A Closed-Chain Approach to Generating Affordance Joint Trajectories for Robotic Manipulators

Janak Panthi, Farshid Alambeigi, *Member, IEEE*, and Mitch Pryor, *Member, IEEE*

***Abstract*—Robots operating in unpredictable environments require versatile, hardware-agnostic frameworks capable of adapting to various tasks. While a recent screw-based affordance approach shows promise, it faces challenges in avoiding undesirable configurations, singularity navigation, and task success prediction. To address these limitations, we propose a novel framework that incorporates gripper orientation control and generates complete joint trajectories in real time for screw-based task affordance execution. Our method models the affordance and manipulator as a closed-chain mechanism, introducing an innovative approach to solving closed-chain inverse kinematics. It encapsulates task constraints and simplifies task definitions, while remaining hardware and robot agnostic, robust to errors, and invariant to the initial grasp. We validate our framework with simulations on a UR5 robot and real-world implementation on a Boston Dynamics Spot robot. Our experiments demonstrate rapid joint trajectory generation (0.0077–0.098 s) for various tasks, including a 420° valve turn with consideration of the gripper orientation. Comparison with the state-of-the-art methods shows a 4x improvement in planning time, reduced joint movement, and achievement of greater task goals. Video demonstrations and the open-source code for this project are available online.**

***Index Terms*—Affordance, closed-chain, constrained manipulation, inverse kinematics (IK), orientation control, virtual chain, virtual joint, virtual kinematic chain (VKC), virtual link.**

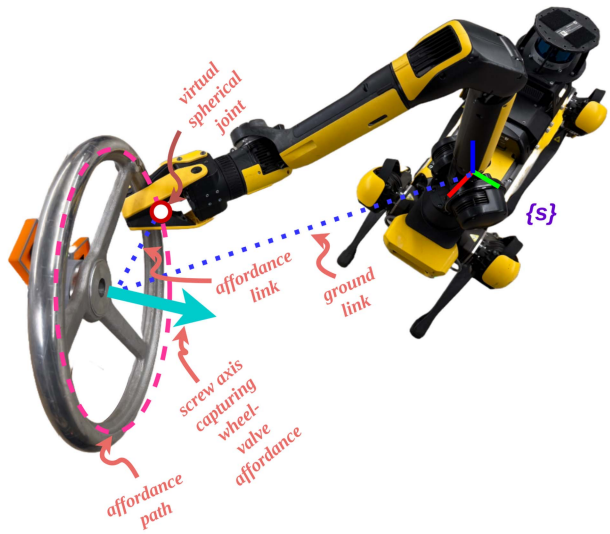


Fig. 1. Manipulator performing a screw-modeled task can be represented as a closed-chain mechanism, comprising the robot, a virtual spherical joint for EE orientation freedom, and a terminal joint that captures the task's screw-modeled affordance.

I. Introduction and Related Work

DEPLOYING robots in real-world environments [1], [2], [3], [4] requires them to complete a large variety of contact manipulation tasks (opening doors, sliding drawers, turning valves, moving objects, etc.) that involve constrained motion along significantly long paths. We must assume the presence of uncertainty, which includes not only the initial location of the manipulated object but also the constrained contact task model's *affordance* [5]. Uncertainties in the affordance may only be apparent once the object is grasped and the task begins. Consequently, during task execution, replanning must be prompt to account for new and updated constraints due to runtime uncertainties. This replanning must also consider specific end-effector (EE) pose requirements, such as rotating the EE in tandem with turning a valve. Failure to address these uncertainties could result in undesirable and potentially hazardous configurations, such as impeding a valve or obstructing a door. Our solution utilizes a novel, versatile framework that seamlessly adapts to diverse manipulation tasks. It leverages the notion of screw-based task affordances [6], including them in the mechanics of the closed chain generated when the task is undertaken; see Fig. 1.

A. Affordances

The term "affordance" originated in psychology literature, describing the functional merit an environment provides to its inhabitants [7]. For example, a book affords reading, a hammer affords hitting, and a valve affords turning. This concept has also found diverse applications in robotics, ranging from learning potential actions for observed objects [8], [9], [10] to planning manipulation based on known effects of actions [11], [12]. There are also efforts focused on detecting affordances via images [8], [10], [13], [14], [15], including articulated objects [16], [17], [18], and also estimating pose and affordance type using

Received 27 February 2025; accepted 2 June 2025. Date of publication 24 June 2025; date of current version 18 July 2025. This article was recommended for publication by Associate Editor K. Lee and Editor M. Yim upon evaluation of the reviewers' comments. *(Farshid Alambeigi and Mitch Pryor contributed equally to this work.) (Corresponding author: Janak Panthi.)*

The authors are with the Walker Department of Mechanical Engineering and Texas Robotics, The University of Texas at Austin, Austin, TX 78712 USA (e-mail: jpanthi@utexas.edu).

Video demonstrations for this project are available at https://youtu.be/Ukv93hbNrOM and the open source code is available online at https://github.com/UTNuclearRoboticsPublic/closed-chain-affordance.

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2025.3582832, provided by the authors.

Digital Object Identifier 10.1109/TRO.2025.3582832

screw theory [19], [20]. While some of these works represent articulated affordances as screw axes, they rely on processed RGB-D image streams of the object and do not directly address physical interaction [6].

To enable the deployment of robots in uncertain physical environments, affordance templates (AT) were introduced for predefined tasks, allowing robots to perform complex tasks via an extended sequence of commands [5]. ATs serve as spatial representations of operator-adjustable robot task objectives and parameters, capturing object affordances within object-centric coordinate frames, and have proven successful in diverse platforms [21]. Early implementations were task-dependent, required human supervision, and defined affordances with a set of EE waypoints, leading to computational inefficiencies. These limitations were addressed in [6], enhancing the framework by adopting screw theory to model affordances, and thus reducing their complexity to just a few parameters. The framework (called AP) demonstrates robustness against runtime uncertainties (perceived as forces and torques by the EE) by dynamically adapting EE velocities during runtime. However, despite its promising capabilities, the system faces two primary limitations that hinder real-world application.

First, the online nature of the AP framework, which involves computing EE velocities in real time for small steps, and its runtime adaptation solely through software compliance, present challenges that cannot be resolved by force control alone. These limitations include navigation around robot singularities, accurate prediction of task success, and selection of optimal starting poses. Our approach to resolving these issues involves real-time complete joint trajectory generation such that a comprehensive overview of the affordance path is consistently available and any changes to avoid undesirable configurations can be promptly made during runtime.

Second, the EE path generated through this framework is essentially a sweep around (or along) the affordance screw axis (for example, for a wheel valve about its rotational axis), and the amount of sweep solely dictates the orientation of the EE. The system lacks the ability to control or free the gripper orientation along the affordance path. This limitation is critical for two reasons. 1) Precise control over gripper orientation along the affordance path is essential for tasks, such as object insertion, precision placement, turning a valve, etc. 2) As we show in Section VI-B, many manipulation tasks, such as sliding a stool along the floor, are feasible with relaxed gripper orientation constraints. Therefore, the range of solutions for such tasks is much more than what is available in the AP work.

An intuitive example underscoring the importance of addressing these limitations is illustrated in Fig. 2, which depicts different ways to turn a valve. Human beings naturally adapt their approach based on the specific application context. In cramped environments, one might choose to turn the valve while keeping the hand orientation constant [see Fig. 2(a)]. When access to the valve surface is restricted, an alternative approach involves pushing on the valve spoke without concern for maintaining a specific hand orientation [see Fig. 2(b)]. In other cases, one might turn the valve such that the entire hand (or only some aspect of its orientation) rotates in tandem with it [see Fig. 2(c)]. For a robot

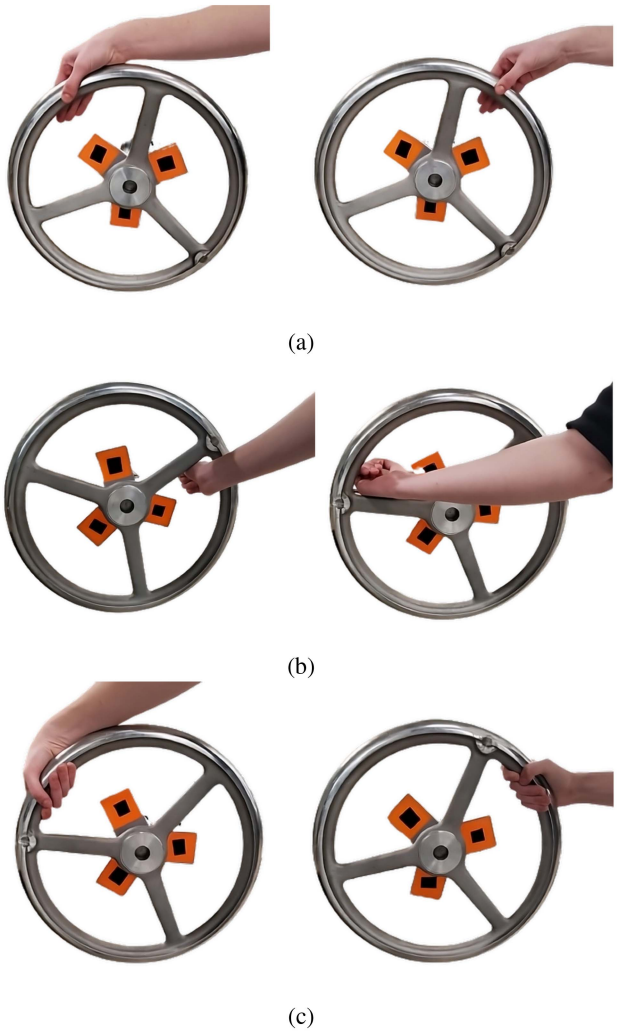


Fig. 2. Methods of turning a valve. (a) Preserving hand orientation. (b) Not preserving hand orientation. (c) Rotating hand in tandem with the valve. In each subfigure, the valve turn start configuration is shown on the left-hand side and the end configuration on the right-hand side.

to effectively perform real-world tasks, it is essential that it has such capabilities in its repertoire. In this work, we address this challenge by incorporating intuitive gripper orientation control (or freedom) during joint trajectory generation. Specifically, we demonstrate the efficacy of our work by performing real-world experiments that involve all the scenarios, as shown in Fig. 2.

Lastly, we also remark that although, like [6], we adopt screw theory to model affordances, a key distinction exists between our methodologies. While they capture the EE movement along the affordance path as the twist of the screw that embodies the affordance, we model the affordance as a joint extending our robot.

B. *Affordance Chain*

In our work, we incorporate the task affordance into an imaginary kinematic chain that extends our robot and forms a unified closed-chain mechanism. The augmentation of imaginary

mechanisms, links, or joints to an existing robot for imposing movements or gathering information is well documented in the literature [22], [23], [24], [25], [26], [27], [28], including the notion of "virtual kinematic chains (VKC)" [25]. Some studies [29], [30] utilize a VKC with ≤ 6 degrees of freedom (DoF) to represent the motion between dual arms for cooperative and/or whole-body control of a mobile manipulator. Similarly, Laurenzi et al. [26] employed a VKC to control a quadruped robot with wheels. Recently, Jiao et al. [27] demonstrated the use of VKCs with optimization and sampling techniques to improve the efficiency of mobile manipulation task planning. Although these works showcase the effectiveness of VKCs, their approaches are task-dependent and result in an open-chain virtual manipulator that requires task priority controllers or optimization techniques. Our approach differs significantly in the following two key aspects.

1) *Task-Independent Modeling:* This stems from our utilization of screw theory to model affordances.
2) *Closed-Chain Kinematics:* We employ a closed-chain kinematics formalism that integrates task constraints.

While previous works, such as [22], [23], and [24], employed "virtual chains" to close open-chain manipulators and perform closed-chain differential kinematics, their definition adhered to specific criteria, including: 1) the virtual chain is open, 2) it includes joints with linearly independent normalized screws, and 3) the presence of the virtual chain does not alter the mobility of the real kinematic chain. An illustrative example is the addition of a virtual prismatic-prismatic-prismatic-spherical (PPPS) chain (comprising three prismatic and three revolute joints) to a robot's kinematic chain, encapsulating its EE spatial motion and transforming it into a closed-chain mechanism [22], [23]. In our work, although we close the manipulator kinematic chain with an imaginary chain, this added chain is not a virtual chain as defined above in that it lacks linearly independent normalized screws and restricts the robot's mobility by confining the EE to the affordance path. Our interest is to encompass the manipulator and the contact task affordance in a unified closed-chain mechanism. Furthermore, although similar to [22], [23], [24], and [31], we leverage the screw-based Davies circulation law [32] for closed-chain velocity kinematics, unlike their reliance on numerical integration techniques, such as Euler integration, we adopt a root-finding method with a novel approach to handling closed-chain closure errors. As the following experiments show, this method offers clear advantages in terms of fast convergence and accuracy.

To avoid ambiguity with existing terminology, such as virtual mechanisms, virtual chains, and VKCs (and to highlight the key role of the pivotal joint in our modeling), we introduce the term *affordance chain* for the imaginary chain that closes the manipulator chain. The affordance chain encapsulates both the task's affordance and the remaining DoF of the EE. By augmenting a manipulator chain with a VKC—that effectively captures both the mechanics of the screw-modeled task and the EE orientation freedom—we achieve a compact mathematical representation of the governing closed-chain constraints. This enables implicit task planning, analogous to turning a "task crank" in a closed-chain mechanism.

C. Constrained Manipulation

Here, we review both machine learning (ML)-based and traditional constrained manipulation approaches. Among ML-based methods, a recent study [33] utilizes deep predictive learning with a modular motion generation model to execute complex manipulation tasks, such as door opening. In [34], the researchers propose a neural planning framework that integrates task and scene representations with sampling-based planners, improving constrained motion planning. Similarly, Lin et al. [35] introduced LAPPLAND, a framework that combines reinforcement learning (RL) and imitation learning for robotic manipulation. LAPPLAND leverages goal-conditioned action primitives for interpretability and has been validated in real-world scenarios including door opening.

