Reactive Planning for Mobile Manipulation Tasks in Unexplored Semantic Environments

Vasileios Vasilopoulos*, Yiannis Kantaros*, George J. Pappas and Daniel E. Koditschek

***Abstract*— Complex manipulation tasks, such as rearrangement planning of numerous objects, are combinatorially hard problems. Existing algorithms either do not scale well or assume a great deal of prior knowledge about the environment, and few offer any rigorous guarantees. In this paper, we propose a novel hybrid control architecture for achieving such tasks with mobile manipulators. On the discrete side, we enrich a temporal logic specification with mobile manipulation primitives such as moving to a point, and grasping or moving an object. Such specifications are translated to an automaton representation, which orchestrates the physical grounding of the task to mobility or manipulation controllers. The grounding from the discrete to the continuous reactive controller is online and can respond to the discovery of unknown obstacles or decide to push out of the way movable objects that prohibit task accomplishment. Despite the problem complexity, we prove that, under specific conditions, our architecture enjoys provable completeness on the discrete side, provable termination on the continuous side, and avoids all obstacles in the environment. Simulations illustrate the efficiency of our architecture that can handle tasks of increased complexity while also responding to unknown obstacles or unanticipated adverse configurations.**

I. INTRODUCTION

Task and motion planning for complex manipulation tasks, such as rearrangement planning of multiple objects, has recently received increasing attention [1]–[3]. However, existing algorithms are typically combinatorially hard and do not scale well, while they also focus mostly on *known* environments [4], [5]. As a result, such methods cannot be applied to scenarios where the environment is initially unknown or needs to be reconfigured to accomplish the assigned mission and, therefore, online replanning may be required [6], resulting in limited applicability.

Instead, we propose an architecture for addressing complex mobile manipulation task planning problems which can handle unanticipated conditions in unknown environments. Fig. 1 illustrates the problem domain and scope of our algorithm. Fig. 2 illustrates the structure of the architecture and the organization of the paper. Our extensive simulation study suggests that this formal interface between a symbolic logic task planner and a physically grounded dynamical controller - each endowed with their own formal properties - can achieve computationally efficient rearrangement of complicated workspaces, motivating work now in progress to give conditions under which their interaction can guarantee success.

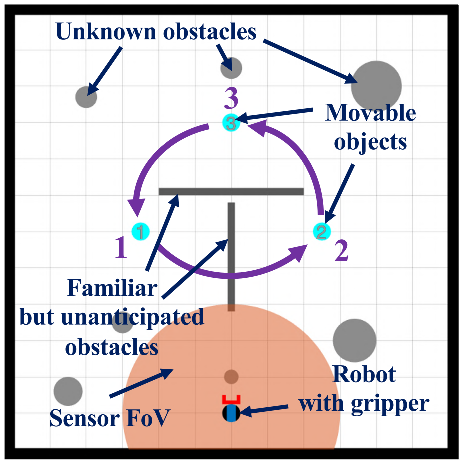

Fig. 1: An example of a task considered in this paper, whose execution is depicted in Fig. 5. A differential drive robot, equipped with a gripper (red) and a limited range onboard sensor for localizing obstacles (orange), needs to accomplish a mobile manipulation task specified by a Linear Temporal Logic (LTL) formula, in a partially known environment (black), cluttered with both unanticipated (dark grey) and completely unknown (light grey) fixed obstacles. Here the task is to rearrange the movable objects counterclockwise, in the presence of the fixed obstacles. Objects' abstract locations (relative to abstract, named regions of the workspace) are known by the symbolic controller both à-priori and during the entire task sequence. Geometrically complicated obstacles are assumed to be familiar but unanticipated in the sense that neither their number nor placement are known in advance. Completely unknown obstacles are presumed to be convex. All obstacles and disconnected configurations caused by the movable objects are handled by the reactive vector field motion planner (Fig. 2) and never reported to the symbolic controller.

A. Mobile Manipulation of Movable Objects

Planning the rearrangement of movable objects has long been known to be algorithmically hard (PSPACE-hardness was established in [8]), and most approaches have focused on simplified instances of the problem. For example, past work on reactive rearrangement using vector field planners such as navigation functions [9] assumes either that each object is actuated [10], [11] or that there are no other obstacles in the environment [12]–[14]. When considering more complicated workspaces, most approaches focus either on sampling-based methods that empirically work well [15], motivated by the typically high dimensional configuration spaces arising from combined task and motion planning [1], [3], or learning a symbolic language on the fly [16]. Such methods can achieve asymptotic optimality [17] by leveraging tools for efficient search on large graphs [18], but come with no guarantee of task completion under partial prior knowledge

* Equal contribution.

The authors are with the GRASP Lab, University of Pennsylvania, Philadelphia, PA 19104. {vvasilo, kantaros, pappasg, kod}@seas.upenn.edu

This work was supported by AFRL grant FA865015D1845 (subcontract 669737-1), AFOSR grant FA9550-19-1-0265 (Assured Autonomy in Contested Environments), and ONR grant #N00014-16-1-2817, a Vannevar Bush Fellowship held by the last author, sponsored by the Basic Research Office of the Assistant Secretary of Defense for Research and Engineering.

