedious and time-consuming, especially in complex robotic manipulation tasks that involve long training times and the segmentation of the process into more manageable subtasks. In addition, inconsistent advice may lead to suboptimal policies when the human trainer lacks expertise. This limitation may also arise when advice is provided at inconsistent time intervals to the RL agent. To the best of our knowledge, this study is the first to present an automated interactive RL approach for continuous procedural tasks. The incorporation of this supervisory tool offers distinct advantages over state-of-the-art approaches. The initial distinction lies in the expansive capacity of constraint-based modeling techniques compared with conventional logical constraints, offering humans a broader scope for conceptualizing and structuring their knowledge. Integrating this broader range of constraints into a system grounded in observation, interpretation, and evaluation enables the emulation of human-like reasoning by the artificial supervisor, thereby avoiding the need for human oversight of the agent's learning process, which can extend over prolonged periods. In addition, this modeling methodology allows for the definition of subtasks and contexts through initial, generic, and final constraints. This approach facilitates the generation of precise evaluative feedback tailored to the given subtask and its context, which is easily comprehensible and interpretable by humans during programming. Finally, the framework facilitates the provision of targeted corrective feedback for assessed subtasks and contexts at predefined intervals, thereby enhancing sample efficiency in large exploration environments.

3. Theoretical framework

3.1. Reinforcement learning

RL [40] is a type of machine learning in which an agent learns to interact with its environment to maximize the rewards it receives in the long term. This interaction learning problem is usually modeled as a Markov Decision Process (MDP).

The MDP defines sequential decision-making as a semi-random and agent-dependent pathway. It also specifies that decisions made and executed (actions) influence not only immediate rewards but also subsequent situations (states) through future rewards. Thus, MDPs comprise three elements: the state the agent is in, the action the agent takes, and the agent's ultimate goal. In their simplest forms, the elements are usually represented by the following tuple (1):

$$[S, A, P(s_{t+1}|s_t, a_t), R(s_t, s_{t+1}, a_t), \gamma] \tag{1}$$

where S denotes the set of possible states of the agent and A is the set of actions. $P(s_{t+1}|s_t, a_t)$ is the probability of transition to future state s_{t+1}, when the agent is in state s_t and applies action a_t. $R(s_t, s_{t+1}, a_t)$ is the reward the agent expects to obtain when it transits from state s_t to state s_{t+1} and it is calculated using the reward function. Finally, γ is the discount factor of the reward function. Fig. 1a shows the basic MDP underlying the decision process of any RL agent.

A policy $\pi(a_t|s_t)$ is a mapping between states and probabilities of selecting each possible action. This process can be defined using sequence (2) as follows:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots \tag{2}$$

3.1.1. Formal definition of a subtask

In contrast to HRL, where a high-level policy, referred to as the task policy, π_Γ, corresponding to the main long-term horizon Γ, and a low-level policy, termed the subtask policy, π_ω, corresponding to the subtask ω, are learned, this study proposes a unified control policy that implicitly comprises both high-level and low-level knowledge. Nevertheless, we rely on the HRL definition to formalize the subtask concept [41].

A subtask is a smaller, distinct component typically used to decompose complex tasks. It comprises a set of actions that contribute to achieving the overarching task, namely, the execution and goal components, respectively. Concerning the execution components, a subtask is defined by an initiation condition I_ω and a termination condition β_ω, which are usually associated with a specific set of states or logical conditions. In addition, regarding the goal components, each subtask is characterized by its own reward r_ω and a potential subgoal g_ω. This subgoal, which can take the form of a state $s \in S$ or an abstract representation thereof, among other possibilities, is commonly linked to the terminal condition β_ω.

3.2. Interactive reinforcement learning

Interactive RL includes the integration of external guidance into the apprenticeship loop. This supportive advice can originate from diverse sources, including pretrained artificial agents with prior task knowledge, domain experts, or non-expert individuals, and may manifest as either corrective or evaluative feedback or both (Fig. 1b).

In this context, at each time step, the supervisor receives updates regarding the state and action taken by the agent. If only corrective

feedback is provided, the trainer may recommend an alternative action or a trend to the agent if the agent's action deviates from its defined objective. In this scenario, the environment itself bestows the agent with rewards. Alternatively, when providing evaluative feedback, the supervisor assigns the agent a numerical reward. Sequences (3) and (4) depict the interaction sequences when the supervisor provides evaluative feedback (f_e) and corrective feedback (f_c) through actions or trends, respectively.

$$s_0, a_0, f_{e_0}, s_1, a_{c_1}, f_{e_1}, s_2, a_2, f_{e_2}, \dots \tag{3}$$

$$s_0, a_0, f_{e_0}, s_1, a_1, f_{e_1}, f_{c_1}, s_2, a_2, f_{e_2}, \dots \tag{4}$$

3.3. Leveraging constraint-based modeling techniques for an intelligent artificial supervisor

Interactive RL leverages the prior knowledge of a task from an artificial or human supervisor to accelerate learning. Typically, these artificial supervisors are derived from pre-existing RL policies, constituting what is commonly termed teacher–student frameworks. Within such frameworks, the supervisory policy is limited to providing corrective actions (f_c) to the RL agent based on predefined criteria to reduce exploration space and improve sample efficiency (Fig. 2a). Although this approach can yield significant gains in tasks characterized by a finite set of state–action pairs, such as grid worlds [42], its efficacy is reduced when handling high-dimensional continuous tasks. This limitation arises from the necessity for an RL policy to exhibit a low standard deviation to serve as an effective supervisor [29]. However, in high-dimensional continuous procedural tasks, learned control policies may be susceptible to environmental stochasticity and random or epistemic uncertainty, potentially resulting in suboptimal policies. Consequently, the prevailing method used to enhance sample efficiency and provide evaluative feedback in scenarios where reward shaping proves challenging is centered upon integrating human involvement within the apprenticeship loop of the RL agent.

Human cognition is more flexible than an RL policy. Drawing upon accumulated life experiences, individuals possess a nuanced understanding of how an RL agent should engage with its environment and the abstraction capacity to conceptualize reward models that may not be readily expressible through conventional logical constraints in RL frameworks (Fig. 2a). This is particularly evident in procedural, variable, and uncertain tasks. For example, as illustrated in Fig. 2b, complex tasks often comprise subtasks delineated by initiation (I_ω) and termination conditions (β_ω). However, these subtasks may overlap, as seen with *Subtasks 2* and *3*, or even nest within one another, as exemplified by *Subtask 4*. In such scenarios, humans possess the interpretative prowess necessary to provide evaluative feedback (f_e) abstracted from a specific subtask while considering the agent's overarching goal. Conversely, during exploration phases in intricate environments characterized by high variability and uncertainty, agents may follow trajectories that make it difficult to immediately classify them as correct or incorrect. As shown in Fig. 2c, multiple trajectories may originate from a common point; however, they diverge in their paths toward a target, highlighting the challenge of delineating sparse rewards without impeding convergence or through overly strict dense rewards hindering generalization. In contrast, humans can perceptively discern an agent's intentions and adaptively provide appropriate rewards, going beyond the constraints of periodic reward assignment.

However, employing a human-in-the-loop approach can be arduous and time-consuming for the individual. In addition, humans typically, rely on visual interfaces to provide evaluative and corrective feedback to agents. In these cases, the evaluation performed by humans is not based on specific data, and if the simulation is ongoing at the time of evaluation, there exists a risk of misplacing evaluative or corrective feedback, potentially resulting in suboptimal policies. Thus, establishing a method to address the temporal disparities between the occurrence of events and the issuance of feedback is crucial for mitigating this risk.

Given these limitations, we suggest replacing human-in-the-loop involvement with an artificial supervisor capable of receiving task-related knowledge transfer from humans. This supervisor relies on constraint-based modeling techniques, offering greater flexibility in modeling human knowledge than with logical constraints alone. These techniques not only cover logical constraints but also allow the establishment of (punctual) qualitative and quantitative temporal constraints and constraints over defined properties or trajectories, thereby addressing the procedural, variable, and uncertain nature of the subtasks described previously:

- Temporal qualitative constraints. These constraints employ Allen interval logic to define temporal relationships among time intervals without requiring numerical quantification of the intervals' start, end, or duration instants. Example (5) illustrates the execution of "*Subtask 1*" immediately following "*Subtask 2*":

$$Subtask1\ [meets]\ Subtask2 \tag{5}$$

- Punctual temporal qualitative constraints. These constraints establish qualitative temporal connections between the initial or concluding moments of the two intervals. In example (6), "*Subtask 1*" concludes simultaneously with the beginning of "*Subtask 2*":

$$Subtask1.End == Subtask2.Start \tag{6}$$

- Temporal quantitative constraints. These constraints are used to set conditions during an interval or the elapsed time between two intervals. Example (7) shows this type of constraint, where the time elapsed between the beginning of "*Subtask 1*" and "*Subtask 2*" must not exceed 0.001 s:

$$Duration(Subtask1.Start, Subtask2.Start) < 0.001\,\mathrm{s} \tag{7}$$

- Constraints on properties. These constraints establish conditions on the values of the properties of the observations. For instance, consider an observation denoted as "*Robot*" with a property referring to the value along the x axis of its tool center point (TCP). For instance, example (8) shows that the TCP value along the x axis of the robot must exceed 0.2 m.

$$Robot.TCP\_x > 0.2\,\mathrm{m} \tag{8}$$

- Constraints on trajectories. These constraints are employed to assess the probable correctness of a trajectory when the reward function is challenging to develop. To achieve this, the motion is modeled using arcs and fuzzy logic. Example (9) shows the motion path that the TCP should follow. The first constraint (Arc) regulates the trajectory's concavity, curvature, and length. The second constraint (Fuzzy) limits the movement direction, indicating how similar the movement is to the 26 classes, each separated by 45°, into which the movement space is discretized. For further details on the definition of these constraints, see [43].

$$\begin{aligned} RobotTCP.Arc(>90, <0.5, <0.2).Fuzzy(&\geq 0.0, \geq 0.0, \geq 0.0, \\ &\geq 0.0, \geq 0.0, \geq 0.0, \geq 0.0, ..., \geq 50 AND < 100) \end{aligned} \tag{9}$$

