frontiers | Frontiers in Physics

TYPE Original Research
PUBLISHED 08 December 2022
DOI 10.3389/fphy.2022.975247

OPEN ACCESS

EDITED BY
William Frere Lawless,
Paine College, United States

REVIEWED BY
Jiuchuan Jiang,
Nanjing University of Finance and Economics, China
Michael Mylrea,
University of Miami, United States

*CORRESPONDENCE
Priyam Parashar,
priyam8parashar@gmail.com

†PRESENT ADDRESS
Priyam Parashar,
Meta AI, Pittsburgh, PA, United States

SPECIALTY SECTION
This article was submitted to Interdisciplinary Physics, a section of the journal Frontiers in Physics

RECEIVED 22 June 2022
ACCEPTED 07 October 2022
PUBLISHED 08 December 2022

CITATION
Parashar P, Goel AK and Christensen HI (2022), Using meta-reasoning for incremental repairs in multi-object robot manipulation tasks.
Front. Phys. 10:975247.
doi: 10.3389/fphy.2022.975247

Using meta-reasoning for incremental repairs in multi-object robot manipulation tasks

Priyam Parashar[1]*†, Ashok K. Goel[2] and Henrik I. Christensen[1]

[1]Contextual Robotics Institute, University of California, San Diego, San Diego, CA, United States, [2]Georgia Institute of Technology, Atlanta, GA, United States

Robots tasked with object assembly by manipulation of parts require not only a high-level plan for order of placement of parts but also detailed low-level information on how to place and pick the part based on its state. This is a complex multi-level problem prone to failures at various levels. This paper employs meta reasoning architecture along with robotics principles and proposes dual encoding of state expectations during the progression of task to ground nominal scenarios. We present our results on table-top scenario using perceptual expectations based in the concept of occupancy grids and key point representations. Our results in a constrained manipulation setting suggest using low-level information or high-level expectations alone the system performs worse than if the architecture uses them both. We then outline a complete architecture and system which tackles this problem for repairing more generic assembly plans with objects moving in spaces with 6 degrees of freedom.

KEYWORDS

cognitive artificial intelligence, cognitive robot architecture, robot system architecture, knowledge-based (KB), task planning, task and motion planning, meta-reasoning

1 Introduction

Industrial robots, i.e., robots producing consumer goods at industry-scale, have remained the fastest growing market in recent times [1]. This reliability and demand are attributed to model-based programming paradigms [2–7] which enable program-and-replay for manipulation tasks. Model-based programming assumes access to completely modeled objects, pre-existing sets of robot-motion plans, and a structured environment that does not change over-time. The requirements of the next Frontier in small-scale robotics, however, cater to a scenario where the end-user wants to program the robot on one instance of the task and expects generalization over different instantiations of the same task or tasks that are similar in terms of objects and actions [8–10]. This problem context brings several realistic, but presently unattainable, robotics challenges that are summarized by the overarching question of "how to transfer known high-level plans for a given task to a similar but different environment represented as low-level observations"?

We focus on the class of multi-object manipulation tasks where a task can only be achieved by correct handling of multiple objects that leverages their affordances. Examples include getting soup out by dipping the ladle with its concave side facing up and attaching a screw to a washer by aligning the screw shaft and with the washer hole. Given such objects affordances (concavity, liquidity; shaft-feature, hole-feature) and rules governing their interactions (contains; inserts), prior work on these tasks reasoning writes out high-level formulae or grammar which should be followed in achievement of the task. However, as pointed out in [11, 12] these formulae do not elaborate "how" the robot should move to accomplish satisfaction of the goal state from any given initial state. Motions depend on the value of the next state but also on low-level percepts in the current state like vision and joint readings. When solved analytically this is a computationally hard process. Thus, we have witnessed a shift towards approximate solutions that leverage data and structured principles [13]. We propose a complementary incremental approach where we can expand the scope of the application of a plan for a task based on trial-and-error. We address the specific problem in which, given a motion planner, a high-level plan for an assembly task, and the plan's successful execution on a specific grounding (i.e., object poses in 3-dimensional space), how might the robot transfer the planner and the plan to a new grounding?

Data driven learning has demonstrably worked very well for applications where noisy pixel or sensor inputs need to be matched to symbols [14, 15] or to a regressed control output [16]. However, the strength of these methods comes from the ability to mine unlimited amounts of data, either through web crawling or simulators. The robotics domain in contrast has limited and specific data, which leads to overfitting and non-generalized task solutions. A hierarchical model which can parse the environment using generic symbols (which can be learned from widely available data) but then uses specific data and learning at lower level to ground those symbols in the given environment and execute plans can bridge this gap [9, 13]. There is a corollary problem to learning policies, then: given manipulation policies and seed sets of states that lead to success, find other connecting states such that any given initial state leads to task success. Our approach does this by learning state-entity sets in which the agent explores an action space through trial-and-error and gathers enough state-data over time to enable robust execution with learned "good states".

More specifically, we want to balance exploitation of high-level knowledge of goals and plans with low-level exploration such that the agent can learn reactive repairs while maintaining goal-driven reasoning and behaviors. Concretely, this paper grounds assembly plans as actions on objects and leverages known object affordances [17–19] as the key state-variable. Object affordances in this chapter are defined as keyframes (position, orientation) with respect to the object's centroid (also known as the object's frame). This affordance-based state description, however, is low level, continuous, and incompatible with traditional task planning. In order to bridge this gap we take a dual-encoding approach leveraging the task domain definitions to also ground plans in high-level pre and post conditions. Thus, our architecture can reason about plans based on low-level sensor observations as well as high-level knowledge inputs. Meta-reasoning and goal-driven reasoning has addressed sophisticated tasks but mostly within disembodied contexts [20–24]. For example, the REM system uses meta-reasoning for transforming a plan for disassembling a device to a plan for assembling it from its components [25]. In contrast we address simpler tasks in an embodied context. This implies a grounding of the plan reasoning and meta-reasoning in visual encodings. The key contributions are an updated architecture for meta-reasoning, a theory for classifying failures in embodied systems, and grounding of meta-reasoning in perceptual expectations. These contributions build on previous several streams of meta-reasoning research which attacked this question of how to account for low-level observations [25–29] as well as robotics research investigating the conceptualization and operationalization of robotics processes as lifted symbolic plans and knowledgebases [2–5], [30]. Note that meta-reasoning itself is also a computationally hard-problem [26, 31] thus we use rule-based methods and data-based approximations in this chapter to ground these components.

The dual-encoding methodology presented in this chapter was observed to be more generic than low-level repair or symbolic repair used alone, as seen in Section 6. Further an application of this architecture with reinforcement learning used to learn action sequences for tasks showed much faster convergence of learning with the structure provided by the meta-reasoner [32]. Our experiments described in Section 7 show that a hierarchical architecture with a high-level repair module solves more instantiations of tasks than one with no high-level repair module [33]. In the next section we present a motivating example. Section 3 gives a quick introduction of relevant concepts and related literature. In Section 4, we present a first experiment which establishes the usefulness of visually grounded lower-level expectations. In Section 5, we present our second experiment which further refines lower-level expectations to account for object configurations and recovery from grounding-level failures.

2 Motivating example

Consider a robot executing a sequential plan to assemble multiple parts together. The robot is given one task of inserting a cylindrical housing into a matching feature on the task-board. The housing has a wider face on one-end which is not compatible with this insertion-feature. The execution sequence for this task is provided in Figure 1. We call this a nominal task sequence.

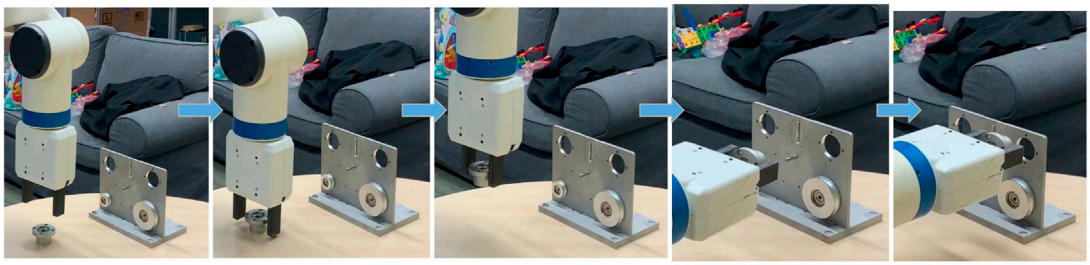


FIGURE 1
Nominal task sequence for inserting housing into the task-board.

