A Hierarchical Deliberative-Reactive System Architecture for Task and Motion Planning in Partially Known Environments

Vasileios Vasilopoulos[1], Sebastian Castro[1], William Vega-Brown[2], Daniel E. Koditschek[3], Nicholas Roy[1]

***Abstract*— We describe a task and motion planning architecture for highly dynamic systems that combines a domain-independent sampling-based deliberative planning algorithm with a global reactive planner. We leverage the recent development of a reactive, vector field planner that provides guarantees of reachability to large regions of the environment even in the face of unknown or unforeseen obstacles. The reachability guarantees can be formalized using contracts that allow a deliberative planner to reason purely in terms of those contracts and synthesize a plan by choosing a sequence of reactive behaviors and their target configurations, without evaluating specific motion plans between targets. This reduces both the search depth at which plans will be found, and the number of samples required to ensure a plan exists, while crucially preserving correctness guarantees. The result is reduced computational cost of synthesizing plans, and increased robustness of generated plans to actuator noise, model misspecification, or unknown obstacles. Simulation studies show that our hierarchical planning and execution architecture can solve complex navigation and rearrangement tasks, even when faced with narrow passageways or incomplete world information.**

I. Introduction

A. Motivation

In this work, we consider a setting in which a highly energetic quadrupedal robot, capable of behaviors like walking, trotting and jumping, is assigned mobile manipulation tasks in an environment cluttered with fixed obstacles and movable objects (see Fig. 2). Solving these tasks requires planning and execution of dynamical pedipulation (nonprehensile manipulation of the environment using general purpose legs) [1] as well as navigation amidst clutter.

Developing computationally and physically viable solutions for these scenarios is challenging, even assuming a deterministic robot in a fully observable world (e.g., PSPACE hardness of the Warehouseman's problem was established in [2]), and it has been well-understood for many years that hierarchical abstractions [3] are required to address the fundamental complexity of such task and motion planning (TAMP) problems [4]. However, when using hierarchical abstract planners, it is difficult to ensure the correctness of the resulting plan, unless every motion primitive in the trajectory is checked for feasibility during planning, significantly impacting the overall computational cost.

[1]Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT, Cambridge, MA 02139 {vvasilo,scastro, nickroy}@csail.mit.edu.

[2]Tagup, Inc. will@tagup.io.

[3]GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104 kod@seas.upenn.edu.

This work was supported by AFOSR grant FA9550-19-1-0265, the ARL/GDRS RCTA project, Coop. Agreement #W911NF-10-2-0016, and the Toyota Research Institute Award LP-C000765-SR.

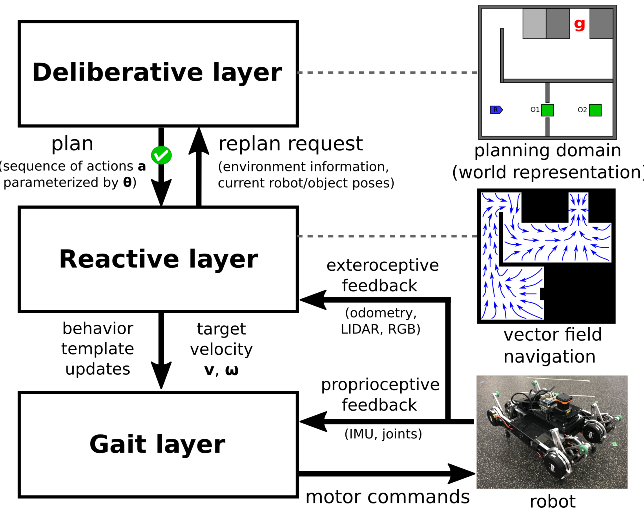


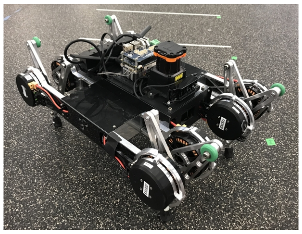
Fig. 1: **Our proposed system architecture**. Given a mobile manipulation task, a *deliberative layer* searches for a sequence of abstract actions, or a *plan*, using *contracts* that describe the reachability guarantees of a global *reactive layer*, which in turn implements these actions and guarantees collision avoidance in complex environments. The reactive layer transmits template commands (such as target velocity or grasping commands) to a *gait layer* that executes high-rate feedback to achieve parameterized steady-state or transitional behaviors on the robot. This architecture allows the deliberative layer to reason about sequencing actions without constructing explicit trajectories through the configuration space, improving computational efficiency while preserving probabilistic completeness.

This paper shows how a deliberative layer and reactive layer can create abstract plans that are correct-by-construction through the use of continuous constraint contracts (C3) between a deliberative and reactive layer [6]. For the reactive layer, we adapt a reactive, vector field planner from our prior work [7] that not only guarantees collision-free convergence to targets but is also robust to environmental uncertainty, even in the presence of unanticipated obstacles.

