frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 14 February 2022
doi: 10.3389/frobt.2021.725780

How to be Helpful? Supportive Behaviors and Personalization for Human-Robot Collaboration

Olivier Mangin[1], *Alessandro Roncone*[2]* *and Brian Scassellati*[1]

[1]Social Robotics Lab, Computer Science Department, Yale University, New Haven, CT, United States, [2]Human Interaction and Robotics Group, Computer Science Department, University of Colorado Boulder, Boulder, CO, United States

The field of Human-Robot Collaboration (HRC) has seen a considerable amount of progress in recent years. Thanks in part to advances in control and perception algorithms, robots have started to work in increasingly unstructured environments, where they operate side by side with humans to achieve shared tasks. However, little progress has been made toward the development of systems that are truly effective in supporting the human, proactive in their collaboration, and that can autonomously take care of part of the task. In this work, we present a collaborative system capable of assisting a human worker despite limited manipulation capabilities, incomplete model of the task, and partial observability of the environment. Our framework leverages information from a high-level, hierarchical model that is shared between the human and robot and that enables transparent synchronization between the peers and mutual understanding of each other's plan. More precisely, we firstly derive a partially observable Markov model from the high-level task representation; we then use an online Monte-Carlo solver to compute a short-horizon robot-executable plan. The resulting policy is capable of interactive replanning on-the-fly, dynamic error recovery, and identification of hidden user preferences. We demonstrate that the system is capable of robustly providing support to the human in a realistic furniture construction task.

Keywords: human-robot collaboration (HRC), partially observable markov decision process (POMDP), partially observable Monte Carlo planning, hierarchical task network (HTN) planning, human-robot interaction (HRI)

OPEN ACCESS

Edited by:
Riccardo Monica,
University of Parma, Italy

Reviewed by:
Mohan Sridharan,
University of Birmingham,
United Kingdom
Mehrnoosh Askarpour,
McMaster University, Canada

***Correspondence:**
Alessandro Roncone
aroncone@colorado.edu

Specialty section:
This article was submitted to Robot and Machine Vision, a section of the journal Frontiers in Robotics and AI

Received: *15 June 2021*
Accepted: *14 December 2021*
Published: *14 February 2022*

Citation:
Mangin O, Roncone A and Scassellati B (2022) How to be Helpful? Supportive Behaviors and Personalization for Human-Robot Collaboration. Front. Robot. AI 8:725780. doi: 10.3389/frobt.2021.725780

1 INTRODUCTION

Recent trends in collaborative robotics are shifting focus on mixed human-robot environments where robots are flexibly adaptable to the rapid changes of the modern manufacturing process and can safely and effectively interoperate with humans. However, albeit considerable progress in robot perception, manipulation and control has improved the robustness and dependability of such platforms, robots are still used as mere recipients of human instructions (Brawer et al., 2018). That is, the human-robot collaboration (HRC) is still fundamentally unbalanced, with the bulk of the perceptual, cognitive and manipulation authority pertaining to the human. To fill this gap, recent works have begun to investigate truly collaborative framework that allow the human and the robot to focus on the part of the task for which they are best suited, and mutually assist each other when needed (Shah and Breazeal, 2010; Hayes and Scassellati, 2015; El Makrini et al., 2017; Roncone et al., 2017; Chang and Thomaz, 2021). Motivated by these results, in this work we focus our efforts on the design and implementation of robot platforms and human–robot interactions that exhibit a variety

of *supportive behaviors* in a mixed-initiative paradigm (Allen et al., 1999). Our goal is for the robot to be capable of assisting the human when they need support the most, provided some degree of knowledge of the task and necessary and sufficient information about the state of the system. Supportive behaviors such as handing over task components, providing tools, cleaning-up unused elements, holding a part during assembly happen to be extremely beneficial for the efficient completion of a task, and are within the realm of possibilities for modern robotic platforms. Such behaviors may also cover information retrieval tasks such as lighting an area of a working space, providing execution time, or detailing parts of the task plan to the user. For the purposes of this paper, we make two design decisions: 1) we focus on how to maximize the usefulness of existing robot platforms, i.e., how to maximally exploit the limited control, perceptive and reasoning skills the robot has in order to best support the human partner; 2) we posit that it is not necessary for the robot to have exhaustive knowledge of the task, nor to be able to completely and comprehensively perceive the environment. Rather, for a system to provide effective support, partial observability of the state of the world and the internal state of the human partner (composed of their beliefs and intents) is sufficient.

A key requirement for collaboration is that peers are able to share a common understanding of the task (Searle, 1990; Shah and Breazeal, 2010). However, an important differentiator between HRC and human-human teaming is that there currently exists a gap between the cognitive capabilities of humans and robots. This makes it both troublesome for the robot to reach the level of task understanding that humans have, and impractical for the human to encode the task model in a way the robot can utilize. To alleviate this issue, in this work we provide shared task modeling through hierarchical task models (HTMs). HTMs are convenient because they are widely used for high-level task planning (Erol et al., 1994), they are close to human intuition (Roncone et al., 2017), and they contain enough information for the robot to be efficient and effective in its support. Depending on the task, an HTM may encode pre- and post-conditions (Georgievski and Aiello, 2015), communicative actions (Brawer et al., 2018), and a variety of operators to combine nodes and subtasks (Hayes and Scassellati, 2016). Notably, these shared task models can provide a substrate for human robot communication and thus foster transparent interactions between peers during task execution (Hoffman and Breazeal, 2004; Brawer et al., 2018). Importantly, the robot is only made aware of the part of the task that matters to it, which simplifies both task planning and plan execution. We trade complete knowledge for adaptability, and optimal planning for good-enough support *by design*.

In conclusion, in this paper we present *a novel framework able to effectively empower a robot with supportive behaviors in a mixed-initiative paradigm*. We show, to our knowledge, the first practical implementation of an HRC application where the robot autonomously chooses to support the human when it deems it appropriate, and selects the right supportive action among the many it is provided with. Our contribution centers around a framework that: 1) maximizes the performance of the mixed human-robot system by leveraging the superior perceptual and manipulative skills of the human while entrusting the robot with the role it is best suited for, i.e., autonomous helper; 2) systematically leverages the human to improve task estimation, disambiguate unobservable states, and align mental models between peers; 3) dynamically adapts to (hidden) user preferences in terms of when and where to provide supportive behaviors, and modifies its policy *during execution* to comply with it. We validate the proposed approach in joint construction tasks that simulate the manufacturing process typical of Small and Medium Enterprises (SMEs), where features such as reconfigurability and ease of deployment are paramount.

In the following sections, we introduce the reader to the state of the art and related works in the field (**Section 2**). Then, we detail the proposed approach, focusing on how it differentiates from relevant research in the topic (**Section 3**). The experimental setup and the experiment design are presented in **Section 4**, followed by the Results (**Section 5**) and Conclusions (**Section 6**).

2 BACKGROUND AND RELATED WORK

This work capitalizes on past research in the field of high-level task reasoning and representation. As detailed in **Section 3**, the core contribution of this paper is a system able to convert human-understandable hierarchical task models into robot-executable planners capable of online interaction with the human. Contrarily to more traditional techniques that leverage full observability in the context of HRC applications [e.g., Kaelbling and Lozano-Pérez (2010); Toussaint et al. (2016)], our system deliberately optimizes its actions based on the interaction dynamics between the human and the robot. We explicitly account for uncertainty in the state of the world (e.g., task progression, availability of objects in the workspace) as well as in the state of the human partner (i.e., their beliefs, intents, and preferences). To this end, we employ a Partially Observable Markov Decision Process (POMDP) that plans optimal actions in the belief space.

To some extent, this approach builds on top of results in the field of task and motion planning [TAMP, see e.g., Kaelbling et al. (1998); Kaelbling and Lozano-Pérez (2013); Koval et al. (2016)]. Indeed, similarly to Kaelbling and Lozano-Pérez (2013) we find approximate solutions to large POMDP problems through planning in belief space combined with just-in-time re-planning. However, our work differs from the literature in a number of ways: 1) the hierarchical nature of the task is not explicitly dealt with in the POMDP model, but rather at a higher level of abstraction (that of the task representation, cf. **Section 3.1**), which reduces complexity at planning stage; 2) we encapsulate the complexity relative to physically interacting with the environment away from the POMDP model, which results in broader applicability and ease of deployment if compared with standard TAMP methods; 3) most notably, we *handle uncertainty in the human-robot interaction* rather than in the physical interaction between the robot and the environment. That is, our domain of application presents fundamental differences with that targeted by TAMP techniques; there is still no shared consensus in the literature on how to model uncertainty about human's beliefs and intents in general, and

the collaboration in particular. Our work contributes to filling this gap.

Planning techniques can enable human robot collaboration when a precise model of the task is known, and might adapt to hidden user preferences as demonstrated by Wilcox et al. (2012). Similarly, partially observable models can provide robustness to unpredicted events and account for unobservable states. Of particular note is the work by Gopalan and Tellex (2015) which, similarly to the approach presented in this paper, uses a POMDP to model a collaborative task. Indeed, POMDPs and similar models (e.g., MOMDPs) have been shown to improve robot assistance (Hoey et al., 2010) and team efficiency (Nikolaidis et al., 2015) in related works. Such models of the task are however computationally expensive and not transparent to the user. Hence, a significant body of work in the fields of human-robot collaboration and physical human-robot interaction focuses on how to best take over the human partner by learning parts of the task that are burdensome in terms of physical safety or cognitive load. Under this perspective, the majority of the research in the field has focused on frameworks for learning new skills from human demonstration [LfD, Billard et al. (2008)], efficiently learn or model task representations (Ilghami et al., 2005; Gombolay et al., 2013; Hayes and Scassellati, 2016; Toussaint et al., 2016), or interpreting the human partner's actions and social signals (Grizou et al., 2013).

No matter how efficient such models are at exhibiting the intended behavior, they are often limited to simple tasks and are not transparent to the human peer. Indeed, evidences from the study of human-human interactions have demonstrated the importance of sharing mental task models to improve the efficiency of the collaboration (Shah and Breazeal, 2010; Johnson et al., 2014). Similarly, studies on human-robot interactions show that an autonomous robot with a model of the task shared with a human peer can decrease the idle time for the human during the collaboration (Shah et al., 2011). Without enabling the robot to learn the task, other approaches have demonstrated the essential capability for collaborative robots to dynamically adapt their plans with respect to the task in order to accommodate for human's actions or unforeseen events (Hoffman and Breazeal, 2004). Likewise, rich tasks models can also enable the optimization of the decision with respect to extrinsic metrics such as risk on the human (Hoffman and Breazeal, 2007) or completion time (Roncone et al., 2017).