While ML-based approaches show promise, they face key challenges, including extensive task-specific training, the sim-to-real gap, and limited generalization to out-of-distribution scenarios [36]. For instance, the modules in [33] are specialized and trained for subtasks, such as door-approaching and door-passing. Methods proposed in [34] require separate training for distinct environments, such as bartender and kitchen, and the method in [35] relies heavily on demonstration data, making it less adaptable to novel tasks.

Large language models (LLMs) have also shown potential for task and motion planning. For example, Ahn et al. [37] grounded LLMs in contextual robot capabilities and environmental states using RL, while Rana et al. [38] employed 3-D scene graphs and used LLMs for semantic search of task-relevant subgraphs, integrating classical path planning with iterative replanning. In [39], the researchers propose RT-2, a vision–language–action model that extends vision–language models by co-fine-tuning them on robotic trajectory data and Internet-scale vision–language tasks, representing robotic actions as text tokens to enable instruction-following robotic policies. However, RT-2 remains limited to in-distribution skills seen in training data and suffers from high computational costs, making real-time inference a challenge. In contrast, our proposed approach requires no training and offers a versatile solution capable of handling a wide range of tasks, demonstrated by over ten real-world tasks in this work. Moreover, as demonstrated via experiments, our method is able to compute long joint trajectories in real time for task execution.

Beyond ML-based methods, traditional open-source planners, such as Open Motion Planning Library (OMPL[1]) and the MoveIt Motion Planning Framework[2], provide general solutions for constrained manipulation, allowing users to impose or relax EE pose constraints. However, these methods do not leverage screw-based modeling/constraints [40] as we do and require manually defined EE waypoints, making task specification cumbersome. Recent works, sequential path stepping (SPS) and direct screw sampling (DSS) from [40], which handle screw constraints similar to our approach, demonstrate superior planning time and success rates compared to MoveIt and OMPL for handling translational constraints. However, like [6], these planners are

[1][Online]. Available: https://ompl.kavrakilab.org
[2][Online]. Available: https://moveit.ai/

unable to independently consider gripper orientation along the screw-modeled task path. Our approach overcomes this limitation and shows superior performance, as demonstrated through quantitative comparison experiments in Section VI-C. Furthermore, our modeling has several unique advantages discussed in Section VII-F.

D. Contributions

We present a novel and efficient approach for generating manipulator joint trajectories to follow an affordance path while maintaining desired EE orientation goals. Not only is the proposed framework fast enough for use in real-time applications, but it is also agnostic to both task and hardware, invariant to the initial grasp, robust, and provides a streamlined and intuitive way to define tasks. This method combines key concepts, including screw theory, affordance chain, Davies circulation law, and closed-chain inverse kinematics (IK). Our main contributions in this study can be summarized as follows.

1) Modeling of the affordance chain and the manipulator together as a single-loop closed-chain mechanism.
2) A novel computationally efficient algorithm to generate manipulator joint trajectories for any given affordance while accounting for gripper orientation freedom or control, based on our new approach to solving closed-chain IK.
3) Validation of the algorithm using both simulation and hardware.

The rest of this article is organized as follows. Section II explains problem definition. In Section III, we start with a brief overview of screw theory, showcasing its practical application in modeling a wheel-valve affordance. Then, in Section IV, we introduce our proposed model, treating the manipulator and affordance chain as a unified closed-chain mechanism. This model is utilized to develop the differential kinematics of a two-link planar robot combined with a rotational affordance, forming a closed-chain four-bar mechanism. Subsequently, we extend this formalism to any j-joint SE(3) robot executing any affordance. Following this, in Section V, we utilize the differential kinematics formalism to develop a closed-chain IK algorithm for generating joint trajectories to execute affordances. Section VI discusses evaluation studies. Simulation results and real-world experiments are presented to validate both the model and algorithm in Section VII. Finally, Section VIII concludes this article.

II. Problem Definition

Given 1) a manipulator description, 2) an initial joint configuration, 3) a *modeled* task affordance screw, and 4) a desired affordance and EE orientation goal, our objective is to compute a joint-space trajectory. This trajectory should smoothly guide the EE along the task affordance path until it reaches the goal, while orienting the EE as desired along the way. For an n-joint manipulator, we seek to determine the trajectory

$$\boldsymbol{Q} = \{q_0, q_1, q_2, \ldots, q_s\} \tag{1}$$

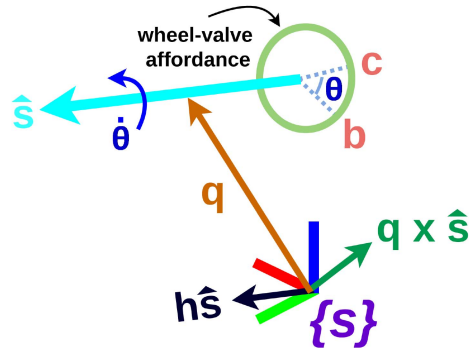


Fig. 3. Visualization of a screw defined in frame $\{s\}$ formed by a screw axis $\hat{s}$, displacement q, and pitch h. As an example, the screw represents the affordance of a wheel valve, rotating it by an angle θ, thereby moving b to c.

where $q_i \in \mathbb{R}^n$ is a set of joint angles denoting a robot configuration, q_0 represents the initial joint configuration, and the joint trajectory consists of $s+1$ points. Note that this work focuses on planning a joint trajectory for executing a screw-modeled task once the manipulator is at the start pose and does not address motion planning for obstacle avoidance in cluttered environments. Before introducing the closed-chain mechanism model that allows us to compute this joint trajectory, in the following section, we discuss the screw representation of an affordance.

III. Modeling Affordances Using Screw Representations

A. Screw Theory

This section briefly reviews the core concepts from screw theory used to define an affordance. For additional details, readers are referred to [41]. Let ω be the three-vector angular velocity of point P with respect to frame $\{s\}$ also expressed as $\hat{\omega}\dot{\theta}$, where $\hat{\omega}$ is the unit axis of rotation and $\dot{\theta}$ is the rate of rotation about that axis. Analogously, the spatial velocity (or twist) of P, defined as $\mathcal{V} = (\omega, v)$ in frame $\{s\}$, where v is its linear velocity, can be interpreted as $S\dot{\theta}$, where S is a unit twist vector, also called a screw and $\dot{\theta}$ is the rate of rotation about that screw. As illustrated in Fig. 3, S can be described by three parameters: $\hat{s}$, the three-vector screw axis; q, the location of $\hat{s}$ from $\{s\}$; and h, the rate of translation to rotation about the screw axis (also called the pitch). The relationship between $\mathcal{V}$ and its corresponding screw S is

$$\mathcal{V} = \begin{bmatrix} \omega \\ v \end{bmatrix} = \begin{bmatrix} \hat{s}\dot{\theta} \\ q \times \hat{s}\dot{\theta} + h\hat{s}\dot{\theta} \end{bmatrix} = \begin{bmatrix} \hat{s} \\ q \times \hat{s} + h\hat{s} \end{bmatrix} \dot{\theta} = S\dot{\theta} \tag{2}$$

where $\hat{s} = \omega/||\omega||$, $\dot{\theta} = ||\omega||$, and $h = \hat{\omega}^T v/\dot{\theta}$.

In the case of pure translation (i.e., where $h \approx \infty$ and $\omega = \mathbf{0}$), the screw axis is chosen as $v/||v||$ and $\dot{\theta}$ interpreted as the linear velocity $||v||$ along that axis. Consequently, $S = \mathcal{V}/||v|| = (0, v/||v||)$.

B. Modeling Affordances With Screw Theory

Consider as an example, the turning of a wheel valve presented in Fig. 3. In this case, the affordance is a pure rotation around a physically intuitive axis, $\hat{s}$, defined in frame $\{s\}$ that turns the valve by an angle θ while taking point b to point c. From (2), the affordance screw S can be computed as

$$S = \begin{bmatrix} \hat{s} \\ q \times \hat{s} \end{bmatrix} \tag{3}$$

where q is the displacement of the screw axis from $\{s\}$. By intuitively identifying $\hat{s}$ and measuring the displacement vector q, we have completely captured the wheel-valve affordance in the form of the affordance screw S. In Section IV, we will demonstrate how S simply forms a screw-based joint in our closed-chain model. Using a similar model as the wheel-valve affordance, we can describe other task affordances, such as opening a drawer (pure translation), turning a screw (helical motion), pushing a door open (pure rotation), etc.

IV. Task Affordance and Robot as a Closed-Chain Mechanism

Consider Fig. 1 in the context of turning a wheel valve. Assuming a nonslipping grasp, we can extend an imaginary link (the *affordance link*) from the real EE to the affordance screw axis, which can be conceptualized as a screw-based joint (the *affordance joint*). If the gripper can grab the valve in any orientation, the EE is "attached" to the affordance link with a spherical joint. We can then define a ground link connecting the affordance joint to the base joint of the robot. We call this imaginary chain extending from the robot EE to the base of the robot the *affordance chain*. The affordance chain and the robot form a closed-chain mechanism in which turning the crank (affordance link) means turning the valve with the manipulator. This model extends to any screw-based affordances manipulated by a robot and enables a natural, intuitive, and affordance-centric EE trajectory planner for moving along the affordance path.

A. Closed-Chain Analysis Using Kirchoff–Davies Formulation

1) Constraint Equation: Before delving into the general formulation, we examine a simple planar case, as illustrated in Fig. 4, where our frame of reference is $\{s\}$. Here, we have a two-link planar manipulator composed of joints represented by screws, S_1 and S_2, tasked with turning a valve whose affordance is denoted by the screw S_4. The affordance chain comprises the virtual EE joint, S_3 (revolute in the planar case), the affordance link connecting it with the affordance joint S_4, and the ground link that connects the affordance joint to the robot's base joint, S_1. The manipulator and affordance chain together form a planar closed-chain four-bar mechanism.

The Kirchoff–Davies circulation law (KDCL) [32] states that the algebraic sum of the relative velocities of adjacent link pairs across a closed kinematic chain is zero. For the mechanism in Fig. 4, if we represent the relative velocities as twists with respect

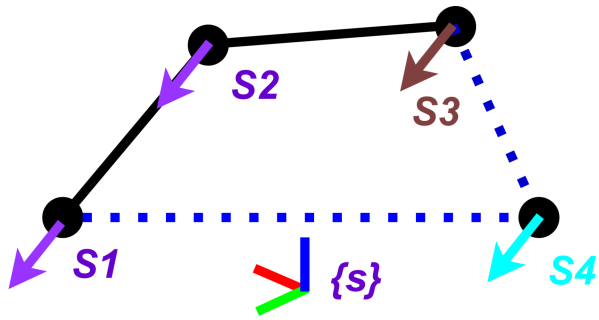


Fig. 4. Two-link planar robot with joints represented by screws S_1 and S_2, executing a task affordance (S_4) with EE orientation freedom (S_3), can be modeled as a four-bar mechanism.

to frame $\{s\}$, then a consequence of the KDCL is

$$\sum_{i=1}^{4} \mathcal{V}_i = 0 \tag{4}$$

where $\mathcal{V}_i$ is the twist vector produced by the ith joint. In other words, using (2), we can write

$$\begin{bmatrix} S_1 & S_2 & S_3 & S_4 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} = \mathbf{0} \tag{5}$$

where S_i is the screw associated with the ith joint, and $\dot{\theta}_i$ its corresponding rate of screw movement. Equation (5) can be concisely written and extended to $SE(3)$ and j joints as

$$N_{(l\times j)}\dot{\theta}_{(j\times 1)} = \mathbf{0}_{(l\times 1)} \tag{6}$$

where l is the length of the screw and N is a matrix with screws S_1 through S_j arranged as its columns, also called the *network matrix* in reference to mechanical networks formed by kinematic chains, adopting the notation from [22]. This constraint equation fully characterizes the "closed-chain" model. In Section IV-A3, we will derive the forward and inverse velocity kinematics relationships. However, first, we must define the primary and secondary joints.

2) Primary and Secondary Joints: A common purpose of closed-chain analysis is to obtain the kinematics of the *secondary* (unactuated) joints given the state of the *primary* (actuated) joints and the geometry of the chain, or *vice versa*. In our context, the designation of primary and secondary joints depends on whether we want to control the gripper orientation along with the affordance execution. Consider, for instance, Fig. 5, where we have our proposed model applied to a 6-DoF manipulator at the initial grasp pose. Recall that we have a virtual spherical joint attaching the EE to the affordance link, which we describe with the mutually orthogonal screws S_7–S_9. S_a is simply the task's affordance screw. For some tasks (pushing a door or moving a chair), minor variations in gripper orientation are acceptable. However, precise orientation control is crucial for certain tasks, such as turning a valve using the methods illustrated in Fig. 2(a) and (c). Conversely, there are scenarios,

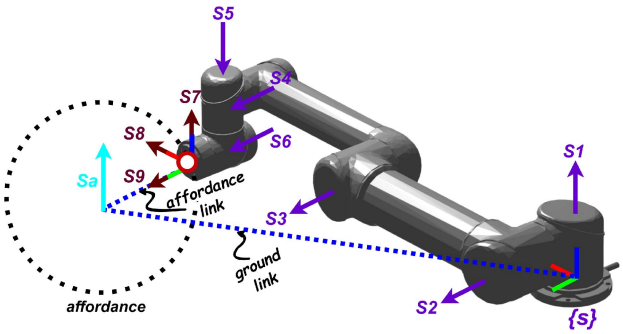


Fig. 5. Six-DoF manipulator with joints S_1–S_6 forms a closed-chain mechanism with its affordance chain, which includes a spherical joint (S_7–S_9) for EE orientation freedom, the task affordance (S_a), and the connecting imaginary links.

such as the valve-turning strategy depicted in Fig. 2(b), where EE orientation is irrelevant. In scenarios requiring the control of affordance alone while freeing the gripper orientation, we define the secondary joint simply as the affordance joint, with the remaining joints, including the robot joints and virtual EE joints, designated as primary. Conversely, for controlling the gripper orientation with the affordance, we consider the affordance and virtual EE joints as the secondary and the robot joints as primary. In tasks requiring control over specific aspects of gripper orientation along with the affordance, the relevant virtual gripper joints are considered along with the affordance. This selection (τ) is simply a matter of choice from the list of closed-chain joints.