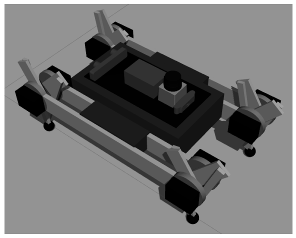
B. Contributions

The contribution of this paper is a hierarchical planning system, shown in Fig. 1, that has the properties of achieving the computational efficiency seen in many task and motion planning approaches, while preserving guarantees of probabilistic completeness that are often sacrificed for computational gains. Our planning system uses the guarantees of an online, vector-field-based reactive layer to define *action contracts*, such as the reachability of target poses, that can be used by an offline deliberative layer. The contracts of each action provide the deliberative planner with knowledge of each action's basin of attraction, allowing it to reason about the effects of sequencing actions without constructing explicit trajectories through configuration space.

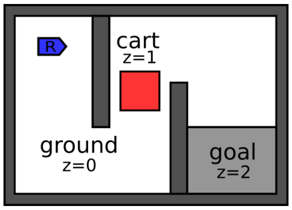
Focusing on the example of a quadruped robot navigating in an environment with static and movable obstacles, we



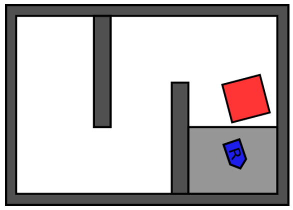
(a) The Minitaur platform.



(b) Simulated Minitaur.



(c) Simple planning problem.



(d) Possible goal state.

Fig. 2: (a) The physical and (b) simulated Minitaur quadrupedal platform [5]. (c) An example of a complex dynamically-constrained domain in which a robot must reach the goal on the lower right, where the heights of the ground plane ($z = 0$), the obstacle ($z = 1$), and the goal ($z = 2$) are all different. The robot can jump only one unit, so the robot must move the cart (red square) to enable a sequence of jumps to reach a goal state (d).

demonstrate computational advantages in the deliberative layer arising from delegating metric details to the reactive (closed loop) controller. First, the reactive layer allows the deliberative planner to plan only in terms of transitions between behaviors, such as grasping and releasing objects. As a result, fewer samples are needed to find a good plan. Second, because the deliberative planner is aware of the domain of convergence for each controller and can produce plans using the adjacency of those basins of reachability, it can construct plans with fewer steps than if it relied solely on fine-grained motion primitives—for example, line-of-sight connections—without sacrificing any correctness guarantees. The difference is especially pronounced when the path requires traversing narrow passages, which are notoriously difficult for sampling-based planners. The reduction in the length of plans dramatically reduces the time required to search for a high-level plan.

C. Organization of the Paper

The paper is organized as follows. Section II summarizes related work. Section III describes the proposed multi-layer architecture, along with its formal guarantees. Section IV describes the mobile manipulation problems addressed in this paper, in two different planning approaches for the deliberative planner: a *local reactive* approach, employing the reactive layer to simply track reference trajectories from the deliberative layer, and our *global reactive* approach with the proposed architecture. Section V describes numerical studies contrasting the performance of the local reactive planner with the global reactive planner in different mobile manipulation scenarios. Section VI provides implementation examples with a simulated robot, and, finally, Section VII concludes with our remarks and ideas for future research.

II. Related Work

Hierarchical abstractions for TAMP have been well studied in the literature. Examples include the use of a deliberative planner that employs a reactive execution layer—such as a motion primitive library [8], or pre-image backchaining with a higher-level planner in deterministic [4] and stochastic [9] settings—to simplify the computational burden of planning. Solutions typically involve a marriage of fast discrete planning tools [10] and sampling- or grid-based discretizations of the continuous action space [11], with significant engineering effort expended on the design of effective heuristics and sampling strategies that exploit task-level and geometric information [12]. Angelic semantics [13] provide a way of describing abstractions that also preserve optimality, but there is no easy way of defining such abstractions in continuous domains. Our prior work [14] provided a step towards tractable planning with complex kinematic constraints, but no appropriate approach exists for the complex legged robot dynamics considered in this paper.

Motivated by the typically high-dimensional configuration spaces arising from combined task and motion planning, most approaches focus either on sampling-based methods that empirically work well [15], [16], or learning a symbolic language on the fly [17]. Such methods require constant replanning in the presence of unanticipated conditions and their search time grows exponentially with the number of configuration variables.

Other approaches focus on the use of reactive temporal logic planning algorithms [18], [19], [20], that can account for environmental uncertainty in terms of incomplete environment models, and also ensure correctness when the robot operates in an environment that satisfies the assumptions modeled in the task specification. Common in these works is the reliance on discrete abstractions of the robot dynamics [21], [22], while active interaction with the environment to satisfy the logic specification is neglected.

III. Vector-Field Task Planning

Our objective is to compute plans for a robot to achieve a goal state, subject to kinodynamic constraints. The active constraints on the dynamics of the world state vary with the robot's behavior, enabling the robot to select different *modes* of its dynamics as it plans to move around the world. For example, a plan for the robot in Fig. 2 might simply be to navigate its workspace, or to make and break contact with the objects in the world as it moves around. Each of these modes corresponds to a different set of constraints.