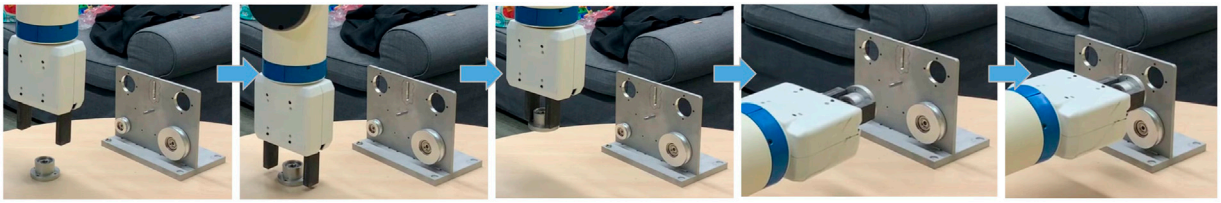


FIGURE 2
Variant setting with upside-down housing placement which leads to failed insertion task.

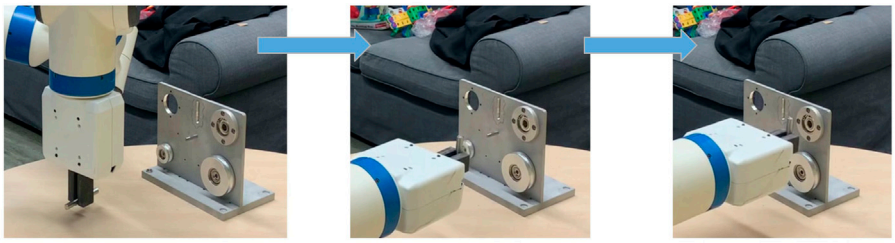


FIGURE 3
Variant setting with a new object whose placement differs from housing and repetition of same insertion task without further reasoning leads to failure.

The robot is now tasked to achieve the same goal of housing being inserted into the task-board in a variant setting, namely one where the initial pose of housing is upside-down on the table. At a high-level the current task remains the same, so if the robot repeats the same sequence as in the nominal setting without further reasoning it fails. Failure occurs because we transition into an incompatible state from which the insertion skill cannot occur as expected. This is shown in Figure 2. This is a simpler class of adaptation where the same objects are being used but their relative poses differ, so the robot's actions need to be goal-driven but also account for these grounded differences.

Now consider the robot is tasked with the goal of inserting a cylindrical shaft into the housing *via* the central hole-feature in it. Again, if the robot does not reason about task-relevant correspondence between the shaft and the housing and just repeats the insertion skill's plan as instantiated based on previous example then it will run into another failure. This situation is shown in Figure 3. This is a more complex case where the skill is the same but the object the skill is being applied to has changed. Now the robot needs to first understand how to adapt known affordances of objects based on previous examples, and next to plan an action sequence to account for the variations.

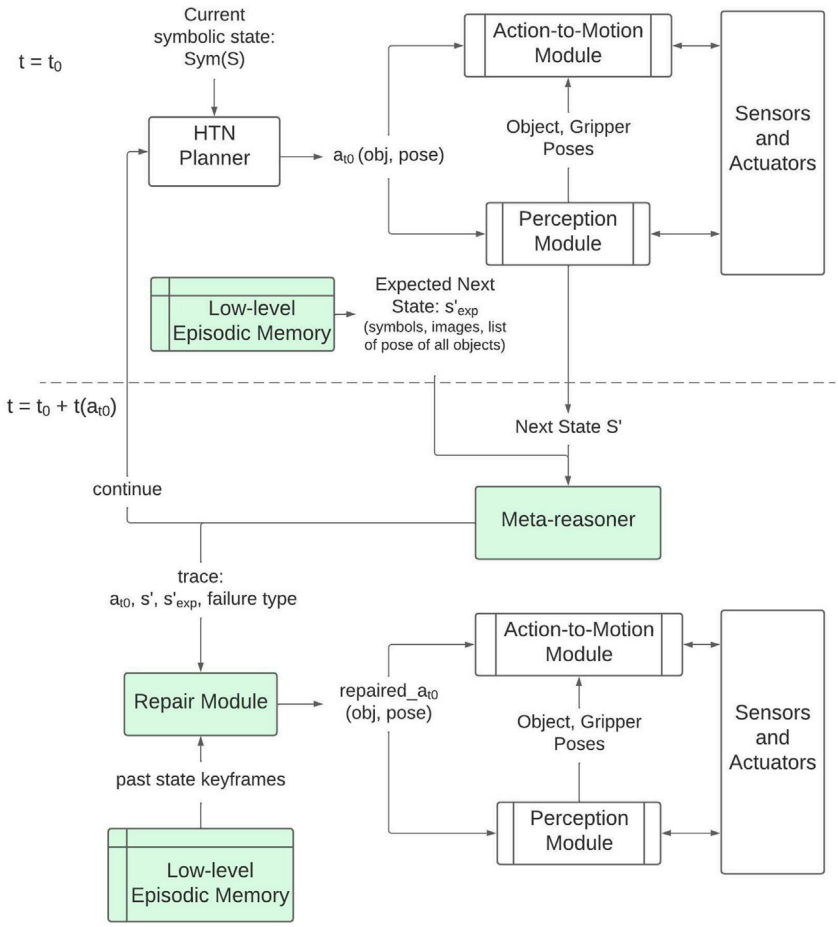


FIGURE 4
Overview of the system components and a temporal flow of how processes unfold across them when failures are identified.

An interesting aspect of this investigation is the entangled relevance of object and grounded pose towards task success. A robot is an acting agent, rather than a reasoning-only agent. A high-level model of these actions would only deign to answer, "use action A on object O." However, we contend that it is not enough to answer, "which object?" but the robot also needs to know what pose of the object with respect to the given task is actuable. A robot perceives its surroundings through 2D and 3D images, and then extracts out relevant semantic symbols as well as the grounded 6-dimensional pose (x, y, z position and r, p, y rotation with respect to base coordinate system) of these symbols in the environment. Each grounded pose of these objects may require a different grasp and alignment from the robots to enable their assembly which is non-trivial to plan since robots have a different physicality and reach in the Euclidean space as compared to humans. Prior work assumes the definition of the action encompasses this reasoning and focuses on high-level sequencing only. We relax this assumption, asking the question how to capture the low-level task-relevance of object affordances from successful high-level plans and use it to generalize said task-plan.

Further these interactions also implicitly respect dynamics like rigid object physics, friction dynamics, and even specific instantiation of objects in the current world. It is computationally intractabe to model each and every one of these aspects separately, thus we use the past traces from successful executions of a task to serve as heuristics on which poses make sense for the given task. Therefore, our focus is on the architecture which enables this learning from failures, incorporation of new evidence into knowledge-base and better planning rather than robustness of the object or affordance features themselves. Our primary goal here is to motivate the fact that the relevance of an object to a task and the function of the object's grounded pose within that task context are entangled together. An agent cannot generalize to a novel situation with an answer to only one of these two questions. Thus, in this chapter we

explore representations encoding object affordances, evaluate if these encodings improve example-based instantiation of novel tasks and present a framework which uses these encodings to facilitate automatic learning from agent's failures.

3 System overview

Figure 4 shows the lifecycle of a typical repair process described in this chapter. The key components of this framework are as follows:

1. A task: A task is a multi-action motion sequence which achieves a given goal state. Given a goal-state the HTN planner chooses the correct task. Thereby given the current state and overarching task it generates the next action towards that task.
2. An action: An action is defined as a skill-label (move, insert, etc.) along with its supporting arguments. In this chapter, we first discuss repair of objects as arguments and then expand scope for repairing embodied poses.
3. A low-level episodic memory: We assume the robot has past episodic memory of at least one successful execution of the task under consideration. This memory is assumed to have low-level information about objects and pose perception based on sensors during execution. Given a task and next action, one can query the expected state after successful execution of that action. One can also query the set of states which successfully lead to execution of a given action.
4. A meta-reasoning component: This component compares the current state with the expected state. It assumes a failure taxonomy to be present and follows rule-based assessment of whether a failure is present or not. It then either asks the deliberative planner to continue or passes on the failure category as well as low-level details to the repair module.
5. A repair module: Given failed action and current task, the repair module generates suggestions for next action which can lead to previously known states which led to success. We assume the HTN knowledgebase has actions which can bridge transformation from current state to "promising states" [32].

The biggest reason for failures in robotics is due to non-determinism over initial state and stochastic execution of actions. In this chapter, we focus on failures induced by missing availability of correct objects and associated table-top rearrangement (Section 6) and wrong positioning of objects (Section 7). Line items three to five described above are core contributions towards generating such repairs for an embodied system.

4 Related work

It is agreed in robotics that hybrid systems that can do both deliberation and reactive revisions pave the way for more complex robotic applications [34, 35], but there does not exist a systematic theory of how to combine distinct levels of planning, reaction, and learning. For instance, ROS [36], a robotics middleware commonly used in research and adopted by some circles in industry [37], uses hybrid planners at both navigation and manipulation level which use both global and local planning behaviors. Parashar et al [38] proposed a hierarchy of failures for such multi-layered architectures, as seen in Figure 5, attempting to scope systematic investigation. This paper is scoped at the level of understanding objects and pose related failures.