- Logical constraints. These constraints define logical relationships using AND, OR, and NOT operators. Example (10) specifies that the TCP value for the robot along the x axis must exceed 0.2 m, whereas along the y axis, it must exceed 0.1 m.

$$Robot.TCP\_x > 0.2\,\mathrm{m}\ AND\ Robot.TCP\_y > 0.1\,\mathrm{m} \tag{10}$$

Thus, this study proposes the following hypothesis: "*an artificial supervisor built upon constraint-based modeling techniques and a system structured around observation, interpretation, and evaluation phases can effectively substitute for and replicate the cognitive capabilities of a human in an interactive RL approach, speeding up the learning process of an RL agent in variable and uncertain procedural tasks*". Consequently, we modify tuple (1) as follows:

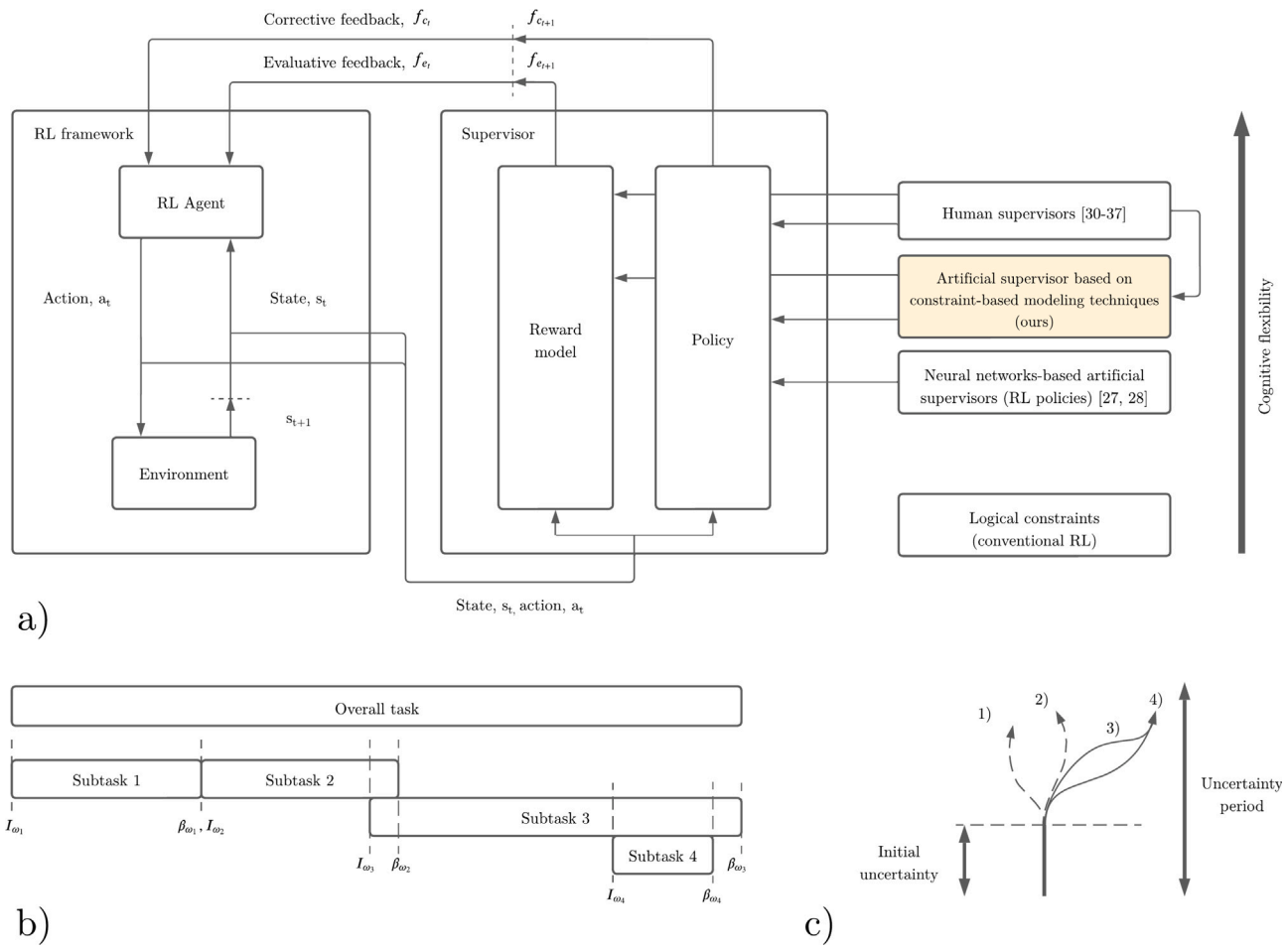


Fig. 2. (a) Expanded interactive RL scheme. (b) Complex task divided into subtasks with nonsequential initiation and termination conditions that require reward shaping. (c) Four trajectories from a common start, initially identical, with only two successfully reaching the target via distinct paths.

$$[S, A, P(s_{t+1}|s_t, a_t), \gamma, C, F] \tag{11}$$

where C represents the set of constraints defined for the artificial supervisor, and each constraint C_i involves a subset of S_i and specifies the allowable combinations of values for these variables. The artificial supervisor function, denoted as F, is expressed as $F : S \times A \rightarrow (f_e, f_c)$, where f_e is the evaluative feedback indicating the reward or penalty for a specific subtask and f_c is the corrective feedback referring to the corrective action, provided to guide the RL agent's exploring.

4. Implementation

4.1. Framework overview

The human should be able to conceptualize and formalize their understanding of the task using constraint-based modeling techniques, thereby facilitating its transfer to an artificial supervisor within the apprenticeship loop of the RL agent. To achieve this objective, we leverage and adapt an established framework known as ULISES.

Previously, this framework was applied to create intelligent learning systems across diverse virtual reality interactive systems [44], diagnose human motor skills during procedural tasks [43], and act as an oracle in testing automated systems. In these applications, ULISES monitors the entire learning or interaction process, providing an overall evaluation and instructive feedback upon completion.

In this study, ULISES was adapted for use in the RL context. Here, the role of the agent involves replicating the behavior of a human expert to oversee the interactions of an RL agent with its environment. Consequently, adjustments were made to allow ULISES to provide corrective and evaluative feedback.

Throughout the learning procedure, the human trainer observes the environment, interprets the agent's actions, and issues evaluative and/or corrective feedback. Similarly, ULISES comprises three basic levels: observation, interpretation, and diagnosis. The Observation Subsystem receives the state space s_t of the RL agent at each time step and generates a ULISES observation element. Concurrently, the Interpretation and Diagnosis Subsystems map the agent's activity into subtasks and contextual elements, denoted as steps and situations, respectively, within the framework. Both steps and situations are derived from ULISES observations. These three elements collectively represent all learning processes in procedural tasks.

Fig. 3 shows the general framework of ULISES-RL. Throughout the interaction between the agent and the environment, data, including the agent's state s_t and action space a_t, are transmitted to ULISES. The cognitive system is established through a tri-tiered abstraction system to offer a generalized representation of the components needed for its subsystems. This approach helps create a tailor-made supervisor that is aligned with the specific task under consideration. These elements specify how ULISES observations are described and interpreted and the criteria followed to assess the agent's actions. In the domain-specific customization process, specific models for observation, interpretation, and tasks are generated. The Observation Model determines communication, and the Task Model resolves the task.

The communication protocol between the different subsystems within these three levels relies on established conventions involving

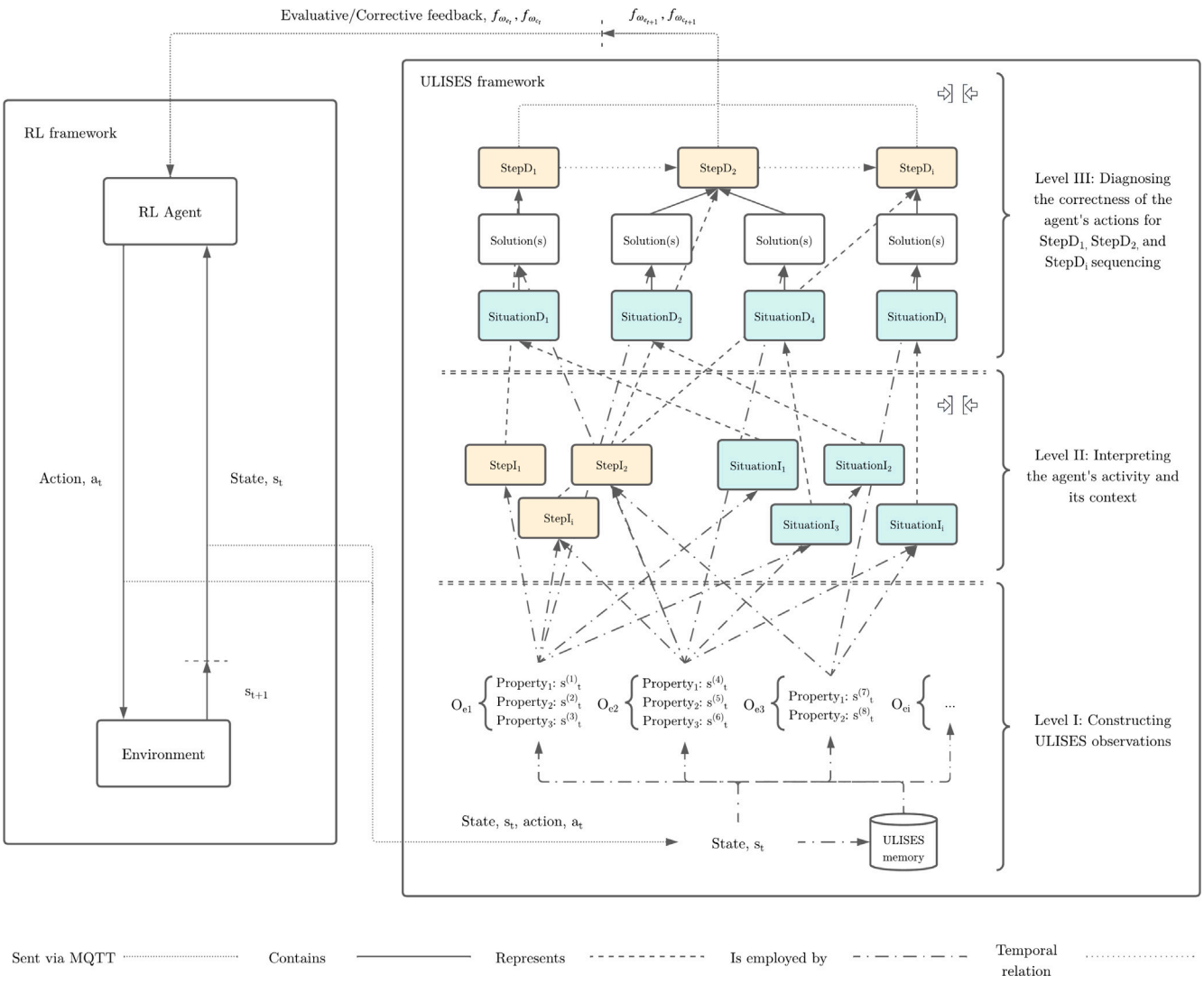


Fig. 3. Overall framework structure. At each time step, the state space s_t is transmitted to the ULISES supervisor to construct ULISES observations (O_e). These observations represent the steps and situations via constraint-based modeling techniques. At the highest level, relations between steps and situations are defined, with potential solutions for each step in a given situation. These solutions are also represented through ULISES observations, possibly using additional observations. Based on the correctness of the agent's actions, the supervisor provides evaluative and corrective feedback to the RL agent.