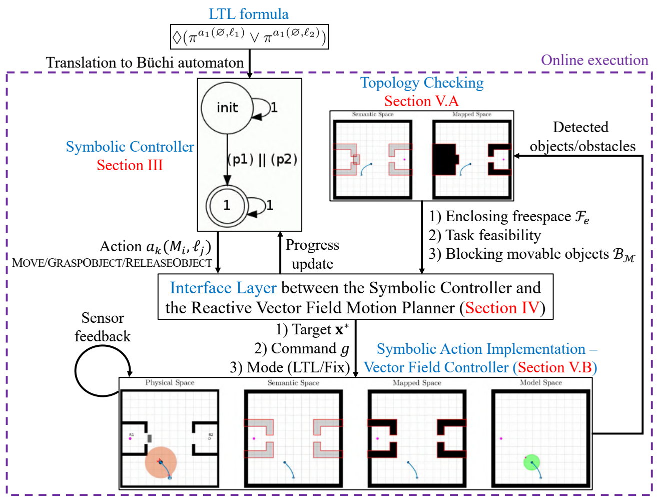
LTL formula
$\Diamond(\pi^{a_1(\varnothing,\ell_1)} \vee \pi^{a_1(\varnothing,\ell_2)})$
Translation to Büchi automaton
Online execution
Symbolic Controller
Section III
init
1
(p1) || (p2)
1
1
Topology Checking
Section V.A
Semantic Space
Mapped Space
Detected objects/obstacles
1) Enclosing freespace $\mathcal{F}_e$
2) Task feasibility
3) Blocking movable objects $\mathcal{B}_\mathcal{M}$
Action $a_k(M_i, \ell_j)$
MOVE/GRASPOBJECT/RELEASEOBJECT
Progress update
Interface Layer between the Symbolic Controller and the Reactive Vector Field Motion Planner (Section IV)
Sensor feedback
1) Target $\mathbf{x}^*$
2) Command g
3) Mode (LTL/Fix)
Symbolic Action Implementation – Vector Field Controller (Section V.B)
Physical Space
Semantic Space
Mapped Space
Model Space

Fig. 2: System architecture: The task is encoded in an LTL formula, translated offline to a Büchi automaton (symbolic controller - Section III). Then, during execution time in a previously unexplored semantic environment, each individual sub-task provided by the Büchi automaton is translated to a point navigation task toward a target $\mathbf{x}^*$ and a gripper command g, through an interface layer (Section IV). This task is executed online by realizing each symbolic action (Section V-B) using a reactive, vector field motion planner (continuous-time controller, [7]) implementing closed-loop navigation using sensor feedback and working closely with a topology checking module (Section V-A), responsible for detecting freespace disconnections. The reactive controller, nominally in an *LTL mode*, guarantees collision avoidance and target convergence when both the initial and the target configuration lie in the same freespace component. On the other hand, if the topology checking module determines that the target is not reachable, the reactive controller either attempts to connect the disconnected configuration space by switching to a *Fix mode* and interacting with the environment to rearrange blocking movable objects, or the interface layer reports failure to the symbolic controller when this is impossible and requests an alternative action.

and their search time grows exponentially with the number of movable pieces [19]. Sampling-based approaches have also been applied to the problem of navigation among movable obstacles (NAMO) [20], where the robot needs to grasp and move obstacles in order to connect disconnected components of the configuration space, with recent extensions focusing on heuristics for manipulation planning in unknown environments [21], [22]. Unlike such methods that require constant deliberative replanning in the presence of unanticipated conditions, this work examines the use of a reactive vector field controller, with simultaneous guarantees of convergence and obstacle avoidance in partially known environments [7], endowed with a narrow symbolic interface to the abstract reactive temporal logic planner whose freedom from any consideration of geometric details affords decisive computational advantage in supervising the task.

B. *Reactive Temporal Logic Planning*

Reactive temporal logic planning algorithms that can account for environmental uncertainty in terms of incomplete environment models have been developed in [23]–[30]. Particularly, [23], [24] model the environment as a transition system which is partially known. Then, a discrete controller is designed by applying graph search methods on a product automaton. As the environment, i.e., the transition system, is updated, the product automaton is locally updated as well, and new paths are re-designed by applying graph search approaches on the revised automaton. A conceptually similar approach is proposed in [25], [26]. The works in [27], [28] propose methods to locally patch paths, as the transition system (modeling the environment) changes so that GR(1) (General Reactivity of Rank 1) specifications [31] are satisfied. Reactive to LTL specifications planning algorithms are proposed in [29], [30], as well. Specifically, in [29], [30] the robot reacts to the environment while the task specification captures this reactivity. Correctness of these algorithms is guaranteed if the robot operates in an environment that satisfies the assumptions that were explicitly modeled in the task specification. Common in all these works is that, unlike our approach, they rely on discrete abstractions of the robot dynamics [32], [33] while active interaction with the environment to satisfy the logic specification is neglected.

C. *Contributions and Organization of the Paper*

This paper introduces the first planning and control architecture to provide a formal interface between an abstract temporal logic engine and a physically grounded mobile manipulation vector field planner for the rearrangement of movable objects in partially known workspaces cluttered with unknown obstacles. We provide conditions under which the symbolic controller is complete (Proposition 1), while exploiting prior results [7] that guarantee safe physical achievement of its sub-tasks when they are feasible, and introduce a new heuristic vector field controller for greedy physical rearrangement of the workspace when they are not. We provide a variety of simulation examples that illustrate the efficacy of the proposed algorithm for accomplishing complex manipulation tasks in unknown environments.

The paper is organized as follows. After formulating the problem in Section II, Section III presents a discrete controller which given an LTL specification generates on-the-fly high-level manipulation primitives, translated to point navigation commands through an interface layer outlined in Section IV. Using this interface, Section V continues with the reactive implementation of our symbolic actions and the employed algorithm for connecting disconnected freespace components blocked by movable objects. Section VI discusses our numerical results and, finally, Section VII concludes with ideas for future work.