Following the notation introduced in our prior work [14], we can define a *planning domain* by a tuple $(h, \mathcal{C})$, where $h : \mathcal{C} \times T\mathcal{C} \rightarrow \mathbb{R}^k$ defines a set of k constraints on the configuration space $\mathcal{C}$ and its tangent bundle $T\mathcal{C}$. Then, a differentiable function $\sigma : [0, T] \rightarrow \mathcal{C}$ is a *feasible path* if $h(\sigma(t), \dot{\sigma}(t)) \geq 0, \forall t \in [0, T]$, where $\dot{\sigma}(t) = d\sigma(t)/dt$. We denote the set of feasible paths by $\Sigma_{\mathcal{C}}$.

Based on this description, we can define a *planning problem* as a tuple $(\mathbf{c}_0, \mathbf{c}^*)$, where $\mathbf{c}_0, \mathbf{c}^* \in \mathcal{C}$ are the initial and goal configurations respectively, and a *solution* to this planning problem as a path $\sigma \in \Sigma_{\mathcal{C}}$ such that $h(\sigma(\tau), \dot{\sigma}(\tau)) \geq 0, \forall \tau \in [0, T]$, $\sigma(0) = \mathbf{c}_0$ and $\sigma(T) = \mathbf{c}^*$.

Solving for such a path $\sigma \in \Sigma_{\mathcal{C}}$ without further assumptions on $h(\sigma(\tau), \dot{\sigma}(\tau))$ is a formally undecidable problem [6], and solving the analogous problem for typical discrete approximations is computationally intractable for scenarios where the robot needs to make and break contact with the environment. A conventional approach is to decompose the problem into a task and motion planning problem: a deliberative layer first solves for a task plan corresponding to a sequence of dynamic modes, parameterized by starting and stopping conditions, and a motion planner generates trajectories within each mode from start to stopping condition.

However, the decomposition into separate task and motion planning problems typically leads to loss of completeness, because the task planner may create motion planning subproblems that are infeasible. We now describe the formal conditions under which a combined deliberative and motion planning layer can compute task plans that preserve probabilistic completeness guarantees of the underlying motion planner, even without first evaluating it.

A. Deliberative layer

We assume that the deliberative planner has knowledge of the entire configuration space, including a description of the world as a collection of objects with geometric information, such as shape and pose, and other properties that constrain the types of actions available with these objects.

We use the *continuous constraint contract* (C3) to represent states, presented in our prior work [6]. The C3 representation is a continuous extension of the SAS+ formalism [23]; as in most planning formalisms, the state of the world is parameterized by the *value* of different *variables*. A state $\mathbf{s} \in \mathcal{S}$ is a collection of variable–value pairs, and represents the set of configurations satisfying the constraint defined by the value assigned to each variable. Each variable v corresponds to a function $\eta_v(\mathbf{c})$ mapping configurations $\mathbf{c}$ to an element of the variable's domain; a state $\{v_1 = p_1, v_2 = p_2\}$ describes the set of configurations $\mathbf{c}$ such that $\eta_{v_1}(\mathbf{c}) = p_1$ and $\eta_{v_2}(\mathbf{c}) = p_2$. There is no requirement that every variable have an assigned value; variables without values represent inactive constraints.

To travel between states, we assume the lower level motion planner can instantiate *actions*, parameterized by a start and goal; the deliberative planner must then choose a sequence of different actions as well as their parameterization.

Theorem 1 (Representing constraints as analytic functions – Included in [6]) *If the kinodynamic constraints h are piecewise-analytic in the sense of Sussmann [24], and the dynamical system is stratified controllable in the sense of Goodwine and Burdick [25], then there is a stratified C3 instance whose actions represent piecewise-analytic vector fields, in which the constraints can be expressed as equalities and inequalities involving only analytic functions.*

Using Theorem 1, the planning problem becomes one of choosing a sequence of vector fields and their parameterizations. We can further take advantage of this result by defining actions as a *contract* between the deliberative and motion layers: formally, we define the requirements and effects of an action a with continuous parameterization $\mathbf{\Theta}_a$ in terms of two functions $g_a : \mathcal{S} \times \mathbf{\Theta}_a \to \mathbb{B}$ and $f_a : \mathcal{S} \times \mathbf{\Theta}_a \to \mathcal{S}$, where $\mathcal{S}$ is the space of possible states $\mathbf{s}$. The function $g_a(\mathbf{s}, \boldsymbol{\theta})$ defines the *requirements* of the action, and $f_a(\mathbf{s}, \boldsymbol{\theta})$ defines its *effects*. If the system is in state $\mathbf{s}$ when executing action a with parameters $\boldsymbol{\theta}$, then the motion planner guarantees that if $g_a(\mathbf{s}, \boldsymbol{\theta}) = 1$ then at some point in the future the system will reach state $\mathbf{s}' = f_a(\mathbf{s}, \boldsymbol{\theta})$.

However, such guarantees are in practice difficult to describe. The easiest guarantee to provide is one where the motion planner is restricted to straight-line actions parameterized by an end point, and enforces reachability by evaluating each straight-line trajectory for violations of the kinodynamic constraints. A deliberative planner using this simplistic motion planner would offer probabilistic guarantees of completeness, but with essentially no computational advantage from the decomposition into deliberative and motion planning. The challenge is to identify a motion planner that can enforce the C3 contracts in a computationally efficient manner.

