Efficient Task Planning for Mobile Manipulation: a Virtual Kinematic Chain Perspective

Ziyuan Jiao[1*] Zeyu Zhang[1*] Weiqi Wang[1] David Han[2] Song-Chun Zhu[1] Yixin Zhu[1] Hangxin Liu[1]

***Abstract*— We present a Virtual Kinematic Chain (VKC) perspective, a simple yet effective method, to improve task planning efficacy for mobile manipulation. By consolidating the kinematics of the mobile base, the arm, and the object being manipulated collectively as a whole, this novel VKC perspective naturally defines *abstract actions* and eliminates unnecessary predicates in describing intermediate poses. As a result, these advantages simplify the design of the planning domain and significantly reduce the search space and branching factors in solving planning problems. In experiments, we implement a task planner using Planning Domain Definition Language (PDDL) with VKC. Compared with conventional domain definition, our VKC-based domain definition is more efficient in both planning time and memory. In addition, abstract actions perform better in producing feasible motion plans and trajectories. We further scale up the VKC-based task planner in complex mobile manipulation tasks. Taken together, these results demonstrate that task planning using VKC for mobile manipulation is not only natural and effective but also introduces new capabilities.**

I. Introduction

As one of the central themes in AI and robotics, task planning is typically solved by searching a feasible action sequence in a domain. Researchers have demonstrated a wide range of successful robotics applications [1, 2] with effective representations or programming languages, such as STRIPS [3], hierarchical task network [4], temporal and-or-graph [5, 6], Markov decision process [7], and PDDL [8].

An effective task planner in robotics generally possesses two characteristics. First, the planning domain must be clearly designed, which includes a set of predicates that truthfully describe the environment states, a set of actions that specify how states transit, and a goal specification that indicates the desired result. However, the definitions of these components are tightly coupled; thus, designing the planning domain could be tedious and error-prone. Second, the abstract notion of symbolic actions should be realizable by motion planners; *i.e.*, the design of these abstract symbols should have practical meaning. These two requirements pose additional challenges in task planning for mobile manipulation; the robot consists of a mobile base and an arm, which possess different motion patterns and capabilities.

* Ziyuan Jiao and Zeyu Zhang contributed equally to this work.
[1] UCLA Center for Vision, Cognition, Learning, and Autonomy (VCLA) at Statistics Department. Emails: {zyjiao, zeyuzhang, weiqi.wang, yixin.zhu, hx.liu}@ucla.edu, sczhu@stat.ucla.edu.
[2] Drexel University, Department of Electrical and Computer Engineering. Email: dkh42@drexel.edu.
The work reported herein was supported by ONR N00014-19-1-2153, ONR MURI N00014-16-1-2007, and DARPA XAI N66001-17-2-4029.

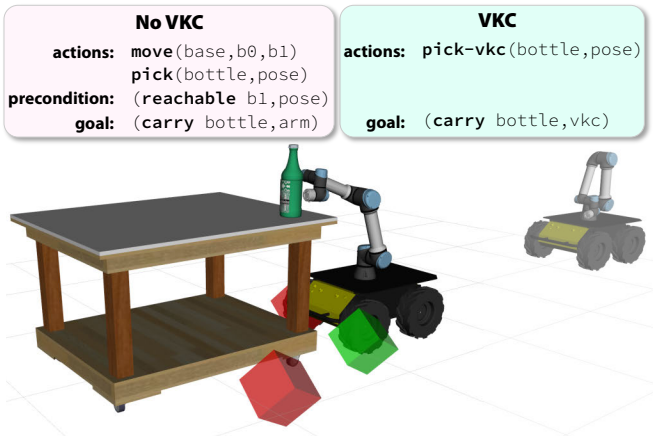


Fig. 1: **A typical task planning setup, wherein the mobile manipulator is to tasked to navigate and pick up the object on the desk.** The VKC-based domain specification reduces the search space by removing the poses of the mobile base near red cubes, resulting in a simpler and more intuitive task planning domain.

To clearly illustrate the above challenges, let us take Fig. 1 as a concrete example, wherein a mobile manipulator is tasked to navigate and pick up the bottle on the desk. A dedicated set of predicates and actions must be specified for the mobile base and the arm; for instance, moving the *base* (`move(·)`) to a configuration, such that the *arm* can pick up the object (`pick(·)`). Of note, finding such a pose oftentimes requires to specify the mobile base and the arm *individually*. However, this separation in the planning domain is *artificial* in nature and ineffectively introduces an *unnecessarily* larger planning space: The valid poses of the mobile base near the goal (*i.e.*, the bottle) must be specified (indicated by the cubes in Fig. 1) in advance, and exactly one (*e.g.*, the green cube) must be selected via sampling or searching under pre-defined heuristic or criteria. This deficiency becomes increasingly evident as the task sequence grows longer and prohibits natural motions that require foot-arm coordination; coordinating the base and arm movements remains challenging even for existing whole-body motion planning methods [9–11], let alone realizing a symbolic task plan with a feasible motion plan.

In this paper, we seek to tackle these challenges in task planning with a focus on domain specification. Specifically, we ask: (i) Is there an intermediate representation to facilitate the specification of the planning domain such that the search can be more natural and efficient, without relying on manually-designed heuristics? (ii) Can this new representation simplify the planning domain in long-horizon tasks while preserving motion feasibility, especially for those who require complex foot-arm coordination?