II. PROBLEM DESCRIPTION

A. *Model of the Robot and the Environment*

We consider a first-order, nonholonomically-constrained, disk-shaped robot, centered at $\mathbf{x} \in \mathbb{R}^2$ with radius $r \in \mathbb{R}_{>0}$ and orientation $\psi \in S^1$; its rigid placement is denoted by $\overline{\mathbf{x}} := (\mathbf{x}, \psi) \in \mathbb{R}^2 \times S^1$ and its input vector $\overline{\mathbf{u}} := (v, \omega)$ consists of a fore-aft and an angular velocity command. The robot uses a gripper to move disk-shaped *movable objects* of known location, denoted by $\tilde{\mathcal{M}} := \{\tilde{M}_i\}_{i \in \{1,\dots,N_M\}}$, with a vector of radii $(\rho_1, \dots, \rho_{N_M}) \in \mathbb{R}^{N_M}$, in a closed, compact, polygonal, typically non-convex workspace $\mathcal{W} \subset \mathbb{R}^2$. The robot's gripper g can either be engaged ($g = 1$) or disengaged ($g = 0$). Moreover, we adopt the perceptual model of our recent physical implementations [7], [34] whereby a sensor of range R recognizes and instantaneously localizes any fixed "familiar" or "unfamiliar" obstacles; see also Fig. 1.

The workspace is cluttered by a finite collection of disjoint obstacles of unknown number and placement, denoted by $\tilde{\mathcal{O}}$. This set might also include non-convex "intrusions" of the boundary of the physical workspace $\mathcal{W}$ into the convex hull of the closure of the workspace $\mathcal{W}$, defined as the *enclosing workspace*. As in [7], [34], [35], we define the *freespace* $\mathcal{F}$ as the set of collision-free placements for the closed ball $\overline{\mathsf{B}(\mathbf{x}, r)}$ centered at $\mathbf{x}$ with radius r, and the *enclosing freespace*, $\mathcal{F}_e$, as $\mathcal{F}_e := \{\mathbf{x} \in \mathbb{R}^2 \,|\, \mathbf{x} \in \text{Conv}(\overline{\mathcal{F}})\}$.

Although none of the positions of any obstacles in $\tilde{\mathcal{O}}$ are à-priori known, a subset $\tilde{\mathcal{P}} \subseteq \tilde{\mathcal{O}}$ of these obstacles is assumed to be "familiar" in the sense of having a recognizable polygonal geometry, that the robot can instantly identify and localize (as we have implemented in [7], [34]). The remaining obstacles in $\tilde{\mathcal{C}} := \tilde{\mathcal{O}} \backslash \tilde{\mathcal{P}}$ are assumed to be strongly convex according to [35, Assumption 2] (and implemented in [7], [34]), but are otherwise completely unknown.

To simplify the notation, we dilate each obstacle and movable object by r (or $r + \rho_i$ when the robot carries an object i), and assume that the robot operates in the freespace $\mathcal{F}$. We denote the set of dilated objects and obstacles derived from $\tilde{\mathcal{M}}, \tilde{\mathcal{O}}, \tilde{\mathcal{P}}$ and $\tilde{\mathcal{C}}$, by $\mathcal{M}, \mathcal{O}, \mathcal{P}$ and $\mathcal{C}$ respectively. For our formal results, we assume that each obstacle in $\mathcal{C}$ is always well-separated from all other obstacles in both $\mathcal{C}$ and $\mathcal{P}$, as outlined in [7, Assumption 1]; in practice, the surrounding environment often violates our separation assumptions, without precluding successful task completion.

B. Specifying Complex Manipulation Tasks

The robot needs to accomplish a mobile manipulation task, by visiting known regions of interest $\ell_j \subseteq \mathcal{W}$, where $j \in \{1, \ldots, L\}$, for some $L > 0$, and applying one of the following three manipulation actions $a_k(M_i, \ell_j) \in \mathcal{A}$, with $M_i \in \mathcal{M}$ referring to a movable object, defined as follows:

- MOVE(ℓ_j) instructing the robot to move to region ℓ_j, labeled as $a_1(\varnothing, \ell_j)$, where $\varnothing$ means that this action does not logically entail interaction with any specific movable object[1].
- GRASPOBJECT(M_i) instructing the robot to grasp the movable object M_i, labeled as $a_2(M_i, \varnothing)$, with $\varnothing$ denoting that no region is associated with this action.
- RELEASEOBJECT(M_i, ℓ_j) instructing the robot to push the (assumed already grasped) object M_i toward its designated goal position, ℓ_j, labeled as $a_3(M_i, \ell_j)$.

For instance, consider a rearrangement planning scenario where the locations of three objects of interest need to be rearranged, as in Fig. 1. We capture such complex manipulation tasks via Linear Temporal Logic (LTL) specifications. Specifically, we use atomic predicates of the form $\pi^{a_k(M_i, \ell_j)}$, which are true when the robot applies the action $a_k(M_i, \ell_j)$ and false until the robot achieves that action. Note that these atomic predicates allow us to specify temporal logic specifications defined over manipulation primitives and, unlike related works [5], [36], are entirely agnostic to the geometry of the environment. We define LTL formulas by collecting such predicates in a set $\mathcal{AP}$ of atomic propositions. For example, the rearrangement planning scenario with three movable objects initially located in regions ℓ_1, ℓ_2, and ℓ_3, as shown in Fig. 1, can be described as a sequencing task [37] by the following LTL formula:

$$\begin{aligned}\phi = &\Diamond(\pi^{a_2(M_1,\varnothing)} \wedge \Diamond(\pi^{a_3(M_1,\ell_2)} \wedge \\ &\Diamond(\pi^{a_2(M_2,\varnothing)} \wedge \Diamond\pi^{a_3(M_2,\ell_3)} \wedge \\ &\Diamond(\pi^{a_2(M_3,\varnothing)} \wedge \Diamond\pi^{a_3(M_3,\ell_1)})))) \qquad (1)\end{aligned}$$

where $\Diamond$ and $\wedge$ refer to the 'eventually' and 'AND' operator. In particular, this task requires the robot to perform the following steps in this order (i) grasp object M_1 and release it in location ℓ_2 (first line in (1)); (ii) then grasp object M_2 and release it in location ℓ_3 (second line in (1)); (iii) grasp object M_3 and release it in location ℓ_1 (third line in (1)). LTL formulas are satisfied over an infinite sequence of states [38]. Unlike related works where a state is defined to be the robot position, e.g., [29], here a state is defined by the manipulation action $a_k(M_i, \ell_j)$ that the robot applies. In other words, an LTL formula defined over manipulation-based predicates $\pi^{a_k(M_i, \ell_j)}$ is satisfied by an infinite sequence of actions $p = p_0, p_1, \ldots, p_n, \ldots$, where $p_n \in \mathcal{A}$, for all $n \geq 0$ [38]. Given a sequence p, the syntax and semantics of LTL can be found in [38]; hereafter, we exclude the 'next' operator from the syntax, since it is not meaningful for practical robotics applications [39], as well as the negation operator[2].

C. Problem Statement

Given a task specification captured by an LTL formula ϕ, our goal is to (i) generate online, as the robot discovers the environment via sensor feedback, appropriate actions using the (discrete) symbolic controller, (ii) translate them to point navigation tasks, (iii) execute these navigation tasks and apply the desired manipulation actions with a (continuous-time) vector field controller, while avoiding unknown and familiar obstacles, (iv) be able to online detect freespace disconnections that prohibit successful action completion, and (v) either locally amend the provided plan by disassembling blocking movable objects, or report failure to the symbolic controller and request an alternative action.

III. SYMBOLIC CONTROLLER

In this Section, we design a discrete controller that generates manipulation commands online in the form of the actions defined in Section II (see Fig. 2), and describe the manner in which this symbolic controller extends prior work [40] to account for manipulation-based atomic predicates. A detailed construction is included in [41, Appendix I].

[1]Although, as will be detailed in Section V, the hybrid reactive controller may actually need to move objects out of the way, rearranging the topology of the workspace in a manner hidden from the logical task controller.

[2]Since the negation operator is excluded, safety requirements, such as obstacle avoidance, cannot be captured by the LTL formula; nevertheless, the proposed method can still handle safety constraints by construction of the (continuous-time) reactive, vector field controller in Section V.

A. Construction of the Symbolic Controller

First, we translate the specification ϕ, constructed using a set of atomic predicates $\mathcal{AP}$, into a Non-deterministic Büchi Automaton (NBA) with state-space and transitions among states that can be used to measure how much progress the robot has made in terms of accomplishing the assigned mission; see [41, Appendix I]. Particularly, we define a distance metric over this NBA state-space to compute how far the robot is from accomplishing its assigned task, or more formally, from satisfying the accepting condition of the NBA, by following a similar analysis as in [40], [42]. This metric is used to generate manipulation commands online as the robot navigates the unknown environment. The main idea is that, given its current NBA state, the robot should reach a next NBA state that decreases the distance to a state that accomplishes the assigned task. Once this target NBA state is selected, a symbolic action that achieves it is generated, in the form of a manipulation action presented in Section II (e.g., 'release the movable object M_i in region ℓ_j'). This symbolic action acts as an input to the reactive, continuous-time controller (see Section V), that handles obstacles and unanticipated conditions.

When the assigned sub-task is accomplished, a new target automaton state is selected and a new manipulation command is generated. If the continuous-time controller fails to accomplish the sub-task (because e.g., a target is surrounded by fixed obstacles), the symbolic controller checks if there exists an alternative command that ensures reachability of the target automaton state (e.g., consider a case where a given NBA state can be reached if the robot goes to either region ℓ_1 or ℓ_2). If there are no alternative commands to reach the desired automaton state, then a new target automaton state that also decreases the distance to satisfying the accepting NBA condition is selected. If there are no such other automaton states, a message is returned stating that the robot cannot accomplish the assigned mission. A detailed description for the construction of this distance metric is provided in [41, Appendix I].

B. Completeness of the Symbolic Controller

Here, we provide conditions under which the proposed discrete controller is complete. The proof for the following Proposition can be found in [41, Appendix I].

Proposition 1 (Completeness) *Assume that there exists at least one infinite sequence of manipulation actions in the set $\mathcal{A}$ that satisfies ϕ. If the environmental structure and the continuous-time controller always ensure that at least one of the candidate next NBA states can be reached, then the proposed discrete algorithm is complete, i.e., a feasible solution will be found.*[3]

[3]Given the current NBA state, denoted by $q_B(t)$, the symbolic controller selects as the next NBA state, a state that is reachable from $q_B(t)$ and closer to the final states as per the proposed distance metric; see also [41, Appendix I]. All NBA states that satisfy this condition are called candidate next NBA states. Also, reaching an NBA state means that at least one of the manipulation actions required to enable the transition from $q_B(t)$ to the next NBA state is feasible.