3) Forward and Inverse Velocity Kinematics: Given the primary joint velocities, we can determine the secondary joint velocities. We start by rewriting (6) as

$$\begin{bmatrix} N_s(l\times\tau) & N_{p(l\times(j-\tau))} \end{bmatrix} \begin{bmatrix} \dot{\theta}_{s(\tau\times 1)} \\ \dot{\theta}_{p((j-\tau)\times 1)} \end{bmatrix} = \mathbf{0}_{(l\times 1)} \tag{7}$$

where N_s denotes the *secondary network matrix* consisting of secondary joint screws as its columns, with the number of columns determined by the selection parameter τ, and $\dot{\theta}_s$ represents the corresponding secondary joint velocities. Similarly, N_p constitutes the *primary network matrix*, comprising primary joint screws as its columns, with $\dot{\theta}_p$ denoting the corresponding primary joint velocities. Equation (7) implies that

$$N_s\dot{\theta}_s = -N_p\dot{\theta}_p. \tag{8}$$

Therefore, the secondary joint velocities are

$$\dot{\theta}_s = -N_s^{\dagger}N_p\dot{\theta}_p \tag{9}$$

where we make use of the pseudoinverse $N_s^{\dagger}$ since N_s is not a square matrix.

Similarly, we can easily rearrange (9) to compute the primary joint velocities given the secondary joint velocities as

$$\dot{\theta}_p = -N_p^{\dagger}N_s\dot{\theta}_s. \tag{10}$$

Note that the forward and inverse velocity kinematics have the same form in this approach. In Section V, we leverage (9) to compute primary joint-space trajectory that achieves a desired secondary joint-space goal. Before proceeding, in the following section, we discuss valuable insights gained from our modeling approach.

B. Insights From Our Modeling Approach

1) Mobility: Our modeling of the affordance chain and manipulator as a closed-chain mechanism enables us to calculate the mobility metric m, representing the DoF of the mechanism, thereby offering further insight into the inherent flexibility and controllability of the system. We can compute it as [22]

$$m = j - l \tag{11}$$

where j is the number of joints and l is the length of the screw vector.

Consider Fig. 5 again, where we have a 6-DoF manipulator executing a task affordance. For this case, with a screw length l of 6 (for $SE(3)$) and a total of ten joints ($j = 10$), the mobility of the chain is calculated as 4 using (11). In other words, there are four controllable aspects of this mechanism, namely, the affordance and the three orientation components of the gripper. This insight is particularly valuable as it precisely delineates the scope and extent of our modeling approach.

2) Role of Affordance Twist: Consider Fig. 5 in the context of controlling both the orientation of the gripper and the affordance. If we compare (10), the closed-chain inverse velocity kinematics equation to the well-known open-chain manipulator IK equation, $\dot{\theta}_p = J^{\dagger}\mathcal{V}_{\text{ee}}$, where $\mathcal{V}_{\text{ee}}$ is the EE twist composed of revolute and prismatic components, $\mathcal{V}_{\text{eer}}$ and $\mathcal{V}_{\text{eep}}$, respectively, and $J^{\dagger}$ is the pseudoinverse of the manipulator Jacobian, our modeling shows that the constraint induced by the affordance replaces $\mathcal{V}_{\text{eep}}$ by the affordance twist $\mathcal{V}_a$ and negates the resultant velocities. This gives us the physical intuition that the effect of a screw-modeled affordance on a manipulator exclusively pertains to the translational motion of the EE, and the affordance twist fully captures that motion

$$\dot{\theta}_p = -N_p^{\dagger}N_s\dot{\theta}_s \tag{12}$$

$$= -J^{\dagger}\{S_7\dot{\theta}_7 + S_8\dot{\theta}_8 + S_9\dot{\theta}_9 + S_a\dot{\theta}_a\} \tag{13}$$

$$= -J^{\dagger}\{\mathcal{V}_{\text{eer}} + \mathcal{V}_a\}. \tag{14}$$

3) Choice of Frame: The screw notation also provides the convenience of defining the network matrices with respect to a frame of choice. For example, consider again that we want to control the gripper orientation as well as the affordance. We could define the network matrices with respect to the gripper frame $\{b\}$ and define the imaginary EE joint such that the three mutually orthogonal screw axes are aligned with the gripper frame at the start of the affordance, and then, the secondary network matrix is written as

$$N_s = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \\ 0 & 0 & 1 & \mathbf{B_a} \\ 0 & 0 & 0 & \\ 0 & 0 & 0 & \\ 0 & 0 & 0 & \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \\ \mathbf{0} & \mathbf{B_a} \end{bmatrix} \tag{15}$$

where $\mathbf{B_a}$ is the affordance screw axis defined in the gripper frame. It is sparser and more elegant than when defined with respect to the base frame, potentially yielding faster computation.

V. Closed-Chain IK Solver

A. Formalism

Since we are interested in generating joint position trajectories to execute an affordance, in this section, we introduce a novel closed-chain IK framework that controls the robot EE along the affordance path. Consider the following function whose roots give us the corresponding primary joint positions, θ_p for desired secondary joint states, θ_{sd}:

$$f(\theta_p) = \theta_{sd} - \theta_s \tag{16}$$

where the length of the vectors θ_{sd} and θ_s, encapsulated in the variable τ, may range from 4 to 1, depending on whether N_s solely comprises the affordance screw or also incorporates the virtual gripper screws. The size of the network matrices, as well as the primary joint angle vector θ_p, will be affected accordingly.

Our formulation in (16) allows us to exploit common root-finding methods for continuous functions. Using the Newton–Raphson method, we employ the following iterative procedure:

$$\theta_p^{i+1} = \theta_p^i - \frac{f(\theta_p^i)}{f'(\theta_p^i)} \tag{17}$$

where

$$f'(\theta_p) = -\frac{d\theta_s}{d\theta_p}. \tag{18}$$

Our approach is to approximate $f'(\theta_p)$ using (9). However, applying numerical methods to continuous functions introduces approximation errors, and in this case, the errors tend to violate the closed-chain nature of the formulation [24], as if opening the chain by moving the end of the ground link. We account for this error with twist ρ, which can be calculated with open-chain forward kinematics, and modify (9) as follows:

$$\dot{\theta_s} = -N_s^{\dagger}\{N_p\dot{\theta_p} + \rho\}. \tag{19}$$

Thus

$$\frac{d\theta_s}{d\theta_p} = -N_s^{\dagger}\{N_p + \frac{\rho}{\dot{\theta_p}}\} = N(\theta_p) \tag{20}$$

where at each iteration, we approximate $\dot{\theta_p^i}$ for a small time interval Δt as

$$\dot{\theta_p^i} = \frac{\theta_p^i - \theta_p^{i-1}}{\Delta t}. \tag{21}$$

Hence, (17) becomes

$$\theta_p^{i+1} = \theta_p^i + N(\theta_p^i)^{\dagger}\{\theta_{sd} - \theta_s^i\}. \tag{22}$$

B. Algorithm for Affordance Trajectory Generation

Given the EE in the initial pose of an affordance, we utilize the above formalism and Algorithm 1 to calculate the joint-space trajectory of the robot that steps the EE along the affordance path while considering its orientation goals until it reaches the affordance goal. The algorithm takes as input desired EE orientation and affordance goals θ_{sdf}, the gripper control variable τ, and the list of screws $cc_$slist of the closed-chain model, which includes the robot Jacobian, imaginary EE joint screws, and affordance joint screw. The computation of the joint trajectory begins with an initial guess of $\theta_{pg} = \mathbf{0}$ and involves solving closed-chain IK. This is accomplished by invoking Algorithm 2 with a small incremental affordance goal (for instance, $\Delta\theta_a = 0.1$ rad) and corresponding gripper orientation goals. Algorithm 2 solves closed-chain IK using (22) while correcting for the closure error ρ by invoking Algorithm 3.

The error-correction procedure in Algorithm 3 involves computing the forward kinematics T_{se} to the end of the ground link, followed by the computation of the skew-symmetric representation of the twist that represents that error, utilizing the matrix logarithm as follows:

$$[\rho_b] = \log\{T_{se}^{-1}\} \tag{23}$$

where ρ_b is the body-frame twist, which can be converted to the space frame $\{s\}$ using the following equation. Here, $\text{adjoint}(T_{se}) \in \mathbb{R}^{6\times6}$ represents the "adjoint representation" of T_{se} used to transform velocities between frames [41]

$$\rho = \text{adjoint}(T_{se})\rho_b. \tag{24}$$

The following scheme is then utilized to adjust both the primary and secondary joint angles using the vector form of ρ:

$$\begin{bmatrix}\theta_p \\ \theta_s\end{bmatrix} \leftarrow \begin{bmatrix}\theta_p \\ \theta_s\end{bmatrix} + \begin{bmatrix}N_p & N_s\end{bmatrix}^{\dagger}\rho. \tag{25}$$

Next, Algorithm 1 records the solution for the current iteration from Algorithm 2 as a point in the differential joint trajectory solution $\boldsymbol{\theta_p}$, i.e., a trajectory that assumes the affordance start configuration as zero for the joint angles. It then updates the guesses θ_{pg} and θ_{sg} for the next iteration with the result from the current iteration, repeating the process until the full trajectory is developed to reach the desired affordance and EE orientation goals. We highlight that this guess-update scheme is vital in minimizing joint movements between two consecutive points in the trajectory.

Note that $\boldsymbol{\theta_p}$ includes joint values for all primary joints, which, in the case of controlling just the affordance, will encompass the virtual gripper joint values as well. To generate a joint trajectory suitable for direct transmission to the robot, we need to extract the joint values corresponding to the robot joints and add the affordance-start-pose robot joint configuration q_0 to align the joint zeros with those of the robot. We do this in Algorithm 1 on Line 22 using the following equation:

$$\boldsymbol{Q} = q_0 + \{0, \theta_{p1}(1:n), \theta_{p2}(1:n), \ldots, \theta_{ps}(1:n)\} \tag{26}$$

where $\theta_{pi}(1:n) \in \boldsymbol{\theta_p}$ such that the robot has n joints.

This planning framework encompasses three parameters: the affordance step $\Delta\theta_a$, which can be fine-tuned to achieve the desired joint trajectory density; the secondary joint goal error

Algorithm 1: Task Affordance Joint Trajectory Generator.

Require: $cc\_slist, q_0, \theta_{sdf}, \tau$
Define affordance step, $\Delta\theta_a$
Determine relevant matrix and vector sizes based on τ
Set start guesses and step goal:
$\theta_{pg} = \mathbf{0}, \theta_{sg} = \mathbf{0}, \theta_{sd} = \mathbf{0}$
Extract final affordance goal, θ_{adf} from θ_{sdf} and compute no. of iterations, m to final goal
Initialize loop counter, $k \leftarrow 0$; success counter, $s \leftarrow 0$
while $k < m$ **do**
  k = k + 1
  Update secondary joint goals for next aff step:
  Update EE orientation goals in θ_{sd}
  Update aff step goal:
  if $k = m$ **then**
    $\Delta\theta_a = \theta_{adf} - \Delta\theta_a * (m-1)$
  end if
  $\theta_{sd}(end) = \theta_{sd}(end) + \Delta\theta_a$
  Call Algorithm 2 with args: $cc\_slist, \theta_{pg}, \theta_{sg}, \theta_{sd}$
  if success **then**
    Record solution, θ_p as a point in $\boldsymbol{\theta_p}$
    Update guesses, θ_{pg}, θ_{sg}
    $s = s + 1$
  end if
end while
Compute robot joint trajectory $\boldsymbol{Q}$ using (26)
return $\boldsymbol{Q}$

Algorithm 2: Closed-Chain IK Solver.

Require: $cc\_slist, \theta_{pg}, \theta_{sg}, \theta_{sd}$
Set max no. of iterations, l, and error thresholds, ϵ_s, ϵ_ρ
Set Δt as small time increment
Capture guesses: $\theta_p \leftarrow \theta_{pg}, \theta_s \leftarrow \theta_{sg}$
Initialize loop counter, $i \leftarrow 0$
Start closure error at $\mathbf{0}$, $\rho \leftarrow \mathbf{0}$
while i< l and $||\theta_{sd} - \theta_s|| > \epsilon_s$ or $||\rho|| > \epsilon_\rho$ **do**
  $i = i + 1$
  Compute N_p, N_s as screw-based Jacobians
  Compute $\dot{\theta}_p$ using (21)
  Compute N using (20)
  Update θ_p using (22)
  Call Algorithm 3 with args, $cc\_slist, N_p, N_s, \theta_p, \theta_s$
end while
success = not($||\theta_{sd} - \theta_{sg}|| > \epsilon_s$ or $||\rho|| > \epsilon_\rho$)
return θ_p, θ_s, success

Algorithm 3: Closure-Error Adjuster.