Our paper is positioned within this growing body of work related to task representations in HRC. We sit on a large body of literature on task and motion planning and POMDP planning with the goal of designing novel human–robot interactions. Unfortunately, little attention has been given to the issue of explicitly tackling the problem of effectively supporting the human partner, and only few works go in this direction. Hayes and Scassellati (2015) presents an algorithm to generate supportive behaviors during collaborative activity, although its results in simulation fall short in terms of providing practical demonstrations of the technique. Grigore et al. (2018) proposes a model to predict supportive behaviors from observed demonstration trajectories and hidden human preferences, but the results are not operationalized and evaluated within a full HRC system. On the other side of the spectrum, a number of works cited above achieve to a certain amount supportive behaviors without explicitly targeting them (Hoffman and Breazeal, 2007; Shah et al., 2011; Gopalan and Tellex, 2015; Toussaint et al., 2016). A limitation of these approaches is that, as mentioned previously, they rely on exact task knowledge that is not always available for complex tasks in practical applications. In this work, we incorporate *unknown* human preferences (i.e., they are not directly provided to the system and they need to be inferred *via* interaction) while *automatically* generating a complex POMDP from a *minimal data abstraction* (i.e., the HTM) and forbidding the robot to explicitly communicate with the human (which we did in our prior work, Roncone et al. (2017)). To the best of our knowledge, to date no work in human-robot collaboration has tackled the problem of high-level decision making in collaborative scenarios with this level of uncertainty of the human (in terms of human state, human beliefs, and human preferences).

3 MATERIALS AND METHODS

This work capitalizes on previous research by the authors. In Roncone et al. (2017), we demonstrated an automated technique able to dynamically generate robot policies from human-readable task models. We then exploited this framework in the context of role assignment: our system was effective in autonomously negotiating allocation of a specific subtask to either the human or the robot during a collaborative assembly. In this work, we expand this approach toward the more general problem of optimally providing support to the human. Similarly to our previous work (Roncone et al., 2017), we employ hierarchical representations of the task at a level of abstraction suitable to naïve human participants and understandable by the robot. The model of the task is provided a priori, although other studies have shown that it is possible to learn task models from human demonstrations (Garland and Lesh, 2003; Hayes and Scassellati, 2014). We then convert this task representation to a robot policy by leveraging the flexibility of POMDP models. This allows the robot to plan under uncertainty and explicitly reason at a high level of abstraction.

It is worth noting that exploiting adaptive planning from POMDPs has been already demonstrated in the context of human-robot collaboration [e.g., Gopalan and Tellex (2015); Nikolaidis et al. (2015)]. Planning under uncertainty is indeed a major requirement for robots to interact with people. Future robotic platforms are most likely to operate in highly unstructured environments, for which even state of the art perception systems are not going to provide full observability or exact estimations. In this work, we push this idea to its limits, inasmuch as we constrain our framework to the condition of being nearly *blind*. That is, the robot is not able to directly observe neither the state of the world (task progression, object locations, etc.), nor the state of the partner (intents, preferences, etc). *This allows to investigate mechanisms of coordination though communication and physical interaction in the environment.*

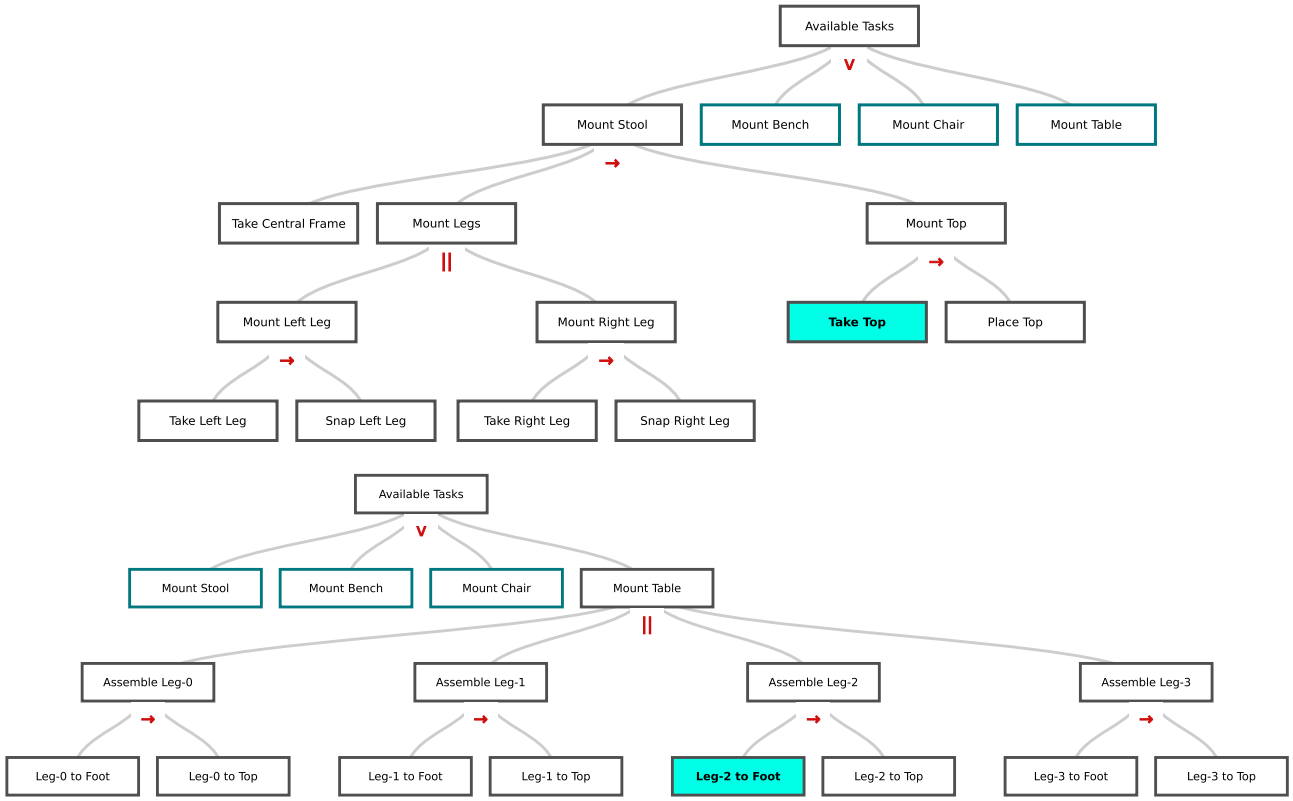

FIGURE 1 | Example HTMs for human-robot assembly tasks. The user can retrieve information on the task execution, and inquire the system about task progression that will be highlighted by the robot during execution (cyan block in picture). The operator type between subtasks (→for sequential tasks, ‖ for parallel tasks, ∨ for alternative tasks) is also available for the user to inspect. **(A)** HTM used in Roncone et al. (2017). **(B)** HTM used in this work. See **Section 5.2** for details.

We achieve that by expanding the technique introduced in Roncone et al. (2017), in that we leverage a high-level task model to *automatically generate* the lower-level POMDP. We then demonstrate how the resulting policy is successful in providing support during realistic human robot collaborations.

3.1 Hierarchical Task Models

Hierarchical structures form an appealing framework for high-level task representations; of particular interest is their capability to enable reuse of components over different tasks. Additionally, their level of abstraction is usually close to human intuition: this facilitates human-robot communication about task execution (Roncone et al., 2017).

Figure 1 depicts some example representations for real-world construction tasks. Similarly to Hayes and Scassellati (2016), we consider HTMs built from primitive actions with operators that combine them into what we refer to as *subtasks* of increasing abstraction. In this work, we assume that information about a set of primitive actions is already available to the robot, and we represent complex tasks on top of this action vocabulary. We assume also that the robot has basic knowledge pertaining these primitive actions. This can range from knowing the type of tools and parts needed to perform an action, to being aware of the fact that supporting the human through holding a part may be beneficial during complex executions. In our previous work (Roncone et al., 2017), we extended the CC-HTM representation introduced in Hayes and Scassellati (2016) with the introduction of a new *alternative* operator (∨ in **Figure 1**). It adjoins the *sequential* (→) and *parallel* (‖) operators. This set of operators proves suitable to capture the complexity of the collaboration, as well as the constraints of a task execution. For example, the parallel operator allows for the two peers to perform two disjoint subtasks as the same time; conversely, the sequential operator constrains them to a specific sequence of execution.

Thanks to their simplicity, HTM models can conveniently be drafted by non-expert workers and remain intuitive to understand. Differently from traditional TAMP approaches (Kaelbling and Lozano-Pérez, 2010, 2013), their high-level of abstraction also enables decoupling of the task planning component from the robot control element. This increases flexibility, in that the robot just needs to be equipped with some motor primitives to match the atomic actions that compose the HTM. From then on, the same library of motor primitives can be used to repurpose the robot to a new task. Furthermore, it becomes easy to port the same task to a new platform with comparable motor and perception skills.