IV. INTERFACE LAYER BETWEEN THE SYMBOLIC AND THE REACTIVE CONTROLLER

We assume that the robot is nominally in an *LTL mode*, where it executes sequentially the commands provided by the symbolic controller described in Section III. We use an *interface layer* between the symbolic controller and the reactive motion planner, as shown in Fig. 2, to translate each action to an appropriate gripper command ($g = 0$ for MOVE and GRASPOBJECT, and $g = 1$ for RELEASEOBJECT), and a navigation command toward a target $\mathbf{x}^*$. If the provided action is $\textsc{Move}(\ell_j)$ or $\textsc{ReleaseObject}(M_i, \ell_j)$, we pick as $\mathbf{x}^*$ the centroid of region ℓ_j. If the action is $\textsc{GraspObject}(M_i)$, we pick as $\mathbf{x}^*$ a collision-free location on the boundary of object M_i, contained in the freespace $\mathcal{F}$.

Consider again the example shown in Fig. 1. The first step of the assembly requires the robot to move object M_1 to ℓ_2 which, however, is occupied by the object M_2. In this case, instead of reporting that the assigned LTL formula cannot be satisfied, we allow the robot to temporarily pause the command execution from the symbolic controller, switch to a *Fix mode* and push object M_2 away from ℓ_1, before resuming the execution of the action instructed by the symbolic controller. For plan fixing purposes, we introduce a fourth action, $\textsc{DisassembleObject}(M_i, \mathbf{x}^*)$, invisible to the symbolic controller, instructing the robot to push the object M_i (after it has been grasped using GRASPOBJECT) toward a position $\mathbf{x}^*$ on the boundary of the freespace until specific separation conditions are satisfied. Hence, an additional responsibility of the interface layer (when in Fix mode) is to pick the next object to be grasped and disassembled from a stack of blocking movable objects $\mathcal{B}_\mathcal{M}$, as well as the target $\mathbf{x}^*$ of each DISASSEMBLEOBJECT action, until the stack $\mathcal{B}_\mathcal{M}$ becomes empty[4]; see Section V-B. Note that if the robot executes a RELEASEOBJECT action before switching to the Fix mode, the interface layer instructs it to first disassemble the carried object M_i and, after disassembling all objects in $\mathcal{B}_\mathcal{M}$, switch back to the LTL mode by re-grasping M_i.

Finally, the interface layer (a) requests a new action from the symbolic controller, if the robot successfully completes the current action execution, or (b) reports that the currently executed action is infeasible and requests an alternative action, if the topology checking module outlined in Section V-A determines that the target is surrounded by fixed obstacles.

V. SYMBOLIC ACTION IMPLEMENTATION

In this Section, we describe the online implementation of our symbolic actions, assuming that the robot has already picked a target $\mathbf{x}^*$ using the interface layer from Section IV. As reported above, in the LTL mode, the robot executes commands from the symbolic controller, using one of the actions MOVE, GRASPOBJECT and RELEASEOBJECT. The robot exits the LTL mode and enters the Fix mode when one or more movable objects block the target destination

[4]The exclusion of the negation operator from the LTL syntax, as assumed in Section II, guarantees that each DISASSEMBLEOBJECT action will *not* interfere with the satisfaction of the LTL formula ϕ, e.g., the robot will not disassemble an object that should not be grasped or moved.

$\mathbf{x}^*$; in this mode, it attempts to rearrange blocking movable objects using a sequence of the actions GRASPOBJECT and DISASSEMBLEOBJECT, before returning to the LTL mode.

The "backbone" of the symbolic action implementation is the reactive, vector field motion planner from prior work [7], allowing either a fully actuated or a differential-drive robot to provably converge to a designated fixed target while avoiding all obstacles in the environment. When the robot is gripping an object i, we use the method from [43] for generating virtual commands for the center $\mathbf{x}_{i,c}$ of the circumscribed disk with radius $(\rho_i + r)$, enclosing the robot and the object. Namely, we assume that the robot-object pair is a fully actuated particle with dynamics $\dot{\mathbf{x}}_{i,c} = \mathbf{u}_{i,c}(\mathbf{x}_{i,c})$, design our control policy $\mathbf{u}_{i,c}$ using the same vector field controller, and translate to commands $\overline{\mathbf{u}} = (v, \omega)$ for our differential drive robot as $\overline{\mathbf{u}} := \mathbf{T}_{i,c}(\psi)^{-1}\,\mathbf{u}_{i,c}$, with $\mathbf{T}_{i,c}(\psi)$ the Jacobian of the gripping contact, i.e., $\dot{\mathbf{x}}_{i,c} = \mathbf{T}_{i,c}(\psi)\,\overline{\mathbf{u}}$.

This reactive controller assumes that a path to the goal always exists (i.e., the robot's freespace is path-connected), and does not consider cases where the target is blocked either by a fixed obstacle or a movable object[5]. Hence, here we extend the algorithm's capabilities by providing a topology checking algorithm (Section V-A) that detects blocking movable objects or fixed obstacles, as outlined in Fig. 2. Based on these capabilities, Section V-B describes our symbolic action implementations. Appendix II in [41] includes a brief overview of the reactive, vector field motion planner, and Appendix III in [41] includes an algorithmic outline of our topology checking algorithm.