Architecturally this paper is informed by cognitive robotics systems like CoSy Project [34] and CRAM [39] and paves a bridge between such hybrid architectures and meta-reasoning components [40, 41]. Given heuristic or expert knowledge about a process, a meta-reasoning system (Figure 6) generates expectations about the state of the world given the actions applied to it, compares the observed state of the world with the expected state, and maps the discrepancy between expectation and observation into one or more repairs at the deliberative level [25, 26]. The recognition of a failure through a comparison of the expected state and the observed state can be challenging if the observations are made through low-level sensors and the expectations are encoded in terms of abstract knowledge representations. We seek a more general strategy for comparing expected and observed states and recognizing failures for the robotics domain.

Our work has some similarity with [23], since they too use HTN plans annotated with expectations to conduct meta-level reasoning over their incomplete plans. However, their expectations are of a conceptual form, abstracted on top of environmental symbols. Jones and Goel [29] present "Empirical Verification Procedures" which ground all high-level concepts and axioms known to the agent in lower-level precepts in a video game. Prior work [32, 42, 43], combines meta-reasoning with reinforcement learning using purely visual form expectations. However, they still use symbolic descriptions or computerized descriptions of visuals which simplifies the perception part of the problem.

Finally, to ground the planning problems we make use of formalism provided by Hierarchical Task Networks. STRIPS planning enables a search-based planning solution in the state-space of the planning domain, however given the repetitive structure of assembly plans, HTNs fare better in terms of efficiency as they allow reuse of expert knowledge. Such procedural and routine-based problems occur in scheduling and logistics regularly, and hierarchical task networks [44] (HTNs) have been used to instead express procedures for completing tasks. HTN planning was formalized and operationalized *via* the SHOP [45] and SHOP2 [46] planners in the International Planning Competition(s). HTNs are a popular way of designing domain-configurable as well as domain-specific planning domains [47], with the procedures defining search control over the state-space to make planning faster. A comprehensive review of different HTN planners is provided in [48].

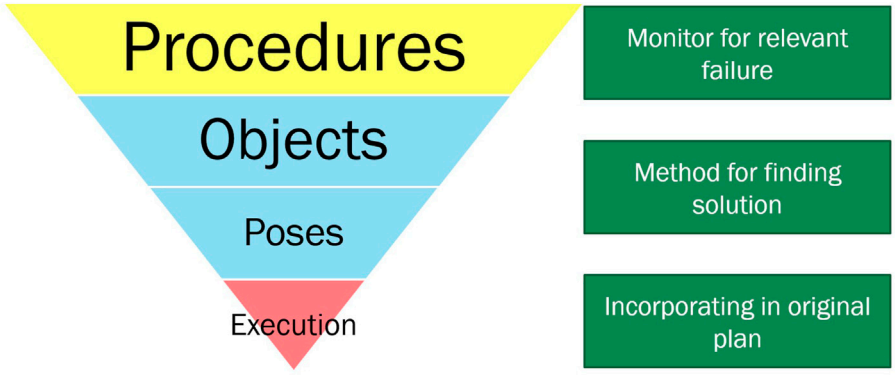


FIGURE 5
The hierarchy of failures (categorised by colour) that occur in a goal-driven, multi-layered robotics architecture. The green boxes represent the three required components to meta-reason about each class of failures.

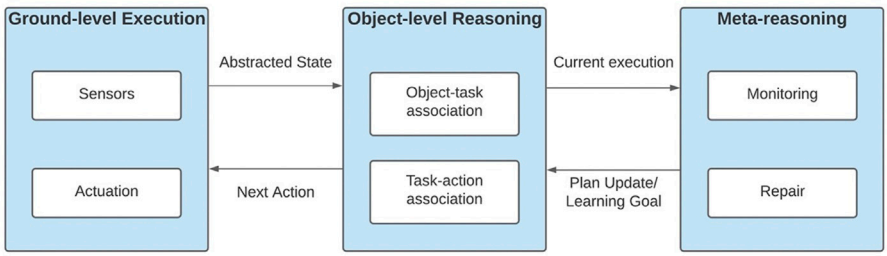


FIGURE 6
High-level system architecture describing the 3 salient components of a meta-reasoning system. Ground-level execution relies on sensors which read the environment state and actuators which manipulate it. The object-level reasoning component which relies on domain knowledge to formulate long-horizon plans given the current state. Finally, the meta-reasoning component which monitors the execution of the object-level plan and launches repairs when failures are identified.

5 Background

5.1 Hierarchical task networks and assembly task domain

We use the HTN formalism [49] for defining our task planning domain and problem. The domain is given as $\mathbf{D} = <\mathcal{P}, \mathcal{T}, \mathcal{M} \ldots >$. $\mathcal{T}$ is a set of tasks in the domain, and $\mathcal{M}$ is a set of methods (recipes) defining how to decompose $t \in \mathcal{T}$ to smaller subtasks which is called its task network. When t cannot be further decomposed then $t \in \mathcal{P}$ and is called a primitive action, which is directly executed by the underlying agent. γ is the set of preconditions defined as grounded symbols and predicates over grounded symbols for each method. A method m can be applied on task t while in state s if $name(m) = t$ and s satisfies $\gamma(m)$. δ is the set of effects that executing primitive-action t will affect on the world-state: if $t \in \mathcal{P}$ is executed in s then next state:

$$s' = \left(s - \delta^{-}(t) + \delta^{+}(t)\right)$$

$\delta^{-}(t)$ and $\delta^{+}(t)$ represent minus and add effects respectively. The assembly task domain that these HTNs operate on is based upon the domain formulation presented in [56] and does not cover those details in the current scope. In the later experiments we explicate relevant aspects of the task planning domain to ground our understanding and discussion.

5.2 Task and motion planning

Task and motion planning (or TAMP) [50, 51] is a task planning formalism extended to account for the continuous-space execution that robots need to do. This is done by extending the next-state equation from previous section and associating a set of continuous-valued valid poses for each agent and object

implicated in state $\mathcal{s}'$. Let $\mathcal{R}$ be the relation connecting an object or agent *oa* with its valid poses under predicate $\mathcal{p}$, then for state description $\mathcal{s}$ containing $\mathcal{p}$ applied on agents or objects:

$$\mathcal{R}(\mathcal{s}) = \mathcal{R}(\mathcal{p}(oa_i))$$

Given such symbol-to-set mapping TAMP does a hybrid optimization over value assignment (as described in previous sub-section) as well as finding a feasible space of continuous execution. However, this symbol-to-set mapping is non-trivial to define. The procedures covered in this chapter formally solve this problem *via* incremental learning of these sets given successful and unsuccessful examples and unlimited time for the robot to conduct trial-and-error. This learning technique is the same as set-expansion techniques used in knowledge-based information retrieval [52–54] except we are learning the set-based relation $\mathcal{R}$ in 6DOF space which does not exhibit any obvious semantic structure.

6 Reasoning across different objects

In this section, we discuss a simple table-top shape drawing task where we ask an agent to draw an alphabet with playing bricks. We wish to evaluate our meta-reasoning architecture that given a variation on the original task along with visual expectations related to its execution, can the agent provide better plan repairs than one with only conceptual expectations. Our experimental setup uses a Baxter with an eye-in-hand camera setup (camera is situated on the wrist). For evaluation we create example failure cases to compare the meta-reasoning cycle which uses dual-encoded expectations (visual + conceptual) against one which only uses the symbolic-level expectations. In the following sections we formulate this problem using task domain description, present representations for encoding visual expectations and discuss the processes which can utilize visual expectations to propose plan repair instead of conceptual expectations. We also explain our implementation for extracting these visual expectations out of an image-stream. In the results section we present a qualitative assessment of our system for failure recovery where failures are induced by changing the environmental conditions to mismatch plan pre-conditions at different depths.

6.1 Problem setup

The problem domain is to use Mega BloksTM to draw shapes on the table-top; for simplicity will be referring to a single unit as a block. Our system considers two different shapes of blocks: 1×1 and 1×2; and supports two different colors: blue and red. Goals are communicated as strings naming the shape to be drawn and relate to a sequence of placements of specific blocks which draw the shape. Each block's physical placement is described by two attributes, its orientation with respect to the table's axis and the location of its centroid in the workspace. When blocks are recognized in an image, they are indexed with a number starting at 0, e.g., b_0 =<color, shape>. To describe the placement of two blocks with respect to each other we use a graph-based format where $\psi_{0,1}$ is a coordinate system transform between the centroid of b_0 and the centroid of b_1. A 1×1 sized Mega Blok is of length 6.1 cm and width 6.1 cm, which we denote as *lb* in the rest of this section.

To codify the pre-conditions and effects of the HTN tasks we use a symbolic state description which include: 1) ob_{grip}: description of the block held in the gripper, Φ if empty and b_j if block with ID j is held in the gripper, and 2) β: set of pairs depicting the required blocks and their mapping to recognized blocks on the table. The overall system uses other variables for planning purposes, but they have been abstracted because they are not relevant to the current discussion. The only primitive action available to planner is place (b_j,x,y) which maps to a heuristic policy under which it grasps and then moves the block b_j to (x,y).