One of the main limitations of HTM-like approaches to HRC is that it is unlikely for the designer to be able to encode the entirety of the information about the actual components of the

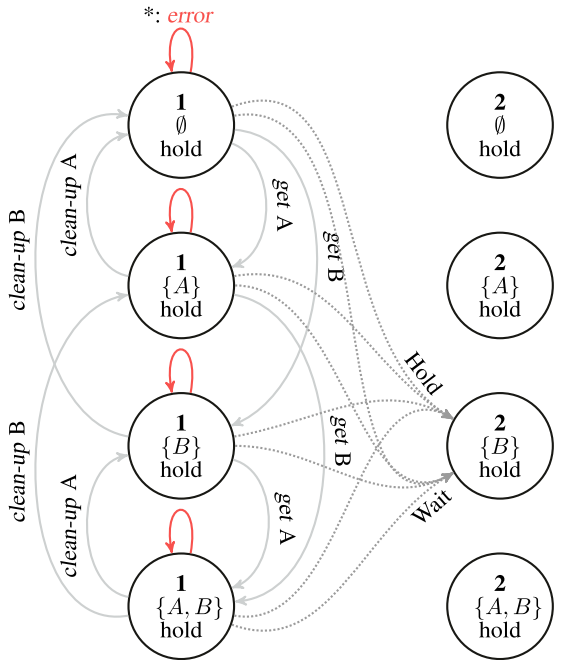

FIGURE 2 | Simplified representation of the restricted model (RM) used in this work. The figure represents an RM associated with a subtask **1** whose successor in the HTM is subtask **2**. This could be any of the terminating nodes in **Figure 1** that are connected *via* a *sequential* operator; please refer to **Section 3.3** to understand how to connect RMs with the *alternative* and *parallel* operators). For the sake of simplicity, the figure only represents actions that are taken starting from subtask **1**. We assume that only two objects "A" and "B" are available and that "A" is consumed by the subtask (like a part of the assembly would be) while "B" is a tool used during the task (e.g., a screwdriver). Each node represents a state, that is a factorization of the HTM subtask, each possible combination of objects on the workspace, and the user preference regarding the *hold* supportive action. Full connections in the graph represent successful transitions for the actions *get* and *clean-up* applied to objects "A" and "B" (that lead to a *none* observation). When taken from other states (e.g., bringing "A" which is already on the workspace), the action would fail, with an *error* observation, and the state would not change. These cases are represented by the red connections. Finally, the dotted connections represent the *hold* and *wait* actions that, from any of the represented states, lead to a transition to the state of the next subtask for which object "B" only is present (i.e., the tool). This means that the transition occurs even if the robot failed to bring all the required tools and parts: we assume here that the human would be able compensate for robot failures. The reward would however be maximal in the transition from state with {*A*, *B*}. To simplify the figure, we omitted the states corresponding to the *no-hold* preference. The graph for the *no-hold* preference is nearly identical except for the fact that the *hold* action fails from these states and hence *wait* is the only action to transition to the next subtask.

task. Although this poses limits to the breadth of applicability of such techniques, we argue that, for a robot to provide effective support, perfect knowledge about task execution is not needed in the first place. A partial HTM, and specifically one that conveniently encodes only the information that matters to the robot, is sufficient for it to operate and interact with the human. For example, the robot does not need to know how to perform a screwing action, nor how to perceive progression of the human screwing. What matters for a supportive robot is what objects are needed to complete said action, and that the human may be facilitated if the robot holds the part steadily. As discussed in **Section 3.2**, our POMDP model complements partial knowledge about the task and the state of the world through interaction with the human partner. It can for example supplant lack of perception about subtask progression by asking the human when the subtask is completed, or by directly moving on to the next subtask if its likelihood of subtask completion is high enough.

3.2 Partially Observable Markov Decision Processes

In this work, we use POMDPs to formulate the decision problem that the robot faces, given a task to solve collaboratively with a human and represented as an HTM as explained in **Section 3.1**. POMDPs are a generalization of Markov decision processes (MDPs), where there is only partial observability of the state of the process. This important relaxation of what defines an MDP allows for a significant gain in flexibility. It is particularly relevant to model-imperfect perception and hidden states such as user preferences. We use such an approach to optimize the robot actions despite incomplete knowledge of the task and uncertainty regarding the dynamics of the collaboration.

More precisely, a POMDP is defined by a 7-tuple (S, A, Ω, T, O, R, γ), where S is a set of states, A is a set of actions, T is a set of state transition probabilities, R: $S \times A \rightarrow$ I R is the reward or cost function, Ω is a set of observations, O is a distribution of observation probabilities, and $\gamma \in [0, 1]$ is the discount factor. Similarly to a MDP, at any given time the system lies in a specific state $s \in S$, which in the case of POMDPs is not directly observable. The agent's action $a \in A$ triggers a state transition to state $s' \in S$ with probability $T(s'|s, a)$ and an observation $o \in \Omega$ with probability $O(o|s', a)$ that depends on the new state s'. Finally, the agent gets a reward $r \in R$ for taking the action a while in state s. In case of POMDPs, the agent's policy is defined on a probability distribution over states b, called the belief state, which accounts for the fact that the agent has no direct access to the real state s. The goal is for the POMDP solver to find a policy $\pi(b)$: $b \rightarrow a$ that maximizes the future discounted rewards over a possibly infinite horizon: $E[\sum_{t=0}^{\infty} \gamma_t r_t]$. Interestingly, actions that do not change the underlying state of the system but only the belief state are also valuable in this context. This proves particularly beneficial for human collaboration, since information-gathering actions belong to this paradigm. For example, this intuition can be used to model communicative actions that trigger observations to disambiguate uncertainty, or to favor low-entropy beliefs with small uncertainty for both the human and the robot. In reality, the belief state is usually very large and continuous; we use a policy that is defined on the history of previous actions and observations that we denote by $h \in H$. Please refer to **Section 3.4** for more information on how we compute a robot policy from POMDP models.

3.3 Restricted Model

We propose an automated technique able to transform task-level HTMs into low-level robot policies through POMDPs. To this end,

we convert each primitive subtask (that is, each leaf composing the HTMs in **Figure 1** into a small, modular POMDP, which we call a *restricted model* (RM). Differently from Shani (2014), the RMs are composed according to the HTM structure and the nature of its operators, in particular: 1) a *sequential* operation will concatenate two RMs with a 100% transition probability; 2) an *alternative* operation between two subtasks will split the robot's belief space in two branches composed of two RMs with a 50% transition probability to each of them; 3) a *parallel* operator will bifurcate the belief space similarly to 2) but will add a 100% transition probability to the subtask not yet performed in each of the branches. In all, this approach allows to conveniently mitigate the computational complexity of planning a global policy in a high-dimensional state and action space (due to the combinatorial nature of HTMs) by focusing on short-horizon planning and the generation of a local policy for a single subtask/RM. That is, each RM is mostly independent from the rest of the problem and can be studied in isolation; however, RMs are still connected to each other by the structure of the HTM and the agent leverages this structure to maintain a belief that goes across subtasks and across RMs. We consider this capability as being core of our contribution, as it enables rich human-robot interactions that are transparent to both the robot policy and the human.

Figure 2 depicts the RM developed in this work, which incorporates information about the state space (affected only by perceptual noise), information about the user preferences (unknown to the robot), and information about the task (in the form of pre- and post-conditions, objects and tools needed, and so forth). As mentioned in **Section 3.1**, its action space corresponds to the set of motor primitives available to the robot. For the purposes of this work, we consider the following supportive actions: 1) *wait* for the human to complete a subtask; 2) *hold* an object to provide support to the human; 3) *bring object* (e.g., constituent parts, small parts, buckets, or tools) onto the workspace; 4) *clean-up object* from the workspace when not needed anymore. These motor primitives are implemented as independent controllers with their own logic: for example, the *wait* controller exploits communication in order to ask the human when the current operation has been completed before moving on to the new subtask. Modularity is employed in order to derive a distinct action for each object involved.

As mentioned earlier, the set of possible observations is minimal by design: a *none* observation (the default), plus a set of *error* observations returned by either the robot itself (e.g., *object-not-found*, *kinematic-error*) or initiated by the human partner (e.g., *wrong-action*). Forcing the system to deal with a limited set of observation is intentional. We additionally resorts to the intuition that, when the robot's execution is correct, the human partner should not be concerned about reinforcing this with positive feedback—which is time consuming and cognitively taxing. Rather, feedback from the human partner should come either if explicitly requested by the robot (to e.g., disambiguate uncertainty), or if the robot's decision making is wrong—and negative feedback should be used to correct its course of actions.

Lastly, the state space S is composed of a set of factored states. It is conceptually divided into three sub-spaces: 1) an *HTM-related state space* $\mathcal{Q}$ pertains to task progression, and is derived directly from the HTM representation. Each of the subtasks in the HTM (i.e., each of the leaves) is assigned an unique state $q \in \mathcal{Q}$; additionally, one final state $\hat{q}$ is associated with a virtual operation that requires the robot to cleanup the workspace. 2) a *subtask-related state space* is defined alongside the controllers that perform each subtask; it is composed of information relating parts and tools (e.g., if tools are present in the workspace, or if parts have been "consumed"). 3) a *human-related state space* encompasses information related to human preferences, beliefs and intents. As shown in **Section 4.3**, in this work we demonstrate how the proposed approach is able to adapt to user preferences even if it is not explicitly made aware of them.

TABLE 1 | Rewards used to train the policy on the POMDP model derived from the HTM.

Event	Reward
Final state reached	100
Subtask transition	10
Missing tool or part on state transition	−15
Uncleaned object on final state	−15
User preference is honored	10
Hold action taken	−2
Wait action taken	0
Other action taken	−1

Instead of a table of rewards for all state transitions, actions, and observations, we present the rewards as triggered by events that can be cumulated. For example, if a final state is reached through a wait action, but the screwdriver is still on the workspace, a reward of 85 is obtained.

It is worth noting how the size of the state space S grows exponentially with the number of preferences and objects, as detailed in **Section 3.4**. In order to account for scalability of the method, we define a *generative POMDP model* that circumvents the issue of explicitly defining the full transition matrix T. Instead, as detailed in **Figure 1**, we generate it as follows. Each action affecting an object changes the state representing its presence in the workspace: for example, bringing a screwdriver makes it available on the workspace with high probability. The *wait* action and eventually the *hold* action trigger the transition from one HTM leaf to the next (according to their order in the sequence, the final operations leading to a transition to the special state $\hat{q}$). This mechanism enforces that transitions between HTM states are transparently synchronized with the human. *Hold* only triggers the transition from states in which the human has preference for holding and fails otherwise. Additionally, the transition from one leaf to the next erases from the state representation all the objects that have been "consumed" by the subtask (typically, the parts that have been used). The initial state is sampled by starting at the initial subtask $q_0 \in \mathcal{Q}$; the workspace is assumed to be free of objects, and the human preferences are randomly set.