A. *Topology Checking Algorithm*

The topology checking algorithm is used to detect freespace disconnections, update the robot's enclosing freespace $\mathcal{F}_e$, and modify its action by switching to the Fix mode, if necessary. In summary, the algorithm's input is the initially assumed polygonal enclosing freespace $\mathcal{F}_e$ for either the robot or the robot-object pair, along with all known dilated movable objects in $\mathcal{M}$ and fixed obstacles in $\mathcal{P}_{\mathcal{I}}$ (corresponding to the index set $\mathcal{I}$ of localized familiar obstacles). The algorithm's output is the detected enclosing freespace $\mathcal{F}_e$, used for the diffeomorphism construction in the reactive controller [7], along with a stack of *blocking movable objects* $\mathcal{B}_{\mathcal{M}}$ and a Boolean indication of whether the current symbolic action is feasible. Based on this output, the robot switches to the Fix mode when the stack $\mathcal{B}_{\mathcal{M}}$ becomes non-empty, and resumes execution from the symbolic controller once all movable objects in $\mathcal{B}_{\mathcal{M}}$ are disassembled. A detailed description is given in [41, Appendix III].

B. *Action Implementation*

We are now ready to describe the used symbolic actions. The symbolic action MOVE(ℓ_j) simply uses the reactive controller to navigate to the selected target $\mathbf{x}^*$, as described in [41, Appendix II]. Similarly, the symbolic action GRASPOBJECT(M_i) uses the reactive controller to navigate

[5]The possibility of an entirely unknown blocking convex obstacle is precluded by our separation assumptions in Section II.

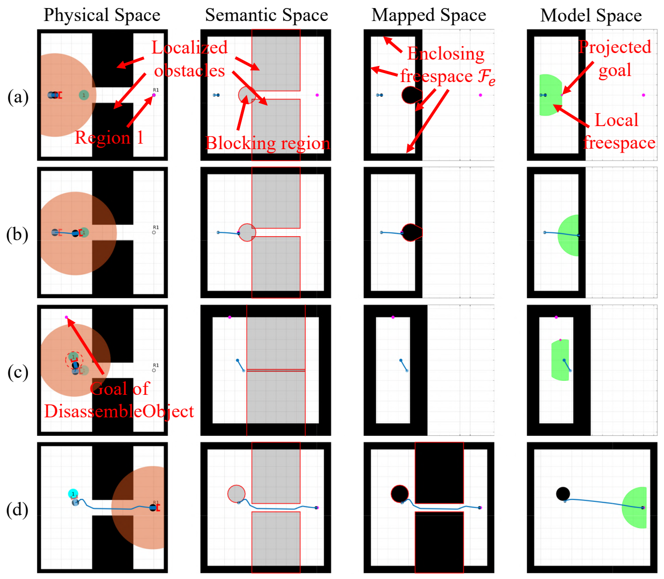

Fig. 3: Demonstration of local LTL plan fixing, where the task is to navigate to region 1, captured by the LTL formula $\phi = \Diamond\pi^{a_1(\varnothing,\ell_1)}$ where ℓ_1 refers to region 1 in the figure. (a) The robot starts navigating to its target, until it localizes the two rectangular obstacles and recognizes that the only path to the goal is blocked by a movable object. (b) The robot switches to the Fix mode, grips the object, and (c) moves it away from the blocking region, until the separation assumptions outlined in Section V-B are satisfied. (d) It then proceeds to complete the task.

to a collision-free location on the boundary of object M_i, and then aligns the robot so that its gripper faces the object, in order to get around Brockett's condition [44]. RELEASEOBJECT(M_i, ℓ_j) uses the reactive controller to design inputs for the robot-object center $\mathbf{x}_{i,c}$ and translates them to differential drive commands through the center's Jacobian $\mathbf{T}_{i,c}(\psi)$, in order to converge to the goal $\mathbf{x}^*$.

Finally, the action DISASSEMBLEOBJECT$(M_i, \mathbf{x}^*)$ is identical to RELEASEOBJECT, with two important differences. First, we heuristically select as target $\mathbf{x}^*$ the middle point of the edge of the polygonal freespace $\mathcal{F}$ that maximizes the distance to all other movable objects (except M_i) and all regions of interest ℓ_j. Second, in order to accelerate performance and shorten the resulting trajectories, we stop the action's execution if the robot-object pair, centered at $\mathbf{x}_{i,c}$ does not intersect any region of interest and the distance of $\mathbf{x}_{i,c}$ from all other objects in the workspace is at least $2(r + \max_{k\in\mathcal{B}_{\mathcal{M}}} \rho_k)$, as this would imply that dropping the object in its current location would not block a next step of the disassembly process. Even though we do not yet report on formal results pertaining to the task sequence in the Fix mode, the DISASSEMBLEOBJECT action maintains formal results of obstacle avoidance and target convergence to a feasible $\mathbf{x}^*$, using our reactive, vector field controller.

VI. ILLUSTRATIVE SIMULATIONS

In this Section, we implement simulated examples of different tasks in various environments using our architecture, shown in Fig. 2. All simulations were run in MATLAB using `ode45`, leveraging and enhancing existing presentation

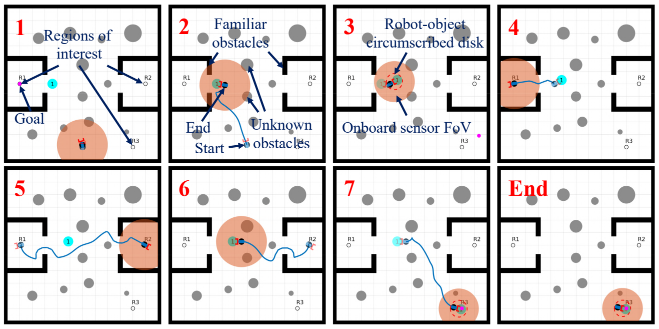

Fig. 4: Executing the LTL formula $\phi = \Diamond(\pi^{a_1(\varnothing,\ell_1)} \wedge \Diamond(\pi^{a_1(\varnothing,\ell_2)} \wedge \Diamond(\pi^{a_2(M_1,\varnothing)} \wedge \Diamond\pi^{a_3(M_1,\ell_3)})))$ in an environment cluttered with known walls (black) and unknown convex obstacles (grey).