In particular, we propose a Virtual Kinematic Chain (VKC) perspective for mobile manipulation, which consolidates the kinematics of the mobile base, the arm, and the object being manipulated into a single kinematic model. By treating the robot as a whole, more abstract actions can be defined to jointly account for both the base and the arm; see `pick-vkc(·)` *vs* `move(·)` and `pick(·)` in Fig. 1. Such an abstraction alleviates the manually-defined *heuristic* of where the robot can reach the goal and the unnecessary definitions of *intermediate goals*, *e.g.*, predicates describing the robot's pose before reaching the goal. As a result, this modification of the planning domain reduces the branching factor, making it scalable to more complex tasks. Crucially, the abstraction introduced by VKC does not sacrifice the success rate to generate a solvable motion planning problem.

In experiments, we implement the VKC-based task planning using PDDL. Compared with a standard PDDL implementation, the VKC perspective simplifies the domain setup and exhibits significant improvements in planning time and memory. Moreover, we demonstrate that abstract actions introduced by VKC are executable at the motion level; they can generate solvable motion planning problems in the form of trajectory optimization or sampling methods. We further validate VKC-based task planning in long-horizon tasks requiring foot-arm coordination. Taking together, our VKC perspective offers a simple yet effective intermediate representation for domain specification in task planning.

A. Related Work

Task and Motion Planning (TAMP) in mobile manipulation: Thanks to the development of PDDL and other planning architectures, complex symbolic task planning can be solved using standard algorithms [2]. Hence, the community has shifted the focus to corresponding a valid symbolic action sequence to feasible motions, which leads to the field of TAMP [12]. While researchers tackle this problem from various angles, such as incorporating motion-level constraints to the task planning [13–15], developing interfaces that communicate between task and motion [16], or inducing abstracted modes from motions [17, 18], it remains a largely unsolved problem. In addition, movements of a mobile base and a manipulator are commanded by two or more separate actions [15, 19, 20], causing increased planning time, less coordinated movements, *etc*. In comparison, the VKC perspective serves as an intermediate representation that benefits the task modeling of mobile manipulation, improves computation efficacy, and facilitates motion planning.

VKC in robot modeling and planning: The idea of VKC could be traced back to 1997 by Pratt *et al*. [21], who proposed VKC to design bipedal robot locomotion [22]. Later, this idea was adopted to chain serial manipulators to form one kinematic chain [23] and to dual-arm manipulation tasks; for instance, connecting parallel structures via rigid-body objects [24], and modeling whole-body control of mobile manipulators [25]. Recently, VKC is also adopted for wheeled-legged robot control [26]. In this paper, we further push the idea of VKC and demonstrate its advantages as an intermediate representation in modeling and planning complex mobile manipulation tasks.

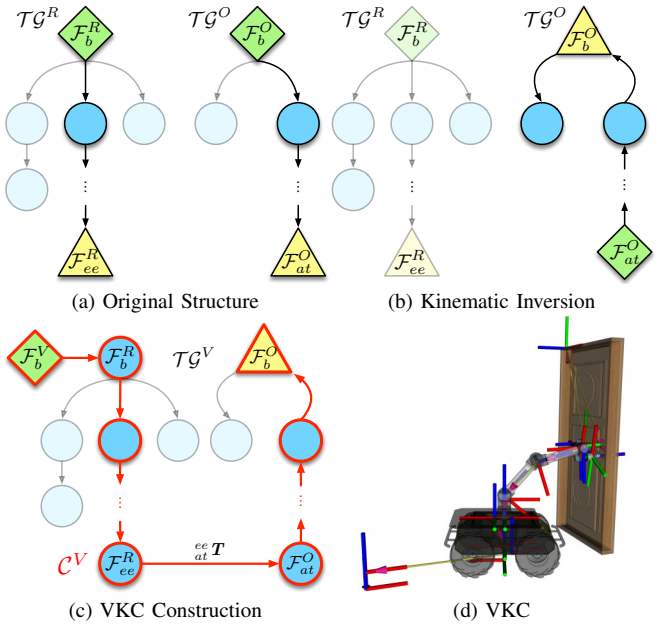


Fig. 2: (a–c) The construction process of a VKC. The green diamond denotes the *root* frame of a kinematic structure, the yellow triangle denotes a *terminal* frame (*e.g.*, robot end-effector or object attachable frame), the blue circle denotes other frames, and the red trail in (c) highlights a constructed virtual kinematic chain. (d) The corresponding VKC in opening the door.

B. Overview

The remainder of this paper is organized as follows. Section II introduces the modeling process of constructing a VKC. A tasking planning framework from the VKC perspective based on PDDL is described in Section III. Section IV further illustrates how the high-level VKC-based task planning facilitates low-level motion planning. In a series of mobile manipulation tasks, we demonstrate the efficacy of VKCs with a high success rate in Section V. We conclude the paper with discussions in Section VI.

II. Virtual Kinematic Chain (VKC) Modeling

The objective of the modeling is to construct a serial VKC $\mathcal{C}^V$ by composing the kinematic of the mobile base, the arm, and the object to be manipulated.

Original Structure: The kinematic models of the robot and the object being manipulated are represented by two kinematic trees, $\mathcal{TG}^R$ and $\mathcal{TG}^O$, respectively. The highlighted area in Fig. 2a indicates the original kinematic chain of interests: $\mathcal{C}^R$ for robots, and $\mathcal{C}^O$ for objects.

Kinematic Inversion: To construct a serial VKC by inserting a virtual joint (*i.e.*, an *attachment*) between the robot's end-effector frame $\mathcal{F}_{ee}^R$ and the object's attachable frame $\mathcal{F}_{at}^O$, one has to invert the kinematic relationship of the object $\mathcal{C}^O$ as shown in the right panel of Fig. 2b. Of note, such an inversion is not equivalent to simply reversing the parent-child relationship between the two adjacent links; the transformation between them must also be properly updated since a joint (*i.e.*, revolute/prismatic) typically constrains the child link's motion w.r.t. *child link's* frame.