6.2 Approach: Hierarchical representation of expectations

In order to scope the search complexity, the high-level planning framework uses lifted symbolic descriptions, however if a failure is noted the meta-reasoner needs access to lower-level, continuous observations which is encoded in the expectations. Our overall planning then is hierarchical in nature, where high-level planner assigns a block symbol to the place action and then a lower-level simple reasoner assigns the requisite (x, y). Leveraging this dual-level planning, we encode a hierarchical schema of expectations. The higher-level level of expectations has block symbols describing the relation between a shape and required block units. The lower-level stores cropped visual grid-maps centred on each block to capture a locally detailed description of its placement. Each grid-map is of length $3lb \times 3lb$ to include the block and some of the surrounding context. We chose a size which records the surrounding context as such mid-level features have been seen to work better for task-level reasoning when compared to hyper-local object-specific features. Readers should note that such a low-level description would require domain knowledge to be encoded since its form is tightly integrated with the goal of the problem itself.

The plans are annotated with expectations at the primitive action level and bubbled up to be associated with the task by assigning the parent task the expectation of the last primitive action in its decomposition. Typical HTN methods/tasks have only symbolic pre-conditions associated with each possible decomposition and then pre and post-conditions associated with each action under it. By bubbling up these grid-based

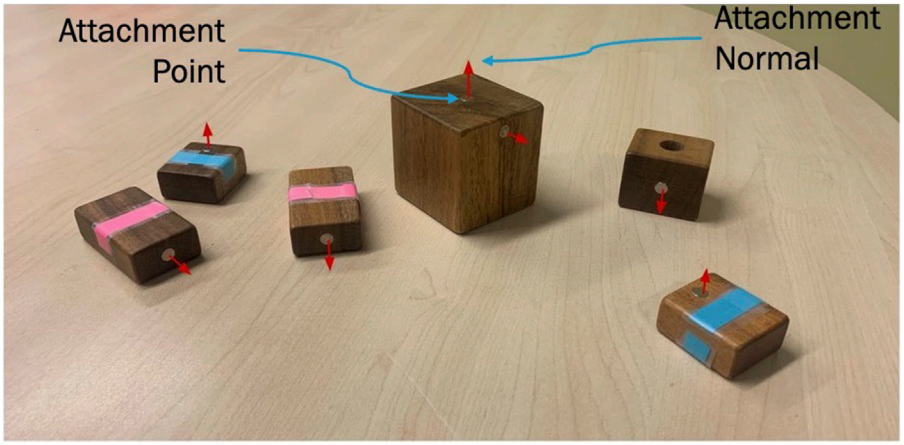


FIGURE 7
Overview of all objects implicated in MagneticAttach task as well as adiagrammatic description of attachment points and attachment normals.

encodings we add more information to our HTN description which also accounts for the final goal of the task.

6.3 Implementation

The HTN plan is executed by an expert kinesthetically driving the agent (hold on to the agent's arm and move it to perform the required actions) to annotate the plan with resulting "ideal profile" of expectations. After each place action the user presses a key to record the block which was placed as well as its 2-dimensional location with respect to the table-top coordinate system. After the full execution, the expectation annotator creates two kinds of databases: one of the complete annotated plan executions, denoted as P_a, which maps each action-step i to demonstrated $< b_j$,x,y > instance. The other one stores the low-level grid-maps and symbolic expectations annotated by the causing action(s) which points back to the parent task itself, denoted by E_{db}. This action-to-expectation mapping can be many-to-one. The second database is key for creating a two-way communication protocol between sensor information and plan knowledge even when symbol grounding fails during run-time.

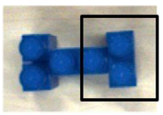
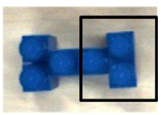
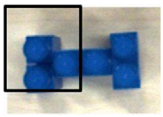
E_{db} is populated using an expectation extractor module. The expectation extractor uses a top-view image feed of workspace, *via* the robot's eye-in-hand setup, to extract expectations associated with each action and task. The block-level description of a symbol is extracted by performing HSV color-thresholding for blob detection on the tabletop view of the symbol under-construction. Once a colored blob is found, its shape is assigned by comparing blob-axis with lb. Next, the visual expectation is the image within a $3lb \times 3lb$ bounding-box centred on the centroid of the blob, and a quantized view of the form of the blob, i.e., a grid-map is created. A grid-map is like an occupancy grid where the occupancy of a cell is decided based on the color presence of the block on a uniformly colored background. The resolution of the grid-map is $0.5 \times lb$.

6.4 Experiment and results

An embodied system can encounter two kinds of failures: logical or physical. By physical we mean misplacement of gripper, wrong state of gripper, etc. This work does not address these failures. In the rest of the paper when we explain our algorithm, we are addressing only the logical failures, i.e., missing blocks, unexpected configuration of blocks, etc. We broadly classify the logical failures into two kinds, one where known entities are observed in an unseen configuration thus going against the explicit nature of pre-conditions, and one where unknown entities are observed breaking the planner's assumptions. We present here an example case where we create both kinds of failures by manipulating the environment. In this example, we have provided the agent with the plans for shape A and H (Figure 7), using two 1×1 blocks for H rather than one 1×2 block. We use these shapes because they possess the kind of form similarities, we want our algorithm to identify. We want to see if our expectations can help in creating connections between pieces of knowledge already stored in our database better than symbolic expectations. Next, the environment is modified to progressively make the failure more difficult for drawing the shape A. We replace required 1×2 block with another:

- Block of same shape but different color
- Set of two 1×1 blocks of same color
- Set of two 1×1 blocks of different color

TABLE 1 Summary of match results. The white square shows which block's resultant placement expectation in shape H was matched.

Type of Replacement	Affected Action-Exp Pair	Grid-map Match	Symbolic Match
1 × 2, red → 1 × 2, blue	BO = {1 × 2, red, 90°}	B4 = {1 × 2, blue, 90°} 	B4 = {1 × 2, blue, 90°} 
1 × 2, red → 1 × 1, red + 1 × 1, red	BO = {1 × 2, red, 90°}	None	None
1 × 2, red → 1 × 1, blue + 1 × 1, blue	BO = {1 × 2, red, 90°}	B2 = {1 × 1, blue, 0°} 	None

The mismatch vector is used to identify the first instance of action in the invoked task which uses the mismatched entity, i.e., block in our case. Next, this action's expectation is retrieved from P_a and a nearest-neighbour algorithm is invoked to find ranked matches from E_{db}. We compare the entries retrieved by symbolic matching and grid-map matching to qualitatively assess the usefulness of our hierarchical expectation representation.

Our results are summarized in Table 1 and compare the grid-map retrievals against symbol expectation matching. The most significant result is shown in row three where due to a grounded encoding of grid-maps its matches were able to search for a visual similarity of form unlike symbolic matching. For row 2, neither found a match since no shape uses {1 × 1, red} blocks in the current HTN plan library.

Our approach lends itself naturally to hybrid execution architectures where reactive learners manipulate raw-data and work in synchrony with deliberative planners which rely on some heuristic or some other form of domain knowledge. While it is easy to think of meta-reasoners as only an additional layer, its strength lies in enabling trading of valuable information across these two layers. It is this strength of the meta-reasoner to form a global view which we believe will be a valuable addition to the long-term autonomy literature in robotics. Specifically, its across-event reasoning can augment the strength of episodic performance exhibited by reactive learners and task-oriented planners.

7 Reasoning across different object poses

The occupancy grid expectations used in the last section are useful for encoding states of the world but run into several problems for generic usage. In a world where an agent is actively interacting with the objects, the agent itself may occlude the sensor which can result in a different expectation which maps to the same underlying state. Further, these relative poses are also entangled with the affordances of the object, i.e., placing two different objects in the same relative pose might not lead to an assembly which was not the case with our previous simpler domain. Finally, while grids work for planar cases, for more complex 3D assembly tasks the quantization can abstract away important low-level state information. In this section, we refine the expectations to be applied to the lower-level state of assembly objects (i.e., position and orientation) and propose a generic backtracking-based repair algorithm over the representation. We use a key point-based object representation since the entire 6-dimensional pose (3-dimensional position and 3-dimensional orientation in free-space) of multiple objects is critical for the success of an assembly task; thus, addressing a more general formulation of the assembly problem.

7.1 Problem setup

An assembly task-and-motion planning domain is defined by a goal-state, i.e., its symbolic and sub-symbolic description. We extend this definition to our modified HTN methods since the original only has a precondition and a network. Continuing the setup in the previous experiment, we assume that the symbolic goal-state is given and the demonstration is used for extracting sub-symbolic states of the objects. As described in Section 7.2.1. Each state's sub-symbolic description includes observed valid poses of all objects and agents implicated in the state-description. This distinction between all valid and observed valid poses is important to note as it distinguishes our work from TAMP. We do not assume all valid poses given, rather build these sets from observations. This is also a weakness as in the current version we do not include known kinematic poses of the agent in the formulation which leads to motion planning failures (discussed in Section 7.5.2). For simplicity of analysis, we only consider two action primitives for this work: *MoveTo* which takes a pose as input and *Grasp* which toggles gripper state.