B. Reactive layer

We now describe a reactive motion planner with a key property: the corresponding C3 contracts can be checked very quickly, without sampling, discretization, or collision-checking. Rather than instantiating a single motion plan, the reactive layer constructs a control policy that is guaranteed to achieve the objectives specified by the deliberative layer, or to return with a failure condition expressing the incorrectness of a presumed constraint in the actual environment.

The reactive layer models the robot as a polygon, and takes as input an estimate of the current reachable set of robot poses, in the form of a polygonal connected component of the robot's workspace, along with a high-level action a with all parameters $\boldsymbol{\theta} \in \mathbf{\Theta}_a$ chosen by the deliberative layer. The reactive layer is implemented using the vector-field-based feedback motion planning scheme introduced in our prior work [7], and its critical advantage is the use of a diffeomorphism construction to deform non-convex environments to easily navigable convex worlds, by employing domain specific knowledge about encountered obstacles.

In the reactive layer we assume that the robot is the only active agent in the world, and behaves like a first-order, nonholonomically-constrained, disk-shaped robot, centered at location $\mathbf{x} \in \mathbb{R}^2$, with radius $r \in \mathbb{R}_{>0}$, orientation $\psi \in S^1$ and input vector $\overline{\mathbf{u}} := (v, \omega)$, consisting of a fore-aft and an angular velocity command. We denote by $\mathcal{W}$ the robot's non-convex polygonal workspace, and by $\mathcal{W}_\mathbf{x} \subseteq \mathcal{W}$ the polygonal region corresponding to the space reachable from the robot's current position $\mathbf{x}$. The workspace is cluttered by a finite collection of disjoint obstacles of unknown shape, number, and placement. Similarly, the *freespace* $\mathcal{F}$ is defined as the set of collision-free placements in $\mathcal{W}$, and we denote by $\mathcal{F}_\mathbf{x} \subseteq \mathcal{F}$ the freespace component corresponding to $\mathcal{W}_\mathbf{x}$.

During online execution, the reactive controller synthesizes an action as a control law by constructing a diffeomorphism $\mathbf{h}$ between $\mathcal{F}_\mathbf{x}$ and a convex *model environment*, where non-convex obstacles are either deformed to topologically equivalent disks or merged to the boundary of $\mathcal{F}_\mathbf{x}$. Then, the robot can navigate by generating virtual commands $\overline{\mathbf{v}} = (\hat{v}, \hat{\omega})$ as in [26], for an equivalent unicycle model (defined in [27, Eqs. (24)-(25)]) that navigates toward the assigned target position $\mathbf{x}^*$ in this model environment, and then mapping the virtual commands to physical inputs (v, ω) through the push-forward of the inverse of $\mathbf{h}$, i.e., $\overline{\mathbf{u}} = [D_{\mathbf{x}}\mathbf{h}]^{-1}\,\overline{\mathbf{v}}$.

Using the language of the deliberative layer, the requirements $g_a(\mathbf{x}, \mathbf{x}^*)$ of a navigation action are satisfied if both the robot and target positions are contained in the same component $\mathcal{F}_{\mathbf{x}}$ of the robot's freespace. More formally, we can decompose $\mathcal{F}$ into a finite collection of connected polygons (possibly with holes), and define a set-valued function $\beta_{\mathcal{F}} : \mathcal{F} \to 2^{\mathcal{F}}$, such that $\mathcal{F}_{\mathbf{x}} \triangleq \beta_{\mathcal{F}}(\mathbf{x}) \subseteq \mathcal{F}$ is the connected component containing $\mathbf{x}$. We describe an implementation of this function in Section IV-C.

Theorem 2 (Target convergence and obstacle avoidance – Corollary of [7, Theorem 2]) *If we define $g_a(\mathbf{x}, \mathbf{x}^*)$ to be equal to 1 when $d(\beta_{\mathcal{F}}(\mathbf{x}), \beta_{\mathcal{F}}(\mathbf{x}^*)) = 0$ and 0 otherwise (with $d(\cdot, \cdot)$ the distance between sets), then the online reactive planner guarantees that the robot will converge to the target $\mathbf{x}^*$ (i.e., $f_a(\mathbf{x}) \triangleq \mathbf{x}^*$), while avoiding all obstacles in its workspace.*

It should be noted that Theorem 2 covers only navigation actions; to navigate across mode boundaries (i.e., across connected components of the configuration space), we use special-purpose local actions (e.g., the action `jump` mentioned in Section IV).

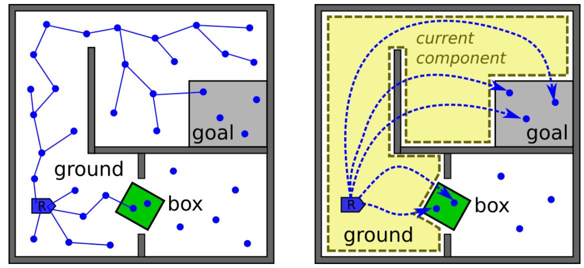
C. Combined deliberative-reactive planning

In practice, given a C3 problem instance describing the permitted actions, the world geometry, and a goal specification, we construct a graph by sampling random parameters for actions. In navigation or manipulation problems, this generally involves sampling candidate placements for objects or for the robot. Importantly, Theorem 1 allows us to sample from the free space of each object independently, rather than sampling from the joint configuration space, without sacrificing completeness. We then perform a direct heuristic search over the planning graph to synthesize a plan. We reduce the computational cost of the search by considering only a reduced set of constraints in the action requirements when sampling, and checking the remaining constraints only when we find a candidate plan to a given state. We refer the reader to [14] for more details on the graph construction and search.