subscriptions, requests, and queries, ensuring that information is exchanged only when necessary. Eventually, ULISES provides an output in the form of evaluative ($f_{\omega_{e_t}}$) and corrective ($f_{\omega_{c_t}}$) feedback, which is transmitted to the RL agent. For a more comprehensive understanding of the individual levels constituting ULISES, a more detailed explanation is provided in the following sections.

4.2. Level I: constructing ULISES observations

The Observation Subsystem of ULISES analyzes the data received from the simulation environment of the RL agent, namely the state space s_t, and models them into ULISES observation elements. These observations, denoted as continuous elements, are generated based on the number of entities in the RL agent simulation environment. For instance, in scenarios involving HRI tasks, observations such as "*Robot*" (O_{e1}) and "*Human*" (O_{e2}) are generated. In addition, each observation can be defined using properties that further characterize its attributes. The ULISES observation component should not be confused with observations accessible to the RL agent when interacting with the environment. In a fully observable MDP, the observations made by the RL agent correspond to the state space s_t and are the properties of each ULISES observation. For example, although the Cartesian coordinates of a robot across the x, y, and z axes may serve as observations for the RL agent, each of these coordinates represents a property within the "*Robot*" observation in ULISES. Fig. 3, at Level I, illustrates how ULISES observations use the different components that constitute the state space s_t to generate their properties.

The state space s_t data are stored in ULISES memory and can be harnessed within the supervisor to generate new observations or properties. This functionality is particularly valuable when the information received in state space s_t lacks the observation property required to establish constraints at higher levels.

The Interpretation and Diagnosis Subsystems leverage the observation elements to realize the real-time interpretation and diagnosis of ongoing actions in the simulation environment. These subsystems provide a framework that can be adapted to any simulation scenario and to any requirement across any domain. Unlike the Interpretation and Diagnosis Subsystems, the Observation Subsystem requires domain-specific integration with the simulation environment. This integration is essential to account for the state space S generated within each simulation context.

The Observation Subsystem leverages the data gathered by the listener and observer agents. The listener agents are responsible for

communicating the simulation engine with the observer agent. In contrast, the observation agent encapsulates the Observation Subsystem, gathering simulation data from listener agents and transforming them into observations.

Within this level, the Observation Subsystem runs the list of specifications of the Observation Model and calls the corresponding observers to check the occurrence at a specific instant. As a result, this subsystem generates new observations when they are detected, extends the duration of the existing observations, and ends them when they are no longer observed.

4.3. Level II: interpreting the reinforcement learning agent's activity and its context

Once the activity within the learning process is observed, the subsequent stage involves its interpretation. Activity interpretation within the framework introduces two components: step and situation elements. The first component encapsulates the attributes necessary for the Interpretation Subsystem to recognize agent activities, which can be understood as the different subtasks inherent in a complex task. The second component outlines the contextual conditions under which each step or subtask can be executed, namely, the circumstances under which each step can be active. Returning to the illustrative guiding example of an HRI task, consider a scenario in which a robot is required to grasp a part, disassemble it, and then place it in another location. For safety reasons, the human counterpart is initially positioned behind the robot until the part is securely grasped. Subsequently, after the robot successfully grips the part, the human selects either the right or left side from which the robot can operate. Meanwhile, the robot remains stationary, awaiting the human's positioning. Once the human has taken his/her place, he/she can specify whether he/she intends to cooperate or collaborate with the robot. In the case of cooperation, both the human and the robot jointly undertake independent disassembly tasks, with the robot responsible for detaching the affixed part and depositing it into a designated container. Conversely, in a collaborative setup, the human assists the robot in extracting the part, after which the human places it into the container. In this example, the task can be segmented into the following steps "*Pick*", "*Wait*", "*Extract*", and "*Place*". At the same time, the situations might encompass the following: "*Human Behind the Robot*", "*Cooperative Human Left*", "*Cooperative Human Right*", "*Collaborative Human Left*", "*Collaborative Human Right*", "*Disassembled Cooperative Human Left*", and "*Disassembled Cooperative Human Right*".

Similar to HRL, where a subtask is determined by an initiation condition I_ω and a termination condition β_ω, steps and situations are defined by three sets of constraints: general, start, and end. Initially, general constraints must be met to detect the occurrence of a step or situation. The start constraints are assessed to verify the beginning of the step or situation, whereas the fulfillment of the end constraints signifies its conclusion, irrespective of the general constraints. A step is only interpreted when the relevant situation context is presented during the exploration phase. Both steps and situations encompass continuous actions and are delineated through various types of constraints. These constraints are applicable to ULISES observations, situations, and steps and encompass all types of restrictions defined in Section 3.3.

Within each interpretation cycle, the interpreter systematically evaluates steps and situations defined by constraints against a designated set of ULISES observations, determining their compliance and retaining the constraint evaluations and corresponding observations in memory. This record is continuously updated to maintain temporal consistency. Beyond constraint assessments, the Interpretation Subsystem determines the interpretation state of a step or situation, simulating the cognitive processes akin to those of a real domain expert. According to this schema, the interpreter determines whether an interval is initiated, concluded, or extended, contingent upon the plausible results of constraint evaluations.

4.4. Level III: evaluating the reinforcement learning agent's performance

The Diagnosis Subsystem evaluates the actions of the RL agent. It collects observations generated by the Observation Subsystem and the interpreted steps and situations from the Interpreter. It is noteworthy that the Diagnosis Subsystem supports different diagnosis techniques. In this study, the implemented supervisor employs a constraint satisfaction-based diagnostic approach that is tailored to the features of the used domains. Specifically, constraint-based modeling provides a framework to specify the execution of certain actions or behaviors within ill-defined domains.

The Diagnostic Subsystem adheres to a standardized procedure that remains agnostic to the specific diagnosis technique employed. Similar to the Interpretation Subsystem, this subsystem maintains the current observations by processing the notification messages received from the Observation Subsystem. These notifications include various updates, such as "*new observation*", "*observation pending*", "*cancel observation*", "*update properties*", "*end observation*", and "*continue observation*". Likewise, the Interpreter informs the Diagnostic Subsystem of any changes in the status of the steps and situations. Once this information has been updated, the Diagnostic Subsystem diagnoses all ongoing situations by generating and sending feedback. The steps and situations delineated at the Interpretation level are interlinked with those at the Diagnostic level. A step can manifest in various situations; similarly, multiple steps may occur in a single situation. For example, consider the HRI disassembly task. Fig. 4 illustrates the correlation among the situations and steps.

At this level, each situation is accompanied by a given set of associated solutions. A solution entails specifying how the steps should be solved in the context of the given situation. To diagnose the activity of the RL agent, both the situation and associated steps must be active. Each step can comprise one or multiple conditions. The conditions are enacted through constraints, and their satisfaction or nonsatisfaction determines whether the supervisor issues a positive or negative reward to the RL agent. Each condition may carry an associated reward or penalty contingent upon compliance with the associated constraint. If multiple conditions cease to be met simultaneously, ULISES consolidates the feedback by sending a single reward or penalty if the rewards or penalties associated with the conditions are equal. However, if the rewards or penalties differ, ULISES provides the reward or penalty with the highest absolute value.

Furthermore, beyond providing evaluative feedback, ULISES can provide corrective feedback. The supervisor facilitates the evaluation of the RL agent's exploration trajectories or trends by analyzing the values stored in its memory. These data can be employed to determine the timing of corrective feedback based on a predefined trend consistency threshold, which can be tailored according to the task requirements.

If the agent's exploration trend is inadequate, ULISES can provide evaluative feedback by sending discrete values according to the task. For tasks with continuous action spaces, ULISES also provides corrective trends through discrete values ranging from -1 to 1 for each of the action components available to the agent. Here, a value of -1 or 1 signifies a decrease or increase in the action value, respectively, and a value of 0 indicates that the supervisor refrains from influencing the agent's actions. However, it is necessary to consider that for these exploration trend hints to be effective, the values provided by the supervisor should be used to adjust the agent's policy parameters via gradient updates [37].

5. Experiments

We chose to evaluate the effectiveness of the proposed framework using a simplified real-world scenario, specifically focusing on the removal of magnetic gaskets from refrigerator doors in an HRI setting (Fig. 5a).