Building on the HTN specifications, for the purpose of this study we categorize the preconditions into two types, those for defining generic applicability (for example, gripper can grasp an object if the object has a grasp-point and gripper is open) and task-specific (for example, gripper should grasp the toy bricks

without blocking the bottom attachment point). In an ideal situation it would be desirable for the domain designers to inject the generic preconditions marking the minimum set of conditions necessary for applying an action to the environment, and for the agent to learn the task-specific constraints based on nominal scenarios, object knowledge and transitions made by assembly skills. Thus, the aim of this study is to learn these task-specific relations over objects and their sub-symbolic groundings, assuming generic preconditions to be given. Please note, we always assume that the nominal plan and traces associated with this nominal plan are already provided to the agent. We focus on the representation of trace, monitoring using expectations based on trace and plan repair.

We model an assembly task as an initial state s_0, a goal state s_g, an attachment (equivalent to a task method) att_g and three main entities: an assembly agent, an active object and a reference object. An active object is the one being manipulated by the assembly agent with respect to the stationary reference object using the action decomposition of att_g leading to s_g. Traces are provided for nominal task settings where poses of objects match underlying assumptions of the plan, the robot is then exposed to a variant setting where the objects are in a different configuration. In the following section, we define the knowledge and representations for capturing task traces. This is followed by a description of how to generate expectations, monitoring over expectations and observations, and the algorithm to repair failures. Finally, we step through our initial experimental result.

7.2 Execution trace: Knowledge and representations

While the task plan considers gripper's poses across the space to ground a plan, the meta-reasoner explicitly collects traces encoding deeper knowledge about how the change in gripper pose is affecting the poses of the object it is operating upon. This bears similarity to how high-level and low-level information is connected in [50, 55]. However, unlike the former we do not assume these relations to be already given rather learn them as part of trace collection and compared to the latter we organize knowledge differently and do not explicitly connect the poses with kinematic constraints of the robot. Thus, for the given attachment att_g an action a_i at ith place resulting in observed state s_0, will have trace-state:

$$T'(i) = \bigcup_{obj_i \in att_g} state(obj_i)$$

Note that this bears similarity to the TAMP equation in Section 4.B. relating sub-symbolic grounding of states to known valid 6DOF states of objects. These traces can now be used to calculate expectations over pose-changes over time for individual objects, as well as for two objects with respect to each other. For the lifted symbols, trace is collected by attaching causal-links to variables which are established by assembly skills by way of computation, perception or motion (see Supplementary Materials).

7.2.1 Object state representation

Each object implicated in a task, i.e., $O(att_g) = \{obj_a, obj_r\}$, is identified by its semantic name which is a string passed as an argument for HTN task methods. Each obj is assigned the following attributes for describing its state:

- Object Pose: $P(obj)$: $|O| \Rightarrow \mathfrak{R}^6$
- Attachment Point: $AP(obj)$: $|O| \Rightarrow \mathfrak{R}^3$
- Attachment Normal: $AA(obj)$: $|O| \Rightarrow \mathfrak{R}^3$, relative to $P(obj)$

Even if implicitly related, all the components are converted to be with respect to the robot's coordinate frame for traces.

Figure 8 shows such a description for the objects in the magnetic robot domain. This representation is based on the preliminary assembly representation in [56] where attachment normals and final pose of objects with respect to world coordinate are specified. Note that here we encode keypoints with respect to the object frame as intrinsic features or affordances of the objects. One can imagine several models, based on this representation, co-existing for a given set of objects. For example, for the toy brick domain, the top and bottom groove locations would count as legitimate attachment-points when creating a 3D structure. On the other hand, for the 2D planar experiment outlined at the beginning of this chapter, we would instead expect a representation where the sides of the bricks count as legitimate attachment-points instead.

7.3 Monitoring for failure

In this work we assume we are only handling objects and pose related failures. Given that the agent only knows about the affordances of each object but not their relative importance to the task or to each other, it is not clear to the agent whether a task is destined for success or failure until the final attachment occurs, especially when the agent has not seen any failures in the past. Thus, we add a verification procedure [29] which is executed by the agent at the end of a task to verify the task was a success or not. If the task is deemed a failure based on verification procedure, then the meta-reasoning process is triggered with traces of past execution, T_{past}, and the current execution trace, T_{curr}. The verification procedure is added as a task method as shown in the code snippet in Supplementary Materials.

In practice though, due to occlusions we detect failure or success after the attachment task by moving the entire sub-assembly to a staging pose. Then we try to detect the assembly on the table, if it is not found then the assembly is verified as

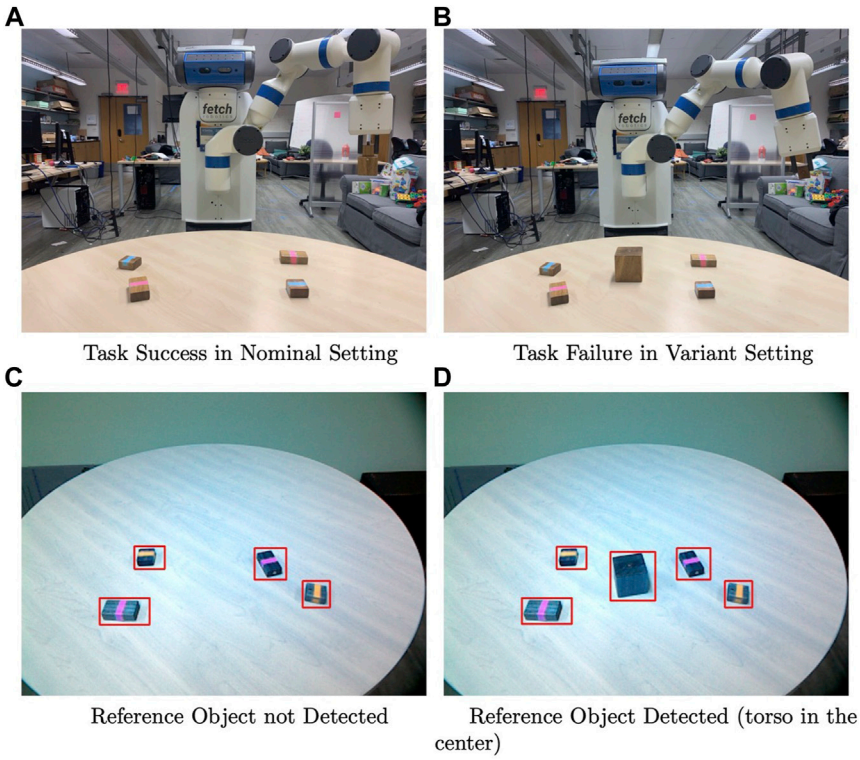


FIGURE 8
An example of a task failure and how visual systems help in monitoring it to verify task completion.

successful. This process is depicted in Figure 9. Note that if obvious or hard failures like mismatched preconditions are observed, then the meta-reasoner can use the algorithms proposed in prior work [32, 33] to repair the domain. In contrast this chapter focuses on non-obvious or soft failures where all the tasks and skills are actuable however do not lead to a successful result.

7.4 Meta-reasoning and repair

If the verification procedure results in a failure, the meta-reasoning cycle is triggered where the meta-reasoner collects past N traces for the same task, generates expectations and invokes a step-by-step comparison with the current trace. We focus on failures at the lowest level, since we have observed this level to be most probable for failures and least explored in related literature. Please note that while the final failure is registered in the form of a logical inconsistency, i.e., task executed but attachment was false, the originating reason for this can be physical or logical.