The spatial relationships between any two frames can be represented by a homogeneous transformation. A formal

inversion of kinematic chains with multiple links is given by:

$$ {}^{i}_{i-1}T_{\text{inv}} = {}^{i}_{i+1}T^{-1} = \begin{bmatrix} {}^{i}_{i+1}R^{T} & -{}^{i}_{i+1}R^{T}\ {}^{i}_{i+1}p \\ \mathbf{0} & 1 \end{bmatrix}, \quad (1) $$

where ${}^{a}_{b}T$ is the transformation from the link a frame to the link b frame. Hence, ${}_{i+1}{}^{i}_{}T$ is the transformation from the link i frame to the link $i+1$ frame in the original kinematic chain, whereas ${}_{i-1}{}^{i}_{}T_{\text{inv}}$ represents the transformation from the link i frame to the link $i-1$ frame in the inverted kinematic chain. Eq. (1) describes the parent link of the link i in the original kinematic chain becomes the child link of the link i in the inverted kinematic chain, where the transformation of the inverted joint is given by ${}_{i-1}{}^{i}_{}T_{\text{inv}}$.

VKC Construction: After inverting the object kinematic chain $\mathcal{C}^O$, a virtual kinematic chain can be constructed by adding a virtual joint (revolute, prismatic, or fixed) between $\mathcal{F}^O_{at}$ and $\mathcal{F}^R_{ee}$, whose transformation is denoted as ${}^{ee}_{at}T$. Next, a virtual base, whose frame is $\mathcal{F}^V_b$, is further inserted to enable a joint optimization of the locomotion and manipulation. Two perpendicular prismatic joints and a revolute joint are added between the virtual base and the robot base to imitate a planar motion between the mobile base and the ground, while ensuring the virtual kinematic chain ($\mathcal{C}^V$, highlighted by red in Fig. 2c) remains serial.

Fig. 2d shows a constructed VKC in opening a door. The above procedure produces a serial kinematic chain: Its base is a virtual fixed link in the environment, and its end-effector is at the link of the door that connects to the ground. The mobile base and the arm become the links embedded in the chain, and their poses during the door opening are calculated implicitly given the end-effector pose (*i.e.*, the angle of the revolute hinge of the door). In other words, the final robot state is jointly optimized with trajectory, task goal, and kinematic constraints, without further efforts on finding the robot goal state or goal space for trajectory generation. This example demonstrates the possibilities to design robot actions and predicates without explicitly specifying the complete robot goal state (see Section III) while still being plausible at the motion level (see Section IV).

III. Task Planning on VKC

Following the classic formalization of task planning, we describe the environment by a set of states $\mathcal{S}$. Possible transitions between these states are defined by $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{S}$, where a transition $t = \langle s, s' \rangle \in \mathcal{T}$ alters the environment state from $s \in \mathcal{S}$ to $s' \in \mathcal{S}$. The goal of the task planning problem is to identify a sequence of transitions that alters the environment from its initial state $s_0 \in S$ to a goal state $s_g \in S_g$, where $S_g \subseteq S$ is a set of goal states.

We primarily consider the task planning problems in mobile manipulation, which require the robot to account for its base, arm, and the object being interacted (*e.g.*, pick and place, door/drawer opening). We formulate the task planning problems and implement the planning domains using PDDL.

In PDDL, the environment state s is described by a set of *predicates* that hold true. Specifically:

- `(vkcState ?r ?q)`: A sub-chain `?r` (*e.g.*, the base, an arm, or even a VKC) of a VKC is at configuration `?q` in joint space.
- `(objConf ?o ?s)`: An object `?o` is at the configuration `?s` in SE(3).
- `(free ?v)`: The robot end-effector is free to grasp.
- `(carry ?o ?v)`: The robot end-effector is carrying an object `?o`.

In this paper, to focus on demonstrating the benefit of task planning with VKC, we pre-sampled feasible configurations `?s` for all objects and corresponding grasping poses.

Transitions in PDDL are modeled by `actions`. Each `action` takes parameters as input and can be called only when its preconditions hold true. After an `action` is called, its effect indicates how the states in the current environment change from preconditions. Thanks to the advantages introduced by the VKC, three simple action definitions—`goto-vkc`, `pick-vkc`, and `place-vkc`—are sufficient to handle various mobile manipulation tasks, from pick-and-place in different setups to foot-arm coordinated and constrained motions (*e.g.*, door/drawer opening). Below is an example of the definitions for three actions; see Figs. 4b and 4c and Section V-A for a comparison between VKC-based PDDL and a standard PDDL for mobile manipulators.

```
(:action goto-vkc
   :parameters (?r ?from ?to)
   :precondition (vkcState ?r ?from)
   :effect (and (vkcState ?r ?to)
                (not (vkcState ?r ?from))))

(:action pick-vkc
   :parameters (?o - obj ?s - state ?v - vkc)
   :precondition (and (objConf ?o ?s)
                      (free ?v))
   :effect  (and (carry ?o ?v)
            (not (objConf ?o ?s))
            (not (free ?v))))

(:action place-vkc
   :parameters (?o - obj ?s - state ?v - vkc)
   :precondition (and (carry ?o ?v)
                      (not (occupied ?s)))
   :effect  (and (not (carry ?o ?v))
                      (objConf ?o ?s)
                      (free ?v)))
```

IV. VKC Facilitates Motion Planning

The conventional task planning setup usually assumes a robot already knows how to execute the actions defined in the task domain and, therefore, does not generate actionable motion trajectories for the robot. However, in practice, this assumption does not always hold as many abstract actions defined in the task domain are difficult to be instantiated at the motion level. This section discusses how the `actions` defined using VKC can properly form a motion planning problem solvable by existing motion planners.