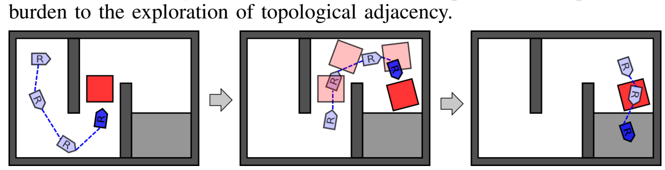
Theorem 3 (Combined probabilistic completeness – Corollary of Theorems 1–2) *If our planning domain contains only modes defined by piecewise-analytic constraints and stratified controllable dynamics, and there exist local actions for navigating across mode boundaries, then the deliberative planner will eventually sample a feasible motion plan, expressed as a sequence of reactive planner actions between connected components of the configuration space.*

IV. System Implementation

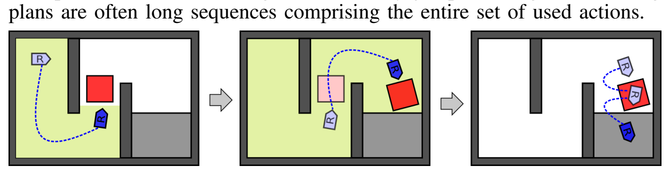
In this Section, we describe the specific class of mobile manipulation problems addressed in this work. While our planner is general purpose for a wide range of problems, we consider the problem of a dynamically complex quadruped robot performing navigation among movable obstacles (NAMO) as in our prior work [28].



(a) In our baseline local reactive planning approach (left), the deliberative planner must conduct an optimized search over the configuration space of robot and object placements in the presumed freespace and is restricted to collision free straight-line paths. Motions that instantiate these paths are generated at runtime by the reactive layer, guaranteeing avoidance of unanticipated obstacles along the way. The global reactive planning approach (right) is guaranteed to generate a collision-free path to any target pose in the robot's current connected component (highlighted in yellow). Actions are now represented by putative robot-connected components and their adjacency relative to robot–object manipulations. This more abstract contract between layers reduces the deliberative planner's computational burden to the exploration of topological adjacency.



(b) Solution to the example from Fig. 2 using the local reactive planning architecture (Section IV-B) [28]. The deliberative planner finds a sequence of collision-free straight-line motion primitives to move the robot to the cart, push the cart near the goal, and finally jump to the goal. Resulting plans are often long sequences comprising the entire set of used actions.



(c) Solution to the example from Fig. 2 using the global reactive planning approach (Section IV-C). Shaded regions indicate the robot's currently occupied connected component, defining the (global) navigation domain for the reactive layer. The dashed lines are purely illustrative, as the actual paths are unknown to the deliberative planner and commanded at runtime by the reactive layer.

Fig. 3: Comparison of local and global reactive planning approaches.

A. Problem Domain Description

Our chosen model abstraction for planning is a 2.5D *semi-planar world* representation, shown in Fig. 2c. All objects in the world, which can be either static or movable, are defined as planar polygons, with a pose in $\mathrm{SE}(2)$ augmented by a z value denoting vertical height. The robot can walk along the polygonal component describing the top of the currently occupied object, jump on and off the ground plane and between objects of varying heights provided the height difference and gap is within its physical capabilities, and manipulate movable objects on the currently occupied object.

As detailed in [29], to demonstrate our proposed architecture, we use five types of actions to model the robot capabilities (`move`, `jump`, `grasp`, `release`, and `push`) and attempt to solve mobile manipulation problems where the reactive layer is instantiated with either a *local reactive approach* that does not allow the deliberative planner to query the reactive planner for motion contracts, or with the proposed *global reactive approach* described in Section III (see Fig. 3). We use the local reactive approach as a baseline comparison because it is equivalent to TAMP planners that provide correctness guarantees, for example, [11], [14], [30].

B. Local Reactive Planning Approach

Given a sequence of actions from the deliberative layer, the local reactive planner only guarantees the feasibility of navigating from a starting pose to a target pose if the path between them is collision-free (Fig. 3b). That is, the reactive planner can move from $\mathbf{x}$ to $\mathbf{x}^*$ if $\texttt{isfeasible}(\mathbf{x}, \mathbf{x}^*) = 1$, with $\texttt{isfeasible}(\mathbf{x}_1, \mathbf{x}_2)$ equal to 1 when $C_{\mathcal{F},r}(P_{sweep}(\mathbf{x}_1, \mathbf{x}_2), \mathbf{c}) \geq 0$ and 0 otherwise, where $P_{sweep}(\mathbf{x}_1, \mathbf{x}_2)$ is a polygon containing the robot polygon at each pose along a geodesic between poses $\mathbf{x}_1$ and $\mathbf{x}_2$ in $\mathrm{SE}(2)$, and $C_{\mathcal{F},r}(P, \mathbf{c})$ checks for collision of a polygon P with any object or obstacle in configuration $\mathbf{c}$. Here, the local reactive layer only plays its intermediating role when recovering the target pose in the face of unanticipated obstacles—or reporting the infeasibility of doing so.