The generated low-level expectations encode constraints over the relative poses of the two objects or sub-assemblies being attached in the given task. Informally, given aggregate and current poses of assembly objects, it expects:

$$aggrpose(obj_1): aggrpose(obj_2) :: currpose(obj_1): currpose(obj_2)$$

Formally, these relations are computed based on aggregation over past traces and the following sets of equations define relations extracted over each trace T_j for task-step i and the expectations averaged over multiple traces.

$$R_p(i,j) = \text{EuclideanDist}(AP(obj_a), AP(obj_r))$$

$$R_n(i,j) = \cos^{-1}(AN(obj_a) \cdot AN(obj_b))$$

$$Exp_{p(i)} = \frac{\sum_{j=1}^{N} R_p(i,j)}{N}$$

$$Exp_{n(i)} = \frac{\sum_{j=1}^{N} R_n(i,j)}{N}$$

If the attachment point and normal in the current trace are significantly different from expected configurations, then the gripper backtracks to the last primitive which assigned its new pose. At this task-step a random good position for the active object's attachment point and corresponding normal is

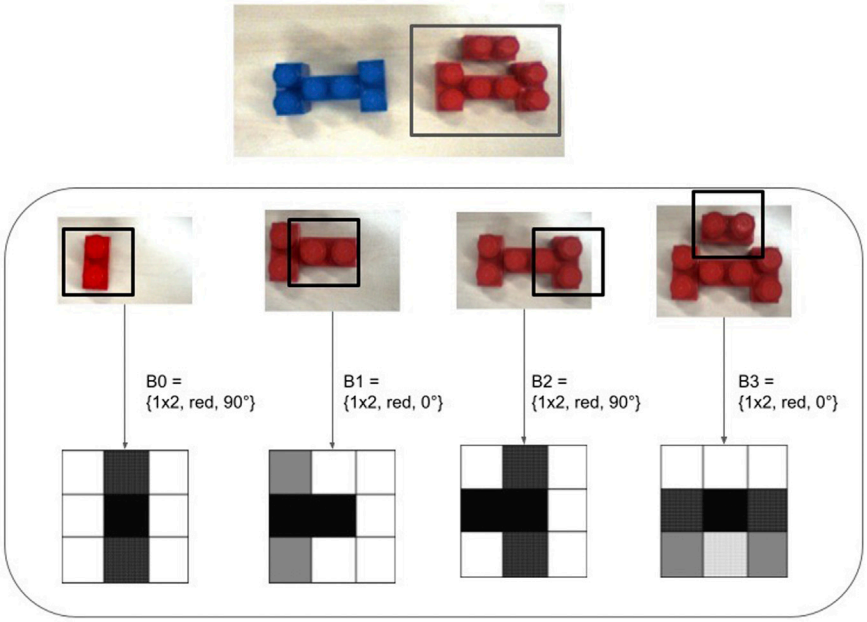


FIGURE 9
Depiction of Hierarchical Expectations, position of centroid removed from symbolic description for brevity. At the top, the darker H shape is blue while the lighter A is red.

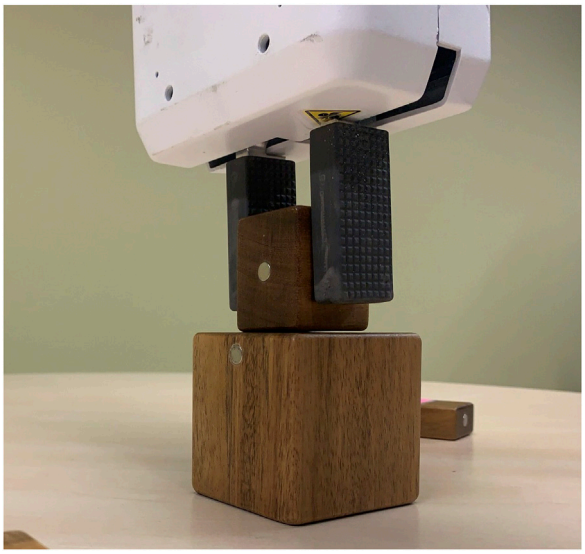


FIGURE 10
Variant setting leading to failure as the attachment points are not aligned and the nominal method did not have explicit actions to align them.

sampled from the traces. We have provided the meta-reasoner with a motion repair module. Given the kinematic relations between the object and the gripper, this is transformed into the related gripper position which is then treated as a repaired argument for the primitive. The repair algorithm keeps backtracking until the last pose which differs from trace if no successful plan is found. The final successful repaired plan, if found, is saved as a new refinement of the task along with starting pose of the objects as a sub-symbolic precondition.

7.5 Experiments and results

7.5.1 Setup

We focus our experiment on the *MagneticAttach* high-level task which is decomposed into a plan as shown in Figure 10. This plan requires grounding for the *MoveTo* primitive poses. These groundings are given by a human instructor *via* a graphical utility aligning the arm with the objects. This grounding is recorded by the task planner using object and gripper poses.

Figure 11 shows the progression of the nominal plan and the changes made to objects. Figure 12 shows the novel variant task which fails given knowledge gap in object grounding. In our results we show how the plan is compared and changed, evaluate transfer over traces which account for different objects as well as configurations, and finally we also outline the complex failures observed during the course of this attempted repair.

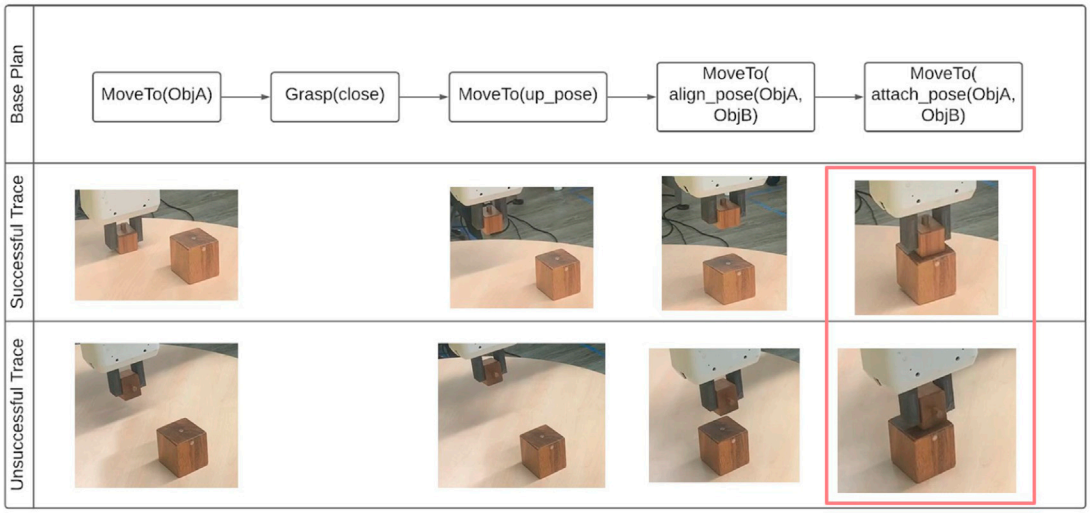


FIGURE 11
A high-level outline of how HTN plans are grounded using experience and compared for repairing object pose input to the execution actions. Meta-reasoner repair process which starts when the top base plan results in failure. The meta-reasoner first compares to see if the current trace was significantly different from nominal traces, if it is then a replacement object pose is sampled from previous traces, the executor backtracks to last action and tries this pose for next action. This sampling and backtracking occurs until a solution is found. Once solution is found, the current trace is rewritten with the new values.

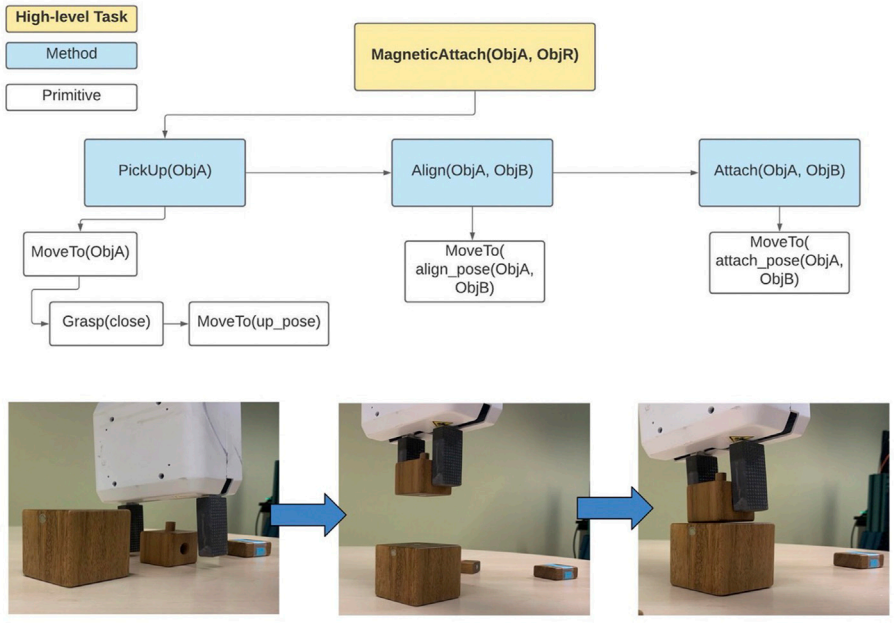
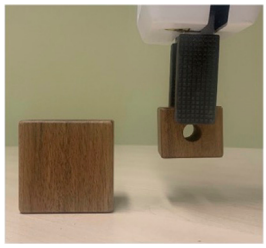
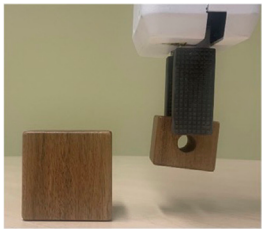
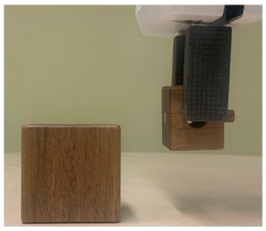


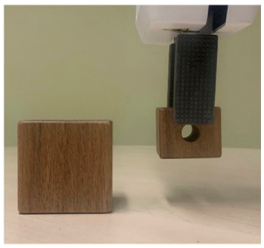
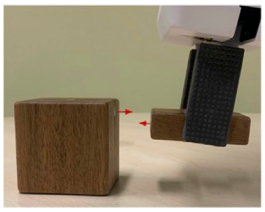
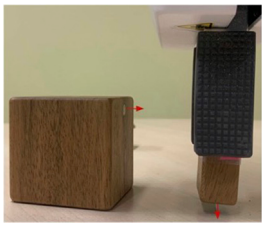
FIGURE 12
Nominal HTN plan decomposing a high-level assembly task (MagneticAttach) to primitives using methods. At the bottom we show the nominal task trace which grounds the high-level plan within the poses of the robot and the objects.