A. From Task to Motion

We start by making the connections between the action semantics and the actual manipulation behaviors, followed by explaining how the predicates and variables in the action definitions are processed by motion planners.

`goto-vkc` ($\mathtt{r}, \mathbf{q}_1, \mathbf{q}_2$): This predicate moves the VKC from the current pose q_1 to a desired pose q_2 for a chain `r`. It represents the tasks that do not require interaction with the environment, wherein the VKC structure remains unchanged. Pure navigation is a typical action falling into this category. For example, `goto-vkc` (`base`, q_1^b, q_2^b) moves the robot to the location specified in q_2^b. Another example is to manipulate a picked object from the current pose q_1 to a certain pose q_2, *i.e.*, `goto-vkc` (`vkc`, q_1, q_2)

`pick-vkc` (`object`, s, `vkc`): This predicate moves the VKC to the `object` to be manipulated and extends the current VKC structure by adding a virtual joint to connect the `object` and the arm's end-effector at state s. Here, the state could be interpreted as a grasping pose, the transformation between the robot gripper and the object to be manipulated (*i.e.*, ${}^{ee}_{at}T$). `pick-vkc` represents the group of tasks that require mobile manipulators to interact with the environment, *e.g.*, picking up an object or grasping a handle.

`place-vkc` (`object`, s, `vkc`): This predicate moves the `object` connected to `vkc` to a goal pose s, while the object to be manipulated is incorporated into the VKC and imposes kinematic constraints to the planner. Once reaching the goal pose, `place-vkc` breaks the current VKC at the virtual joint where it connects the mobile manipulator and the `object`, and the `object` will be placed at where it was disconnected from VKC. `place-vkc` represents the group of tasks that mobile manipulators stop interacting with the environment, such as placing an object on the table.

In motion planning, configuration space Q describes the environment state. Q's dimension n equals to VKCs' degrees of freedom. A collision-free subspace $Q_{\text{free}} \subseteq Q$ is the space that VKCs can traverse freely without colliding with the environment or itself. The problem of motion planning on VKC is equivalent to finding a collision-free path $\mathbf{q}_{1:T} \in Q_{\text{free}}$ from the initial pose $\mathbf{q}_1 \in Q_{\text{free}}$ to reach the final state $\mathbf{q}_T \in Q_{\text{free}}$. Each action predicate requires to form a motion planning problem due to the kinematic structure changes.

B. Optimization-based Motion Planning

Finding a collision-free path $\mathbf{q}_{1:T} \in Q_{\text{free}}$ for given tasks can be formulated by trajectory optimization, *e.g.*, CHOMP [27] and TrajOpt [28]. The objective function of the trajectory optimization can be formally expressed as:

$$\min_{\mathbf{q}_{1:T}} \sum_{t=1}^{T-1} ||W_{\text{vel}}^{1/2} \ \delta\mathbf{q}_t||_2^2 + \sum_{t=2}^{T-1} ||W_{\text{acc}}^{1/2} \ \delta\dot{\mathbf{q}}_t||_2^2, \tag{2}$$

where $\mathbf{q}_{1:T}$ is the trajectory sequence $\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_T\}$, and $\mathbf{q}_t$ the VKC state at the t^{th} time step. We penalize the overall velocities and acceleration of every joint with diagonal weights W_{vel} and W_{acc} for each joint, respectively.

Meanwhile, the constructed VKC should also be subject to kinematic constraints of the robot and the environment:

$$h_{\text{chain}}(\mathbf{q}_t) = \mathbf{0}, \ \forall t = 1, 2, \ldots, T, \tag{3}$$

including forward kinematics and closed chain constraints. We can formulate the task goal as an inequality constraint:

$$||f_{\text{task}}(\mathbf{q}_T) - \mathbf{g}||_2^2 \leqslant \xi_{\text{goal}}, \tag{4}$$

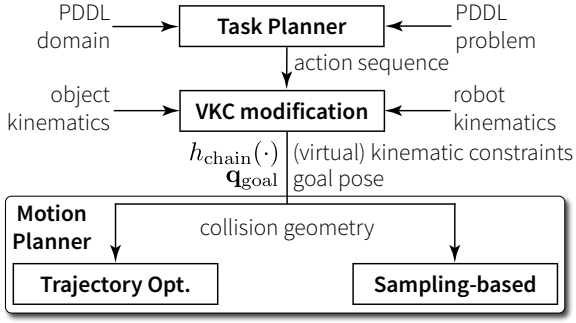


Fig. 3: **The computing logic of instantiating the `actions` in a task plan to trajectories at the motion level.** Each action symbol encodes a (virtual) kinematic chain and a goal pose, which are sufficient for a motion planner given the environmental constraints.

which bounds the element-wise squared ℓ^2 norm between the final state in the goal space $f_{\text{task}}(\mathbf{q}_T)$ and the task goal $\mathbf{g} \in \mathbb{R}^k$ ($k \leqslant n$) with a tolerance ξ_{goal}. The function $f_{\text{task}} : Q \to \mathbb{R}^k$ is a task-dependent function that maps the joint space of a VKC to the goal space that differs from task to task. This definition relaxes hard constraints of goal state and optimized the other $n - k$ states with objective function Eq. (2). Of note, Eqs. (3) and (4) are not the only forms of constraints that a VKC-based approach can incorporate; in fact, it is straightforward to add additional task constraints to the same optimization problem in Eq. (2), depending on various task-specific requirements. In this paper, we further impose several additional safety constraints, including joint limits, bounds for joint velocity and acceleration, and link-link and link-object collisions. A more detailed analysis of the optimization framework can be found in Jiao *et al*. [29].