Interestingly, the design choice of limiting the perception of the state of the world naturally conforms to the statistical nature of a POMDP approach. That is, adding uncertainty in the model makes it ultimately more robust to actual uncertainty in the collaborative interaction. Without loss of generality, the RM can allow for unexpected transitions in order to account for actions of the human that are not observed by the robot—e.g., when the human partner fetches a required component by themselves,

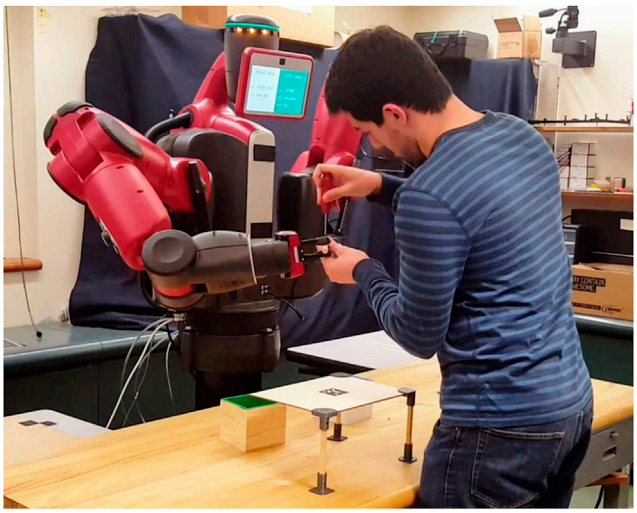

FIGURE 3 | The experimental setup, in which a human participant engages in a joint construction task with the Baxter Robot. In the picture, the robot is supporting its human partner by holding a leg of the table while the human is screwing. See **Section 4.2** for information about the task.

unexpected failures, or missing objects. To model this, and to avoid the robot to be stalled in a wrong belief, we introduce low probability random transitions between all the state features in S.

Finally, rewards are provided in the following cases: 1) when the robot proposes to hold, and the human has preference for holding; 2) when there is a transition between a subtasks and its successor; 2) at completion of the full task. Additionally, each action taken by the robot has an intrinsic cost, and a negative reward is also given when the human has to bring or clean an object which was not taken care of by the robot. **Table 1** provides a summary of the rewards used in the experiment from **Section 5.2**.

3.4 POMDP Planner

In this work, we implement a planner based on POMCP (Silver and Veness, 2010), which is able to plan from generative models and can handle very large state spaces. This is achieved through Monte-Carlo estimation of the beliefs by using a set of particles to represent each belief. A policy is learned based on all the visited histories, which constrains exploration to feasible states only.

Namely, n particles can approximate a belief over states as follows: each particle i is in a given state s_i, and they collectively represent the belief:

$$b = \frac{1}{n}\sum_{i=1} n\delta_{s_i}, \tag{1}$$

Where δ_s is the indicator function on the state s. A pseudo-code algorithm detailing how the belief is updated in practice is shown in Algorithm 1; a similar process is applied for updating the belief when the robot gets an observation from the environment. In this case, new states that led to an observation different than the one the robot obtained are then discarded. Using particles for belief representation and Monte-Carlo techniques for value estimation addresses the issue of the belief space being too large to be explicitly represented in its entirety. In a realistic HRC domain—such as those detailed in **Section 5**—there are typically thousands of states and tens of actions, but the amount of plausible states at any time is limited. Hence, representing the $|\mathcal{S}|$-dimensional belief is not feasible, but at the same time, despite large state spaces, beliefs are sparse. This is well represented through sets of particles, that naturally conform to sparse representations. Further, this approach only requires a generative model of the transitions instead of representing the full transition matrix, whose dimension is $|\mathcal{S}|^2 \times |\mathcal{A}| \times |\mathcal{O}|$.

Algorithm 1. Algorithm to simulate potential future hypotheses about the consequence of a robot's action *via* Monte Carlo sampling. Given a belief on the current state, our method repeatedly samples from the set of particles in the belief and applies the logic illustrated in **Figure 1** to the state that the particle holds [match_action(a)]. This process yields a new particle that is added to the updated belief on the following state. Once the new belief contains the desired number of particles, the process stops.

Function `Step` (*old_belief b, action a*) **:**
  while *size(new_belief)* $\leq n$ **do**
    $s \leftarrow$sample_one_particle(b);
    if *random()* $< \varepsilon$ **then**
      $s' \leftarrow$random_state();
    else
      $s' \leftarrow$match_action(a);
    end
    add_particle(s', new_belief);
  end
return new_belief;

The computational complexity of the planning is bound to an exploration-exploitation trade-off. Namely, the planner explores a tree whose branching factor is equal to the number of actions multiplied by the number of observations. Therefore, an important parameter to control the complexity of the problem is the horizon of the exploration. More precisely, we define the horizon either in terms of the number of transitions or as the number of *HTM subtasks* that the exploration accounts for. Such optimizations result in locally optimal decisions for what concerns a fixed number of subtasks. In order to limit computational complexity, it is also possible to remove part of the stochasticity introduced in **Section 3.3** for what concerns the transitions and the observations—although it still occurs in the belief transitions that hence still represent all the possible hypothesis. By doing so, we prevent the planner from exploring feasible but rare events. In case these rare events occur during live interaction with the human, the online component of the planner is able to re-compute a new policy. This makes the algorithm robust to unexpected events without penalizing the exploration. That is, we artificially simplify the model of the interaction at the offline planning step, but we then compensate for its imperfections online—i.e., during task execution.

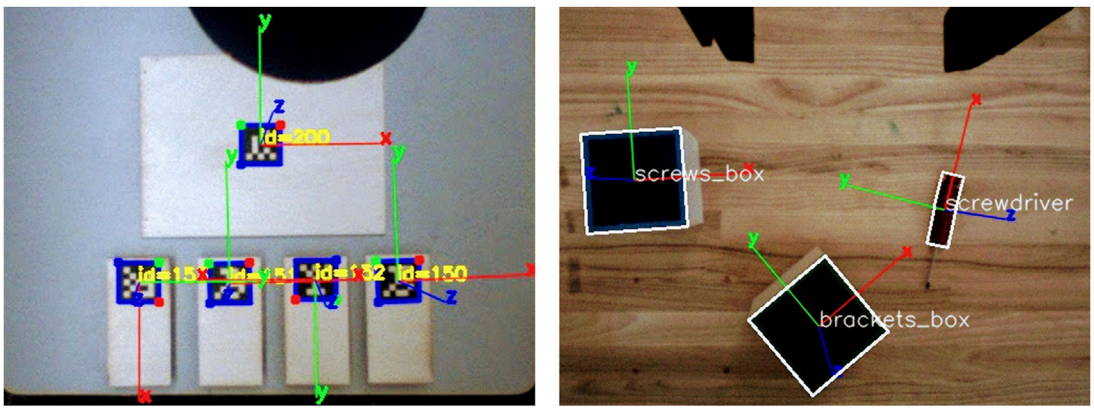

FIGURE 4 | Snapshot of the camera streams from the left and right end effectors **(A,B)**. The left arm uses a fiducial marker tracking system based on Garrido-Jurado et al. (2014), whereas the right arm implements the HSV-based 3D reconstruction software detailed in **Section 4.1.1**. See **Section 4.2** below for a description of the objects composing the construction task.

4 IMPLEMENTATION

4.1 Experimental Setup

The experimental evaluation is carried out on a Baxter Research Robot (cf. **Figure 3**), using the Robot Operating System (ROS Quigley et al., 2009). As mentioned in **Section 3**, even though we do not concern with improving the physical capabilities of the platform, we leverage the state of the art in robot perception and control in order to build up a set of basic capabilities for the robot to effectively support its human partner. The resulting framework, originally developed in Roncone et al. (2017), exposes a library of high level actions, that are the only interface through which the POMDP planner can send information to—and retrieve information from—the Baxter system and the experimental setup.

The system presented in Roncone et al. (2017) provides multiple, redundant communication channels to interact with the human partner *"on human terms"*Breazeal, 2002). Among the list of available layers, for the purposes of this work we employ: 1) a *Text-to-Speech (TTS)* channel, used to verbally interact with the human; 2) a *Feedback* channel, shown in the robot's head display, which provides feedback about its internal states and intents (see **Figure 3**); 3) an *Error* channel that allows the human to send error messages to the robot, and is triggered by pressing one of the buttons on the robot's end effectors. In addition to this, a fourth channel has been implemented, in the form of a *Speech-to-Text (STT)* system able to convert human sentences into robot-readable commands. It employs the Google Cloud STT API Goo (2021) combined with a text parser that relies on a dictionary shared in advance with the human participant.

Both arms are able to perform precise, closed loop visual servoing tasks thanks to a pair of cameras their end effectors are equipped with. The left arm is equipped with a vacuum gripper, able to pick up flat surfaces with constant texture, whereas the right end-effector is a parallel electric gripper capable of performing more complex grasping tasks. We maximize the usage of both arms by leveraging their respective embodiments: to this end, two different perceptual systems have been employed (cf. **Figure 4**). The perception system for the left arm (**Figure 4A**) is provided by ARuco (Garrido-Jurado et al., 2014), a library capable of generating and detecting fiducial markers that are particularly suitable for being positioned on flat surfaces. For what concerns the right arm (**Figure 4B**), a custom color-based pose estimation algorithm has been implemented. It is detailed in the following section.

4.1.1 6D Object Reconstruction From Single View

We consider here the scenario in which the end-effector is vertically placed on top of the pool of objects, and the objects are in the field of view of the camera. In order to be able to grasp a variety of objects with the parallel gripper installed on the Baxter's right arm (see **Figure 4B**), the following two steps are to be performed: objects need to be firstly detected in the camera view, and then their position and orientation has to be reconstructed in the 3D operational space of the robot. For what concerns the former, a number of different computer vision techniques can be employed. In this work, we utilize a Hue-Saturation-Value (HSV) color segmentation algorithm: that is, each object is detected thanks to its color in the HSV color space, and its bounding box is stored for later use. After an object has been detected, it is necessary to estimate its 3D pose in the world reference frame. We assume here that the object's physical sizes (width and height) are known, and that the matrices of intrinsic and extrinsic parameters K and $[R|T]$ are available. Notably, whilst K can be estimated *via* a prior camera calibration step, the extrinsic parameters are computed thanks to the robot's kinematics and the knowledge of the current joint configuration. In this context, a standard perspective transformation can be applied in order to estimate the 3D position of a point $P_w = [X\ Y\ Z\ 1]^T$ in the world reference frame from its corresponding image point $p_c = [u\ v\ 1]^T$ in the camera reference frame. The following equation holds:

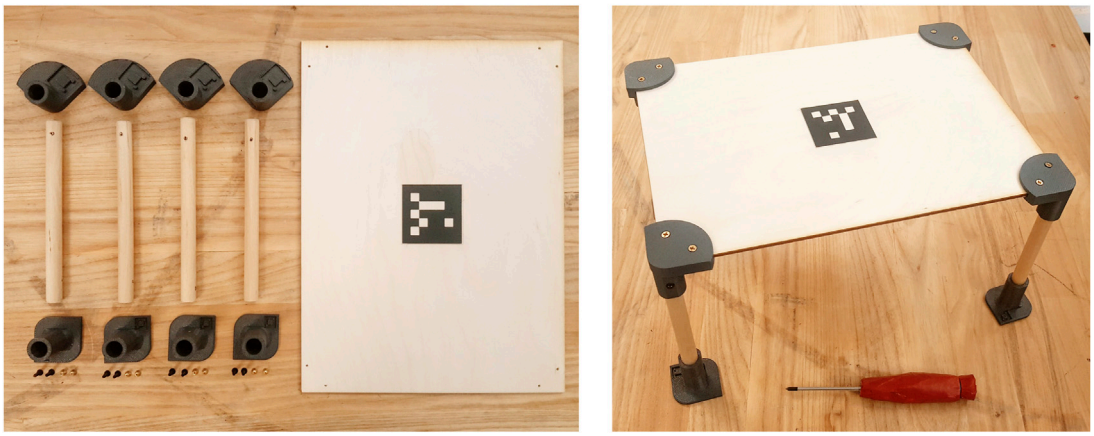

FIGURE 5 | (A) The table building task is composed of one plywood tabletop, four dowels that act as legs, four brackets (top 3D printed objects in figure) and four feet (bottom 3D printed objects in figure). A total of 16 screws are needed to secure parts together. Both the tabletop and the legs have been pre-drilled to facilitate assembly. **(B)** The table after the construction task is completed. The only tool required for the assembly is a screwdriver (red object in bottom of figure).

$$s\ p_c = K[R|T]\ P_w, \tag{2}$$

Where s is a scale factor[1]. The perspective transformation equation is then applied to estimate the pose of the mass center of the object by iteratively minimizing the reprojection error of its corners *via* a Levenberg-Marquardt algorithm (Marquardt, 1963), using the OpenCV computer vision library (Bradski, 2000).

The technique proposed here is subject to a number of estimation and computational errors, in particular if the distance between the camera and the desired object is significant. In spite of that, we capitalize on the fact that the algorithm is employed in a visual servoing setup, in which the robot refines its estimation the closer it gets to the object. That is, even if the initial pose reconstruction may be defective, it is continuously updated with a frequency of 30 Hz and refined until the end-effector reaches the object. The authors acknowledge that more advanced 3D reconstruction techniques could be used, such as exploiting a depth sensing camera properly calibrated with respect to the robot—e.g., Jiang et al. (2011). The main advantage of the proposed solution is however to employ a compact, self-contained estimation step that does not rely on external equipment or burdensome calibration. We consider this an important asset of our approach, that facilitates re-use and applicability to novel domains.

4.2 Experiment Design

As detailed in **Sections 1** and **4.1**, we perform our experiments in a collaborative scenario where human participants engage in a construction task with the Baxter Robot (see **Figure 3**). The collaborative task the two peers are engaged with is the joint construction of a miniaturized table (cf. **Figure 5**). The table is part of an open-source effort to standardize experiments in HRC and improve reproducibility (Zeylikman et al., 2018). It is composed of five structural elements (the tabletop and four legs) and eight custom 3D-printed linkages (four brackets secure the legs to the tabletop, whereas four feet are used to stabilize the structure). A total of 16 screws are required for the assembly. A screwdriver is the only tool needed to build the table, which has an approximate size of 30 cm × 21 cm × 15 cm when completed. Particular attention has been placed in the conceptualization of the task. The main goal is to tailor the design of the table to the typical constraints human-robot collaboration experiments present. We purposely aim for: 1) ease of reuse; 2) ease of retrieval of the constituent parts; 3) scalability; 4) proximity with real-world HRC applications, that are typically characterized by a combination of complex actions to be performed by the human partner (which often involve the use of tools), and simpler tasks the robot is usually assigned to. The choices taken at the design stage allow us to comply with these requirements: the brackets used in this experiment belong to a larger library of linkages that has been made available online[2], whereas the table shown in **Figure 5** is only one of the many designs allowed by our solution.

For the purposes of this work, the two partners have distinct, non overlapping roles: the human (hereinafter referred to as the *builder*) is in charge of performing actions that require fine manipulation skills (e.g., screwing) or complex perception capabilities (such as inserting the top of the table onto a bracket); the robot (also called the *helper*) is instructed to back the builder with the supportive actions described below. Importantly, the flexibility of the POMDP planner allows for a certain slack in terms of role assignment and task allocation by design. As detailed in **Section 3**, the planner is automatically able to comply with unlikely states of the system, and to re-plan

[1]Please refer to Hartley and Zisserman (2004) for more information on the perspective transformation problem in detail.

[2]`scazlab.github.io/HRC-model-set` hosts CAD models, specifications for 3D printing, tutorials for assembling example designs, and reference links for purchasing parts.

accordingly. As a practical consequence to this, we are in the position of allowing the human participant to take charge of some supportive actions if he so chooses. That is, we disclose to the builder that they are allowed to retrieve parts and tools by themselves, even though we do not enforce it on them—nor we convey that it is part of her duties as a participant of the experiment. The robot helper is provided with a set of basic capabilities encapsulated into a library of high-level actions. The supportive actions it has been instructed to perform are the following: 1) retrieve parts (e.g., tabletop, screws, or legs); 2) retrieve the tool (namely, the screwdriver); 3) cleanup the workplace from the objects that are not going to be needed in the future; 4) hold structural parts in order to facilitate the builder's actions. To comply with the Baxter's limited manipulation capabilities, we positioned the smaller components (i.e., the screws and the 3D printed objects) in apposite boxes, to be picked up by the parallel gripper—see *screws_box* and *brackets_box* in **Figure 4B**. Similarly, the legs of the table have been equipped with a specific support in order to be picked up by the vacuum gripper on the left arm (cf. **Figure 4A**).

It is worth noting how this specific task is particularly advantageous for the purposes of this work thanks to 1) its simplicity and 2) the need for the human participant to perform the same actions multiple times. We deliberately designed a task that does not require any particular skill from the builder, while being easy to understand and remember. Although it may be tedious for the user, the need of performing multiple actions of the same type is beneficial in terms of showcasing the online user adaptation capabilities introduced in **Section 1**. As detailed in **Section 5** below, one of the assets of the proposed system is to be able to abide by the builder's preferences: in such a scenario, the robot is able to receive eventual negative feedback in case of wrong action, replan accordingly, and exhibit the effects of such replanning *within the same task execution*, i.e., without having to perform a new task from scratch.

4.3 Experimental Evaluation

We demonstrate the proposed system in a live interaction with human participants. The robot is in charge of backing the user with the right supportive action at the right moment by virtue of a partial observation of the state of the world, the complete knowledge of the task execution plan, and the HTM-to-POMDP planner presented in **Section 3**. We devised two distinct experimental conditions, in a within-subjects design. For all the conditions, the skill set and capabilities of the robot do not vary, but the user preferences are explicitly altered, unbeknownst to the robot. That is, the robot is exposed to a change in the state of the system—composed of the world plus the human—that it could not observe, but needs to deduce either by actively gathering information from the builder, or by building upon feedback coming from her. Our experiments are designed to evaluate the following hypotheses:

- H1. The proposed system is capable of operating under uncertainty and maximizing returns even in the presence of high combinatorial complexity;
- H2. The robot is able to personalize its behavior to varying human preferences and replan for mistakes on the fly, thus maximizing the capability of supporting the human partner.

In the following sections, we detail the two experimental conditions. Please refer to **Section 5** for a comparative evaluation.

4.3.1 Condition A—Full Support

In this scenario, the robot expects to support the human to the best of its capabilities, that is by performing all the actions it is allowed to. Firstly, the builder is introduced to the platform and the construction task. The experimenter then proceeds to illustrate the Baxter's capabilities (i.e., providing parts, retrieving tools, holding objects and cleaning up the workspace) and the interaction channels the human is supposed to employ during task execution. Next, the experimenter communicates to the user that the robot is supposed to perform all the supportive actions by itself, but also that the participant is free to take charge of some actions if she so chooses or if the robot fails. No information is given in terms of what to expect from the robot, or how the human-robot interaction is supposed to occur.

4.3.2 Condition B—Adaptation to User Preferences (No Holding Actions Required)

This condition involves the same interaction between the human and the robot as Condition A. As detailed in **Section 4.3**, the independent variable we tweak in our within-subjects experiments is the user preference for what concerns the support that the human participants expects from the robot. In this scenario, the human worker is told to prefer not to have parts held by the robot while screwing. Since the robot is unaware of this, it may still perform the holding action even if not required. In case this happens, the human is instructed to negatively reward the robot by sending an error signal to the Baxter.

5 RESULTS

The framework detailed in **Sections 3** and **4.1** has been released under the open-source LGPLv2.1 license, and is freely available on GitHub[3]. A number of C++ based ROS packages has been made available for robot-related software, whereas the planner has been encapsulated into a ROS-independent Python package. In the following sections, we evaluate the proposed approach. Firstly, we perform a series of off-line experiments to assess if the proposed model can derive effective policies against a variety of tasks and experimental conditions. We show how our method outperforms ad-hoc policies on simulated interactions from these models (**Section 5.1**). Lastly, we validate our system in a live

[3]`github.com/scazlab/human_robot_collaboration` hosts the source code for the robot controllers, whereas `github.com/ scazlab/ task-models` hosts the HTM to POMDP planner.

interaction with human participants. We demonstrate how effective task policies can be computed that enable supportive behaviors during collaborative assembly tasks (**Section 5.2**).

5.1 Synthetic Evaluation in Simulation

In this section, we present a quantitative evaluation of the proposed approach during off-line explorations. To this end, we focus on the two most important aspects that are involved in the design of effective human-robot interactions: 1) the flexibility of our method against a variety of task structures (**Section 5.1.1**) 2) its adaptability to custom user preferences (**Section 5.1.2**).