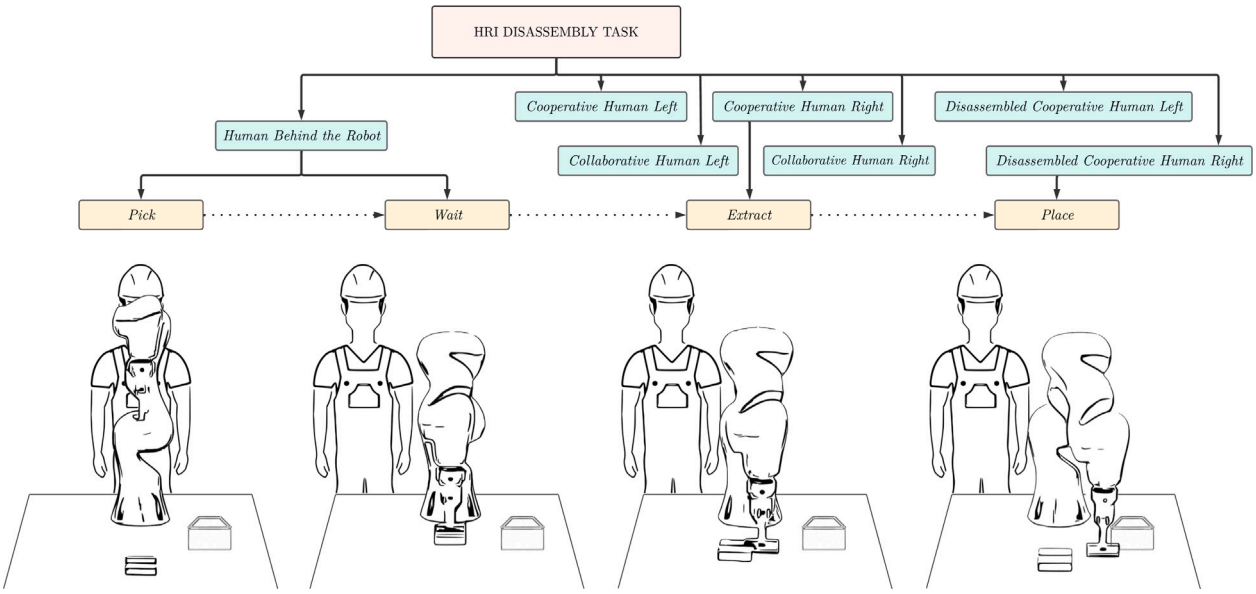


Fig. 4. Interrelations and sequencing of situations and steps in an HRI disassembly task. The task begins with the worker behind the robot for safety, activating the "*Human Behind the Robot*" situation. Two steps occur: the robot picks or grasps the part and then awaits the human's decision on the working side. When the worker positions himself, the robot proceeds to "*Extract*", which can occur in four situations. If the worker disassembles alone and is to the robot's right, the "*Extract*" step follows the "*Cooperative Human Right*" situation. After extraction, the robot places the part in a container on its left, evaluated under the "*Disassembled Cooperative Human Right*" situation.

Building on the groundwork established in our previous study [45], we conducted a proof-of-concept study focused on the execution of contact-rich tasks contingent upon the HRI level. To streamline the complexities associated with the extraction process, we devised two distinct rigid components: a stationary base affixed to a refrigerator door and a peg inserted within a corresponding slot and grasped by the robotic end-effector. The dimensions of the base were 0.2 m × 0.1 m × 0.03 m, and while those of the slot were 0.2 m × 0.02 m × 0.01 m. The slot was precisely aligned with the upper surface of the base at a depth of 0.01 m.

The robot is tasked with extracting the peg in the direction of the human or in the opposite to his/her location depending on whether he/she is working "collaboratively" or "cooperatively", respectively, as depicted in Fig. 5b. To replicate human presence within the simulation environment, actual measurements of an individual's upper body were collected in the laboratory via skeleton tracking. The motion capture process employed an Intel RealSense L515 camera positioned on the laboratory ceiling, providing an overhead view of the surroundings, in conjunction with YOLOv7 pose estimation.[1] To introduce a degree of uncertainty into the simulation environment, we introduced a perturbation to enhance the policy's adaptability to image capture noise. This perturbation applied random noise with magnitudes of up to 0.02 m in the x, y, and z components of key body points within the skeleton. Likewise, the friction (μ) between the base and the peg was randomized between 0 and 0.1.

In compliance with safety protocols, the human worker initiates the task while being positioned behind the robot and subsequently selects a side at a randomized time step (Fig. 5c). Consequently, the robot waits for the human to act accordingly. As the disassembly begins to introduce a slightly further increment in the environment's uncertainty, the worker may randomly transition from Point 2 to Point 3 or vice versa. Similarly, a random transition between the HRI modes from "cooperative" to "collaborative" or vice versa occur. In this demonstration, the robot is depicted as already grasping the peg at the beginning of the episode. This task serves as a simplification of the example presented in Section 4 to explain how the framework works.

[1] https://github.com/RizwanMunawar/yolov7-pose-estimation.

Based on the information provided, the intrinsic components of the RL framework are defined as follows.

State space: The selected observations encompass the robot's TCP's relative position at the current time step with respect to the initial position across x, y, and z spatial axes ($tcp_{rel_{x,y,z}}$) and its value in the previous time step ($ptcp_{rel_{x,y,z}}$). Using relative positions is recommended to enhance the agent's generalization capability by eliminating the reliance on specific axis configurations. As a result, the disassembly part does not need to maintain a fixed position. However, for this particular task, its position was presumed to remain constant owing to the alignment facilitated by the pistons of the disassembly chain (Fig. 5a). The collision forces between the peg and the base for all three spatial axes ($F_{x,y,z}$) were measured. Furthermore, the state space included the relative position of each key body point of the human with respect to the robot's TCP's position ($j_{i_{rel_{x,y,z}}}$). Lastly, the state space considers four more aspects: the relative position between the target (where the part is to be placed if working cooperatively) and the robot's TCP's position in the x and y axes ($target_{rel_{x,y}}$); the number of complete time steps relative to the total duration of the episode (t_{ep}); the progress of peg extraction, expressed as a percentage (p_{extr}); and the type of HRI mode used by the human and robot (HRI_{mode}). Thus, the state space S was defined as follows:

$$S = [tcp_{rel_{x,y,z}}, ptcp_{rel_{x,y,z}}, F_{x,y,z}, j_{i_{rel_{x,y,z}}}, target_{rel\,x,y}, t_{ep}, p_{extr}, HRI_{mode}], \tag{12}$$

Action space: The choice of an action space significantly affects the robustness and performance of the learned policy. Employing a task space approach enables sending the pose commands in Cartesian coordinates to the robot's internal controller, potentially enhancing robustness and expediting the learning process by augmenting sample efficiency [46]. Therefore, action space A is defined as follows:

$$A = [\Delta x, \Delta y], \tag{13}$$

where Δx and Δy are translational motions along the Cartesian x and y axes, respectively.

5.1. Observation model

The observations and properties used by ULISES are not exclusively reliant on data transmitted from the environment where the

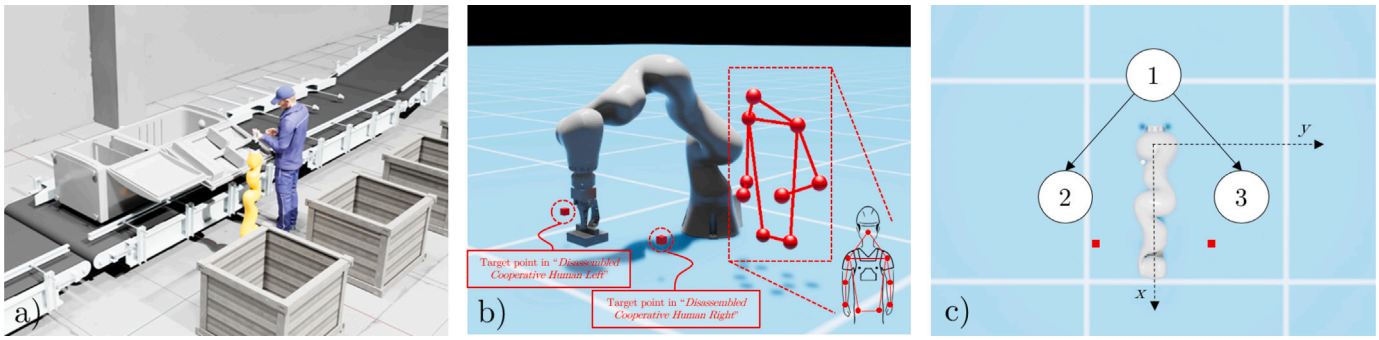


Fig. 5. (a) Transformation of the actual disassembly line into an HRI environment incorporating the collaborative robot. (b) Depending on whether working on cooperative or collaborative mode, the robot is tasked with extracting the peg to either the opposite or same side as the worker, respectively. Furthermore, when operating in cooperative mode, following the extraction, the robot must approach the nearest red cube, which represents the location of the tank where the peg must be deposited. (c) Top view of the simulation environment. To ensure safety, the worker initiates the task while being positioned behind the robot (Point 1) until the robot successfully grasps the gasket. Subsequently, the worker randomly selects one side of the robot (Point 2 or Point 3) to operate cooperatively or collaboratively.

Table 1
Observations and properties defined in the HRI disassembly Observation Model.

Observation	Properties
Robot	TCPPositionX/Y/Z TCPPreviousPositionX/Y/Z Velocity ForceX/Y/Z TimeStep Displacement HRIMode TargetEuclideanDistance TrendX/Y
Human	TF_i X/Y/Z TFMinEuclideanDistanceX/Y
RobotTCP	TCPPositionX/Y/Z

RL agent operates; instead, the supervisor can internally generate the required information for diagnosis. These observation properties are stored in memory, facilitating access to subsequent calculations by the supervisor. Table 1 lists the observations and their corresponding properties for constructing the Observation Model tailored to the HRI disassembly task. Specifically, the "*Velocity*" property represents the derivative of the TCP position. Furthermore, to compute the "*TargetEuclideanDistance*", "*TFMinEuclideanDistanceX*" and "*TFMinEuclideanDistanceY*" properties, ULISES calculates the Euclidean distances between the target location and the robot's TCP position, and all human tracking features (TFs) and the robot's TCP, respectively. Subsequently, for the minimum Euclidean distance between the human and the robot, ULISES selects the x and y-components of the nearest TF of the human worker. Lastly, the "*TrendX*" and "*TrendY*" properties represent the derivative computed between the TCP position along the x and y axes at the current time step and its value from 10-time steps ago, respectively.

At times, particularly during the exploration phase, a robot's movements may lack precision, thereby making it difficult to discern its intent. In cases where these types of imprecise motions occur and the occurrence or non-occurrence of an observation cannot be definitively determined, we propose employing a fuzzy logic approach. Specifically, we generate the fuzzy observation "*RobotTCP*", which only requires the position of the TCP in Cartesian coordinates. This observation is used during the "*Place*" step, where executing different trajectories may lead the robot to reach the same target point. ULISES internally calculates the realized trajectory from the angle, radius, and length of the motion. For more details, please refer to [43]. This observation is used at the Interpretation level to capture the intentionality of the RL agent, thereby facilitating its diagnosis at the Diagnosis level.