Require: $cc\_slist, N_p, N_s, \theta_p, \theta_s$
Compute forward kinematics to chain's end link, T_{se}
Compute skew-symmetric form of ρ using (24)
Adjust θ_p, θ_s using (25)
return θ_p, θ_s

threshold ϵ_s; and the closed-chain closure error threshold ϵ_ρ—both of which can be adjusted to meet the desired planner accuracy. When controlling both affordance and gripper orientation, ϵ_s is composed of ϵ_{sa} for affordance error threshold and ϵ_{so} for orientation error threshold. ϵ_ρ consists of $\epsilon_{\rho\omega}$ for the angular component and $\epsilon_{\rho v}$ for the linear component.

Remark 1: Equation (26) is the solution to the problem defined in Section II.

Remark 2: While we have developed this framework to generate joint trajectories for affordance execution, and its versatility extends beyond this application. Its foundation in closed-chain mechanism modeling allows for direct application to various single-loop closed-chain mechanisms in general.

VI. Evaluation Studies

We validated our proposed framework through both simulation and real-robot experiments. The following sections describe these studies in detail.

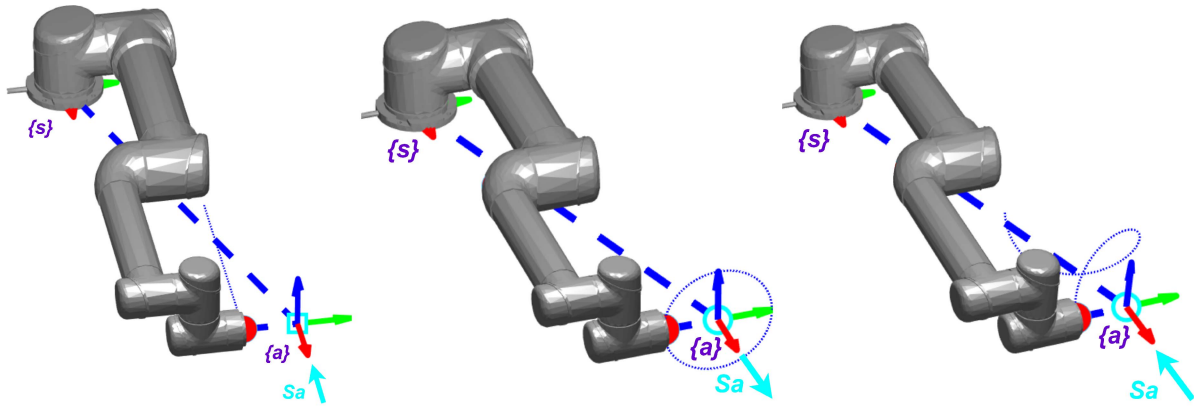
A. Simulation Studies

For simulation studies, we considered a UR5 robot performing a representative set of affordances. The simulation experiments involved executing different task affordances in MATLAB with a UR5 manipulator and exploring the effect of changing planner parameters (ϵ_s and $\Delta\theta_a$) on the rate and nature of convergence. We present results showcasing pure rotational, translational, and screw motion tasks under affordance control (i.e., $\tau = 1$), aiming to capture common manipulation scenarios. For a video demonstration of one such experiment where the robot performs these motion types, the reader is referred to online.[3]

Fig. 6(a)–(c) shows the initial robot configuration, affordance (represented by the screw S_a), and EE paths for the conducted simulation experiments, along with an overlay of the closed-chain model. Note that while the affordance type varies between tasks, the closed-chain model itself remains constant. For the pure rotational affordance, as shown in Fig. 6(b), and screw motion affordance, as shown in Fig. 6(c), we looked at two cases with affordance steps of 0.1 rad (5.7°) and 0.5 rad (28.6°), respectively, each covering a total affordance path of 5 rad (287°). Similarly, for the pure translational affordance of Fig. 6(a), we present two cases with affordance steps of 1 and 5 cm, respectively, each encompassing a total affordance path of 50 cm.

We evaluated the performance of our planner by varying the solution accuracy of our closed-chain IK solver (using ϵ_s) between $\pm 1\%$, $\pm 5\%$, and $\pm 10\%$. Since each affordance-step-IK computation might require a different number of iterations to converge to a solution, we tracked the minimum and maximum number of iterations needed for the solver to converge for each affordance trajectory. This allowed us to assess the tradeoff between solution accuracy and computational efficiency in the IK solver within the context of the planner. This insight is valuable

[3][Online]. Available: https://youtu.be/Ukv93hbNrOM



(a) (b) (c)

Fig. 6. Start pose, affordance (S_a defined in frame $\{s\}$), affordance path, and closed-chain model overlay for simulation experiments: (a) pure translation, (b) pure rotation, and (c) screw motion.

TABLE I
IMPACT OF AFFORDANCE STEP SIZES AND ACCURACIES ON STEP ITERATIONS

Affordance Step(rad)	Accuracy(%)	Iterations (min)	Iterations (max)
0.1	1	8	14
0.1	5	6	9
0.1	10	4	7
0.5	1	9	13
0.5	5	7	10
0.5	10	7	10

Pure Translation with Total Affordance 0.5m

Affordance Step(m)	Accuracy(%)	Iterations (min)	Iterations (max)
0.01	1	7	30
0.01	5	5	20
0.01	10	4	16
0.05	1	8	30
0.05	5	5	19
0.05	10	4	18

Screw Motion with Total Affordance 5 radians

Affordance Step (rad)	Accuracy (%)	Iterations (min)	Iterations (max)
0.1	1	10	13
0.1	5	7	9
0.1	10	5	7
0.5	1	10	13
0.5	5	8	9
0.5	10	8	9

because, given our closed-chain model, the EE is guaranteed to remain on the affordance path. Any loss of accuracy only affects the affordance goal; for instance, in the case of valve turning, this is just the degrees it is turned. For many real-world applications, this might not be critical, and as long as the EE remains on the affordance path, the tasks could be accomplished without the need for accuracy. The results of this study are documented in Table I . For all cases, the closed-chain closure error threshold (ϵ_ρ) was set at $1e^{-4}$ using the L2 metric.

For the three affordance types at $\pm 1\%$ accuracy requirement, we also tracked the affordance and closed-chain closure errors against the number of iterations to converge to various affordance steps. Our objective was to better understand the convergence characteristics of our IK solver.

TABLE II
AFFORDANCE CONTROL EXPERIMENT PLANNING TIMES

Task	Task Description			Planning Time(s)
	Aff Type	Goal	Steps	
Open drawer	Translation	0.29 m	0.05 m	0.0155 ± 9e-05
Close drawer	Translation	0.2 m	0.05 m	0.0077 ± 1e-04
Move stool	Rotation	0.5 π	0.15 rad	0.0375 ± 2e-04

Drawer tasks IK method: Pseudoinverse, Move stool: Transpose.

TABLE III
AFFORDANCE AND EE ORIENTATION CONTROL EXPERIMENT PLANNING TIMES

Task	Aff Goal	Planning Time(s)
Case 1	$0.5\pi(90°)$	0.0975 ± 2e-04
Case 2	$0.25\pi(45°)$	0.0622 ± 4e-04
Case 3	$1.75\pi(315°)$	0.0861 ± 7e-04
Case 4	$2.33\pi(420°)$	0.0643 ± 5e-04

For all tasks, Aff Type: Rotation, Step: 0.2 Rad. Affordance rate: 0.2 rad/0.3 s = 0.667 rad/s, IK Method: Transpose.

B. Real-Robot Experimental Studies

We applied the proposed framework using C++ and ROS2 to a manipulator-integrated Boston Dynamics Spot robot arm to perform real-world manipulation tasks. Our objective with real-world implementation was twofold, and our studies are accordingly split into the following two categories. A video demonstration of these tasks is available online.[4]

1) *Affordance Control:* To demonstrate our ability to perform common articulated and nonarticulated manipulation tasks while allowing for minor gripper orientation freedom, i.e., $\tau = 1$. For these studies, we performed drawer-opening and closing tasks and moved a stool out of the way. The start configurations for these tasks, as well as the affordance screws, are shown in Fig. 7.

[4][Online]. Available: https://youtu.be/Ukv93hbNrOM

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

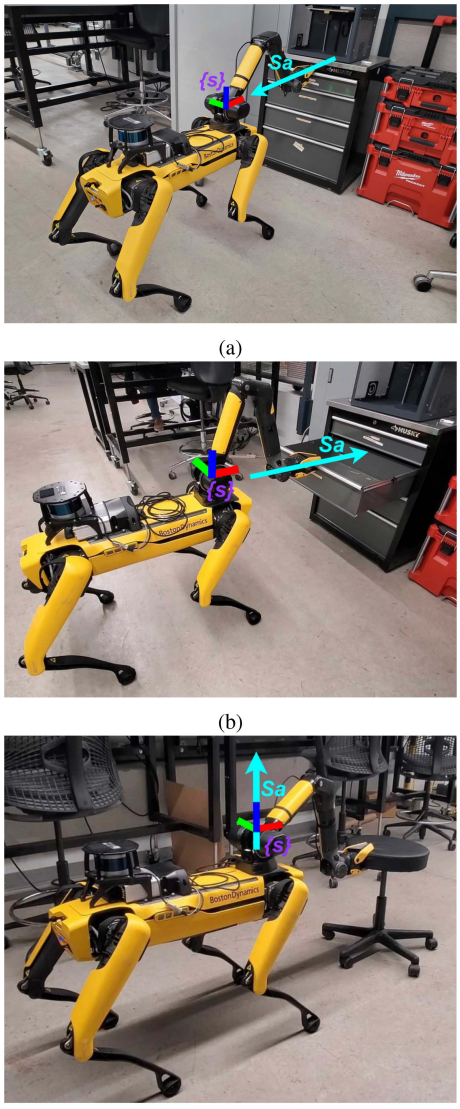


Fig. 7. Affordance control experimental setup showing robot start pose and affordance screw, S_a defined in frame $\{s\}$: (a) pulling a drawer, (b) pushing a drawer, and (c) moving a stool.

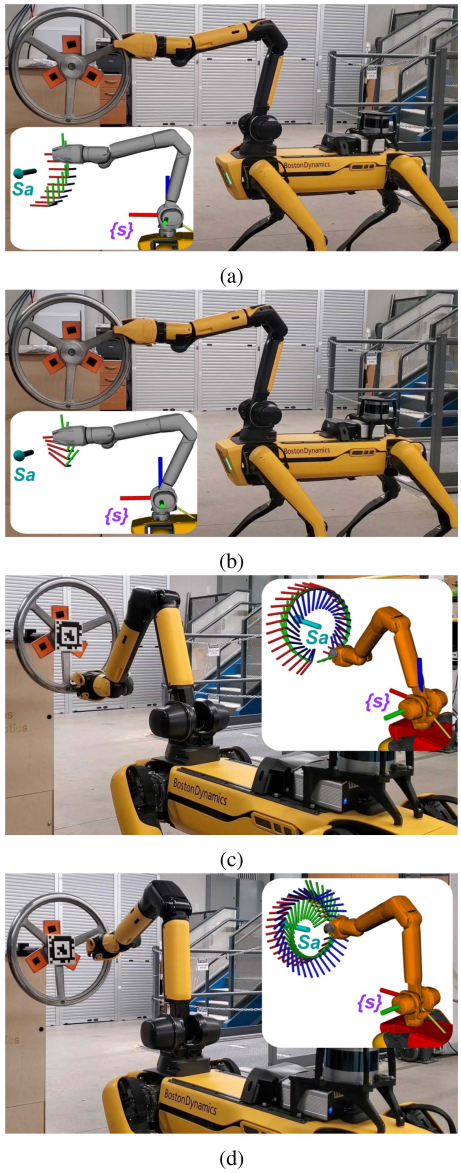


Fig. 8. Affordance and EE orientation control experimental setup showing robot start pose, planned paths, and affordance screw, S_a defined in frame $\{s\}$: (a) case 1: EE orientation fixed in world, (b) case 2: pitch changing in tandem, roll and pitch fixed, (c) case 3: roll changing in tandem, yaw and pitch free, and (d) case 4: EE orientation free.

2) *Affordance and EE Orientation Control:* To demonstrate our ability to precisely control or free the orientation of the EE along the affordance path, i.e., $\tau \geq 1$. For these studies, we turned to the various valve-turning approaches we introduced in Section I-A and conducted those experiments using the Spot robot. Specifically, we examined the four different cases listed in the following. The start configurations, affordance screws, and planned EE paths for these tasks are shown in Fig. 8.
   a) *Fixed EE Orientation in World Frame:* The EE orientation remains unchanged relative to the fixed frame, $\{s\}$. This is similar to preserving hand orientation in Fig. 2(a). We performed a 90° valve turn using this approach.
   b) *EE Pitch Changing with Valve Rotation, Roll and Yaw Fixed in World Frame:* The EE pitch changes in tandem with the valve rotation, while the roll and yaw remain fixed relative to $\{s\}$. This is similar to rotating the entire hand in tandem in Fig. 2(c). We performed a 45° valve turn using this approach.
   c) *EE Roll Changing with Valve Rotation, Pitch and Yaw Free:* The EE roll changes in tandem with the valve rotation, while both the pitch and yaw are free to change in any manner. This is similar to rotating the hand in tandem in Fig. 2(c) while allowing some freedom along hand pitch and yaw. We performed a 315° valve turn using this approach.
   d) *Free EE Orientation:* The EE orientation along the affordance path does not matter and is free to change

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

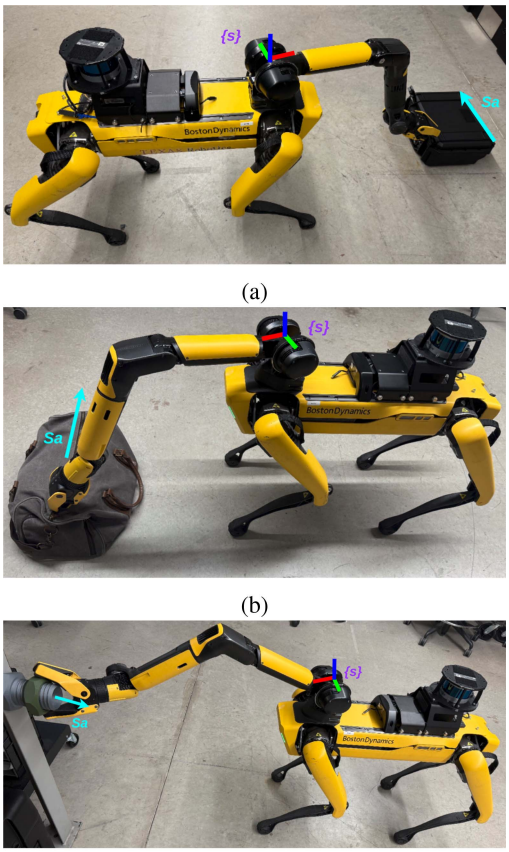


Fig. 9. Comparison experiment task setup showing robot start pose and affordance screw, S_a defined in frame $\{s\}$: (a) opening a pelican case, (b) unzipping a dufflebag, and (c) unfastening a nut.

in any manner. This is similar to not preserving hand orientation in Fig. 2(b) where we turn the valve by pushing on the spoke. We performed a 420° valve turn using this approach.