C. Sampling-based Motion Planning

Alternatively, motion planning on VKC can also be viewed as a search procedure in the configuration space $\mathcal{Q}_{\text{free}}$. Given a path planning problem within $\mathcal{Q}_{\text{free}}$, a sampling-based method would attempt to find a set of collision-free way points that start from an initial configuration $\mathbf{q}_0 \in \mathcal{Q}_{\text{free}}$ and end in the goal configuration $\mathbf{q}_{\text{goal}} \in \mathcal{Q}_{\text{free}}$.

Rapidly-exploring Random Tree (RRT) is a probabilistically complete search algorithm that incrementally expands a collection of directional nodes $\mathcal{T}$ to explore space [30]. In this paper, we adopted a RRT-connect algorithm [31] from the Open Motion Planning Library (OMPL) [32] as our sampling-based motion planner, which initiates exploration from $\mathbf{q}_0$ and $\mathbf{q}_{goal}$ concurrently.

Unlike the optimization-based method mentioned in Section IV-B, way-points collected by RRT-connect are not smoothed by an objective function during search; instead, interpolation was performed after the search is complete for a smooth trajectory to be executed on a mobile manipulator.

Fig. 3 summarizes the computing logic of instantiating the `actions` to motion trajectories. The action sequence produced by the task planner encodes how the VKC changes over each action and its desired goal pose. Together with environmental constraints (*e.g.*, the actual robot kinematics and the objects' geometry), the information provided by the VKC-based task planner is sufficient for a typical motion planner to produce a feasible trajectory from $\mathbf{q}_0$ to $\mathbf{q}_{\text{goal}}$.

V. Experiment

We conduct a series of experiments to evaluate the efficacy of the proposed VKC perspective for planning mobile manipulation tasks in simulations. The first experiment compares the designs of PDDL definition with VKC or without VKC and their corresponding planning efficiency. Since the action definitions can be arbitrarily abstract at the symbolic task level, we further validate the VKC-based action design in the second experiment that it indeed provides sufficient information for motion planners to produce feasible trajectories. Finally, in the third experiment, we showcase how the VKC perspective empowers more complex task planning.

A. Simplifying Task Domain

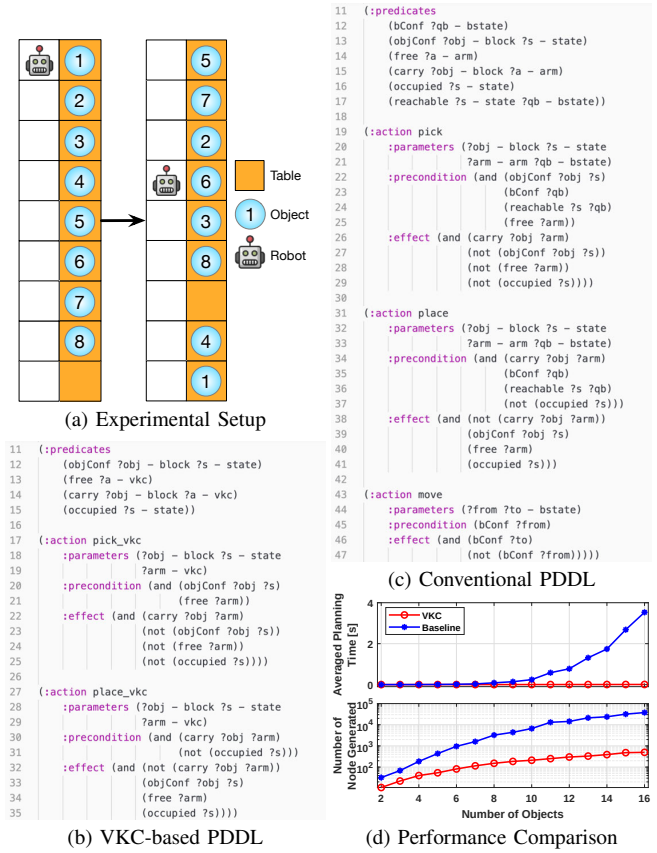
Since the VKC perspective treats the base, the arm, and the object to be manipulated as a whole, designing the planning domain becomes much simpler. In this experiment, we focus on an object-arrangement task, where the robot is tasked to re-arrange m objects on $m+1$ tables into the desired order while satisfying the constraint that each table can only support one object. Fig. 4a shows a typical example of this task's initial and goal configuration with $m = 8$ objects, randomly sampled in each experimental trial.

Fig. 4b shows a PDDL domain designed by the actions mentioned in Section III, which requires less predicates and provides more abstract actions compared with those designed by conventional domain definition shown in Fig. 4c. Specifically, the conventional method would require (i) more predicates to describe the mobile base's states and thus more complex preconditions for actions, (ii) one more action to control the mobile base, and (iii) more parameters for other actions. To solve for a task plan, we adopt the Iterated Width Search (IWS) algorithm [33]; it is a width-limited version of the Breadth First Search (BFS) that repeatedly runs with increasing width limits until a feasible task plan is found. If no feasible task plan could be found within the maximum width limit of the IWS, a traditional BFS with no width limit will be deployed to search for a solution.

In experiments, we run 50 trials for each setup; see the result summary in Fig. 4d. As the task complexity increases, the average planning time and the number of nodes generated in search (*i.e.*, memory required) increase relatively slowly for the VKC-based task plan. In comparison, the baseline using conventional methods increases much more rapidly.