C. Global Reactive Planning Approach

The crucial advantage of the global reactive controller developed in [7] is that it guarantees successful navigation to any pose in its *connected component of the freespace*, $\mathcal{F}$. Recall the function $\beta_{\mathcal{F}}(\mathbf{x})$ defined in Section III-B that maps each robot pose to the connected component of the freespace containing that pose. In the general reactive planning setting, $\beta_{\mathcal{F},o}(\mathbf{c})$ depends on the configuration $\mathbf{c}$ of each object, and returns the connected component of the freespace of object o, conditioned on the pose of each other object. The reactive layer then defines, for any goal pose $\mathbf{o}^*$, a closed-loop controller with an attractor basin that includes the polygon $\beta_{\mathcal{F},o}(\mathbf{c}^*)$. Formally, for the purposes of the deliberative layer, $\texttt{isfeasible}(\mathbf{o}, \mathbf{o}^*) = 1$, if $\mathbf{o} \in \beta_{\mathcal{F},o}(\mathbf{c}^*)$.

We describe the global reactive planner as a C3 domain, and use the sampling-based planning algorithm described in Section III-C to search for a sequence of transitions between adjacent basins of attraction created by invocations of the reactive layer. To determine adjacency, we explicitly construct the polygonal connected components of the freespace containing each robot and object using the Boost Geometry library [31]. Two polygonal components are adjacent if the distance between them is small enough to be traversed by a jumping or manipulation action.

V. Numerical Experiments

In this Section, we present scenarios that describe common task specifications that can be solved using our system, and we perform qualitative and quantitative analyses contrasting the performance of the deliberative planner with a local reactive and a global reactive planner.

A. Known Environment Scenario

In the scenario in Fig. 4, the robot must move an object to a goal while navigating an increasingly dense set of randomly generated obstacles known by the deliberative layer. Fig. 4d shows that planning times increase with the number of obstacles at a higher rate using the local reactive approach. Despite the added overhead of decomposing the environment into connected components when using the global reactive planning approach, more samples are needed to successfully solve these tasks using the local reactive system. Also, the success rate of planning with the local reactive approach decreases with environmental complexity. For all generated worlds in which the global reactive approach found a plan, we planned 10 times more with the local reactive approach. In the case of zero obstacles, average success rate was 100%, gradually decreasing to 66.5% for 20 obstacles.

B. Doorways Scenario

The scenario in Fig. 5 explores how planning time scales with environmental complexity, both in terms of static obstacles and movable objects. Since the complexity of walls is abstracted away with the global reactive layer, this scenario can be solved with less samples (and therefore in less time) than with the local approach, as shown in Fig. 5b. Moreover, extraneous objects that do not block a doorway have a significantly lower impact on planning time, as seen in the inflection points for the "3 Doorways" and "5 Doorways" lines at 3 and 5 objects, respectively.

VI. Simulation Experiments

We demonstrate our proposed system architecture on a Ghost Minitaur [5] quadrupedal robot using the Gazebo simulator[1]. In our implementation, the gait layer abstracts the details of determining how to move the limbs or negotiate uneven terrain. Specifically, we employ the steady-state behaviors *"Walk"* and *"Push-Walk"* from [28], to either navigate the workspace or use the robot's front limbs as a virtual gripper when manipulating movable objects respectively. In addition, we use a set of four transitional behaviors: *"Mount"*, *"Dismount"*, *"Jump-Up"* and *"Jump-Across"*, adapted from [32], to mount and dismount objects, or jump on platforms or across gaps.

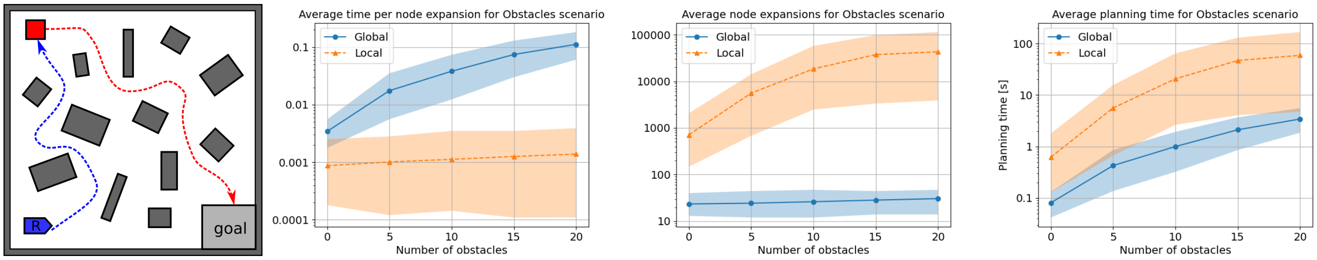
A. Doorways Scenario

We show the robot executing plans generated using our method on the *Doorways* scenario of Section V-B. Fig. 5d demonstrates that the global reactive approach allows the deliberative layer to find plans even in the presence of a complex space punctured by a large number of obstacles. Here, the robot has prior knowledge of all fixed walls, but no prior information on the location of the cylindrical obstacles; it must discover and avoid them using an onboard LIDAR.