To evaluate our framework, we recorded several key metrics during all of these experiments. First, we tracked the actual joint states of the robot to demonstrate our method's ability to generate smooth joint trajectories. We also measured planning times to show the efficiency of the framework. The planning process was run five times for each task to calculate the average planning time, reported as the mean ± standard deviation.

Inspired by Wolovich and Elliott [42], who introduced the Jacobian transpose method for solving IK in the context of open-chain serial manipulators, we explored approximating $N(\theta_p^i)^\dagger$ in (22) with the transpose, $N(\theta_p^i)^T$. In these experiments, we implemented both the pseudoinverse and transpose methods, observing that the transpose method often led to faster solutions. Therefore, in our report of planning times, we specify whether the pseudoinverse or transpose method was used.

In addition, we monitored the predicted versus actual EE trajectories and calculated the deviation between the two. This was done to demonstrate that, despite potential errors due to runtime uncertainties or localization of the affordance screw, the tasks could still be accomplished. It is worth noting that the actual and predicted EE trajectories were sampled at different rates due to the nature of the ROS2 data collection framework. Thus, to compute the trajectory error, we first identified the two closest points on the actual trajectory for each point on the predicted trajectory. We then applied linear interpolation between these actual points to obtain a more accurate match and computed the Euclidean distance between the predicted point and the closest interpolated actual position to quantify the error.

In addition to these metrics, for the affordance and EE orientation control studies, we recorded the predicted versus actual EE orientation trajectories to show that we could generate trajectories that allow precise EE orientation control or freedom along the affordance path and that the predicted orientation was achieved during task execution.

Based on the insight from simulation studies and given the nature of the experiments, we opted to set the accuracy of the planner at 10%. The closure error threshold was set at $1e^{-6}$ using the L2 metric. We opted for affordance steps that allowed the task to be executed at a reasonable and smooth pace, but these steps could be increased to further reduce planning time if needed. Lastly, the time difference between joint trajectory points was set at 0.3 s.

C. State-of-The-Art Comparison Studies

To assess the performance of our closed-chain affordance (CCA) planner against the state-of-the-art methods, we implemented the screw-based planners, SPS and DSS, as discussed in Section I-C, alongside CCA [with affordance control, Section VI-B)] on the Boston Dynamics Spot quadruped. We conducted experiments, including common manipulation tasks, representing all three screw types: opening a Pelican case (rotational motion), unzipping a duffle bag for inspection (translational motion), and unfastening a nut (screw motion). The experimental setup is shown in Fig. 9. Each task was performed five times from randomized start configurations to different task goals, resulting in a total of 15 tasks per planner. The same starting configurations and goals were used for all three planners to ensure a fair comparison.

We evaluated three key performance metrics that have real-world implications.

1) *Joint Distance*, the total Euclidean path length traveled in joint space, computed as the sum of Euclidean distances between consecutive joint configurations in the planned trajectory. This metric assesses the motion efficiency of each planner.
2) *Maximum Task Goal*, the highest task (affordance) goal achieved during planning (e.g., maximum Pelican case opening angle), measuring the planner's capability to achieve extended task goals.
3) *Planning Time*, the time required to generate a plan, reflecting the planner's computational efficiency.

For a fair, direct comparison, we also implemented self-collision and joint-limit checking for CCA using MoveIt[5] and

[5][Online]. Available: https://moveit.picknik.ai/main/doc/examples/planning_scene/planning_scene_tutorial.html

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

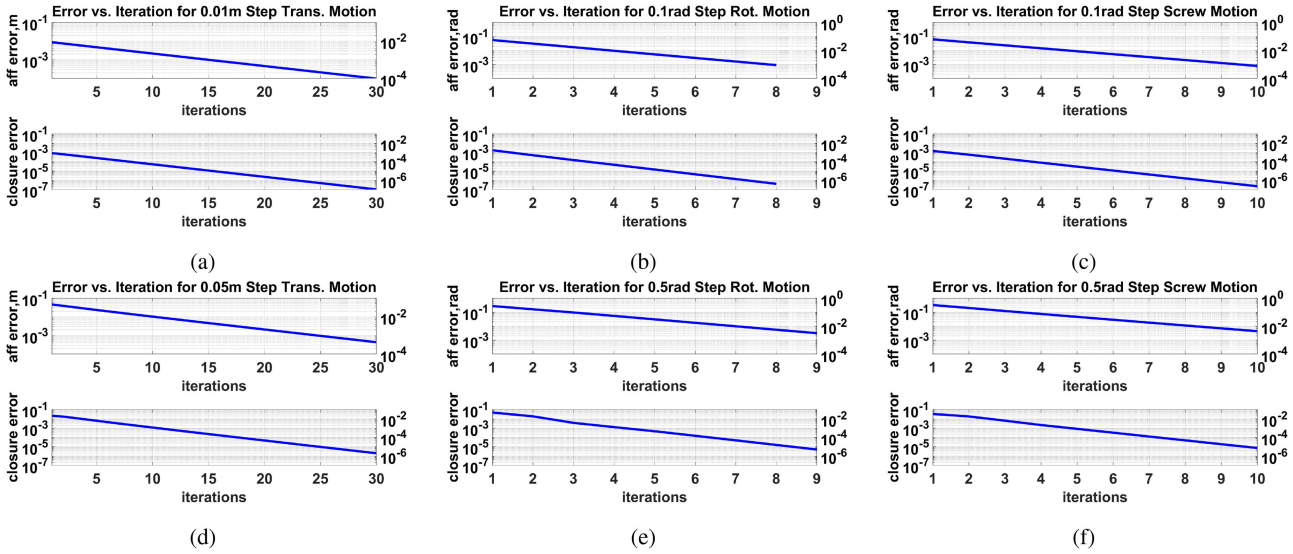


Fig. 10. Affordance error (as absolute deviation from the desired affordance goal) and closed-chain closure error (as L2 norm of the closure error twist) plotted against number of iterations for IK computation of the robot configuration an affordance step away from the start configuration. The plots are shown for various affordance steps and types: (a) 1 cm step pure translation, (b) 0.1 rad step pure rotation, (c) 0.1 rad step screw motion, (d) 5 cm step pure translation, (e) 0.5 rad step pure rotation, and (f) 0.5 rad step screw motion.

included it in our planning time calculations. To have consistent references for measuring joint distances across identical tasks, we first executed DSS, the planner utilizing a sampling-based motion planner underneath, to determine an achievable task goal. We then performed the same tasks using the other two planners. The detailed settings used for these planners during the experiments are provided in the Appendix. A video demonstration of representative comparison tasks performed using CCA is available online.[6]

VII. Results and Discussion

In this section, we present and analyze the results from our simulation and real-robot experimental studies.

A. Simulation Results and Discussion

Consider the results for pure rotation from Table I. For a 0.1-rad affordance step, we observe that the minimum number of iterations needed for the solver to converge increases from 4 to 6 to 8 as the accuracy requirement is tightened from $\pm 10\%$ to $\pm 5\%$ to $\pm 1\%$, respectively. Similarly, the maximum number of iterations required rose from 7 to 9 to 14. This trend is consistent for the 0.5-rad case, as well as for all pure translation and screw motion cases. As expected, the maximum and minimum number of iterations and thus the computation times are significantly reduced at the cost of some accuracy. Note that in the 0.1-rad step cases, the increase in iterations follows a more discernible pattern compared to the 0.5-rad step cases. This is likely because larger step sizes result in an initial guess for our root-finding method that is farther from the solution, making the solver more susceptible to convergence irregularities. It is trivial to see that the overall computation time for the entire trajectory can also be saved by employing larger affordance steps as tolerated by our applications. However, very large steps can (and should be) avoided, as they might result in the solver not converging given the nature of the underlying root-finding method. Note that the smaller the affordance steps, the denser and smoother the joint trajectories become due to the update scheme mentioned in Section V-B.

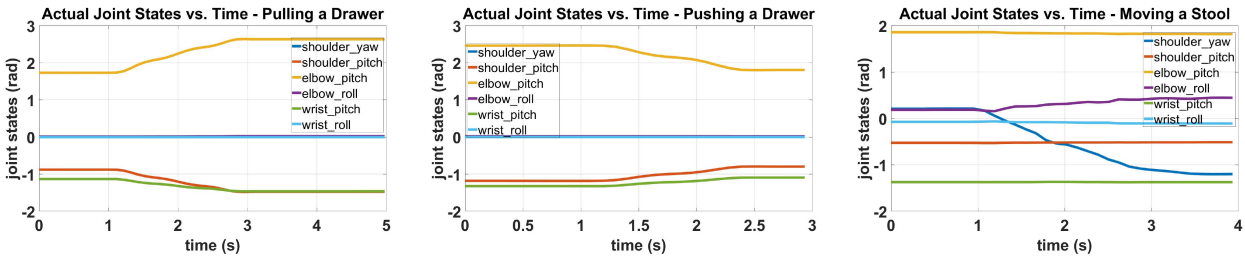
For $\pm 1\%$ task accuracy requirement and different affordance step sizes, we present the error plots for the first step in the affordance trajectories. Fig. 10(a) and (d) shows the error plots for the pure translational case of 1 and 5 cm steps, respectively. Similarly, Fig. 10(b) and (e) shows the error plots for the pure rotation case of 0.1 and 0.5 rad steps, respectively. Finally, Fig. 10(c) and (f) shows the error plots for the screw motion case of 0.1 and 0.5 rad steps, respectively. A remarkable observation from our study was the exponential convergence behavior across all cases. Although the Newton–Raphson method typically converges quadratically, it can exhibit exponential (linear) convergence when the derivative is singular at the root [43]. Since our model involves a redundant closed-chain mechanism, the constraint Jacobian at the solution is not full rank [44], leading to the observed exponential convergence (see Appendix D for the mathematical proof). This is illustrated by the linear nature of the semilog error versus iterations plots in Fig. 10(a)–(f).

B. Real-Robot Affordance Control Results and Discussion

We observed smooth joint movements during all affordance control tasks, as shown in Fig. 11(a)–(c). Regarding trajectory planning times, as listed in Table II, the drawer opening and closing tasks took 0.0155 and 0.0077 s, respectively, while moving the stool took 0.0375 s.

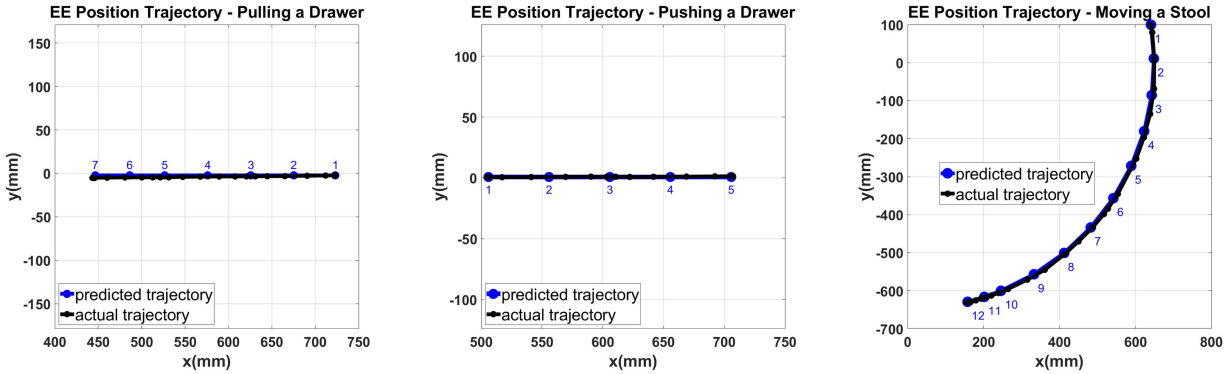
When examining the predicted versus actual EE trajectories for these tasks [see Fig. 12(a)–(c)], we found that all three

[6][Online]. Available: youtu.be/Ukv93hbNrOM



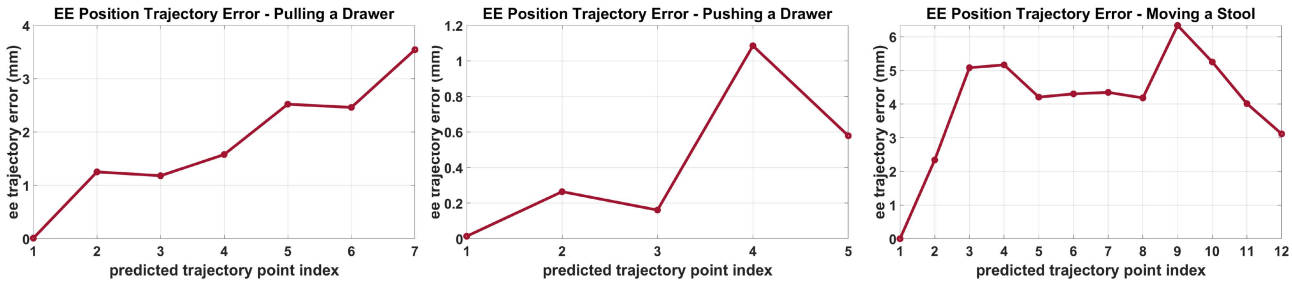
(a) (b) (c)

Fig. 11. Joint states as a function of time for affordance control experiments, demonstrating smooth movement: (a) pulling a drawer, (b) pushing a drawer, and (c) moving a stool.