5.2. Interpretation model

The HRIMode property is delineated as a floating-point variable derived from a Boolean value, where 0.0 indicates a "cooperative" interaction and 1.0 indicates a "collaborative" interaction. If operating in "cooperative" mode, the robot is required to remove the part to the opposite side of the worker and subsequently deposit it into a trash receptacle. Conversely, when operating in "collaborative" mode, the robot is tasked with removing the part toward the worker's side before delivering it to the worker. Table A.4 in Appendix A lists the steps and situations outlined in the Interpretation Model for the task.

5.3. Task model

Table B.5 in Appendix B outlines the task resolution considering the defined steps and situations. For this task, it was determined that the corrective action provided by ULISES should be ±0.015 m, based on the positional trajectories recorded in the "*TrendX*" and "*TrendY*" properties of the "*Robot*" observation.

5.4. Training and evaluation

The proposed framework is universally compatible with any RL agent. In this study, training was conducted using soft actor-critic (SAC) [47], deep deterministic policy gradient (DDPG) [48], and proximal policy optimization (PPO) [49], which are the most commonly employed RL agents in contact-rich manipulation tasks [10].

For training and evaluation, the physical environment was replicated using the Omniverse Isaac Sim simulator via the skrl library [50]. All training and evaluation were executed using ULISES on a laptop equipped with a 2,7 GHz Intel Core i7-10850H CPU. The RL environment was run on a separate laptop with a 3,20 GHz Intel Xeon W-11855M CPU, 128 GB of RAM, and an NVIDIA RTX A5000 GPU with 24 GB of VRAM. Additional details about the hyperparameters selected for the agents are presented in Table 2. The agents employ a two-layer hidden neural architecture that includes 128 and 64 neurons. Fig. 6 shows the mean and standard deviation of the rewards obtained over 10 training sessions for each agent when only evaluative feedback is provided against instances where both evaluative and corrective feedback are used.

Examining the SAC curves, in cases relying solely on evaluative feedback, the learning curve stabilized at approximately 120,000 time steps, and the algorithm converged to a local minimum. However, the integration of corrective feedback resulted in the stabilization of the curve occurring approximately 60,000 time steps. This sample efficiency improvement was also evident in the DDPG curves, where, with evaluative feedback alone, the learning curve stabilized at approximately 80,000 time steps and converged to a local minimum.

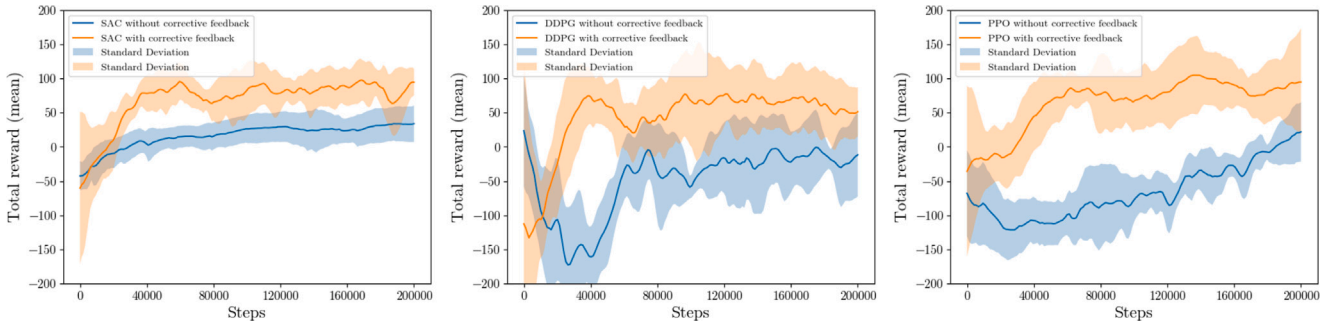


Fig. 6. Mean reward and standard deviation of the rewards perceived by SAC, DDPG, and PPO during the HRI disassembly task simulation training.

Table 2
Choice of hyperparameters for actor-critic agents.

Agent	Parameters	Value
SAC	Memory size	15 625
	Batch size	4096
	Discount factor γ	0.99
	Learning rate η	$5 \cdot 10^{-4}$
	Initial entropy value	1
DDPG	Memory size	15 625
	Batch size	4096
	Discount factor γ	0.99
	Learning rate η	$5 \cdot 10^{-4}$
	Noise type	Ornstein–Uhlenbeck
	θ	0.15
	σ	0.2
	Base scale	0.1
PPO	Rollouts	2048
	Learning epochs	16
	ϵ	0.2
	Mini batches	512
	Discount factor γ	0.99
	Learning rate η	$5 \cdot 10^{-4}$

By integrating corrective feedback, the learning process stabilized at approximately 40,000 time steps, indicating a notable acceleration of convergence by approximately 50% for both actor-critic agents. In addition, the RL agents demonstrated enhanced performance, obtaining a higher mean reward value, and achieving better results when executing the entire task. Finally, for the PPO agent, the learning process continued for 200,000 time steps when only the evaluative feedback was used. In contrast, with the combination of evaluative and corrective feedback, the curve stabilized at approximately 60,000 time steps.

Nonetheless, the noise evident in the SAC learning curve when corrective feedback was integrated is worth highlighting compared with the more stable curve observed with evaluative feedback alone. These fluctuations may stem from the corrective feedback provided by ULISES. By providing a predefined and empirically selected numerical value as corrective action for both components within the agent's action space, there exists a possibility that these actions may not always be optimally suited to the states they are intended for. Consequently, although these recommendations may enhance the sample efficiency, they can disrupt the convergence of the algorithm. This observation prompts us to consider how ULISES provides corrective actions. One possible approach involves the artificial supervisor becoming aware of this scenario and gradually reducing its corrective intervention as the RL agent acquires more knowledge. Alternatively, as mentioned in Section 4.4, another viable option could involve the supervisor imparting directional cues rather than precise numerical values.

Despite these considerations, 10 evaluations were conducted with each agent using the best neural weights derived from the best learning curves. Each evaluation included 200,000 time steps. The average success rates of 97.3%, 86.5% and 94.8% were achieved for SAC, DDPG, and PPO, respectively.

5.5. Experiments on a real system

The performance of the learned policy was evaluated using a KUKA LBR Iiwa robot (Fig. 7). The framework [51] was used to deploy the RL control policy in a real-world setting. To ensure safe disassembly and provide safety measures against potential collisions with humans, a compliance controller operating in the end-effector position space was implemented. If the threshold of $F_{max} = 3\,\text{N}$ was exceeded, the robot entered a safety stop.

Evaluations were conducted using SAC weights because they demonstrated the best performance in the simulation. For this purpose, two pairs of disassembly parts, each with tolerances of 10^{-3} and 5×10^{-4} m, influencing the friction during the extraction, were used. Twenty extraction attempts were performed for each type of part under varying conditions. These conditions included the worker's position (left, right, or transitioning between sides) and nature of the interaction between the worker and the robot (cooperative, collaborative, or transitioning between these modes). Table 3 presents the results of these evaluations for each part and condition. The policy achieved a disassembly success rate of 91.88%. However, the task success rate decreased slightly during transitions between the two HRI modes or when moving from one side to the other. In cases of significant human movement, the skeleton tracking system occasionally misidentifies the robot's joints as those of a human. This misidentification can lead to a policy of executing motions based on the robot's location rather than a person's. To address this issue, a more sophisticated monitoring system should be implemented. The simulation and experimental results are available at https://youtu.be/94Hnh0PluxU.

Despite these minor limitations, the results demonstrate the effectiveness of the proposed framework. Compared with our previous study [45], in which the robot performed only peg extraction cooperatively, the success rate remained largely unaffected and, in some instances, even improved. For example, with the 10^{-3} m tolerance part, the success rate in our previous study was 95%, whereas in this case, it was 93.75%. However, as the tolerance decreased, such as with the 5×10^{-4} m part, the previous study showed a decline to 84.5%, whereas in the current study, the success rate remained higher at 90%. In the previous study, the reward function was a complex mathematical expression; however, these findings suggest that using simpler binary rewards, combined with corrective feedback, can improve sample efficiency and agent performance. Nonetheless, these tests are still far from simulating the complexity of a real disassembly plant, so further task complexity is required before transitioning to production environments.

6. Discussion

The research and application of RL in robotics have experienced exponential growth in recent years; however, its full potential remains

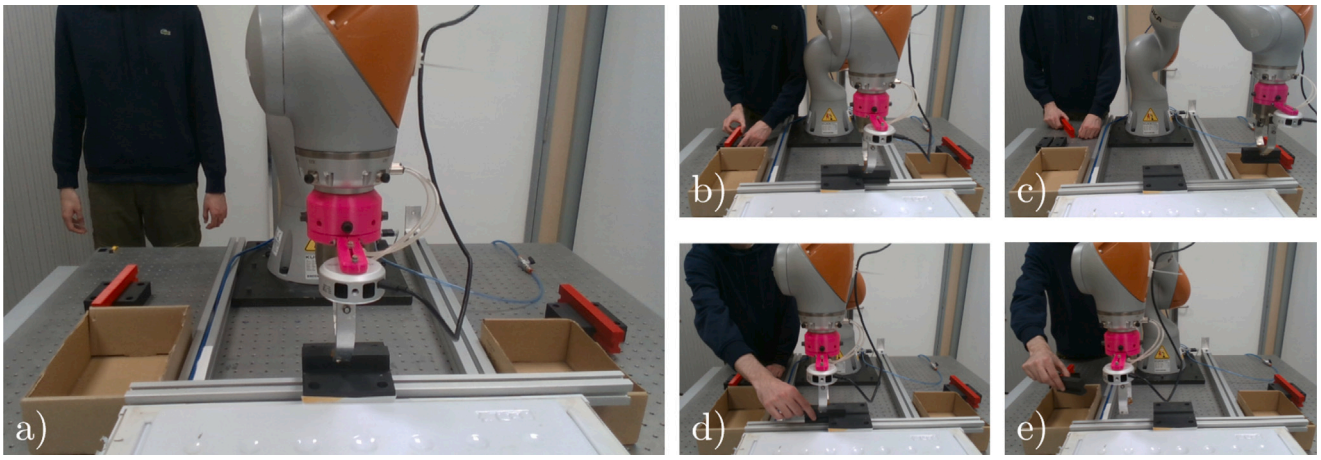


Fig. 7. Real-world experiments showcasing cooperative and collaborative subtasks: (a) subtask "*Wait*"; (b) subtask "*Extract*" working cooperatively; (c) subtask "*Place*" working cooperatively; (d) subtask "*Extract*" working collaboratively; (e) subtask "*Place*" working collaboratively.