B. Unknown Environment Scenario

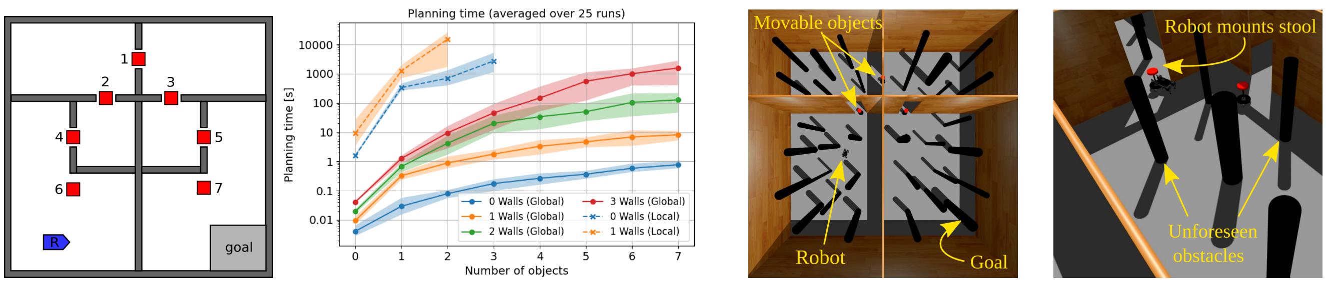
In this scenario, shown in Fig. 6, the robot must move from its starting pose to a specified goal pose. In the absence of obstacles, the lowest-cost solution involves directly moving along a hallway to the goal. However, as the obstacle density increases, it may become difficult to plan around obstacles, or the hallway may be blocked altogether. In this case, there is a higher-cost alternative in which the robot can push a movable object near the top of the environment and navigate a longer path to the goal. We qualitatively show that the global approach can handle unanticipated obstacles without

[1] Video of these simulations is included in the video submission and online in `https://youtu.be/Ta5sVFkNnxo`. The files for simulating Minitaur in Gazebo can be found in `https://github.com/KodlabPenn/kodlab_gazebo` and a C++ implementation of the reactive layer is included in `https://github.com/KodlabPenn/semnav`.



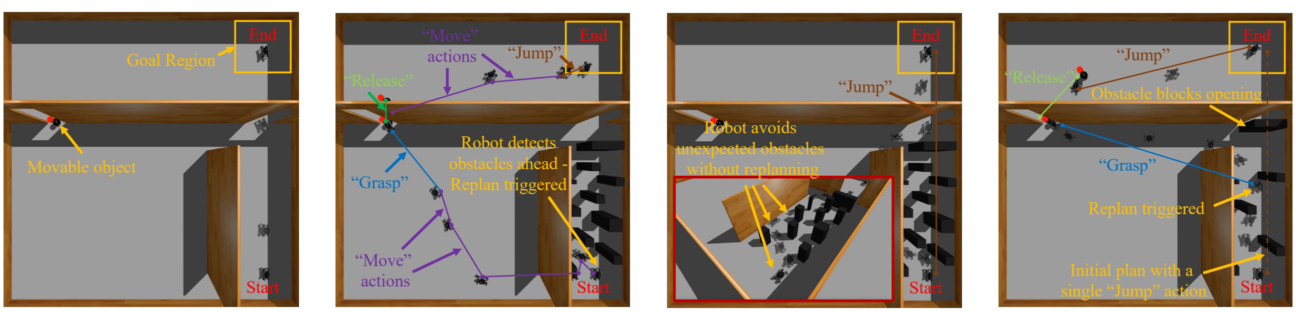
(a) Scenario overview (b) Expansion time vs. complexity (c) Node expansions vs. complexity (d) Planning time vs. complexity

Fig. 4: **Known Environment Scenario**. The robot (blue) must move a block (red) to the goal in the lower right corner of the map. Obstacles (grey) are randomly generated, ranging from 0 to 20 in number, to explore the effects of obstacle density on planning time (a). With the global reactive approach, even though each individual graph node expansion becomes more expensive as obstacle density increases (b), the relative decrease in node expansions (c) results in an overall decrease in planning time (d). Shaded areas denote 5th to 95th percentiles.



(a) Scenario overview (b) Planning time vs. complexity (c) Gazebo scenario (d) Simulation details

Fig. 5: **Doorways Scenario**. The robot (blue) must traverse an increasingly complex set of walls and push objects out of the doorways to reach the goal (a). Planning time increases with number of doorways and obstacles, which both add complexity to the problem. Additionally, the contract provided by the global reactive approach significantly reduces planning times for equivalent problems compared to when utilizing the local reactive approach. Shaded areas denote 5th to 95th percentiles. (b). Walls are added in the following order: first, the vertical wall containing doorway 1, then the horizontal wall with doorways 2 and 3, and finally the center walls containing doorways 4 and 5. Objects are randomly placed in locations 1–7, ensuring that first all existing doorways are blocked before placing extraneous objects in free space. We simulated this scenario in Gazebo (see the accompanying video submission), adding random cylindrical obstacles that were unknown to the deliberative planner (c), (d). By leveraging the proposed global reactive planning approach, the robot is able to navigate the environment and manipulate movable objects to reach a goal despite the unforeseen obstacles.