TABLE 2 A comparison of calculated R_p and R_n across different poses of the same object. Column 1 is the nominal plan grounded in instructed poses. Column 2 and 3 are variant poses from the test task. R_p is a good indicator of failure at the end of the task, however R_n is more sensitive to gross variations in object's pose.

			
Rp	0.2276	0.3355	0.3409
	0.2248	0.3286	0.3734
	0.0575	0.0392	0.0720
	0.0023	**0.0058**	**0.0412**
Rn	3.123	3.129	**1.483**
	3.14	3.133	**1.494**
	3.13	3.127	**1.492**
	3.13	3.14	**1.483**

The bold values indicate higher distinction power between task-states.

TABLE 3 A comparison of calculated Rn across different objects. Column 1 is the nominal plan grounded in instructed poses for Object 1. Column 2 and 3 are nominal and variant poses for Object 2 operated on by the same MagneticAttach task. R_n traces are still able to differentiate between wrong and nominal traces, given only the object attachment information about Object 2.

			
Rn	3.123	3.141	**1.571**
	3.14	3.142	**1.569**
	3.13	3.137	**1.568**
	3.13	3.143	**1.568**

Bold values indicate higher distinction power between task-states.

7.5.2 Repair result

The meta-reasoner backtracks one step each time, samples a known good pose from past traces which resulted in a successful task and plans a motion for it. This pose replaces the argument for next action in the plan. We observed as long as the gripper pose associated with the object is reachable from current configuration this repair can find a repair.

However, we observed two key failures which our approach does not model:

1. Collision Failure: In some cases, since we are not modeling the grasp to be task-informed, the gripper can cover the attachment point of the object. In such cases, even if our repair algorithm provides a good pose for the object, the robot cannot plan for it since the gripper would collide with the reference object in this configuration.
2. Reachability Failure: In certain cases, even if the gripper is not covering the attachment point, the robot arm joint configuration required for a good object pose is unreachable from the backtracked pose. This is due to singularity issues in arm motion planning and limited workspace.

7.5.3 Monitoring for same object's variant poses

Table 2 presents the values for R_p and R_n for the object configurations shown in the header. We see that by considering both values for traces, we can establish better similarity between successful tasks.

7.5.4 Monitoring across objects

Different objects may have their attachment normals aligned with different axes of the object's coordinate frame. In Table 3 we present the R_n values comparing nominal trace for one object to nominal and wrong configuration for another object. We observe that by modeling attachment normals separately we can still establish similarity between successful and failing tasks.

8 Conclusion

We started this paper by describing a research gap in traditional industrial robotics' planning around the issue of small-scale, heterogeneous assembly scenarios. The traditional methods only operate at high-level modeling of plans assuming low-level poses of objects are fixed, however such mono-level architectures and systems do not work well when an agent is required to adapt and learn in unstructured environments. We also highlighted a research gap in metareasoning literature with respect to grounding methods of failure monitoring and repair in action and perception that are critical for an embodied agent like an industrial robot.

In this investigation, we outlined lower-level task representations and reasoning processes to include them in the meta-reasoning architecture. These task representations focus on the action and object representation components which are unique to embodied agents. This gave rise to representations of expectations in meta-reasoning which encode relations between physical goal-state and acting processes rather than encoding the meta-relations between the reasoning processes and their arguments, as is typically done in the traditional meta-reasoning literature. This motivated an updated architecture and processes to encode, store and use these representations in an action and object-centric manner.

Our key finding is that extending the metareasoning architecture with the lower-level expectations adds flexibility to otherwise rigid model-based planners. We posit that this also enables a crisper modeling of different knowledge bases and processes involved in a robotics planning and execution process. Using only conceptual expectations does not capture state changes on the ground and our experiments suggest using both conceptual and visual expectations solves more kinds of failures than conceptual expectations alone. We conclude that knowledge transfer using metareasoning makes a robotic system more flexible than one with only classical planning. On the other hand, previous results [42] suggest that learning with metareasoning requires more structured knowledge but less data.

Our overall goal and motivation with this line of work was to enable adaptive agents which can conduct long-term reasoning but also learn from low-level data and thus explore the environment in a meaningful way. We believe that while learning paradigms can bring significant improvement to what we believe a robot is capable of, a learned component is only as good as the quality of data it is based on. We conclude that designing transitional agents can enable a bridge to systematically collect real-world data by specifically mining failure-events.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary Materials, further inquiries can be directed to the corresponding authors.

Author contributions

This research is part of PP research thesis. Primarily advised by HC on the robotics end, and advised by AG on the cognitive system end.

Funding

The work was Funded by HC and Contextual Robotics Institute, UC San Diego—Funding *via* RPDC and Dr. Nahid Sidki for supporting part of this research.

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.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fphy.2022.975247/full#supplementary-material

References

1. IFR. *World robotics report*. Frankfurt, Germany: International Federation of Robotics (2020). [Internet]Available from: https://ifr.org/ifr-press-releases/news/record-2.7-million-robots-work-in-factories-around-the-globe.

2. Huckaby JO. *Knowledge transfer in robot manipulation tasks*. Atlanta, GA, United States: Georgia Institute of Technology (2014).

3. Huckaby JO, Christensen HI. A taxonomic framework for task modeling and knowledge transfer in manufacturing robotics. In: *Workshops at the twenty-sixth AAAI conference on artificial intelligence* (2012).

4. Huckaby J, Christensen H. Modeling robot assembly tasks in manufacturing using SysML. In: Proceeding of the ISR/Robotik 2014; 41st International Symposium on Robotics; June 2014; Munich, Germany. IEEE (2014). p. 1–7.

5. Huckaby J, Vassos S, Christensen HI. Planning with a task modeling framework in manufacturing robotics. In: Proceeding of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems; November 2013; Tokyo, Japan. IEEE (2013). p. 5787–94.

6. Wang R, Guan Y, Song H, Li X, Li X, Shi Z, et al. A formal model-based design method for robotic systems. *IEEE Syst J* (2019) 13(1):1096–107. doi:10.1109/jsyst.2018.2867285

7. IFR. *Advances in programming lower cost of adoption [Internet]*. Frankfurt, Germany: IFR International Federation of Robotics (2021). [cited 2022 Aug 21]. Available from: https://ifr.org/post/advances-in-programming-lower-cost-of-adoption.

8. Devin C, Abbeel P, Darrell T, Levine S. Deep object-centric representations for generalizable robot learning. In: Proceeding of the 2018 IEEE International Conference on Robotics and Automation (ICRA); September 2018. IEEE (2018). p. 7111–8.

9. Ghalamzan EAM, Paxton C, Hager GD, Bascetta L. An incremental approach to learning generalizable robot tasks from human demonstration. In: Proceeding of the 2015 IEEE International Conference on Robotics and Automation (ICRA); May 2015; Seattle, WA, USA. IEEE (2015). p. 5616–21.

10. Koert D, Maeda G, Lioutikov R, Neumann G, Peters J. Demonstration based trajectory optimization for generalizable robot motions. In: Proceeding of the 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids); November 2016; Cancun, Mexico. IEEE (2016). p. 515–22.

11. Fitzgerald T, Goel AK, Thomaz AL. Human-guided object mapping for task transfer. *ACM Trans Hum Robot Interact* (2018) 7(2):1–24. doi:10.1145/3277905

12. Fitzgerald T, Goel A, Thomaz A. Abstraction in data-sparse task transfer. *Artif Intelligence* (2021) 300:103551. doi:10.1016/j.artint.2021.103551

13. Wells AM, Dantam NT, Shrivastava A, Kavraki LE. Learning feasibility for task and motion planning in tabletop environments. *IEEE Robot Autom Lett* (2019) 4(2):1255–62. doi:10.1109/lra.2019.2894861

14. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, et al. *An image is worth 16x16 words: Transformers for image recognition at scale* (2020). arXiv preprint arXiv:201011929.