(a) (b) (c)

Fig. 12. Actual versus predicted EE position trajectories for affordance control experiments: (a) pulling a drawer, (b) pushing a drawer, and (c) moving a stool.



(a) (b) (c)

Fig. 13. Euclidean error between predicted and actual EE position trajectories plotted against the indices of predicted trajectory points shown in Fig. 12 for affordance control experiments: (a) pulling a drawer, (b) pushing a drawer, and (c) moving a stool.

tasks were successfully completed despite some deviation errors introduced due to runtime uncertainties. Specifically, the error between the predicted and actual trajectories was up to 1.2 mm for pushing the drawer [see Fig. 13(b)] and 4 mm for pulling the drawer [see Fig. 13(a)]. However, for moving the stool [see Fig. 13(c)], the error was up to 6.5 mm, likely due to the frictional uncertainties of rolling a stool on a rough floor.

C. Real-Robot Affordance and EE Orientation Control Results and Discussion

Similar to the affordance control experiments, we observed smooth joint movements during all affordance and EE orientation control tasks, as shown in Fig. 14(a)–(d). Regarding trajectory planning times, as listed in Table III, Case 1 (90° valve turn) took 0.0975 s, Case 2 (45° valve turn) took 0.0622 s, Case 3 (315° valve turn) took 0.0861 s, and Case 4 (420° valve turn) took 0.0643 s. It is noteworthy that by allowing freedom in some or all aspects of gripper orientation, as in Cases 3 and 4, which involve much longer trajectories compared to Cases 1 and 2, we were able to generate trajectories much faster. Intuitively, by allowing aspects of the gripper orientation to vary (i.e., by increasing the number of columns in N_p and decreasing in N_s), we expand the solution space [linked to the Jacobian's null space in (20)], enabling faster convergence of our root-finding method (22). In practice, the resulting deviations in orientation are minimal, as our update scheme (discussed in Section V-B) starts with a guess close to the solution.

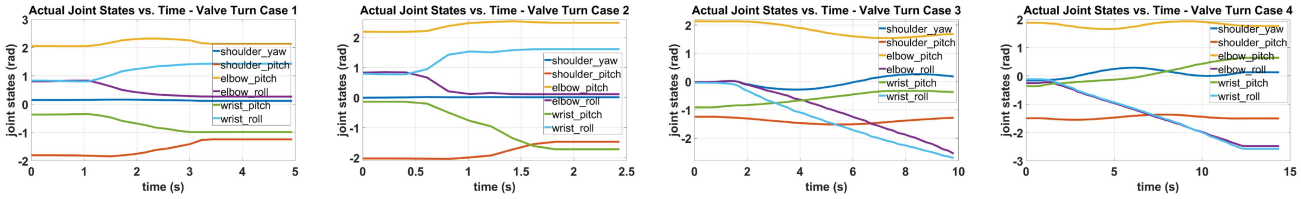


Fig. 14. Joint states as a function of time for affordance and EE orientation control experiments, demonstrating smooth movement. (a) Case 1: EE orientation fixed in world. (b) Case 2: pitch changing in tandem, roll and pitch fixed. (c) Case 3: roll changing in tandem, yaw and pitch free. (d) Case 4: EE orientation free.

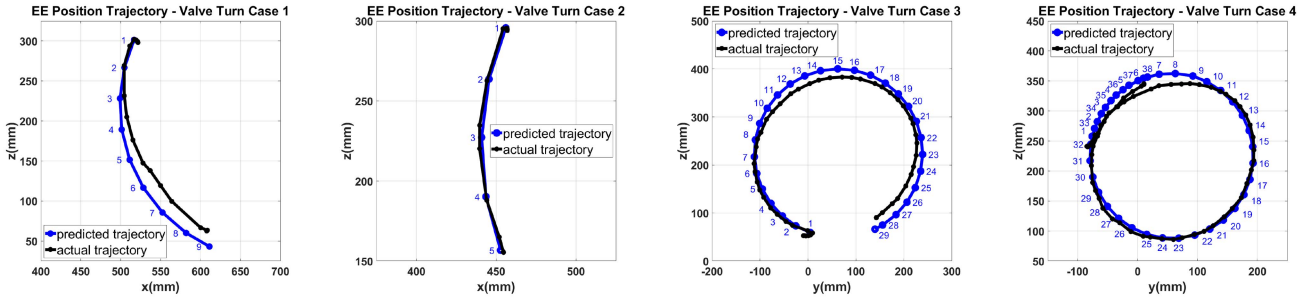


Fig. 15. Actual versus predicted EE position trajectories for affordance and EE orientation control experiments. (a) Case 1: EE orientation fixed in world. (b) Case 2: pitch changing in tandem, roll and yaw fixed. (c) Case 3, roll changing in tandem, yaw and pitch free. (d) Case 4: EE orientation free.

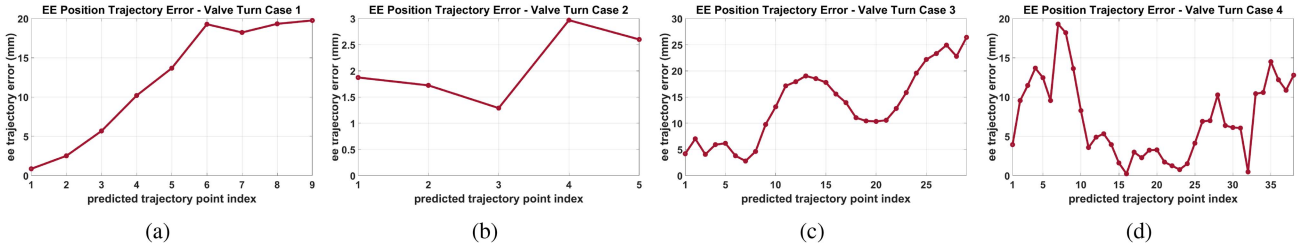


Fig. 16. Euclidean error between predicted and actual EE position trajectories, plotted against the indices of predicted trajectory points shown in Fig. 15 for affordance and EE control experiments. (a) Case 1: EE orientation fixed in world. (b) Case 2: pitch changing in tandem, roll and pitch fixed. (c) Case 3, roll changing in tandem, yaw and pitch free. (d) Case 4: EE orientation free.

When examining the predicted versus actual EE trajectories for these tasks [see Fig. 15(a)–(d)], we found that all tasks were successfully completed despite large deviation errors introduced due to affordance screw localization as well as runtime uncertainties. There were two sources of localization errors.

1) Locating the screw axis necessitated detection of an AprilTag through the hand camera. Since the affordance screw's location and orientation are estimated based on the AprilTag, and AprilTags are known to have pose estimation errors [45], these errors propagate to the screw and, ultimately, affect the predicted EE trajectories.
2) The arm had to retract to a pose where the AprilTag was visible before transitioning to the affordance start pose. During this process, the whole body controller was active, causing the Spot robot's body to move to compensate for the arm's motion. This movement would shift our "fixed" frame $\{s\}$ (arm base link) as well.

We observe from Fig. 16(a)–(d) that the trajectory error ranges from a maximum of 3–28 mm between the four valve-turn experiments. We attribute this remarkable error handling and robustness to two key factors. First, we are working in the joint space, and the output from our framework is joint trajectories starting from the current (affordance-start) state of the robot. Therefore, even if there are affordance localization errors, the robot still attempts to perform the motion. Second, the robot has mechanical compliance, which is helpful in combating runtime uncertainties.

When analyzing the actual EE orientation trajectory for Case 1, where we expect the EE orientation to remain fixed in the world frame [see Fig. 17(a)], we observe that the robot maintains the roll and yaw within a remarkable maximum deviation of

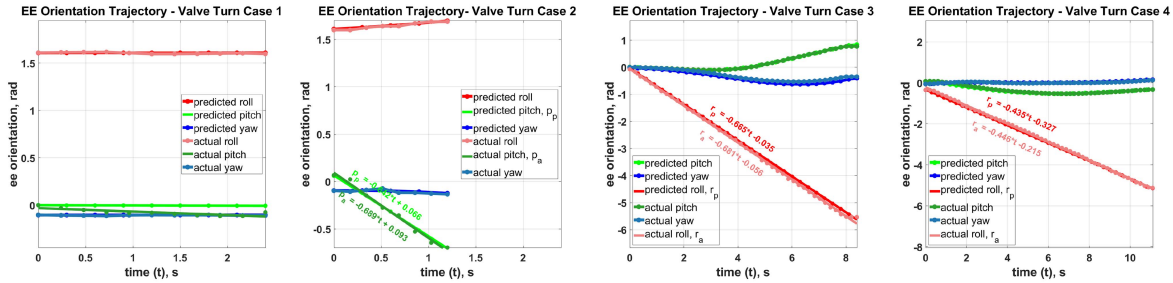


Fig. 17. Actual versus predicted EE orientation trajectories for affordance and EE orientation control experiments. (a) Case 1: EE orientation fixed in world, as *XYZ* Euler angles. (b) Case 2: pitch changing in tandem, roll and yaw fixed, as *XYZ* Euler angles. (c) Case 3: roll changing in tandem, yaw and pitch free, as *YZX* Euler angles. (d) Case 4: EE orientation free, *YZX* Euler angles.

0.01 rad (0.57°) and the pitch within 0.1 rad (5.7°). For Case 2, where we expect the EE pitch to change at the same rate as the affordance while the roll and yaw stay constant, this is precisely what we observe. The roll and yaw plots are horizontal straight lines, whereas the pitch plot is a straight line with a significant slope. The absolute value of the rate of change of the actual pitch from Fig. 17(b) is 0.689 rad/s, compared with the predicted rate of 0.642 rad/s and the affordance rate of 0.667 rad/s (see Table III).

Note that although we start at similar configurations for both Cases 1 and 2, we can only achieve a 45° valve turn in Case 2 because this approach leads to the wrist pitch joint limit of −1.83 rad [see Fig. 14(b)], whereas in Case 1, we can achieve a 90° turn. We chose this setup for the two experiments to highlight that some approaches are preferable in certain applications over others.

For Case 3, where we expect the EE roll to change at the same rate as the affordance, while pitch and yaw can change freely, we observe in Fig. 17(c) that the roll plot is a straight line with a significant slope, as expected, whereas the pitch and yaw plots are nonlinear. The absolute rate of change of the actual roll is 0.681 rad/s, compared to the predicted rate of 0.665 rad/s and the affordance rate of 0.667 rad/s (see Table III). We emphasize that by allowing freedom in the EE pitch and yaw, we achieve a remarkable 315° valve turn. If we had used the Case 2 approach, we would have been limited to only a 135° turn due to the wrist roll joint reaching its limit.

For Case 4, instead of using the valve circumference, we push against a spoke to turn the valve, and our goal is to ensure the EE remains on the affordance path, but its orientation is not constrained. Using this approach, we can remarkably generate a joint trajectory that turns the valve by 420° at once. The orientation freedom is highlighted by the nonlinear nature of the pitch and yaw plots in Fig. 17(d). Although the roll plots are linear, indicating a constant rate of change, which is likely due to the smooth nearest-solution-finding nature of our framework, we do not expect them to match the affordance rate since we have not constrained the EE orientation. This is precisely what we observe: the absolute rates of change (actual: 0.446 rad/s and predicted: 0.435 rad/s) are significantly different from the affordance rate of 0.667 rad/s (see Table III).

D. State-of-The-Art Comparison Results and Discussion

SPS failed to plan for full trajectories in 53% of the test cases. However, in those where it succeeded, as shown in Fig. 18(a), the joint distances were within 0.05 rad of those in the DSS trials. We attribute this similarity to the same screw-based modeling constraints underlying both planners. When we compare the joint distances for the DSS-generated joint trajectories against CCA, which successfully planned for all trials [see Fig. 18(b)], we observe that on average, CCA achieved the same tasks while requiring the manipulator to travel only 46% of the joint distance compared to DSS. Moreover, in 80% of the cases, the CCA joint distances were less than those of DSS. To highlight this important finding, we juxtapose the DSS and SPS goal configurations with CCA for a representative Pelican opening experiment case in Fig. 19.

When examining the maximum achievable task goals across the three experimental categories [see Fig. 20(a)–(c)], we observe that CCA consistently planned for higher task goals than SPS or DSS for the same tasks and starting configurations. Regarding planning time, in cases where all three planners succeeded (see Fig. 21), SPS was significantly faster than DSS, consistent with prior findings [40]. In those same cases, CCA was, on average, 4.3 times faster than SPS and 11.3 times faster than DSS.

In summary, CCA planned trajectories that required substantially less joint movement to complete the tasks. This efficiency likely stems from its unique affordance control feature (discussed in Section VI-B) which allows minor deviations in the EE orientation. Due to the underlying Newton–Raphson method and guess update scheme (discussed in Section V-B), this feature leads to finding solutions that minimally deviate from the start configuration. By minimizing joint movements, the distance to joint limits is also extended, enabling greater task goals. Finally, the closed-chain constrained solution space, explored using the Newton–Raphson method, likely contributes to its superior computational efficiency over SPS and DSS.