5.1.1 Task Structures

In order to demonstrate how the proposed method is able to provide effective support to a human partner in real-world collaborative tasks, we evaluate it on three task models, derived from distinct HTMs. The HTMs differ in the number of subtasks to solve, and in the type of relational operators between subtasks. They therefore illustrate how we can derive policies from various task models. All task models in this section are characterized by primitive subtasks that require a combination of: 1) a set of tools that the robot needs to initially bring and then clean at the end of the task; 2) a shared supportive action '*a*'; 3) another supportive action, that can be either '*b*' or '*c*'. Let denote 'B' the subtask that involves supportive actions '*a*' and '*b*' and 'C' the one involving '*a*' and '*c*'. The first HTM, denoted as *sequential* task, consists of a sequence of 20 subtasks 'B'. The second task model is denoted as *uniform*; it consists of an alternative between 24 subtasks, each composed of a sequence of four subtasks, each of type 'B' or 'C'. In other words, for each episode the current task is randomly chosen among any sequence of three subtask of type 'B' or 'C'. The last HTM, denoted as *alternative* task, is an alternative between only 4 sequences of four subtasks: 'BCCC', 'BBBB', 'CBBC', and 'CCBC'. It thus introduces a dependency between the successive required actions.

We compare the performance of the proposed approach against two hand-coded policies. A *random* policy initially brings all the required tools (which is always a successful strategy); it then takes action '*a*', and after that it randomly chooses between action '*b*' and '*c*' until one succeeds. It finishes by cleaning the workspace. When observing a failure (except on '*b*' and '*c*'), the policy simply repeats the last action. The *repeat* policy is instead designed for the *sequential* task. Similarly to the *random* policy, it starts by providing the required tools, and then it repeats actions '*a*' and '*b*' 20 times and then cleans the workspace. Similarly to *random*, it repeats failed actions until success. **Figure 6** presents the average return of each policy on the three conditions. Although *repeat* is very efficient on the *sequence* task, it is unfitted to the others and fails on the two other tasks. The *random* policy is suboptimal on all but the *uniform* task but can still solve them with a few failures. On the other side, the *POMCP* policy that is learned from each task model matches or outperforms the other policies, providing full support to **H1**. This experiment hence demonstrates that we can leverage knowledge about the task structure to automatically derive efficient policies for a wide variety of tasks characterized by a wide range of combinations.

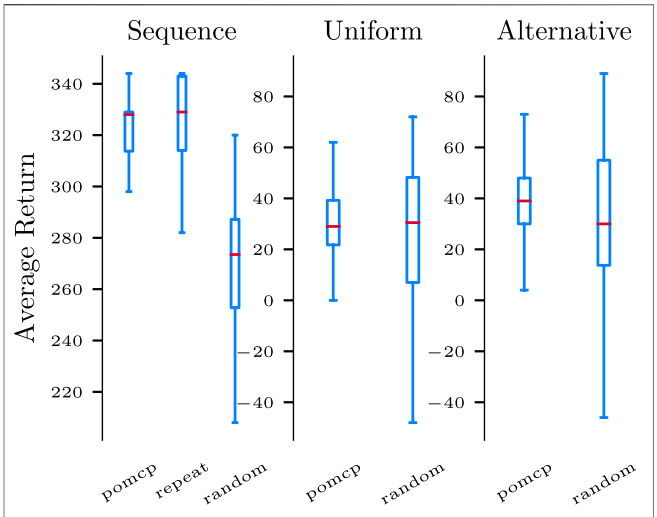

FIGURE 6 | The *POMCP* policy matches or outperforms ad-hoc strategies against three different HTMs (*sequential*, *uniform*, and *alternative* task). The results are the distribution of returns over 100 evaluation episodes from the two hand-coded policies (*repeat* and *random*) as well as the *POMCP* policy derived from the POMDP. On the last two tasks, the *repeat* policy fails in most of the cases and is stopped by an upper bound on the horizon of the episode. It gets a very low average return (between −300 and −400) that has been omitted from the figure to better compare the other policies.

5.1.2 User Preferences

Being able to comply with—and dynamically adapt to—custom user preferences is crucial for a robot that needs to provide the best support to its partner. A prompt and personalized response allows for a more natural interaction and a less cognitively demanding execution, which ultimately result in a more efficient collaboration. To this end, we present a system that is successfully able to account for user preferences. As detailed in **Section 4.3**, the participant is allowed to choose if the robot should provide support by holding the structure while the human is screwing. We compare our approach against two hard-coded policies: the most proactive strategy (i.e., always offering to hold for support) and the most conservative one (i.e., never proposing to support the human). **Figure 7** collects results from 6,000 simulated interactions, in which the human has a preference for "holding" p_H that ranges from 0.0 ("I never want the robot to hold') to 1.0 ("I always want the robot to hold when needed"). The two hard-coded policies are only optimal when they match with the expected user preference; in the intermediate scenarios, their performance degrades quickly as the uncertainty on the actual user preference increases. Conversely, our system is able to adapt to whether the human partner would like the robot to provide hold support or not, and outperforms both strategies in the majority of conditions with high average returns. Importantly, the *POMCP* policy is capable of adapting on the fly in intermediate cases: when the probability of holding preference p_H is between the extreme cases (i.e., $0.0 < p_H < 1.0$), the user is free to "change their mind" and signal this to the robot through an error message. These results validate H2 in

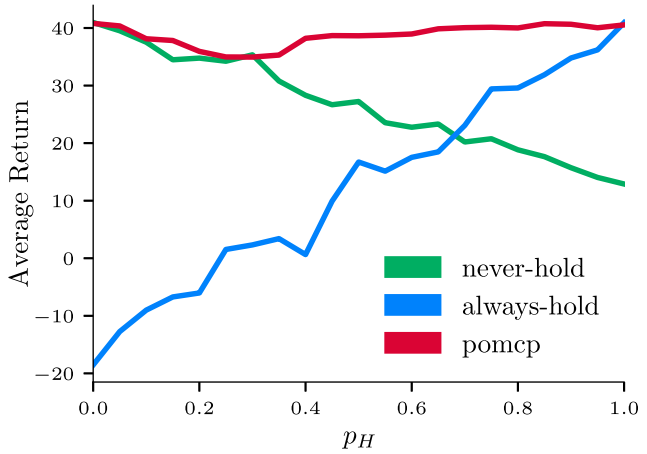

FIGURE 7 | Comparison of the proposed strategy (red) against an "Always Hold" (green) and a "Never Hold" (blue) strategy during simulated interactions with varying degree of user preferences. For each of them, average return values and standard deviations with respect to the probability of the "Hold" preference p_H are shown. The three strategies are tested against a single "Assemble Leg" subtask (see **Figure 1**, bottom) with 20 different values of p_H, ranging from 0.0 (i.e., the human never wants the robot to hold) to 1.0 (i.e., the human always like the robot to hold). For each of the 20 different preferences, the results are averaged from 100 simulated interactions, for a total of 6,000.

simulation; we refer the reader to **Section 5.2** for comparable results in the real world.

5.2 Live Interaction With Human Participants

The synthetic experiments described in **Section 5.1** demonstrate the ability of the algorithm to yield good rewards in a range of scenarios including variable task complexities (H1) and user preferences (H2). These experiments, however, do not validate our algorithm in a real human-robot interaction, which is what the majority of prior work stops at. Although the simulation demonstrates that the algorithm successfully reaches high reward, they cannot guarantee that these rewards accurately approximate—or even correlate with—the actual utility of having a robot supporting a human during a live interaction. In this section, we address this issue through an experimental scenario where a human participant is engaged in the joint construction of furniture with the Baxter robot.

We define the construction task detailed in **Section 4.2** as a sequence of eight subtasks, whose HTM is shown in **Figure 1**, bottom. All the subtasks depicted in figure require high dexterity and perception skills, and thus need to be performed by the human builder exclusively. For each of the four legs, the builder is in charge of firstly screwing the linkages (bracket and foot) onto the leg, and subsequently screwing the bracket onto the tabletop. As introduced in **Section 4.2**, retrieving parts and tools from their respective pool has been designed as one of the supportive actions the helper robot can choose from. Further, the Baxter robot is allowed to hold parts in order to facilitate the participant's work, and to clean up the workstation when it deems it appropriate. For more information about the actual interaction, we refer the reader to the accompanying video, which has been summarized in **Figure 8** (full resolution available at youtu.be/OEH-DvNS0e4).

The system has been evaluated with four participants; each participant performed the task in both Condition A and B, in a within-subject design (see **Section 4.3** for a description of the experimental conditions). In all the demonstrations performed, the robot was successful in providing support to the human with minimal overhead on them: as shown in **Table 2**, the user feedback is minimal and only necessary in case of robot mistakes. In all, we register a reduction of task

TABLE 2 | Evaluation of the human-robot collaboration.

	Condition A—*hold*	Condition B—*no-hold*
Average completion time	649 [s]	747 [s]
Average number of robot actions	21	14

For both conditions, average completion time and average number of robot actions are shown. We register an overall improvement in task completion time when the robot is allowed to better support the human through "hold" actions (which results in an increased number of robot actions per tasks).

TABLE 3 | Example histories of actions and observations during the interaction, for the two conditions.

	Condition A—*hold*			Condition B—*no-hold*			
	actions	**obs**	$\widehat{p_H}$	**actions**	**obs**	$\widehat{p_H}$	
1	bring screws	none	0.34	bring screws	none	0.43	1
2	bring leg	none	0.38	bring leg	none	0.46	2
3	bring screwdriver	fail	0.36	bring screwdriver	none	0.44	3
4	bring top	none	0.37	bring joints	fail	0.38	4
5	bring joints	none	0.39	bring top	none	0.38	5
6	bring screwdriver	fail	0.43	bring joints	none	0.35	6
7	bring screwdriver	none	0.42	hold	fail	0.03	7
8	hold	none	0.41	bring leg	none	0.03	8
9	hold	none	0.86	wait	none	0.02	9
10	bring leg	none	0.97	wait	none	0.01	10
11	hold	none	0.97	wait	none	0.01	11
12	hold	none	0.99	bring leg	none	0.01	12
13	bring leg	none	1.0	wait	none	0.01	13
14	hold	none	1.0	wait	none	0.02	14
15	hold	none	1.0	bring leg	none	0.03	15
16	bring leg	none	1.0	wait	none	0.03	16
17	hold	none	1.0	wait	none	0.04	17
18	hold	none	1.0	clear joints	none	0.04	18
19	clear screws	none	1.0	clear screws	none	0.04	19
20	clear joints	none	1.0	clear screwdriver	none	0.01	20
21	clear screwdriver	none	1.0	wait	none	0.01	21
22	wait	none	1.0				22

$\widehat{p_H}$ is the estimation of the probability for the "hold" preference in the robot's internal belief. In Condition A, the human prefers the robot to hold the structure: thanks to a "none" observation when "hold" is offered (step 8, left), the internal estimation of human preference progressively grows to 1.0 over time. Conversely, in Condition B the robot offers to "hold" and receives an error message (step 7, right). This observation progressively lowers p_H to zero, and the robot did not perform this supportive behavior in the future.