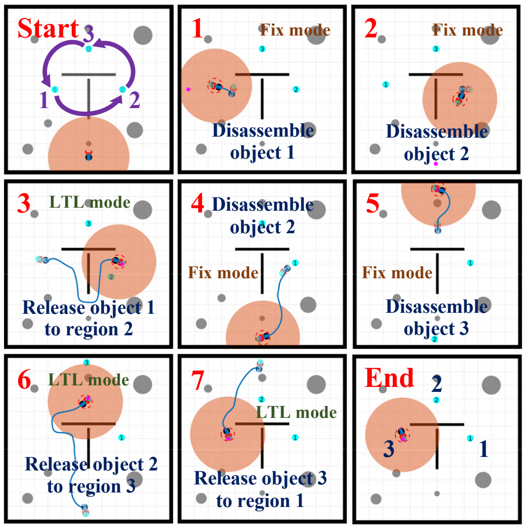

Fig. 5: An illustrative execution of the problem depicted in Fig. 1. The task, specified by the LTL formula (1), requires the counterclockwise rearrangement of 3 objects in an environment cluttered with some unanticipated familiar (initially dark grey and then black upon localization) and some completely unknown (light grey) fixed obstacles.

infrastructure[6]. The discrete controller and the interface layer are implemented in MATLAB, whereas the reactive controller is implemented in Python and communicates with MATLAB using the standard MATLAB-Python interface. For our numerical results, we assume perfect robot state estimation and localization of obstacles using the onboard sensor, which can instantly identify and localize either the entirety of familiar obstacles or fragments of unknown obstacles within its range[7]. The reader is referred to the accompanying video submission for visual context and additional simulations.

A. Demonstration of Local LTL Plan Fixing

Fig. 3 includes a demonstration of a simple task, encoded in the LTL formula $\phi = \Diamond\pi^{a_1(\varnothing,\ell_1)}$, i.e., eventually execute the action Move to navigate to region 1, demonstrating how the Fix mode for local rearrangement of blocking movable objects works.

[6]See `https://github.com/KodlabPenn/semnav_matlab`.

[7]The reader is referred to [7], [34] for examples of physical realization of such sensory assumptions, using a combination of an onboard camera for obstacle recognition and pose estimation, and a LIDAR sensor for extracting distance to nearby obstacles.

B. Executing More Complex LTL Tasks

Fig. 4 includes successive snapshots of a more complicated LTL task, captured by the formula

$$\phi = \Diamond(\pi^{a_1(\varnothing,\ell_1)} \wedge \Diamond(\pi^{a_1(\varnothing,\ell_2)} \wedge \Diamond(\pi^{a_2(M_1,\varnothing)} \wedge \Diamond\pi^{a_3(M_1,\ell_3)})))$$

which instructs the robot to first navigate to region 1, then navigate to region 2, and finally grasp object 1 and move it to region 3, in an environment cluttered with both familiar non-convex and completely unknown convex obstacles. Before navigating to region 1, the robot correctly identifies that the movable object disconnects its freespace and proceeds to disassemble it. After visiting region 2, it then revisits the movable object, grasps it and moves it to the designated location to complete the required task. The reader is referred to the video submission for visual context regarding the evolution of all planning spaces [7] (semantic, mapped and model space) during the execution of this task, as well as several other simulations with more movable objects, including (among others) a task where the robot needs to patrol between some predefined regions of interest in an environment cluttered with obstacles by visiting each one of them infinitely often.

C. Execution of Rearrangement Tasks

Finally, a promising application of our reactive architecture concerns rearrangement planning with multiple movable pieces. Traditionally, such tasks are executed using sampling-based planners, whose offline search times can blow up exponentially with the number of movable pieces in the environment (see, e.g., [19, Table I]). Instead, as shown in Fig. 5, the persistent nature of our reactive architecture succeeds in achieving the given task online in an environment with multiple obstacles, even though our approach might require more steps and longer trajectories in the overall assembly process than other optimal algorithms [17]. Moreover, the LTL formulas for encoding such tasks are quite simple to write (see (1) for the example in Fig. 5), instructing the robot to grasp and release each object in sequence; the reactive controller is capable of handling obstacles and blocking objects during execution time. Our video submission includes a rearrangement example with 4 movable objects, requiring more steps in the assembly process.

VII. Conclusion

In this paper, we propose a novel hybrid control architecture for achieving complex tasks with mobile manipulators in the presence of unanticipated obstacles and conditions. Future work will focus on providing end-to-end (instead of component-wise) correctness guarantees, extensions to multiple robots for collaborative manipulation tasks, as well as physical experimental validation.

References

[1] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "Sampling-based methods for factored task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1796–1825, 2018.

[2] W. Vega-Brown and N. Roy, "Asymptotically optimal planning under piecewise-analytic constraints," in *The 12th International Workshop on the Algorithmic Foundations of Robotics*, 2016.

[3] A. Krontiris and K. Bekris, "Dealing with difficult instances of object rearrangement," in *Robotics: Science and Systems*, 2015.

[4] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 639–646.

[5] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Towards manipulation planning with temporal logic specifications," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 346–352.

[6] L. P. Kaelbling and T. Lozano-Perez, "Hierarchical task and motion planning in the now," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1470–1477.