(a) No unforeseen obstacles (b) Local Reactive Planning (c) Global Reactive Planning (d) Replanning needed

Fig. 6: **Unknown Environment Scenario**. Minitaur must move to a goal region at the top right of the environment. The lowest-cost solution involves navigating along the hallway on the right, which is successfully executed using both the local reactive and the global reactive approach (a). Random unanticipated obstacles (black) may appear in this hallway and are only detected and localized when the robot approaches them within a specified distance. Using the local reactive approach, the robot quickly abandons the initially evaluated plan because some of the initial waypoints lie in obstacle space, and replans. It then unnecessarily switches to a higher-cost plan, involving manipulating the movable object and navigating a longer path to the goal (b). Using the global reactive approach, the robot either avoids all interior obstacles without changing its initially executed plan (c), or requests an alternative plan when detecting that the requirements of the contract are violated, i.e., the robot and the goal lie in different connected components of the freespace (d).

triggering a full replan, unless a newly localized obstacle blocks the hallway, violating the contract between the layers. For these simulations, we assume that the robot possesses a sensor of fixed range (set at 3 m), for localizing obstacles.

VII. Conclusion

Our hierarchical planner exhibits the greatest gains in efficiency when finding long plans with a small number of actions; problems that require evaluating combinatorially many transitions remain an open challenge. Our contract-based approach to action modeling could be combined with recent improvements in sampling strategies, search algorithms, and planning heuristics. In addition, our approach could be applied to other classes of robotic platforms for which local controllers can be devised. Finally, we note that one shortcoming of our approach is the lack of a mechanism for the deliberative planner to correct the reactive planner if it makes a locally suboptimal decision. In principle, we could defer the selection of the level of abstraction at which to search for a plan to execution time, or even interleave searches at different levels of abstraction to capitalize on the relative strengths of each representation.

References

[1] M. T. Mason and K. M. Lynch, "Dynamic manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1993.

[2] 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.

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

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

[5] Ghost Robotics, "Ghost Robotics Minitaur," 2016. [Online]. Available: http://www.ghostrobotics.io/minitaur/

[6] W. Vega-Brown and N. Roy, "Task and motion planning is PSPACE-complete," in *The AAAI Conference on Artificial Intelligence*, 2020.

[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, 2020.

[8] A. Majumdar and R. Tedrake, "Funnel Libraries for Real-Time Robust Feedback Motion Planning," *The International Journal of Robotics Research*, vol. 36, no. 8, 2017.

[9] L. P. Kaelbling and T. Lozano-Pérez, "Pre-image backchaining in belief space for mobile manipulation," in *Robotics Research*, 2017.

[10] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, 2018.

[11] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *International Conference on Automated Planning and Scheduling*, 2020.

[12] F. Lagriffoul and B. Andres, "Combining task and motion planning: A culprit detection problem," *The International Journal of Robotics Research*, vol. 35, no. 8, 2016.

[13] B. Marthi, S. Russell, and J. Wolfe, "Angelic hierarchical planning: Optimal and online algorithms," in *International Conference on Automated Planning and Scheduling*, 2008.

[14] 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.

[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 9th International Workshop on the Algorithmic Foundations of Robotics*, 2010.

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

[17] 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*, 2018.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, 2008.

[23] C. Bäckström and B. Nebel, "Complexity results for SAS+ planning," *Computational Intelligence*, vol. 11, no. 4, 1995.

[24] H. Sussmann, "Subanalytic sets and feedback control," *Journal of Differential Equations*, vol. 31, no. 1, pp. 31–52, 1979.

[25] B. Goodwine and J. Burdick, "Controllability of kinematic control systems on stratified configuration spaces," *Transactions on Automatic Control*, vol. 46, no. 3, pp. 358–368, 2001.

[26] O. Arslan and D. E. Koditschek, "Sensor-Based Reactive Navigation in Unknown Convex Sphere Worlds," *The International Journal of Robotics Research*, 2018.

[27] V. Vasilopoulos and D. E. Koditschek, "Reactive Navigation in Partially Known Non-Convex Environments," in *The 13th International Workshop on the Algorithmic Foundations of Robotics*, 2018.

[28] V. Vasilopoulos, T. T. Topping, W. Vega-Brown, N. Roy, and D. E. Koditschek, "Sensor-Based Reactive Execution of Symbolic Rearrangement Plans by a Legged Mobile Manipulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[29] V. Vasilopoulos, S. Castro, W. Vega-Brown, D. E. Koditschek, and N. Roy, "Technical Report: A Hierarchical Deliberative-Reactive System Architecture for Task and Motion Planning in Partially Known Environments," *arXiv: 2202.01385*, 2022.

[30] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010.

[31] B. Schling, *The Boost C++ Libraries*. XML Press, 2011.

[32] T. T. Topping, V. Vasilopoulos, A. De, and D. E. Koditschek, "Composition of templates for transitional pedipulation behaviors," in *International Symposium on Robotics Research*, 2019.