This result is evident. As we can see in Fig. 4d, planning in the non-VKC version of the task domain requires exploring more nodes at each depth level to find a plausible pose for the mobile base. It also requires more actions to accomplish the task, which further yields a deeper depth during the search. Suppose there are N nodes on average to be generated at each depth level of the search algorithm, and a feasible solution is found at depth d, the total number of nodes being generated is N^d. In theory, when the search algorithm performs in the VKC domain, the total number of generated node is $(c_1 N)^{c_2 d}$, where $c_1 \leqslant 1, c_2 \leqslant 1$. In the task with 16 objects, our experiment empirically finds $c_1 = 0.75$ and $c_2 = 0.22$ on average over 50 trails.

Taken together, the results in the first experiment demonstrated that VKC-based task planning requires much fewer explorations in both width and depth during the search algorithm, therefore achieving higher efficacy with less memory.



(a) Experimental Setup

```
(:predicates
    (objConf ?obj - block ?s - state)
    (free ?a - vkc)
    (carry ?obj - block ?a - vkc)
    (occupied ?s - state))

(:action pick_vkc
    :parameters (?obj - block ?s - state
                 ?arm - vkc)
    :precondition (and (objConf ?obj ?s)
                       (free ?arm))
    :effect (and (carry ?obj ?arm)
                 (not (objConf ?obj ?s))
                 (not (free ?arm))
                 (not (occupied ?s))))

(:action place_vkc
    :parameters (?obj - block ?s - state
                 ?arm - vkc)
    :precondition (and (carry ?obj ?arm)
                       (not (occupied ?s)))
    :effect (and (not (carry ?obj ?arm))
                 (objConf ?obj ?s)
                 (free ?arm)
                 (occupied ?s)))
```

(b) VKC-based PDDL

```
(:predicates
    (bConf ?qb - bstate)
    (objConf ?obj - block ?s - state)
    (free ?a - arm)
    (carry ?obj - block ?a - arm)
    (occupied ?s - state)
    (reachable ?s - state ?qb - bstate))

(:action pick
    :parameters (?obj - block ?s - state
                 ?arm - arm ?qb - bstate)
    :precondition (and (objConf ?obj ?s)
                       (bConf ?qb)
                       (reachable ?s ?qb)
                       (free ?arm))
    :effect (and (carry ?obj ?arm)
                 (not (objConf ?obj ?s))
                 (not (free ?arm))
                 (not (occupied ?s))))

(:action place
    :parameters (?obj - block ?s - state
                 ?arm - arm ?qb - bstate)
    :precondition (and (carry ?obj ?arm)
                       (bConf ?qb)
                       (reachable ?s ?qb)
                       (not (occupied ?s)))
    :effect (and (not (carry ?obj ?arm))
                 (objConf ?obj ?s)
                 (free ?arm)
                 (occupied ?s)))

(:action move
    :parameters (?from ?to - bstate)
    :precondition (bConf ?from)
    :effect (and (bConf ?to)
                 (not (bConf ?from)))))
```

(c) Conventional PDDL

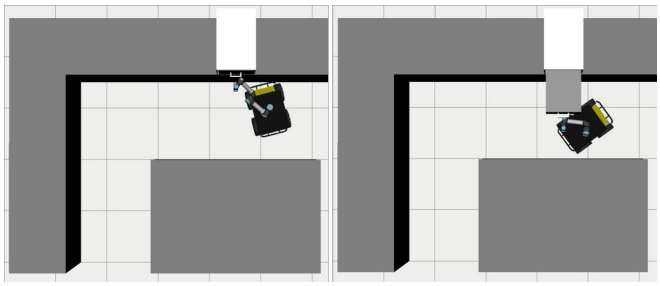
(d) Performance Comparison

Fig. 4: **VKC-based domain specification improves the task planning efficacy.** (a) An example setup of re-arranging 8 objects on 9 tables; one table can only support one object. (b) The VKC-based PDDL specification has less variables and more abstract actions than (c) a conventional PDDL specification. (d) The VKC-based domain specification allows a solver to search for a feasible plan for tasks of re-arranging 2 to 16 objects with significantly less time and generated nodes in search (*i.e.*, less memory).

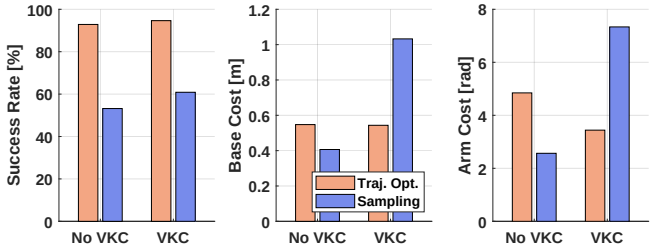
B. Improving Mobile Manipulation

In general, actions that are more abstract and with fewer variables in the planning domain specification would lead to more efficient task planning, but simultaneously could result in less success rate in generating feasible plans at the motion level. In this experiment, we validate that the VKC-based task planning provides efficacy at the task level and maintains a high success rate at the motion level. Based on the generated task plans (*i.e.*, action sequences) and the encoded information (as described in Section IV), we apply a trajectory optimization-based motion planner and a sampling-based motion planner and evaluate how well they can produce feasible motion trajectories for the given task.

Specifically, we consider the task of pulling opening a drawer; see Fig. 5a. The task plans: (i) `place-vkc` (`drawer`, s^1_{d}, `vkc`), (ii) `move` (q^0_{b}, q^1_{b}) + `place` (`drawer`, s^1_{d}, `arm`, q^1_{b}), are produced by two PDDLs specified with and without VKC, respectively. We compare the success rate of executing the trajectories planned by trajectory optimization and sampling motion planning methods described in Sections IV-B and IV-C, as well as the base and arm cost measured by the distances they travel; see Fig. 5b.



(a) The drawer opening task.