Table 3
Evaluations on a real system involving the disassembly of parts with varying tolerances, performed cooperatively and collaboratively, with the worker positioned to either the left or right of the robot or moving between both sides.

Tolerance between parts	Coop.				Col.				Coop.- Col.		Col.- Coop.	
	L	R	L-R	R-L	L	R	L-R	R-L	L	R	L	R
10^{-3} m	(19/20) 95%	(20/20) 100%	(18/20) 90%	(18/20) 90%	(20/20) 100%	(20/20) 100%	(19/20) 95%	(19/20) 95%	(18/20) 90%	(19/20) 95%	(17/20) 85%	(18/20) 95%
5×10^{-4} m	(19/20) 95%	(18/20) 90%	(18/20) 90%	(17/20) 85%	(20/20) 100%	(20/20) 100%	(16/20) 80%	(19/20) 95%	(17/20) 85%	(18/20) 90%	(17/20) 85%	(17/20) 85%

Interaction level: Cooperative (Coop.), Collaborative (Col.). **Worker location relative to the robot**: left side (L), right side (R), from left to right side (L-R), from right to left side (R-L).

largely untapped. Much of the current research in this field has focused on basic manipulation or motion planning tasks, which are not feasible in complex industrial environments. To equip robots with the capabilities required for such contexts, it is necessary to enable them to chain together various subtasks effectively, thereby enabling the performance of more intricate operations. Achieving this objective hinges on exploring two active lines of research in this machine learning control technique, namely reward shaping and sample efficiency [9,10].

In this regard, interactive RL is a promising alternative. This paradigm integrates an external artificial or human supervisor into the learning loop of RL agents. In particular, in scenarios involving continuous action-space pairs, current studies tend to adopt human-in-the-loop approaches. Humans leverage their cognitive capabilities and the models generated from their life experiences to provide evaluative and corrective feedback as appropriate. However, prolonged learning sessions that span hours render human supervision impractical. Consequently, this study proposes a shift away from human-in-the-loop setups to employing an artificial supervisor using constraint-based modeling techniques. This approach offers greater flexibility in terms of encapsulating human cognition than conventional RL, which relies on logical constraints. By abstracting complex tasks into manageable subtasks and contexts, the artificial supervisor facilitates the specification of rewards, penalties, and corrective actions for the RL agent in uncertain environments.

The proposed interactive RL framework employs an adapted version of the ULISES framework, which was initially designed for tutoring and testing tasks. Comprising three hierarchical levels (observation, interpretation, and diagnosis), ULISES provides human-like reasoning capabilities: observing the environment, interpreting events, and providing rewards or penalties based on evaluations. The proposed framework allows any instructional designer to define rewards without programming knowledge. To validate the framework's efficacy, a procedural disassembly task (segmented into subtasks) within an HRI work environment was tested. The artificial supervisor provided evaluative and corrective feedback to improve the sample efficiency and increase the mean cumulative task reward by approximately 50%. Despite the promising outcomes, this study identified several avenues for future research.

The first area of future research will involve exploring the framework's efficacy in more intricate tasks comprising more subtasks. It would be advisable to analyze how a policy's learning of initial subtasks may impact its ability to master subsequent tasks and vice versa. Of particular interest is the potential influence of the neural network's memorization capacity on task performance. In this sense, considering approaches that incorporate recurrent neural networks or applying the proposed framework within HRL could offer insights into effectively addressing these challenges.

Another avenue for future exploration lies in deepening the ability to model observations using fuzzy logic. Although this study presents a basic example, we believe that such modeling can be extended to complex environments in which the robot's trajectories are not intuitively clear. An example scenario is the disassembly of a real magnetic gasket. These gaskets usually have varying geometries and adhesion forces. In these cases, defining a mathematical reward function to guide the robot learning may not be a straightforward or interpretive task.

Lastly, in continuous state action pair tasks, a shift toward providing hints rather than specific corrective actions may be advantageous. Although the proposed framework sends hints, such as +1, −1, or 0, representing action augmentation, reduction, or further exploration by the agent, respectively, the current setup only sends specific predefined corrective actions. These predefined actions, albeit suboptimal, contribute to the noise observed in the curve depicted in Fig. 6. To allow the RL agent to incorporate these hints, gradient updates are required to adjust the agent's policy parameters. Such an implementation holds significant value, especially considering that human supervisors typically focus on individual actions [37]. In contrast, ULISES possesses

the capability to can provide corrective trends for all parameters constituting the agent's action space, thus offering a more comprehensive guidance and learning approach.

7. Conclusions

This paper proposes a framework for automated interactive RL that replaces the need for human-in-the-loop approaches with an artificial supervisor that features constraint-based modeling techniques. The supervisor provides evaluative and corrective feedback to the RL agent during procedural task learning. Unlike conventional RL, which relies primarily on logical constraints, constraint-based modeling techniques offer enhanced flexibility in representing human knowledge about a task. This includes temporal constraints and considerations of properties represented by fuzzy logic. For its implementation, we leveraged and adapted the ULISES tool, which was previously used in tutoring and testing tasks. ULISES emulates human cognitive processes across three principal stages: observing the environment, interpreting ongoing activities, and assessing their correctness within a given context. Although the evaluation of this framework demonstrates promising results in procedural tasks that are divisible into subtasks, further research efforts are required to assess its potential in even more complex and realistic scenarios.

CRediT authorship contribution statement

Íñigo Elguea-Aguinaco: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Conceptualization. **Aitor Aguirre-Ortuzar:** Writing – review & editing, Methodology, Conceptualization. **Unai Izagirre-Aizpitarte:** Writing – review & editing, Conceptualization. **Ibai Inziarte-Hidalgo:** Visualization. **Simon Bøgh:** Writing – review & editing, Supervision. **Nestor Arana-Arexolaleiba:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Inigo Elguea-Aguinaco reports financial support was provided by Basque Government Department of Economic Development, Sustainability, and Environment.

Acknowledgments

This work has been partially funded by the Basque Government Department of Economic Development, Spain, Sustainability and Environment through the Bikaintek 2020 program.

Appendix A. Interpretation model

This section gathers all the steps and situations, along with their corresponding constraints, as defined in the Interpretation Model.

Appendix B. Task model

This section presents all the solutions defined for each step and its corresponding situation in the HRI disassembly task.

Data availability

No data was used for the research described in the article.

Table A.4
Definition of steps and situations that compose the HRI disassembly interpretation model.

Step(s)	Observation(s)	Instance(s)	Constraint(s)
Wait	Human	h	*General:* h.TFMinEuclideanDistanceX < 0.0
Description: As a safety measure, the human remains positioned behind the robot while the part is being grasped. Subsequently, upon grasping the part, the robot is required to wait for the worker to position him/herself adjacent to the robot. The "*Wait*" step stays active while the worker remains behind the robot. Assuming the robot's location at the coordinate origin, the human is deemed to be positioned behind the serial manipulator if the value of his/her TF closest to the robot's TCP in the x component is less than 0.			
Extract	Robot	r	*General*: r.Displacement ≥ -0.11 AND r.Displacement ≤ 0.11
Description: The base, into which the peg is inserted, spans a length of 0.2 m. Given that the peg is centrally aligned and grasped by the robot, a displacement within the range of −0.1 to 0.1 m is necessary for the peg's extraction. However, to enable ULISES to diagnose each action at every time step, the step must remain consistently active. On occasion, an action executed by the RL agent may result in the robot concluding the episode outside these defined limits. Consequently, it becomes necessary to slightly extend these limits within the Interpretation Model to ensure the step remains active.			
Place	Robot	r	*Start*: r.Displacement ≤ -0.1 OR r.Displacement ≥ 0.1, *General*: r.TargetEuclideanDistance > 0.01
Description: In contrast to the "*Wait*" and "*Extract*" steps, the "*Place*" step is subject to an initial condition that initiates its activation. Precisely, this step becomes active upon the completion of the extraction process and persists in this state while the Euclidean distance between the robot TCP and the target point is above 0.01 m.			
Situation(s)	Observation(s)	Instance(s)	Constraint(s)
Human Behind Robot	Human	h	*General*: h.TFMinEuclideanDistanceX < 0.0
Description: The "*Human Behind Robot*" situation remains active as long as the human location stays below 0. Assuming the robot is located at the origin of the coordinate system, during this period, the x component of the TF of the human nearest to the robot's TCP would exhibit a negative value.			
Cooperative Human Left	Robot Human	r h	*Start*: h.TFMinEuclideanDistanceX > 0.0, *General*: r.HRIMode == 0.0 AND h.TFMinEuclideanDistanceY > 0.0, *End*: r.Displacement ≤ -0.11 OR r.Displacement ≥ 0.11
Description: The variable HRIMode is equal to 0.0, so the robot works cooperatively. The activation of the "*Cooperative Human Left*" situation occurs when the y component of the TF nearest to the robot's TCP registers a positive value, indicating its location to the left of the robot.			

(*continued on next page*)

Table A.4 (*continued*).