E. Current Limitations of Our Planner

In this work, we do not focus on planning in cluttered environments. Our planner only accounts for self-collision and does

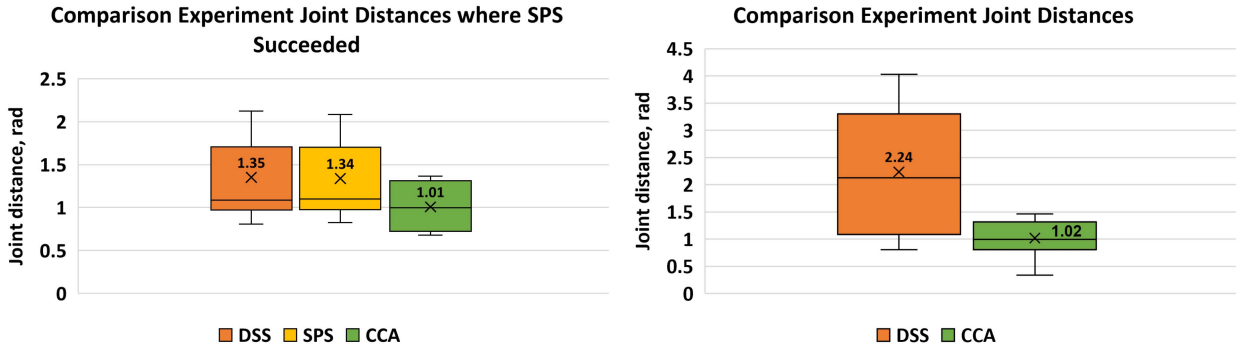


Fig. 18. Joint distance, calculated as the sum of Euclidean distances between consecutive joint configurations in the trajectory, for the comparison experiment tasks.: (a) for cases where SPS succeeded and (b) for all cases.

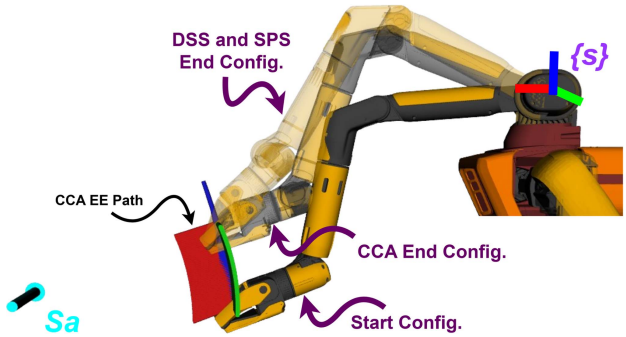


Fig. 19. End configurations for a representative pelican opening experiment, starting from the same initial configuration, as planned by different planners. CCA requires less joint movement to perform the same task. Task affordance represented by screw S_a in frame $\{s\}$.

not consider collisions with external objects. SPS has a similar limitation, whereas DSS addresses this using open-source sampling-based motion planners. In addition, DSS can handle multiple chained or unchained screw-modeled tasks, while our approach currently addresses one screw-modeled task at a time. Finally, we did not study or analyze the class of singularities that our closed-chain mechanism may encounter. Expanding our planner to address these limitations is a direction for future work.

F. Summary of the Benefits of Our Framework

In the following, we summarize notable features of our framework.

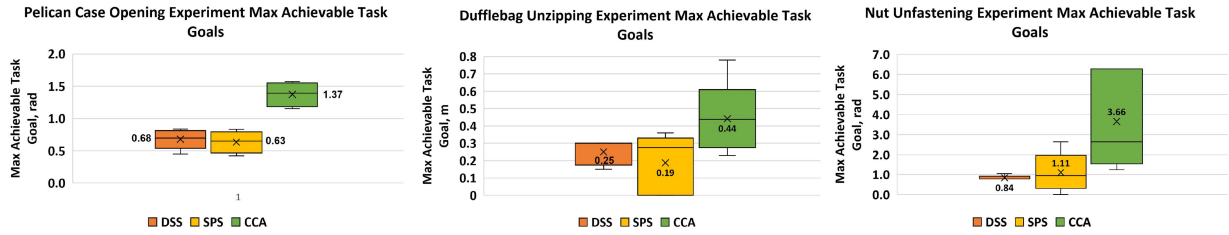
1) Task and Robot Agnosticism: From a modeling standpoint, as detailed in Section IV, our closed-chain mechanism is a concatenation of the robot kinematic chain and the affordance chain, where the robot could be any open-chain serial manipulator and the affordance be any screw-modeled task. Furthermore, from a mathematical standpoint, the secondary network matrix N_s encompasses all screws relevant to defining a task—such as those related to the affordance and gripper orientation—and the primary network matrix N_p encompasses all primary screws responsible for executing that task (e.g., screws describing the joints of a robot). This underscores the generality of our framework across any robot-affordance combination, making it robot- and task-independent.

2) Streamlined Task Definition: From a user standpoint, as outlined in Section V-B, the input to our planner boils down to furnishing the affordance description in the form of a screw and goal, and indicating whether and what aspect of the gripper orientation to control. Conceptually, the task is simply to turn the "affordance crank" in our closed-chain model. Moreover, unlike waypoint-based methods, which may require explicit specification of multiple intermediate EE poses, our approach leverages the continuous screw representation to implicitly plan the trajectory. Thus, our framework eliminates the need for EE waypoints to define screw-modeled task affordances and offers a streamlined, intuitive, and affordance-centric approach to task conceptualization and execution. Furthermore, task constraints, such as affordance limits, are ingrained in our model such that any affordance goal beyond the robot's reachability simply results in a partial joint trajectory solution that brings the EE to the affordance's maximum extent.

3) Grasp Invariance: As our approach connects the robot EE to the affordance link using a virtual spherical joint, the model remains independent of the EE pose at the beginning of the affordance. This enables the robot to initiate the task in any configuration without compromising the validity of the model.

4) Robustness: For the experiments discussed in Section VI-I-C, we saw that our framework is able to complete the tasks even where there are affordance localization and runtime errors causing up to 28 mm deviation between the predicted and actual EE trajectories. We attributed this robustness to our framework's output being joint trajectories starting from the current state of the robot and to the mechanical compliance of the robot itself. Although we have yet to study the upper limit of this robustness and investigate the contribution of each of these factors, we find that the framework performs remarkably well in the face of uncertainties in this implementation, including maintaining the desired EE orientation.

5) Gripper Orientation Control: By incorporating gripper orientation control or freedom through the parameter τ, we can seamlessly include gripper orientation goals along with the affordance. This capability is vital for various contact tasks and can facilitate solution exploration and avoidance of undesirable



(a) (b) (c)

Fig. 20. Highest planned task (affordance) goals achieved by different planners during comparison experiments: (a) Pelican case opening, (b) dufflebag unzipping, and (c) nut unfastening.

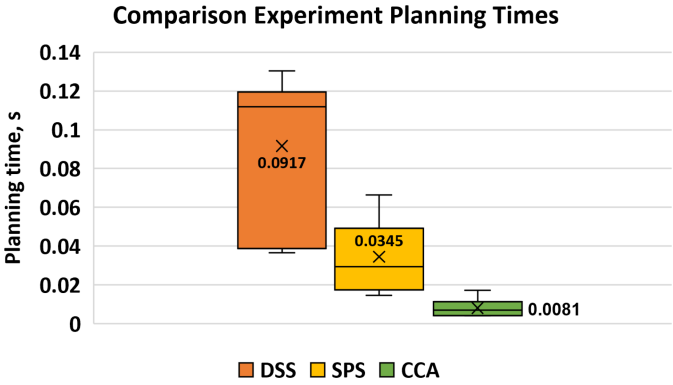


Fig. 21. Planning times of different planners for tasks in the comparison experiments where all three succeeded.

configurations as demonstrated in the affordance and EE control experiments in Section VII-C. This feature is intuitive and accessible through our framework.

6) Computational Efficiency: In Sections VII-B and VII-C, we demonstrated our capability to generate trajectories to perform the tested set of contact tasks in a matter of 0.0077–0.098 s, including long trajectories, such as the 420° valve turn. The tasks were executed smoothly and rapidly, with a 0.3-s time interval between consecutive trajectory points. Thus, our computational efficiency implies that in real time, for every timestep in the trajectory, we are able to generate complete long-horizon trajectories. This is valuable in monitoring and promptly responding to runtime uncertainties during task execution. We also showed in Section VII-D that our method is significantly faster than the state of the art.

VIII. Conclusion and Future Work

We proposed a novel robot-and-task-agnostic framework that offers a computationally efficient and intuitive approach to generate joint trajectories for executing screw-modeled task affordances, while considering the gripper orientation freedom or control. Validation of the framework's efficacy was demonstrated through simulations of a UR5 robot and real-robot implementations on a Boston Dynamics Spot quadruped, covering common articulated and nonarticulated manipulation tasks. Real-world experiments showed robustness of the framework against affordance localization errors and runtime uncertainties. Comparison with the state-of-the-art showed our method is 4x faster and generates joint trajectories that require less joint movement and achieve greater task goals.

In this study, our proposed model comes into effect after the robot reaches the starting pose to perform a task affordance. For future endeavors, we aim to enhance our model to include the approach motion. This will allow us to calculate at once the joint trajectory needed to transition from any arbitrary pose to the starting pose and execute the task affordance. A potential method could utilize Bezier curves to generate smooth trajectories to approach the task's start pose while prioritizing collision avoidance and other constraints [46]. In addition, we utilize this framework to perform more tasks modeled with sequential screws, such as opening a door by turning a knob and pushing the door open. This involves incorporating a causal chaining of affordances and a more intricate robot model that includes both the arm and the body. While the current implementation utilized a space-frame-based closed-chain affordance framework, we acknowledge the potential for even faster computation with sparser matrices in a body-frame setup, as discussed in Section IV-B. Therefore, our future work will involve testing a body-frame-based closed-chain affordance framework.

This work was implemented on a mechanically compliant and sophisticated robot. However, our future investigations will extend to evaluating its performance on other robots, including noncompliant and more cost-effective alternatives. Lastly, we identified our work as a solution to the limitations inherent in the screw primitive approach proposed by Pettinger et al. [6]. However, the software-compliant nature of their system has advantages in addressing runtime uncertainties. For robots that do not have mechanical compliance, we would like to incorporate software compliance into the real-time implementation of our framework to combat runtime uncertainties.

Appendix

To facilitate reproducibility of our experiments, here we provide additional details that pertain to our evaluation studies.

A. Simulation Studies

1) For pure translation tasks [see Fig. 6(a)]
   $S_a = \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}^T$.
2) For pure rotation tasks [see Fig. 6(b)]
   $\hat{s_a} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ at $\begin{bmatrix} 0.817 & 0.291 & -0.006 \end{bmatrix}^T$ m.
3) For screw motion tasks [see Fig. 6(c)]

$\hat{s_a} = \begin{bmatrix}-1 & 0 & 0\end{bmatrix}^T$ with a 5-cm pitch at $\begin{bmatrix}0.817 & 0.291 & -0.006\end{bmatrix}^T$ m.

4) Start configuration for all of these tasks, $q_0 = (0, 0, 0, 0, 0, 0)$.

B. Real-World Implementation Studies

1) Pulling a drawer, [see Fig. 7(a)]
   $S_a = \begin{bmatrix}0 & 0 & 0 & -1 & 0 & 0\end{bmatrix}^T$
   $q_0 = (-0.0008, -0.8798, 1.7327, 0.0127, -1.1322, -0.0027)$.
2) Pushing a drawer, [see Fig. 7(b)]
   $S_a = \begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\end{bmatrix}^T$
   $q_0 = (0.0080, -1.1822, 2.4639, 0.0203, -1.3232, -0.0005)$.
3) Moving a stool, [see Fig. 7(c)]
   $\hat{s_a} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$ at $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$ m
   $q_0 = (0.2084, -0.5254, 1.8599, 0.1858, -1.3719, -0.0743)$.
4) Valve turn case 1, [see Fig. 8(a)]
   $\hat{s_a} = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$ at $\begin{bmatrix}0.6940 & -0.0252 & 0.2199\end{bmatrix}^T$ m
   $q_0 = (0.1513, -1.8213, 2.0578, 0.8435, -0.3443, 0.7954)$.
5) Valve turn case 2, [see Fig. 8(b)]
   $\hat{s_a} = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$ at $\begin{bmatrix}0.6262 & -0.0652 & 0.2223\end{bmatrix}^T$ m
   $q_0 = (0.0140, -2.0268, 2.1910, 0.8096, -0.1487, 0.8129)$.
6) Valve turn case 3, [see Fig. 8(c)]
   $\hat{s_a} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ at $\begin{bmatrix}0.6172 & 0.0636 & 0.2247\end{bmatrix}^T$ m
   $q_0 = (0.0177, -1.2799, 2.1361, 0.0438, -0.8449, -0.0771)$.
7) Valve turn case 4, [see Fig. 8(d)]
   $\hat{s_a} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ at $\begin{bmatrix}0.6163 & 0.0574 & 0.2258\end{bmatrix}^T$ m
   $q_0 = (-0.1408, -1.5098, 1.8449, -0.2122, -0.2910, -0.1452)$.
8) Trajectory planning was performed on a Dell XPS 17 9710 equipped with an Intel Core i7-11800H processor @ 2.30 GHz (16 CPUs) and 32 GB of DDR4 RAM (3200 MHz, 2x16 GB).

C. State-of-The-Art Comparison Studies

1) *CCA Parameters:*
   a) Secondary joint goal accuracy: 1%.
   b) Affordance step: resulting in 50 trajectory points.
2) *SPS and DSS Parameters:*
   a) IK Solver: KDL.[7]
   b) DSS motion planner: PRM [47].
   c) Goal joint tolerance: 0.001, default.
   d) Goal orientation tolerance: 0.001, default.
   e) Goal position tolerance: 0.0001, default.
   f) IK search resolution: 0.005, default.
3) For these experiments, all three planners were compiled with CMAKE_BUILD_TYPE set to RELEASE, enabling compiler optimizations for improved performance.