(b) Success rates and the corresponding base and arm movements.

Fig. 5: **Instantiating the task plans to motions** in (a) a drawer opening task. The domains, one with VKC and the other without, are specified similar to Figs. 4b and 4c. The generated task plans are processed by an optimization-based and a sampling-based motion planner. (b) Task success rates, and base and arm costs. Failure cases for sampling include time-out for both sub-tasks: 5 mins for reach, 50 seconds for open

The trajectory optimization-based motion planner can produce feasible trajectories for the given task with high success rates; the produced trajectories are more efficient in terms of shorter base and arm traveling distances. Typically, sampling-based motion planners would struggle in incorporating kinematic and safety constraints due to naturally unconstrained configuration spaces, which need extra effort to accommodate extra kinematic constraints [34], making it less suitable for such tasks. However, it is still more successful in producing feasible trajectories under the VKC specification compared with the setting without VKC. The most significant drawbacks of sampling-based motion planners are the high execution costs and violation of safety limits.

C. Solving Tasks with Multiple Steps

Complex multi-step mobile manipulation tasks with long action sequences can also be easily accomplished using the action set introduced by the VKC-based task planner described in Section III. These actions contain high-level task semantics that could be adapted to various tasks; *e.g.*, attaching to the doorknob could be expressed by a `pick-vkc` action, and open the door to a certain angle could be expressed by a `place-vkc`.

Fig. 6a qualitatively shows a complex multi-step task planning using the VKC-based domain specification and instantiating that to motions. For a more fair comparison, in addition to the initial and goal state of the environment, both the VKC and non-VKC methods are provided with the (identical) grasping poses for all movable objects, but not the corresponding robot state. In this task, a mobile manipulator needs to (i) grasp the stick, (ii) fetch the cube under the table using the stick, which is otherwise challenging to reach, (iii) move the cube outside, (iv) place the stick down, (v) grasp the cabinet and open it, (vi) place the cube inside the cabinet, and (vii) close the cabinet door. At each trial, the mobile manipulator is randomly placed in the environment.

Fig. 6b illustrates that the above complex multi-step task can be accomplished by using only two abstract actions defined based on VKC, one action in each step. Without the VKC perspective, significantly more effects must first be devoted to designing the planning domain. Furthermore, to ensure successful planning of actions that require foot-arm coordination, each step may require several actions to be executed together; see Table I for a comparison between the two setups. Even after the additional efforts of specifying base pose from a feasible region, its accumulated success rate at the motion level produced by the corresponding actions still underperforms the VKC version, shown in Fig. 6c. Without VKC, the motion planner particularly suffers at step 2 when the robot needs to fetch the cube in a confined space, as it requires the planner to deliver proper navigation and manipulation with excellent foot-arm coordination (*i.e.*, coordinating `move` and `place`). In sum, this experiment demonstrates that VKC-based mobile task planning for mobile manipulation tasks is advantageous by simplifying domain specification and improving motion planning.

TABLE I: **Actions and predicates in the defined planning domains.** Without VKC, more actions must be specified, and extra predicates are required for generating a feasible task plan.

Setup	Group	Notation	Description
VKC	Actions	`pick-vkc`(o,s,v) `place-vkc`(o,s,v)	see Section III
	Predicates	`Graspable`(o,v)	Check if robot v is able to grasp object o v
		`RigidObj`(o)	Check if object o is rigid object
		`ArtiObj`(o)	Check if object o is articulated object
		`ToolObj`(o)	Check if object o could be used as a tool
		`Occupied`(s)	Check if a position s being occupied
		`Carried`(o)	Check if an object o is carried by robot
		`ContainSpace`(o,s)	Check if a position s being contained in the object o
Non-VKC	Actions	`move`(s_1,s_2)	Move robot from s_1 to s_2
		`pick`(o,s_1,s_2)	Pick the object o at location s_1 given robot state s_2
		`place`(o,s_1,s_2)	Place the object o at location s_1 given robot state s_2
		`open`(o,s_1,s_2)	Open the object o at location s_1 given robot state s_2
		`close`(o,s_1,s_2)	Close the object o at location s_1 given robot state s_2
	Extra Predicates for Non-VKC	`HasTool`(s)	Check if the robot at current state s holding a tool
		`AbleToPick`(s)	Check if the robot at state s is able to do pick action
		`Reachable`(o,s)	Check if object o is reachable by mobile base at state s

VI. Discussion and Conclusion

We present a VKC perspective that improves the domain specifications for mobile manipulation task planning. By integrating the kinematics of the mobile base, the arm, and the object to be manipulated into a single VKC, more abstract action symbols become possible and fewer predicates/variables/intermediate goals are required in designing the planning domain. In a series of experiments, we demonstrate that the VKC-based domain specification using PDDL supports more efficient task planning, works better with existing motion planners, and scales up to more complex tasks compared with the one without VKC. We argue the proposed VKC perspective has significant potential in promoting mobile manipulation in real-world daily tasks. Our future work will explore TAMP on VKC.

Acknowledgement: We thank Jiang Xin of the UCLA ECE Department for discussions on trajectory optimization.