15. Park D, Hoshi Y, Kemp CC. A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *IEEE Robot Autom Lett* (2018) 3(3):1544–51. doi:10.1109/lra.2018.2801475

16. Nair S, Rajeswaran A, Kumar V, Finn C, Gupta A. *R3M: A universal visual representation for robot manipulation [internet]* (2022). arXiv; 2022 [cited 2022 Aug 21]. Available from: http://arxiv.org/abs/2203.12601.

17. Simeonov A, Du Y, Tagliasacchi A, Tenenbaum JB, Rodriguez A, Agrawal P, et al. *Neural descriptor fields: SE(3)-Equivariant object representations for manipulation* (2021). [cited 2022 May 10]; Available from: https://arxiv.org/abs/2112.05124v1.

18. Murali A, Liu W, Marino K, Chernova S, Gupta A. Same object, different grasps: Data and semantic knowledge for task-oriented grasping. In: *Conference on robot learning* (2020).

19. Manuelli L, Gao W, Florence P, Tedrake R. Kpam: KeyPoint Affordances for category-level robotic manipulation. In: T Asfour, E Yoshida, J Park, H Christensen, O Khatib, editors. *Robotics research*. Cham: Springer International Publishing (2022). p. 132–57. (Springer Proceedings in Advanced Robotics).

20. Cox MT. Perpetual self-aware cognitive agents. *AIMag* (2007) 28(1):32.

21. Cox MT. A model of planning, action and interpretation with goal reasoning. *Adv Cogn Syst* (2016) 5:57–76.

22. Cox MT, Alavi Z, Dannenhauer D, Eyorokon V, Muñoz-Avila H, Perlis D. Midca: A metacognitive, integrated dual-cycle architecture for self-regulated autonomy. *Proc AAAI Conf Artif Intelligence* (2016) 30:3712–8. doi:10.1609/aaai.v30i1.9886

23. Dannenhauer D, Muñoz-Avila H. Raising expectations in GDA agents acting in dynamic environments. In: Proceedings of the 24th International Conference on Artificial Intelligence; July 2015 (2015). p. 2241–7.

24. Dannenhauer D, Muñoz-Avila H, Cox MT. Informed expectations to guide GDA agents in partially observable environments. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence; July 2016 (2016). p. 2493–9.

25. Murdock JW, Goel AK. Meta-case-based reasoning: Self-improvement through self-understanding. *J Exp Theor Artif Intelligence* (2008) 20(1):1–36. doi:10.1080/09528130701472416

26. Murdock JW, Goel AK. *Self-improvement through self-understanding: Model-based reflection for agent self-adaptation*. Amazon (2011).

27. Stroulia E, Goel AK. Functional representation and reasoning for reflective systems. *Appl Artif Intelligence* (1995) 9(1):101–24. doi:10.1080/08839519508945470

28. Stroulia E, Goel AK. Evaluating PSMs in evolutionary design: The A UTOGNOSTIC experiments. *Int J human-computer Stud* (1999) 51(4):825–47. doi:10.1006/ijhc.1999.0331

29. Jones JK, Goel AK. Perceptually grounded self-diagnosis and self-repair of domain knowledge. *Knowledge-Based Syst* (2012) 27:281–301. doi:10.1016/j.knosys.2011.09.012

30. Goel AK, Rugaber S. Gaia: A CAD-like environment for designing game-playing agents. *IEEE Intell Syst* (2017) 32(3):60–7. doi:10.1109/mis.2017.44

31. Conitzer V, Sandholm T. *Definition and complexity of some basic metareasoning problems* (2003). arXiv:cs/0307017 [Internet]. [cited 2021 Feb 9]; Available from: http://arxiv.org/abs/cs/0307017.

32. Parashar P, Goel AK, Sheneman B, Christensen HI. Towards life-long adaptive agents: Using metareasoning for combining knowledge-based planning with situated learning. *Knowledge Eng Rev* (2018) 33:e24. doi:10.1017/s0269888918000279

33. Parashar P, Naik A, Hu J, Christensen HI. A hierarchical model to enable plan reuse and repair in assembly domains. In: Proceeding of the 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE); August 2021; Lyon, France. IEEE (2021). p. 387–94.

34. Christensen HI, Kruijff GJM, Wyatt JL *Cognitive systems*, Vol. 8. Springer Science & Business Media (2010).

35. Kortenkamp D, Simmons R, Brugali D. Robotic systems architectures and programming. In: *Springer handbook of robotics*. Springer (2016). p. 283–306.

36. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, et al. Ros: An open-source robot operating system. In: *ICRA workshop on open source software*, 3 (2009).

37. ROS-Industrial. *ROS-Industrial goals and background*.

38. Parashar P. *Using meta-reasoning for failure detection and recovery in assembly domain [internet]*. San Diego: University of California (2021). [cited 2022 May 11]. Available from: https://www.proquest.com/openview/9e3639b5696fdd09488ed817b5786710/1?pq-origsite=gscholar&cbl=18750&diss=y.

39. Beetz M, Mösenlechner L, Tenorth M. CRAM—a cognitive robot abstract machine for everyday manipulation in human environments. In: Proceeding of the 2010 IEEE/RSJ international conference on intelligent robots and systems; October 2010; Taipei, Taiwan. IEEE (2010). p. 1012–7.

40. Muñoz-Avila H, Cox MT. Case-based plan adaptation: An analysis and review. *IEEE Intell Syst* (2008) 23(4):75–81. doi:10.1109/mis.2008.59

41. Cox M, Raja A. *Metareasoning: A manifesto*. Cambridge, MA, United States: BBN Technical (2007).

42. Parashar P, Sheneman B, Goel AK. Adaptive agents in minecraft: A hybrid paradigm for combining domain knowledge with reinforcement learning. In: *International conference on autonomous agents and multiagent systems* (2017). p. 86–100.

43. Ulam P, Goel AK, Jones J, Murdock W. Using model-based reflection to guide reinforcement learning. *Reasoning, Representation, Learn Comp Games* (2005) 107.

44. Erol K, Hendler J, Nau DS. *HTN planning: Complexity and expressivity*. Seattle, WA, United States: AAAI (1994). p. 1123–8.

45. Nau DS, Cao Y, Lotem A, Munoz-Avila H. SHOP: Simple hierarchical ordered planner. In: Proceedings of the 16th international joint conference on artificial intelligence - volume 2; July 1999. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. (1999). p. 968–73. (IJCAI'99).

46. Nau DS, Au TC, Ilghami O, Kuter U, Murdock JW, Wu D, et al. SHOP2: An HTN planning system. *J Artif Intell Res* (2003) 20:379–404. doi:10.1613/jair.1141

47. Nau D, Au TC, Ilghami O, Kuter U, Wu D, Yaman F, et al. Applications of SHOP and SHOP2. *IEEE Intell Syst* (2005) 20(2):34–41. doi:10.1109/mis.2005.20

48. Georgievski I, Aiello M. *An overview of hierarchical task network planning* (2014). Mar 28 [cited 2020 May 16]; Available from: https://arxiv.org/abs/1403.7426v1.

49. Erol K, Hendler JA, Nau DS. *Semantics for hierarchical task-network planning*. College Park, MD, United States: Maryland Univ College Park Inst for Systems Research (1995).

50. Garrett CR, Chitnis R, Holladay R, Kim B, Silver T, Kaelbling LP, et al. Integrated task and motion planning. *Annu Rev Control Robot Auton Syst* (2021) 4(1):265–93. doi:10.1146/annurev-control-091420-084139

51. Kaelbling LP, Lozano-Pérez T. Hierarchical task and motion planning in the now. In: Proceeding of the 2011 IEEE international conference on robotics and automation; May 2011; Shanghai, China. IEEE (2011). p. 1470–7.

52. Jindal P, Roth D. Learning from negative examples in set-expansion. In: Proceeding of the 2011 IEEE 11th International Conference on Data Mining; December 2011; Vancouver, BC, Canada. IEEE (2011). p. 1110–5.

53. Sarmento L, Jijkuon V, de Rijke M, Oliveira E. "More like these": Growing entity classes from seeds. In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management - CIKM '07; January 2007; Lisbon, Portugal. New York, NY, United States: ACM Press (2007). [cited 2022 Aug 21]. p. 959. Available from: http://portal.acm.org/citation.cfm?doid=1321440.1321585.

54. Zhang X, Chen Y, Chen J, Du X, Wang K, Wen JR. Entity set expansion via knowledge graphs. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval; August 2017; New York, NY, United States: Association for Computing Machinery (2017). p. 1101–4. (SIGIR '17). Available from. doi:10.1145/3077136.3080732

55. Dornhege C, Eyerich P, Keller T, Brenner M, Nebel B. Integrating task and motion planning using semantic attachments. In: Proceedings of the 1st AAAI Conference on Bridging the Gap Between Task and Motion Planning. Washington, DC, United States: AAAI Press (2010). p. 10–7. (AAAIWS'10-01).

56. De Mello LH, Sanderson AC. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Int Conf robotics automation* (1989) 7:56–7.