[7] V. Vasilopoulos, G. Pavlakos, S. L. Bowman, J. D. Caporale, K. Daniilidis, G. J. Pappas, and D. E. Koditschek, "Reactive Semantic Planning in Unexplored Semantic Environments Using Deep Perceptual Feedback," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4455–4462, 2020.

[8] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, "On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE-Hardness of the "Warehouseman's Problem"," *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 76–88, 1984.

[9] E. Rimon and D. E. Koditschek, "Exact Robot Navigation Using Artificial Potential Functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[10] L. L. Whitcomb, D. E. Koditschek, and J. B. D. Cabrera, "Toward the Automatic Control of Robot Assembly Tasks via Potential Functions: The Case of 2-D Sphere Assemblies," in *IEEE International Conference on Robotics and Automation*, 1992, pp. 2186–2192.

[11] C. S. Karagoz, H. I. Bozma, and D. E. Koditschek, "Coordinated Navigation of Multiple Independent Disk-Shaped Robots," *IEEE Transactions on Robotics*, vol. 30, no. 6, p. 1289–1304, December 2014.

[12] H. Isil Bozma and D. E. Koditschek, "Assembly as a noncooperative game of its pieces: analysis of 1D sphere assemblies," *Robotica*, vol. 19, no. 01, pp. 93–108, 2001.

[13] C. S. Karagoz, H. I. Bozma, and D. E. Koditschek, "Feedback-based event-driven parts moving," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1012–1018, 2004.

[14] O. Arslan, D. P. Guralnik, and D. E. Koditschek, "Coordinated robot navigation via hierarchical clustering," *IEEE Transactions on Robotics*, vol. 32, no. 2, p. 352–371, 2016.

[15] J. van den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha, "Path planning among movable obstacles: A probabilistically complete approach," in *The 8th International Workshop on the Algorithmic Foundations of Robotics*, 2010.

[16] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018.

[17] W. Vega-Brown and N. Roy, "Admissible abstractions for near-optimal task and motion planning," in *International Joint Conference on Artificial Intelligence*, 2018.

[18] J. Wolfe, B. Marthi, and S. Russell, "Combined task and motion planning for mobile manipulation," in *International Conference on Automated Planning and Scheduling*, 2010.

[19] W. Vega-Brown and N. Roy, "Anchoring abstractions for near-optimal task and motion planning," in *RSS Workshop on Integrated Task and Motion Planning*, 2017.

[20] M. Stilman and J. Kuffner, "Planning Among Movable Obstacles with Artificial Constraints," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1295–1307, 2008.

[21] H.-N. Wu, M. Levihn, and M. Stilman, "Navigation Among Movable Obstacles in unknown environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1433–1438.

[22] M. Levihn, J. Scholz, and M. Stilman, "Planning with movable obstacles in continuous environments with uncertain dynamics," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 3832–3838.

[23] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Revising motion planning under linear temporal logic specifications in partially known workspaces," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 5025–5032.

[24] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[25] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal motion planning for hybrid systems in partially unknown environments," in *The 16th International Conference on Hybrid Systems: Computation and Control*, 2013, pp. 353–362.

[26] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 583–599, 2016.

[27] S. C. Livingston, R. M. Murray, and J. W. Burdick, "Backtracking temporal logic synthesis for uncertain environments," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 5163–5170.

[28] S. C. Livingston, P. Prabhakar, A. B. Jose, and R. M. Murray, "Patching task-level robot controllers based on a local μ-calculus formula," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 4588–4595.

[29] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[30] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit, "Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles," *Autonomous Robots*, vol. 42, no. 4, pp. 801–824, 2018.

[31] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive (1) designs," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, 2006, pp. 364–380.

[32] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 864–874, 2005.

[33] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.

[34] V. Vasilopoulos, G. Pavlakos, K. Schmeckpeper, K. Daniilidis, and D. E. Koditschek, "Reactive Navigation in Partially Familiar Planar Environments Using Semantic Perceptual Feedback," *Under review, arXiv: 2002.08946*, 2020.

[35] O. Arslan and D. E. Koditschek, "Sensor-Based Reactive Navigation in Unknown Convex Sphere Worlds," *The International Journal of Robotics Research*, vol. 38, no. 1-2, pp. 196–223, July 2018.

[36] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "SMC: Satisfiability Modulo Convex Programming," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1655–1679, 2018.

[37] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *44th IEEE Conference on Decision and Control, European Control Conference, (CDC-ECC)*, Seville, Spain, 2005, pp. 4885–4890.

[38] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, Cambridge, MA, 2008.

[39] M. Kloetzer and C. Belta, "A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[40] Y. Kantaros, M. Malencia, V. Kumar, and G. J. Pappas, "Reactive temporal logic planning for multiple robots in unknown environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 11 479–11 485.

[41] V. Vasilopoulos, Y. Kantaros, G. J. Pappas, and D. E. Koditschek, "Technical Report: Reactive Planning for Mobile Manipulation Tasks in Unexplored Semantic Environments," in *arXiv: 2011.00642*, 2020.

[42] Y. Kantaros and M. M. Zavlanos, "Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 812–836, 2020.

[43] V. Vasilopoulos, W. Vega-Brown, O. Arslan, N. Roy, and D. E. Koditschek, "Sensor-Based Reactive Symbolic Planning in Partially

Known Environments," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 5683–5690.

[44] R. W. Brockett, "Asymptotic stability and feedback stabilization," in *Differential Geometric Control Theory*. Birkhauser, 1983, pp. 181–191.