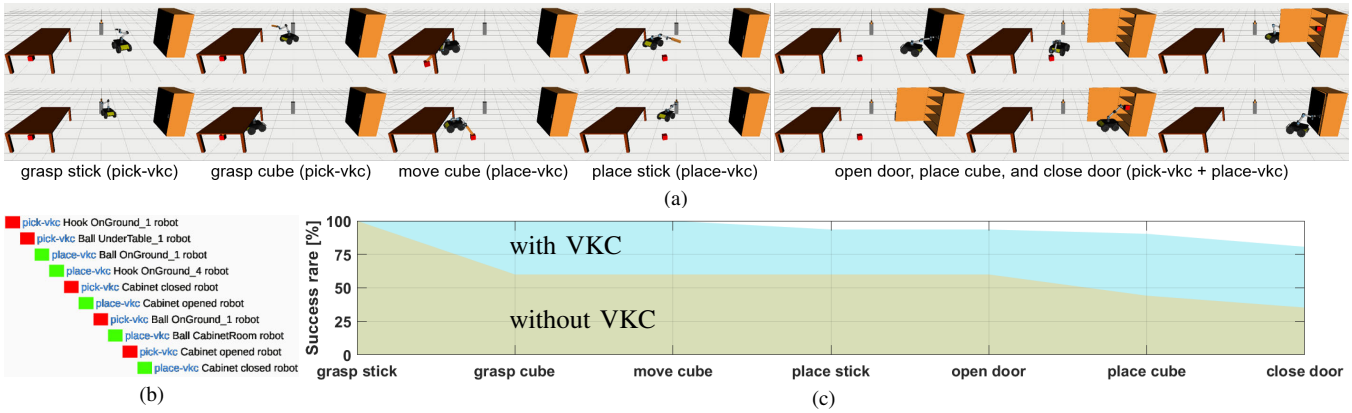


Fig. 6: (a) The VKC-based task planner can easily scale up to a complex multi-step task, which can be (b) succinctly expressed by merely two actions defined based on the VKCs. (c) More abstract action definitions introduced by VKC instantiate better at the motion level, possessing an excellent foot-arm coordination in each step of the task. Without VKC, to ensure successful planning for tasks that require foot-arm coordination, several actions must be executed together to complete certain steps in the task.

REFERENCES

[1] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[2] E. Karpas and D. Magazzeni, "Automated planning for robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 417–439, 2020.

[3] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[4] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "Shop2: An htn planning system," *Journal of Artificial Intelligence Research*, vol. 20, pp. 379–404, 2003.

[5] M. Edmonds, F. Gao, H. Liu, X. Xie, S. Qi, B. Rothrock, Y. Zhu, Y. N. Wu, H. Lu, and S.-C. Zhu, "A tale of two explanations: Enhancing human trust by explaining robot behavior," *Science Robotics*, vol. 4, no. 37, 2019.

[6] H. Liu, C. Zhang, Y. Zhu, C. Jiang, and S.-C. Zhu, "Mirroring without overimitation: Learning functionally equivalent manipulation actions," in *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[7] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[8] M. Fox and D. Long, "Pddl2. 1: An extension to pddl for expressing temporal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[9] K. Shankar, *Kinematics and Local Motion Planning for Quasi-static Whole-body Mobile Manipulation*. PhD thesis, California Institute of Technology, 2016.

[10] D. M. Bodily, T. F. Allen, and M. D. Killpack, "Motion planning for mobile robots using inverse kinematics branching," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2017.

[11] S. Chitta, B. Cohen, and M. Likhachev, "Planning for autonomous door opening with a mobile manipulator," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2010.

[12] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, 2021.

[13] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2011.

[14] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2011.

[15] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *International Journal of Robotics Research (IJRR)*, vol. 37, no. 1, pp. 104–136, 2018.

[16] 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 *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2014.

[17] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning.," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[18] M. Toussaint, K. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[19] J. Bidot, L. Karlsson, F. Lagriffoul, and A. Saffiotti, "Geometric backtracking for combined task and motion planning in robotic systems," *Artificial Intelligence*, vol. 247, pp. 229–265, 2017.

[20] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez, "Learning to guide task and motion planning using score-space representation," *International Journal of Robotics Research (IJRR)*, vol. 38, no. 7, pp. 793–812, 2019.

[21] J. Pratt, P. Dilworth, and G. Pratt, "Virtual model control of a bipedal walking robot," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 1997.

[22] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *International Journal of Robotics Research (IJRR)*, vol. 20, no. 2, pp. 129–143, 2001.

[23] N. Likar, B. Nemec, and L. Žlajpah, "Virtual mechanism approach for dual-arm manipulation," *Robotica*, vol. 32, no. 6, 2014.

[24] Y. Wang, C. Smith, Y. Karayiannidis, and P. Ögren, "Cooperative control of a serial-to-parallel structure using a virtual kinematic chain in a mobile dual-arm manipulation application," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[25] Y. Wang, C. Smith, Y. Karayiannidis, and P. Ögren, "Whole body control of a dual-arm mobile robot using a virtual kinematic chain," *International Journal of Humanoid Robotics*, vol. 13, no. 01, 2016.

[26] A. Laurenzi, E. M. Hoffman, M. P. Polverini, and N. Tsagarakis, "An augmented kinematic model for the cartesian control of the hybrid wheeled-legged quadrupedal robot centauro," *Robotics and Automation Letters (RA-L)*, 2019.

[27] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: gradient optimization techniques for efficient motion planning," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2009.

[28] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research (IJRR)*, vol. 33, no. 9, pp. 1251–1270, 2014.

[29] Z. Jiao, Z. Zhang, X. Jiang, D. Han, S.-C. Zhu, Y. Zhu, and H. Liu, "Consolidating kinematic models to promote coordinated mobile manipulations," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[30] S. M. Lavalle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," in *Intenational Workshop on Algorithmic and Computational Robotics*, 2000.

[31] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2000.

[32] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[33] N. Lipovetzky and H. Geffner, "Width-based algorithms for classical planning: New results," in *Proc. ECAI*, 2014.

[34] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *International Journal of Robotics Research (IJRR)*, vol. 38, no. 10-11, pp. 1151–1178, 2019.