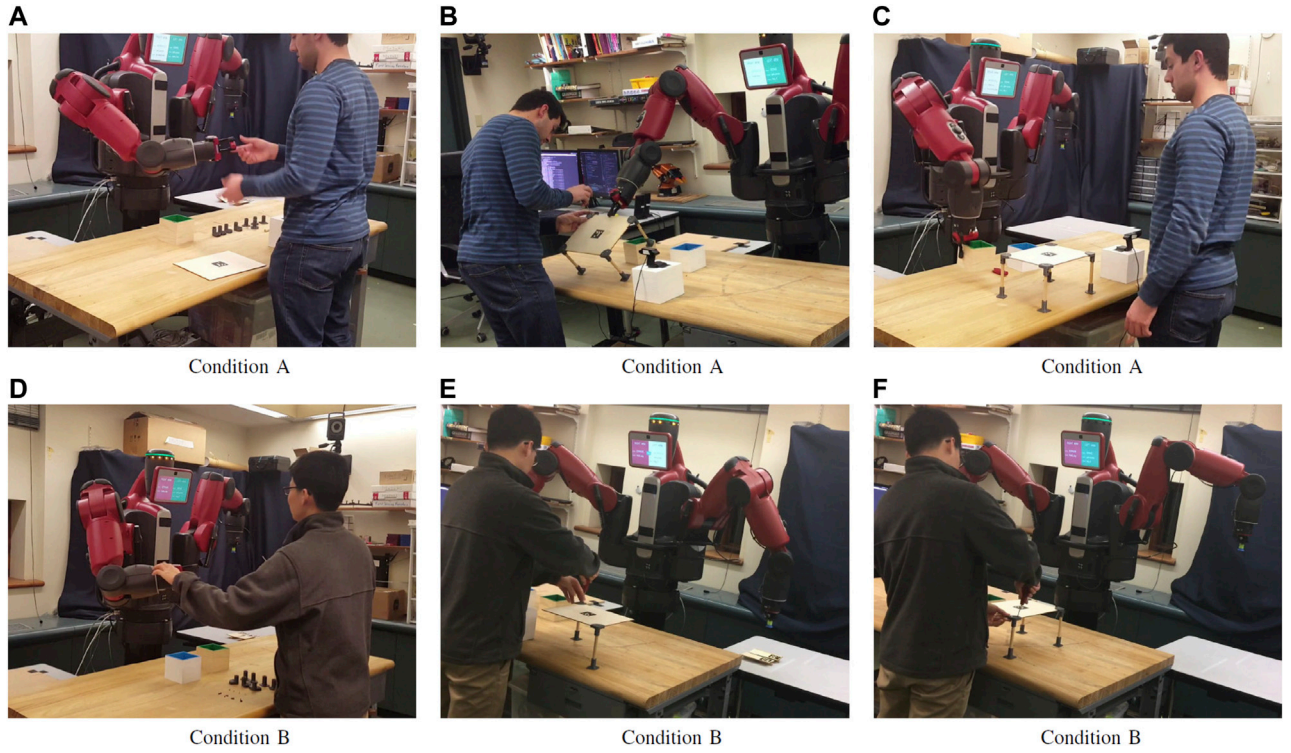

FIGURE 8 | Snapshots acquired during the collaborative assembly of the table in Condition **A** (full support, top) and Condition **B** (no holding required, bottom). Condition **A (A–C)**: 1) Baxter provides the tool to the participant; 2) the robot supports the human by holding the tabletop while the human screws the leg in place; 3) the user has finished his task and observes the robot freeing the workspace from the box of linkages. Condition **B (D–F)**: 1) The user signals to the robot that the holding action is not required by pressing the error button, and the robot acknowledges back by signaling into its display that it received this information; 2) The hold action is not performed any more, but other actions such as the retrieval of the leg are still performed; 3) The human participant completes the execution of the task without the help of the robot, as requested.

completion time (13.2% on average) in Condition A. That is, when the robot is allowed to provide more support to the human and is thus intervening more in the task, the overall team performance improves despite the fact that the robot is operating under high uncertainty (thus further validating H1).

More interesting is to evaluate the extent to which the planner is able to recover from robot failures and policy errors in the presence of unobservable human preferences (H2). To this end, we highlight two example trajectories in **Table 3**. They correspond to actual trajectories, one from Condition A, where the robot is expected to hold the parts, and one from Condition B, where the user will signal her preference about not wanting parts to be held. The interaction channels described in **Section 4.1** allow for a certain degree of flexibility for what concerns the type of communication the builder and the helper can engage into. In particular, the Baxter is allowed to gather information about the user preference by engaging in a hold action in order to better estimate this variable through the feedback coming from the user. This would ultimately disambiguate its uncertainty, because the builder would communicate failure in case her preference is a "no-hold." For the purposes of this work, we examine here three prototypical interactions, color-highlighted in **Table 3** with their corresponding preference update in the third column of each table:

—*Robot-initiated failure detection*: as described in **Section 3**, the robot is not allowed to directly perceive the state of the world and the progression of the task. Still, it is possible for it to detect an action failure by using its own internal sensors. If this is the case, e.g., when the robot tries to pick up the screwdriver but the gripper is empty, the system is able to re-plan its execution and re-schedule the action at a later stage. As highlighted in **Table 3**, blue sequence, the robot does not necessarily repeat the same action right after the failure is detected, since other actions may have higher priority at that point.

—*Successful hold action* (green sequence in **Table 3**): the robot starts with a non-zero estimation $\widehat{p_H}$ of probability for the hold preference. If, while performing an "hold" action, it does not receive negative feedback from the user (observation: "none"), $\widehat{p_H}$ increases as it becomes more likely that the builder wants the robot to hold. This is further enforced with subsequent "hold" actions.

—*User-initiated failure*: in Condition B, the system experiences an user-initiated failure while proposing to hold the

part (observation: "fail," red sequence in **Table 3**). As a consequence of this, the probability of the hold action in the belief distribution $\widehat{p_H}$ decreases, and the robot will not perform this action in the future. Rather than hold parts for the user, it will wait for her to complete the action, and will move on to the next step when she communicates completion of the subtask.

One last aspect worth elaborating on is the fact that, in order for the robot to be perceived as an effective collaborator, transparency of the system during interaction is paramount. The human needs to be able to access (to a certain degree) what the robot's internal state is, what it thinks about the task progression and, importantly, how it intends to act next. Failure to deliver transparency results in user frustration and task inefficiency. Within this context, the overlapping, redundant interaction channels (cf. **Section 4.1**) are beneficial in guaranteeing a transparent exchange of information between the two partners. This is particularly important in case of unexpected deviations from the robot's nominal course of actions—that is, robot failures. Both in case of a robot-initiated failure and a human-initiated error signal (**Figures 8D–F**), the system was able to acknowledge the user about its error state through the Baxter's head display and/or speech utterances. In this way, it was always evident to the user that the robot failed, and eventually why it failed (in case the failure was not of robot-initiated).

6 DISCUSSION AND FUTURE WORK

In this work, we present a system able to convert high-level hierarchical task representations into low-level robot policies. We demonstrate robustness to task representations with varying complexity, as well as a certain degree of customization with respect to task-relevant variables such as user preferences and task completion time. Further, we provide demonstration of our technique in a mixed-initiative human-robot collaboration. As evident in the accompanying video, the human maintains full control throughout the task execution, but the robot acts independently, anticipates human needs, and does not wait to be told what to do. The paradigm in which we operate is neither to attempt implementing the ideal system that never fails (or does not contemplate the occurrence of failures), nor shaping the environment in such a way that it prevents the robot to fail. Rather, we decidedly embrace the idea that robots' perception and actions are inherently faulty, and errors during operation are possible and *expected to occur*. The approach we present does not intend to compete with the optimized assembly lines that takes months to design, but provides an easy-to-deploy, reconfigurable paradigm, suitable for small and medium enterprises.

To our knowledge, this work is the first attempt at a *practical* demonstration of supportive behaviors in a realistic human-robot collaborative scenario. In addition, we fundamentally differ from past research on the topic, where collaboration typically translates to the human and the robot tasked with parallel, non overlapping subtasks and structured, transactional interactions. Rather, our experiment shows a fully integrated interaction, where the human and the robot *physically engage in shared-environment collaboration*.

A more extensive evaluation of the scalability of the proposed framework to a broader domain of applications is the major direction for future work. Although the simulated interactions shown in **Section 5.1.1** proved its feasibility in theory, it remains to be seen how much the approach can scale up to more complex tasks in practice. In particular, we plan on leveraging the flexibility of the HTM representation to: 1) model more complex task structures; 2) apply the method to different interaction paradigm. Furthermore, our previous work introduced a model that allows the robot to effectively exploit basic communication capabilities in order to target the problem of task allocation and information gathering (Roncone et al., 2017; Brawer et al., 2018). Bringing this level of interaction in the current setup is also a direction of convergence.

In this work, we assume that the high level representations our system relies on are either already available or easy enough to generate. It is however a matter of future work to explore how this assumption holds in various application domains, and whether sufficiently precise models can be learned from spoken instructions or user demonstrations. Our method also relies on existing controllers for the supportive actions. Interesting directions of future work would entail the ability for users to teach new primitives to the robot (e.g., by demonstration) and then combine these primitives into more complex task models.

An important factor to consider when applying high-level task planners to HRC is that there is no theoretical guarantee that the reward parameters set in the system correctly approximate the value of the robot's behavior to the user. We consider this as being one of the main obstacles that researchers face when applying research from POMDP planning to HRI. Consistently with prior work, in this paper we have made the decision to focus on task completion as the main metric to measure collaboration performance (as seen in e.g., Roncone et al. (2017)); however, there may be other sources of value in support that are not directly measured by completion time—such as reduced cognitive load or physical fatigue on the human partner. As a consequence, the design of the reward function explicitly relates to task completion time, either directly or indirectly. This conveniently allows to set reward parameters based on empirical measures, which constitutes a promising calibration procedure for the system. Some reward parameters are directly related to measurable time (such as the reward for advancing to the next subtask which is identified by the average completion time for that subtask, or the reward for fetching a tool which is related to the time it would take the human to fetch it). Others can be estimated indirectly (e.g., the reward for respecting the hold preference can be computed by measuring how much time users are ready to sacrifice to get their preference followed). Exploring whether this constitutes a sound and efficient calibration mechanism is out of the scope of this paper but an important avenue to be explored in future work.