[7][Online]. Available: https://orocos.org/wiki/orocos/kdl-wiki.html

D. Proof of Exponential Convergence

In the following, we present the proof (adapted from [48]) of the exponential convergence of the error discussed in Section VII-A.

We rewrite the Newton–Raphson procedure of (17) as

$$\Psi(\theta_p) = \theta_p - \frac{f(\theta_p)}{f'(\theta_p)} \tag{27}$$

such that $\Psi(\theta_p) \in C^2(\mathbb{R})$

Using Taylor approximation

$$\theta_p^{i+1} \approx \Psi(\theta_p^*) + (\theta_p^i - \theta_p^*)\Psi'(\theta_p^*) + \frac{1}{2}(\theta_p^i - \theta_p^*)^2\Psi''(\theta_p^*) \tag{28}$$

where θ_p^* is the root value. Since $\Psi(\theta_p^*) = \theta_p^*$, and if $e_{i+1} = \theta_p^{i+1} - \theta_p^*$ and $e_i = \theta_p^i - \theta_p^*$ are the errors, then

$$e_{i+1} \approx e_i\Psi'(\theta_p^*) + \frac{1}{2}e_i^2\Psi''(\theta_p^*). \tag{29}$$

Differentiating (27) w.r.t. θ_p, we get

$$\Psi'(\theta_p^*) = \frac{f(\theta_p^*)f''(\theta_p^*)}{(f'(\theta_p^*))^2}. \tag{30}$$

Since we have a redundant closed-chain mechanism, the Jacobian $f'(\theta_p^*)$ in (18) is rank-deficient, leading to an ill-conditioned expression in (30). Assuming sufficient differentiability, we invoke L'Hôpital's rule

$$\lim_{\theta_p \to \theta_p^*} \frac{f(\theta_p)f''(\theta_p)}{(f'(\theta_p))^2} = \lim_{\theta_p \to \theta_p^*} \frac{f'(\theta_p)f''(\theta_p) + f(\theta_p)f'''(\theta_p)}{2f'(\theta_p)f''(\theta_p)}. \tag{31}$$

Invoking L'Hôpital's rule again

$$= \lim_{\theta_p \to \theta_p^*} \frac{f''(\theta_p)^2 + 2f'(\theta_p)f'''(\theta_p) + f(\theta_p)f^{iv}(\theta_p)}{2\left(f'(\theta_p)f'''(\theta_p) + f''(\theta_p)^2\right)} = \frac{1}{2}. \tag{32}$$

Hence, (29) becomes

$$e_{i+1} \approx \frac{1}{2}e_i + O(e_i^2). \tag{33}$$

Since the first term dominates, the error at each iteration decreases linearly from the previous iteration, resulting in an overall exponential decay.

ACKNOWLEDGMENT

The authors would like to thank Los Alamos National Laboratory for funding the work that motivated this research effort.

REFERENCES

[1] S. J. Jorgensen et al., "Deploying the NASA Valkyrie humanoid for IED response: An initial approach and evaluation summary," in *Proc. IEEE-RAS 19th Int. Conf. Humanoid Robots (Humanoids)*, 2019, pp. 1–8.
[2] Y. Yokokohji, "The use of robots to respond to nuclear accidents: Applying the lessons of the past to the Fukushima Daiichi nuclear power station," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 4, pp. 681–710, 2021.
[3] M. Chiou et al., "Robot-assisted nuclear disaster response: Report and insights from a field exercise," in *Proc. 2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 4545–4552.
[4] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, 2019, Art. no. eaat8414.

[5] S. Hart, P. Dinh, and K. A. Hambuchen, "Affordance templates for shared robot control," in *Proc. 2014 AAAI Fall Symp. Ser.*, 2014 pp. 81–82.

[6] A. Pettinger, F. Alambeigi, and M. Pryor, "A versatile affordance modeling framework using screw primitives to increase autonomy during manipulation contact tasks," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7224–7231, Jul. 2022.

[7] J. J. Gibson, "The theory of affordances," in *The Ecological Approach to Visual Perception*, vol. 1. Hilldale, NJ, USA: Lawrence Erlbaum, 1977, pp. 119–135.

[8] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from RGB-D videos," *Int. J. Robot. Res.*, vol. 32, no. 8, pp. 951–970, 2013.

[9] S. Rezapour Lakani, A. J. Rodríguez-Sánchez, and J. Piater, "Towards affordance detection for robot manipulation using affordance for parts and parts for affordance," *Auton. Robots*, vol. 43, pp. 1155–1172, 2019.

[10] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *Proc. 2015 IEEE Int. Conf. Robot. Autom.*, 2015, pp. 1374–1381.

[11] E. Ugur, E. Şahin, and E. Oztop, "Unsupervised learning of object affordances for planning in a mobile manipulation platform," in *Proc. 2011 IEEE Int. Conf. Robot. Autom.*, 2011, pp. 4312–4317.

[12] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei, "Deep affordance foresight: Planning through what can be done in the future," in *Proc. 2021 IEEE Int. Conf. Robot. Autom.*, 2021, pp. 6206–6213.

[13] Y. Zhu, A. Fathi, and L. Fei-Fei, "Reasoning about object affordances in a knowledge base representation," in Proc. Comput. Vis.–ECCV 13th Eur. Conf., Zurich, Switzerland, Sep. *6–12, 2014, Proc., Part II 13*, 2014, pp. 408–424.

[14] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," in *Proc. 2016 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 2765–2770.

[15] T.-T. Do, A. Nguyen, and I. Reid, "AffordanceNet: An end-to-end deep learning approach for object affordance detection," in *Proc. 2018 IEEE Int. Conf. Robot. Autom.*, 2018, pp. 5882–5889.

[16] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song, "Category-level articulated object pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3706–3715.

[17] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg, "Articulated object interaction in unknown scenes with whole-body mobile manipulation," in *Proc. 2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 1647–1654.

[18] K. Mo, L. J. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani, "Where2Act: From pixels to actions for articulated 3D objects," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 6813–6823.

[19] A. Jain, R. Lioutikov, C. Chuck, and S. Niekum, "ScrewNet: Category-independent articulation model estimation from depth images using screw theory," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 13670–13677.

[20] A. Jain, S. Giguere, R. Lioutikov, and S. Niekum, "Distributional depth-based estimation of object articulation models," in *Proc. Conf. Robot Learn.*, 2022, pp. 1611–1621.

[21] S. Hart, P. Dinh, and K. Hambuchen, "The affordance template ROS package for robot task programming," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 6227–6234.

[22] A. Campos, R. Guenther, and D. Martins, "Differential kinematics of serial manipulators using virtual chains," *J. Braz. Soc. Mech. Sci. Eng.*, vol. 27, pp. 345–356, 2005.

[23] C. H. Dos Santos, R. Guenther, D. Martins, and E. R. De Pieri, "Virtual kinematic chains to solve the underwater vehicle-manipulator systems redundancy," *J. Braz. Soc. Mech. Sci. Eng.*, vol. 28, pp. 354–361, 2006.

[24] H. Simas, R. Guenther, D. Da Cruz, and D. Martins, "A new method to solve robot inverse kinematics using Assur virtual chains," *Robotica*, vol. 27, no. 7, pp. 1017–1026, 2009.

[25] N. Likar, B. Nemec, and L. Žlajpah, "Virtual mechanism approach for dual-arm manipulation," *Robotica*, vol. 32, no. 6, 2014, Art. no. E3.

[26] A. Laurenzi, E. M. Hoffman, M. P. Polverini, and N. G. Tsagarakis, "An augmented kinematic model for the cartesian control of the hybrid wheeled-legged quadrupedal robot Centauro," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 508–515, Apr. 2020.

[27] Z. Jiao et al., "Efficient task planning for mobile manipulation: A virtual kinematic chain perspective," in *Proc. 2021 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8288–8294.

[28] J. Woolfrey and D. Liu, "An optimal dynamic control method for robots with virtual links," in *Proc. 2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 12843–12848.

[29] 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 *Proc. 2015 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 2372–2379.

[30] Y. Wang, C. Smith, Y. Karayiannidis, and P. Øgren, "Whole body control of a dual-arm mobile robot using a virtual kinematic chain," *Int. J. Humanoid Robot.*, vol. 13, no. 01, 2016, Art. no. 1550047.

[31] J.-S. Zhao and S.-T. Wei, "Kinematics of articulated planar linkages," *Front. Mech. Eng.*, vol. 7, 2021, Art. no. 774814.

[32] T. Davies, "Kirchhoff's circulation law applied to multi-loop kinematic chains," *Mechanism Mach. Theory*, vol. 16, no. 3, pp. 171–183, 1981.

[33] H. Ito, K. Yamamoto, H. Mori, and T. Ogata, "Efficient multitask learning with an embodied predictive model for door opening and entry with whole-body control," *Sci. Robot.*, vol. 7, no. 65, 2022, Art. no. 8177.

[34] A. H. Qureshi, J. Dong, A. Baig, and M. C. Yip, "Constrained motion planning networks X," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 868–886, Apr. 2022.

[35] N. Lin et al., "Manipulation planning from demonstration via goal-conditioned prior action primitive decomposition and alignment," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1387–1394, Apr. 2022.

[36] Í. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, and N. Arana-Arexolaleiba, "A review on reinforcement learning for contact-rich robotic manipulation tasks," *Robot. Comput.-Integr. Manuf.*, vol. 81, 2023, Art. no. 102517.

[37] A. Brohan et al., "Do as I can, not as I say: Grounding language in robotic affordances," in *Proc. Conf. Robot Learn.*, 2022.

[38] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Sünderhauf, "SayPlan: Grounding large language models using 3D scene graphs for scalable robot task planning," in *Proc. Conf. Robot Learn.*, 2023, pp. 23–72.

[39] B. Zitkovich et al., "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Conf. Robot Learn.*, 2023.

[40] A. Pettinger, J. Panthi, F. Alambeigi, and M. Pryor, "Efficient constrained motion planning using direct sampling of screw-constraint manifolds," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 9793–9809, 2025, doi: 10.1109/TASE.2024.3513160.

[41] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge, MA, USA: Cambridge Univ. Press, 2017.

[42] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *Proc. IEEE 23rd Conf. Decis. Control*, 1984, pp. 1359–1363.

[43] D. W. Decker, H. B. Keller, and C. T. Kelley, "Convergence rates for Newton's method at singular points," *SIAM J. Numer. Anal.*, vol. 20, no. 2, pp. 296–314, 1983.

[44] E. Sariyildiz and H. Temeltaş, "A new formulation method for solving kinematic problems of multiarm robot systems using quaternion algebra in the screw theory framework," *Turkish J. Elect. Eng. Comput. Sci.*, vol. 20, no. 4, pp. 607–628, 2012.

[45] S. Cho and E. Kim, "Analysis in long-range AprilTag pose estimation and error modeling," in *Proc. 35th Int. Tech. Meeting Satell. Division Inst. Navigation (ION GNSS 2022)*, 2022, pp. 1338–1349.

[46] X. Zhao, Z. Cao, W. Geng, Y. Yu, M. Tan, and X. Chen, "Path planning of manipulator based on RRT-connect and Bezier curve," in *Proc. IEEE 9th Annu. Int. Conf. CYBER Technol. Autom., Control, Intell. Syst.*, 2019, pp. 649–653.

[47] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

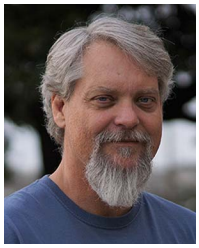
[48] M. Embree, "Lecture 39: Root finding via Newton's method," CAAM 453 Numerical Analysis I., Rice University, Nov. 2009, Accessed: Feb. 21, 2025. [Online]. Available: https://www.sci.utah.edu/beiwang/teaching/cs6210-fall-2016/lecture39.pdf

Janak Panthi received the B.S. degree in engineering and applied mathematics from Loras College, Dubuque, IA, USA, in 2015, and the M.S. degree in mechanical engineering in 2024 from The University of Texas at Austin, Austin, TX, USA, along with a certification of expertise in robotics, where he is currently working toward the Ph.D. degree with the Nuclear & Applied Robotics Group.

Prior to graduate studies, he spent six years in industry, working in design, application engineering, and R&D. His current research focuses on enabling quadruped robots to perform contact and manipulation tasks for radiation survey applications.

Farshid Alambeigi (Member, IEEE) received the M.Sc. degree in robotics and the Ph.D. degree in mechanical engineering from Johns Hopkins University, Baltimore, MD, USA, in 2017 and 2019, respectively.

He is an Assistant Professor with the Walker Department of Mechanical Engineering, The University of Texas at Austin (UT Austin), Austin, TX, USA. He is also a core Faculty Member with Texas Robotics, Austin. At UT Austin, he directs the Advanced Robotic Technologies for Surgery lab. His research interests include surgical robotic systems for various minimally invasive or endoscopic procedures, surgical autonomy, and surgineering.



Mitch Pryor (Member, IEEE) received the M.Sc. and Ph.D. degrees in mechanical engineering from The University of Texas at Austin, Austin, TX, USA, in 1999 and 2002, respectively.

He is a Research Professor with The University of Texas at Austin. He is a core Faculty Member with Texas Robotics, Austin, and a Co-founder of the Nuclear & Applied Robotics Group and the Drilling & Rig Automation Group. Both are interdisciplinary research efforts to deploy long-term autonomous robots in hazardous, uncertain environments to perform manufacturing, material handling, and related tasks.