Step(s)	Observation(s)	Instance(s)	Constraint(s)
Cooperative Human Right	Robot Human	r h	*Start*: h.TFMinEuclideanDistanceX > 0.0, *General*: r.HRIMode == 0.0 AND h.TFMinEuclideanDistanceY < 0.0, *End*: r.Displacement ≤ −0.11 OR r.Displacement ≥ 0.11
Description: The activation of the "*Cooperative Human Right*" situation occurs when HRIMode is equal to 0.0, and the *y* component of the TF nearest to the robot's TCP registers a negative value, indicating its location to the right of the robot.			
Collaborative Human Left	Robot Human	r h	*Start*: h.TFMinEuclideanDistanceX > 0.0, *General*: r.HRIMode == 1.0 AND h.TFMinEuclideanDistanceY > 0.0
Description: The variable HRIMode is equal to 1.0, so the robot is working collaboratively. On the other hand, the *y* component of the TF closest to the robot's TCP registers a positive value, indicating that the closest part of the human to the robot's TCP is to its left.			
Collaborative Human Right	Robot Human	r h	*Start*: h.TFMinEuclideanDistanceX > 0.0, *General*: r.HRIMode == 1.0 AND h.TFMinEuclideanDistanceY < 0.0
Description: The activation of the "*Collaborative Human Right*" situation occurs when HRIMode is equal to 1.0, and the *y* component of the TF nearest to the robot's TCP registers a positive value.			
Disassembled Cooperative Human Left	Robot Human	r h	*Start*: r.TCPPositionY ≤ −0.1, *General*: r.HRIMode == 0.0 AND h.TFMinEuclideanDistanceY > 0.0
Description: The "*Disassembled Cooperative Human Left*" situation becomes active once the part is disassembled to the opposite side of the human and remains under this status while the HRIMode equals 0.0 and the worker is positioned to the left of the robot.			
Disassembled Cooperative Human Right	Robot Human	r h	*Start*: r.TCPPositionY ≥ 0.1, *General*: r.HRIMode == 0.0 AND h.TFMinEuclideanDistanceY < 0.0
Description: The "*Disassembled Cooperative Human Right*" situation becomes active once the part is disassembled to the opposite side of the human and remains under this status while the HRIMode equals 0.0 and the worker is positioned to the right of the robot.			

Table B.5
Definition of the HRI disassembly task model.

Situation: Step	Condition(s)	Observation(s)	Instance(s)	Constraint(s)
Human Behind Robot: Wait	Static	Robot	r	r.Velocity == 0.0
Description: While the human maintains a position behind the robot, the robot is required to wait. If the robot refrains from movement, it receives a reward of +1; however, if it exhibits any non-zero speed, it incurs a penalty of −2.				
Cooperative Human Left: Extract	TCP direction	Robot	r	r.TCPPositionY < r.TCPPositionPreviousY
Description: In cooperative mode, the robot is tasked with extracting the peg to the opposite side of the human, positioned to its left. The extraction direction of the part is deemed correct if the value of the robot's TCP in the *y* component is lower than that in its previous time step. If the TCP value at the current time step is lower than that at the previous time step, the supervisor assigns a positive reward of +1. Conversely, a penalty of −2 is provided otherwise.				
Cooperative Human Left: Extract	On time	*STEPS::Extract	e	Duration(e) < 500
Description: One of the objectives of the task is to expedite the extraction process. To achieve this, each episode is allotted 500 time steps. Failure to complete the extraction within this timeframe incurs a penalty of −50 from the supervisor, prompting an automatic restart of the episode. **Note*: The *STEPS* prefix indicates that the instance is a step, not an observation.				
Cooperative Human Left: Extract	Safety distance	Human	h	h.MinEuclideanDistance > 0.15
Description: In HRI settings, ensuring human safety takes precedence. When the robot operates in "cooperative" mode, the extraction must proceed toward the opposite side of the worker to prevent potential collisions. Consequently, if the extraction direction is incorrect and the distance between the robot's TCP and the closest part of the human's body falls below 0.15 m, the supervisor issues a negative reward of −50, prompting the automatic restart of the episode.				
Cooperative Human Left: Extract	Incorrect disassembly	Robot	r	r.Displacement < 0.1
Description: In the "cooperative" mode, if the extraction is carried out towards the side of the human, it is deemed an erroneous disassembly. In such cases, the supervisor issues a negative reward of −50, and the episode is restarted.				

(*continued on next page*)

Table B.5 (*continued*).

Situation: Step	Condition(s)	Observation(s)	Instance(s)	Constraint(s)
Cooperative Human Left: Extract	Correct disassembly	Robot STEPS::Extract STEPS::Place	r e p	r.Displacement ≥ −0.1 AND e [overlaps] p
Description: In the "cooperative" mode, if the extraction is performed towards the opposite side of the human, it is considered a correct disassembly. In such instances, the supervisor provides a reward of +20, and the episode continues with the "*Place*" step.				
Cooperative Human Right: Extract	TCP direction	Robot	r	r.TCPPositionY > r.TCPPositionPreviousY
Description: In the "cooperative" mode with the human positioned on the right, the extraction direction is deemed correct if the y component of the robot's TCP in the current time step is greater than that of the previous time step. If this happens, the supervisor issues a reward of +1. Conversely, if the condition is not met, a penalty of −2 is provided.				
Cooperative Human Right: Extract	On time	STEPS:: Extract	e	Duration(e) < 500
Description: Refer to condition "*On time*" from "*Cooperative Human Left: Extract*".				
Cooperative Human Right: Extract	Safety distance	Human	h	h.MinEuclideanDistance > 0.15
Description: Refer to condition "*Safety distance*" from "*Cooperative Human Left: Extract*".				
Cooperative Human Right: Extract	Incorrect disassembly	Robot	r	r.Displacement > −0.1
Description: Refer to condition "*Incorrect disassembly*" from "*Cooperative Human Left: Extract*".				
Situation: Step	Condition(s)	Observation(s)	Instance(s)	Constraint(s)
Cooperative Human Right: Extract	Correct disassembly	Robot STEPS::Extract STEPS::Place	r e p	r.Displacement ≥ 0.1 AND e [overlaps] p
Description: Refer to condition "*Correct disassembly*" from "*Cooperative Human Left: Extract*".				
Collaborative Human Left: Extract	TCP direction	Robot	r	TCPPositionY > r.TCPPositionPreviousY
Description: In "collaborative" mode, the robot is tasked with extracting the peg to the human's side, located to its left. The extraction direction of the part is deemed correct if the value of the robot's TCP in the y component is greater than that in its previous time step. If the TCP value at the current time step is higher than that at the previous time step, the supervisor assigns a positive reward of +1. Conversely, a penalty of −2 is provided otherwise.				
Collaborative Human Left: Extract	On time	STEPS::Extract	e	Duration(e) < 500
Description: Refer to condition "*On time*" from "*Cooperative Human Left: Extract*".				
Collaborative Human Left: Extract	Incorrect disassembly	Robot	r	r.Displacement > −0.1
Description: In the "collaborative" mode, if the extraction is carried out to the opposite side of the human, it is deemed an erroneous disassembly. In such cases, the supervisor issues a negative reward of −50, and the episode is restarted.				
Collaborative Human Left: Extract	Correct disassembly	Robot	r	r.Displacement ≥ 0.1
Description: In the "cooperative" mode, if the extraction is performed towards the side of the human, it is considered a correct disassembly. In such instances, the supervisor provides a reward of +20, and the episode is restarted.				
Collaborative Human Right: Extract	TCP direction	Robot	r	r.TCPPositionY < r.TCPPositionPreviousY
Description: In the "collaborative" mode with the human positioned on the right, the extraction direction is deemed correct if the y component of the robot's TCP in the current time step is lower than that of the previous time step. If this happens, the supervisor issues a reward of +1. Conversely, if the condition is not met, a penalty of −2 is provided.				
Collaborative Human Right: Extract	On time	STEPS::Extract	e	Duration(e) < 500
Description: Refer to condition "*On time*" from "*Cooperative Human Left: Extract*".				

(*continued on next page*)

Table B.5 (*continued*).

Situation: Step	Condition(s)	Observation(s)	Instance(s)	Constraint(s)
Collaborative Human Right: Extract	Incorrect disassembly	Robot	r	r.Displacement < 0.1
Description: Refer to condition "*Incorrect disassembly*" from "*Collaborative Human Left: Extract*".				
Collaborative Human Right: Extract	Correct disassembly	Robot	r	r.Displacement ≤ −0.1
Description: Refer to "*Correct disassembly*" from "*Collaborative Human Left: Extract*".				
Disassembled Cooperative Human Left: Place	Approach	*FUZZY:: RobotTCP	rtcp	rtcp.Arc(>50.0, <0.1, <0.2). Fuzzy(≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥50.0 AND <100.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0)
Description: After the peg is removed, the robot's TCP will be approximately at coordinates [0.65, −0.1]. The target location for placing the peg is [0.55, −0.25]. The Arc(>50.0, <0.1, <0.2) property specifies that the angle should exceed 50 degrees, the radius should be less than 0.1 m, and the linear distance should be less than 0.2 m. Additionally, the fuzzy membership degree should primarily correspond to the tenth of the twenty-six classes into which the motion space is divided, corresponding to <−1, −1, 0>. If the trajectory of the robot's TCP during the "*Place*" step meets these conditions, the supervisor provides a reward of +1; otherwise, a penalty of −2 is given. **Note*: The *FUZZY* prefix indicates that the observation is of fuzzy type.				
Situation: Step	Condition(s)	Observation(s)	Instance(s)	Constraint(s)
Disassembled Cooperative Human Left: Place	On time	STEPS::Place	p	Duration(p) < 500
Description: Refer to condition "*On time*" from "*Cooperative Human Left: Extract*".				
Disassembled Cooperative Human Left: Place	Reached	Robot	r	r.TargetEuclideanDistance ≤ 0.02
Description: The agent is considered to have reached the target point when the Euclidean distance between the TCP and the target is 0.02 m or less. Upon reaching this criterion, the supervisor issues a reward of +20 as an acknowledgment of the achievement.				
Disassembled Cooperative Human Right: Place	Approach	FUZZY:: RobotTCP	rtcp	rtcp.Arc(>300.0, <0.1, <0.2). Fuzzy(≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥50.0 AND <100.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0, ≥0.0)
Description: After the peg is removed, the robot's TCP will be approximately at coordinates [0.65, 0.1]. The target location for placing the peg is [0.55, 0.25]. The Arc(>300.0, <0.1, <0.2) property specifies that the angle should exceed 300°, the radius should be less than 0.1 m, and the linear distance should be less than 0.2 m. Additionally, the fuzzy membership degree should primarily correspond to the eighth of the twenty-six classes into which the motion space is divided, corresponding to <-1, 1, 0>. If the trajectory of the robot's TCP during the "*Place*" step meets these conditions, the supervisor provides a reward of +1; otherwise, a penalty of −2 is given.				
Disassembled Cooperative Human Right: Place	On time	STEPS::Place	p	Duration(p) < 500
Description: Refer to condition "*On time*" from "*Cooperative Human Left: Extract*".				
Disassembled Cooperative Human Right: Place	Reached	Robot	r	r.TargetEuclideanDistance ≤ 0.02
Description: Refer to condition "*Reached*" from "*Disassembled Cooperative Human Left: Place*".				