In conclusion, in this work we presented an interactive and effective HTM-to-POMDP method that will open the door to a wide array of user studies to assess the quality and the effectiveness of the interaction between the human and the robot. The degree of proaction shown by the robot during collaboration can significantly lower the barrier to entry for non-expert users, in that users can immediately see what the robot is capable of, without over- or under-estimating its skill set. In this regard, an extensive user study would help solidifying this intuition, and assessing how useful the proposed system is in setting expectations for naive users. Finally, although our prior work showed a general user preference toward our system (Roncone et al., 2017), a broader user study would prove statistical significance in terms of reduced levels of stress and cognitive load to the user. Importantly, this would also allow to highlight potential friction points that can be leveraged to better design the human-robot interaction.

DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://github.com/ScazLab/human_robot_collaboration and https://github.com/ScazLab/task-models.

ETHICS STATEMENT

The studies involving human participants were reviewed and approved by Institutional Review Board at Yale University. The patients/participants provided their written informed consent to participate in this study.

AUTHOR CONTRIBUTIONS

OM and AR contributed to the conception and design of the study, development and release of the software, and analysis of the results. OM, AR, and BS wrote sections of the manuscript, contributed to revision of it, and approved the submitted version.

FUNDING

This work was supported by the Office of Naval Research (ONR) award #N00014-18-1-2776, the National Science Foundation (NSF) under grant nos. 2033413, 1955653, 1928448, 1936970, 1813651, PI Scassellati, and the Army Research Laboratory under grant nos W911NF-21-2-0290, W911NF-21-2-0123, PI Roncone.

REFERENCES

Allen, J. E., Guinn, C. I., and Horvtz, E. (1999). Mixed-initiative Interaction. *IEEE Intell. Syst.* 14, 14–23. doi:10.1109/5254.796083

Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2008). *Robot Programming by Demonstration*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1371–1394. doi:10.1007/978-3-540-30301-5_60

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's J. Softw. Tools*.

Brawer, J., Mangin, O., Roncone, A., Widder, S., and Scassellati, B. (2018). "Situated Human–Robot Collaboration: Predicting Intent from Grounded Natural Language," in Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on (IEEE).

Breazeal, C. (2002). *Designing Sociable Robots*. MIT Press.

Chang, M. L., and Thomaz, A. (2021). "Valuable Robotic Teammates: Algorithms that Reason about the Multiple Dimensions of Human-Robot Teamwork," in Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction, 580–582.

El Makrini, I., Merckaert, K., Lefeber, D., and Vanderborght, B. (2017). "Design of a Collaborative Architecture for Human-Robot Assembly Tasks," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE), 1624–1629. doi:10.1109/iros.2017.8205971

Erol, K., Hendler, J., and Nau, D. S. (1994). "HTN Planning: Complexity and Expressivity," in In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94) (Seattle, Washington: AAAI Press), 1123–1128.

Garland, A., and Lesh, N. (2003). *Learning Hierarchical Task Models by Demonstration*. Mitsubishi Electric Research Laboratory.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. (2014). Automatic Generation and Detection of Highly Reliable Fiducial Markers under Occlusion. *Pattern Recognition* 47, 2280–2292. doi:10.1016/j.patcog.2014.01.005

Georgievski, I., and Aiello, M. (2015). Htn Planning: Overview, Comparison, and beyond. *Artif. Intelligence* 222, 124–156. doi:10.1016/j.artint.2015.02.002

Gombolay, M., Wilcox, R., and Shah, J. (2013). "Fast Scheduling of Multi-Robot Teams with Temporospatial Constraints," in Proceedings of Robotics: Science and Systems, Berlin, Germany. doi:10.15607/rss.2013.ix.049

Goo (2021). Google Cloud Speech API. Availableat: https://cloud.google.com/speech/docs/.

Gopalan, N., and Tellex, S. (2015). "Modeling and Solving Human-Robot Collaborative Tasks Using POMDPs," in RSS Workshop on Model Learning for Human-Robot Communication.

Grigore, E., Roncone, A., Mangin, O., and Scassellati, B. (2018). "Preference-Based Assistance Prediction for Human–Robot Collaboration Tasks," in Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on (IEEE).

Grizou, J., Lopes, M., and Oudeyer, P.-Y. (2013). "Robot Learning Simultaneously a Task and How to Interpret Human Instructions," in Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on (IEEE), 1–8. doi:10.1109/devlrn.2013.6652523

Hartley, R. I., and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. second edn. Cambridge University Press. 0521540518.

Hayes, B., and Scassellati, B. (2016). "Autonomously Constructing Hierarchical Task Networks for Planning and Human-Robot Collaboration," in International Conference on Robotics and Automation (ICRA). doi:10.1109/icra.2016.7487760

Hayes, B., and Scassellati, B. (2014). "Discovering Task Constraints through Observation and Active Learning," in Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on, 4442–4449. doi:10.1109/iros.2014.6943191

Hayes, B., and Scassellati, B. (2015). "Effective Robot Teammate Behaviors for Supporting Sequential Manipulation Tasks," in Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. doi:10.1109/iros.2015.7354288

Hoey, J., Poupart, P., Bertoldi, A. v., Craig, T., Boutilier, C., and Mihailidis, A. (2010). Automated Handwashing Assistance for Persons with Dementia Using Video and a Partially Observable Markov Decision Process. *Computer Vis. Image Understanding* 114, 503–519. doi:10.1016/j.cviu.2009.06.008

Hoffman, G., and Breazeal, C. (2004). "Collaboration in Human-Robot Teams," in In Proc. of the AIAA 1st Intelligent Systems Technical Conf. doi:10.2514/6.2004-6434

Hoffman, G., and Breazeal, C. (2007). Cost-Based Anticipatory Action Selection for Human-Robot Fluency. *IEEE Trans. Robot.* 23, 952–961. doi:10.1109/tro.2007.907483

Ilghami, O., Munoz-Avila, H., Nau, D. S., and Aha, D. W. (2005). "Learning Approximate Preconditions for Methods in Hierarchical Plans," in Proceedings of the 22nd international conference on Machine learning (Shanghai, China: ACM), 337–344. doi:10.1145/1102351.1102394

Jiang, Y., Moseson, S., and Saxena, A. (2011). "Efficient Grasping from RGBD Images: Learning Using a New Rectangle Representation," in 2011 IEEE International Conference on Robotics and Automation, 3304–3311. doi:10.1109/icra.2011.5980145

Johnson, M., Bradshaw, J. M., Feltovich, P. J., Jonker, C. M., Van Riemsdijk, M. B., and Sierhuis, M. (2014). Coactive Design: Designing Support for Interdependence in Joint Activity. *J. Human-Robot Interaction* 3, 43–69. doi:10.5898/jhri.3.1.johnson

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and Acting in Partially Observable Stochastic Domains. *Artif. Intelligence* 101, 99–134. doi:10.1016/s0004-3702(98)00023-x

Kaelbling, L. P., and Lozano-Pérez, T. (2010). "Hierarchical Task and Motion Planning in the Now," in Proceedings of the 1st AAAI Conference on Bridging the Gap Between Task and Motion Planning (vancouver, canada: AAAI Press), 33–42. AAAIWS'10-01.

Kaelbling, L. P., and Lozano-Pérez, T. (2013). Integrated Task and Motion Planning in Belief Space. *Int. J. Robotics Res.* 32, 1194–1227. doi:10.1177/0278364913484072

Koval, M. C., Pollard, N. S., and Srinivasa, S. S. (2016). Pre- and post-contact Policy Decomposition for Planar Contact Manipulation under Uncertainty. *Int. J. Robotics Res.* 35, 244–264. doi:10.1177/0278364915594474

Marquardt, D. W. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *J. Soc. Ind. Appl. Mathematics* 11, 431–441. doi:10.1137/0111030

Nikolaidis, S., Lasota, P., Ramakrishnan, R., and Shah, J. (2015). Improved Human-Robot Team Performance through Cross-Training, an Approach Inspired by Human Team Training Practices. *Int. J. Robotics Res.* 34, 1711–1730. doi:10.1177/0278364915609673

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., et al. (2009). "ROS: an Open-Source Robot Operating System," in ICRA Workshop on Open Source Software.

Roncone, A., Mangin, O., and Scassellati, B. (2017). Transparent Role Assignment and Task Allocation in Human Robot Collaboration. IEEE International Conference on Robotics and Automation (ICRA). doi:10.1109/icra.2017.7989122

Searle, J. (1990). *Collective Intentions and Actions*, 19. MIT press, 401–416.

Shah, J., and Breazeal, C. (2010). An Empirical Analysis of Team Coordination Behaviors and Action Planning with Application to Human-Robot Teaming. *Hum. Factors* 52, 234–245. doi:10.1177/0018720809350882

Shah, J., Wiken, J., Williams, B., and Breazeal, C. (2011). "Improved Human-Robot Team Performance Using Chaski, a Human-Inspired Plan Execution System," in Proceedings of the 6th international conference on Human-robot interaction (ACM), 29–36. doi:10.1145/1957656.1957668

Shani, G. (2014). Task-Based Decomposition of Factored POMDPs. *IEEE Trans. Cybern.* 44, 208–216. doi:10.1109/tcyb.2013.2252009

Silver, D., and Veness, J. (2010). "Monte-Carlo Planning in Large POMDPs," in *Advances in Neural Information Processing Systems 23*. Editors J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta (Curran Associates, Inc.), 2164–2172.

Toussaint, M., Munzer, T., Mollard, Y., Wu, L. Y., Vien, N. A., and Lopes, M. (2016). "Relational Activity Processes for Modeling Concurrent Cooperation," in 2016 IEEE International Conference on Robotics and Automation (ICRA), 5505–5511. doi:10.1109/icra.2016.7487765

Wilcox, R., Nikolaidis, S., and Shah, J. (2012). Optimization of Temporal Dynamics for Adaptive Human-Robot Interaction in Assembly Manufacturing. *Proc. Robotics: Sci. Syst.* doi:10.15607/RSS.2012.VIII.056

Zeylikman, S., Widder, S., Roncone, A., Mangin, O., and Scassellati, B. (2018). "The HRC Model Set for Human-Robot Collaboration Research," in Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on (IEEE). doi:10.1109/iros.2018.8593858

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.