References

[1] R.E. Andersen, S. Madsen, A.B. Barlo, S.B. Johansen, M. Nør, R.S. Andersen, S. Bøgh, Self-learning processes in smart factories: Deep reinforcement learning for process control of robot brine injection, Procedia Manuf. 38 (2019) 171–177.

[2] M. Hildebrand, R.S. Andersen, S. Bøgh, Deep reinforcement learning for robot batching optimization and flow control, Procedia Manuf. 51 (2020) 1462–1468.

[3] A. Orsula, S. Bøgh, M. Olivares-Mendez, C. Martinez, Learning to grasp on the moon from 3D octree observations with deep reinforcement learning, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2022, pp. 4112–4119.

[4] N. Vithayathil Varghese, Q.H. Mahmoud, A survey of multi-task deep reinforcement learning, Electronics 9 (9) (2020) 1363.

[5] J. Luo, E. Solowjow, C. Wen, J.A. Ojea, A.M. Agogino, Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2018, pp. 2062–2069.

[6] Z. Hou, Z. Li, C. Hsu, K. Zhang, J. Xu, Fuzzy logic-driven variable time-scale prediction-based reinforcement learning for robotic multiple peg-in-hole assembly, IEEE Trans. Autom. Sci. Eng. 19 (1) (2020) 218–229.

[7] X. Cheng, S. Liu, Dynamic obstacle avoidance algorithm for robot arm based on deep reinforcement learning, in: 2022 IEEE 11th Data Driven Control and Learning Systems Conference, DDCLS, IEEE, 2022, pp. 1136–1141.

[8] J.C. Kiemel, T. Kröger, Learning collision-free and torque-limited robot trajectories based on alternative safe behaviors, in: 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids), IEEE, 2022, pp. 223–230.

[9] M. Hutsebaut-Buysse, K. Mets, S. Latré, Hierarchical reinforcement learning: A survey and open research challenges, Mach. Learn. Knowl. Extr. 4 (1) (2022) 172–221.

[10] Í. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, N. Arana-Arexolaleiba, A review on reinforcement learning for contact-rich robotic manipulation tasks, Robot. Comput.-Integr. Manuf. 81 (2023) 102517.

[11] C.C. Beltran-Hernandez, D. Petit, I.G. Ramirez-Alpizar, K. Harada, Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach, Appl. Sci. 10 (19) (2020) 6923.

[12] G. Schoettler, A. Nair, J.A. Ojea, S. Levine, E. Solowjow, Meta-reinforcement learning for robotic industrial insertion tasks, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 9728–9735.

[13] A.A. Apolinarska, M. Pacher, H. Li, N. Cote, R. Pastrana, F. Gramazio, M. Kohler, Robotic assembly of timber joints using reinforcement learning, Autom. Constr. 125 (2021) 103569.

[14] Y. Wang, C.C. Beltran-Hernandez, W. Wan, K. Harada, Robotic imitation of human assembly skills using hybrid trajectory and force learning, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 11278–11284.

[15] J. Eschmann, Reward function design in reinforcement learning, Reinf. Learn. Algorithms: Anal. Appl. (2021) 25–33.

[16] T. Ren, Y. Dong, D. Wu, K. Chen, Learning-based variable compliance control for robotic assembly, J. Mech. Robotics 10 (6) (2018) 061008.

[17] T. Davchev, K.S. Luck, M. Burke, F. Meier, S. Schaal, S. Ramamoorthy, Residual learning from demonstration: Adapting dmps for contact-rich manipulation, IEEE Robot. Autom. Lett. 7 (2) (2022) 4488–4495.

[18] N. Lin, Y. Li, K. Tang, Y. Zhu, X. Zhang, R. Wang, J. Ji, X. Chen, X. Zhang, Manipulation planning from demonstration via goal-conditioned prior action primitive decomposition and alignment, IEEE Robot. Autom. Lett. 7 (2) (2022) 1387–1394.

[19] M. Braun, S. Wrede, Incorporation of expert knowledge for learning robotic assembly tasks, in: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, Vol. 1, IEEE, 2020, pp. 1594–1601.

[20] Y. Wang, C.C. Beltran-Hernandez, W. Wan, K. Harada, Hybrid trajectory and force learning of complex assembly tasks: A combined learning framework, IEEE Access 9 (2021) 60175–60186.

[21] C. Arzate Cruz, T. Igarashi, A survey on interactive reinforcement learning: Design principles and open challenges, in: Proceedings of the 2020 ACM Designing Interactive Systems Conference, 2020, pp. 1195–1209.

[22] W.B. Knox, P. Stone, Interactively shaping agents via human reinforcement: The tamer framework, in: Proceedings of the Fifth International Conference on Knowledge Capture, 2009, pp. 9–16.

[23] A.L. Thomaz, C. Breazeal, et al., Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance, in: Aaai, Vol. 6, Boston, MA, 2006, pp. 1000–1005.

[24] H.B. Suay, S. Chernova, Effect of human guidance and state space size on interactive reinforcement learning, in: 2011 Ro-Man, IEEE, 2011, pp. 1–6.

[25] F. Cruz, S. Magg, C. Weber, S. Wermter, Training agents with interactive reinforcement learning and contextual affordances, IEEE Trans. Cogn. Dev. Syst. 8 (4) (2016) 271–284.

[26] I. Sheidlower, A. Moore, E. Short, Keeping humans in the loop: Teaching via feedback in continuous action space environments, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2022, pp. 863–870.

[27] J.F. Allen, G. Ferguson, Actions and events in interval temporal logic, J. Logic Comput. 4 (5) (1994) 531–579.

[28] J. Lin, Z. Ma, R. Gomez, K. Nakamura, B. He, G. Li, A review on interactive reinforcement learning from human social feedback, IEEE Access 8 (2020) 120757–120765.

[29] F. Cruz, S. Magg, Y. Nagai, S. Wermter, Improving interactive reinforcement learning: What makes a good teacher? Connect. Sci. 30 (3) (2018) 306–325.

[30] M. Zimmer, P. Viappiani, P. Weng, Teacher-student framework: a reinforcement learning approach, in: AAMAS Workshop Autonomous Robots and Multirobot Systems, 2014.

[31] I. Moreira, J. Rivas, F. Cruz, R. Dazeley, A. Ayala, B. Fernandes, Deep reinforcement learning with interactive feedback in a human–robot environment, Appl. Sci. 10 (16) (2020) 5574.

[32] H. Ritschel, E. André, Real-time robot personality adaptation based on reinforcement learning and social signals, in: Proceedings of the Companion of the 2017 Acm/Ieee International Conference on Human-Robot Interaction, 2017, pp. 265–266.

[33] D. McDuff, A. Kapoor, Visceral machines: Risk-aversion in reinforcement learning with intrinsic physiological rewards, 2018, arXiv preprint arXiv:1805.09975.

[34] I. Akinola, Z. Wang, J. Shi, X. He, P. Lapborisuth, J. Xu, D. Watkins-Valls, P. Sajda, P. Allen, Accelerated robot learning via human brain signals, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2020, pp. 3799–3805.

[35] S.K. Kim, E.A. Kirchner, L. Schloßmüller, F. Kirchner, Errors in human-robot interactions and their effects on robot learning, Front. Robotics AI 7 (2020) 558531.

[36] S.C. Akkaladevi, M. Plasch, S. Maddukuri, C. Eitzinger, A. Pichler, B. Rinner, Toward an interactive reinforcement based learning framework for human robot collaborative assembly processes, Front. Robotics AI 5 (2018) 126.

[37] C. Celemin, J. Ruiz-del Solar, An interactive framework for learning continuous actions policies based on corrective feedback, J. Intell. Robot. Syst. 95 (2019) 77–97.

[38] C. Celemin, J. Ruiz-del Solar, J. Kober, A fast hybrid reinforcement learning framework with human corrective feedback, Auton. Robots 43 (2019) 1173–1186.

[39] E. Chisari, T. Welschehold, J. Boedecker, W. Burgard, A. Valada, Correct me if i am wrong: Interactive learning for robotic manipulation, IEEE Robot. Autom. Lett. 7 (2) (2022) 3695–3702.

[40] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.

[41] S. Pateria, B. Subagdja, A.-h. Tan, C. Quek, Hierarchical reinforcement learning: A comprehensive survey, ACM Comput. Surv. 54 (5) (2021) 1–35.

[42] E. Ilhan, J. Gow, D. Perez, Student-initiated action advising via advice novelty, IEEE Trans. Games 14 (3) (2021) 522–532.

[43] A. Aguirre, A. Lozano-Rodero, L.M. Matey, M. Villamañe, B. Ferrero, A novel approach to diagnosing motor skills, IEEE Trans. Learn. Technol. 7 (4) (2014) 304–318.

[44] A. Aguirre, A. Lozano-Rodero, M. Villamañe, B. Ferrero, L.M. Matey, OLYMPUS: An intelligent interactive learning platform for procedural tasks., in: GRAPP/IVAPP, 2012, pp. 543–550.

[45] Í. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, N. Arana-Arexolaleiba, Goal-conditioned reinforcement learning within a human-robot disassembly environment, Appl. Sci. 12 (22) (2022) 11610.

[46] O. Spector, M. Zacksenhouse, Deep reinforcement learning for contact-rich skills using compliant movement primitives, 2020, arXiv preprint arXiv:2008.13223.

[47] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.

[48] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.

[50] A. Serrano-Munoz, D. Chrysostomou, S. Bøgh, N. Arana-Arexolaleiba, Skrl: Modular and flexible library for reinforcement learning, J. Mach. Learn. Res. 24 (254) (2023) 1–9.

[51] A. Serrano-Muñoz, Í. Elguea-Aguinaco, D. Chrysostomou, S. BØgh, N. Arana-Arexolaleiba, A scalable and unified multi-control framework for KUKA LBR iiwa collaborative robots, in: 2023 IEEE/SICE International Symposium on System Integration, SII, IEEE, 2023, pp. 1–5.