Proceedings of the ASME 2021
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC-CIE2021
August 17-19, 2021, Virtual, Online

DETC2021-71814

POINT-TO-POINT PATH PLANNING BASED ON USER GUIDANCE AND SCREW LINEAR INTERPOLATION

Riddhiman Laha[1],*, Anjali Rao[2], Luis F.C. Figueredo[1], Qing Chang[3], Sami Haddadin[1], Nilanjan Chakraborty[4]

[1]Munich School of Robotics and Machine Intelligence, Technical Universität München, Germany
[2]Vicarious AI, CA
[3]Mechanical and Aerospace Engineering, University of Virginia, VA
[4]Department of Mechanical Engineering, Stony Brook University, NY

ABSTRACT

Despite the increasing number of collaborative robots in human-centered manufacturing, currently, industrial robots are still largely preprogrammed with very little autonomous features. In this context, it is paramount that the robot planning and motion generation strategies are able to account for changes in production line in a timely and easy-to-implement fashion. The same requirements are also valid for service robotics in unstructured environments where an explicit definition of a task and the underlying path and constraints are often hard to characterize. In this regard, this paper presents a real-time point-to-point kinematic task-space planner based on screw interpolation that implicitly follows the underlying geometric constraints from a user demonstration. We demonstrate through example scenarios that implicit task constraints in a single user demonstration can be captured in our approach. It is important to highlight that the proposed planner does not learn a trajectory or intends to imitate a human trajectory, but rather explores the geometric features throughout a one-time guidance and extend such features as constraints in a generalized path generator. In this sense, the framework allows for generalization of initial and final configurations, it accommodates path disturbances, and it is agnostic to the robot being used. We evaluate our approach on the 7 DOF Baxter robot on a multitude of common tasks and also show generalization ability of our method with respect to different conditions.

Keywords: Motion Planning, Manipulation, Kinematics, Task-Space Planning, Redundancy Resolution

1. INTRODUCTION

Within the last decades, the manufacturing industry has been transformed. Collaborative robots have been increasingly adopted in industries enabling human-robot interaction and collaboration [1]. Robotics technology and human-robot interaction studies have also been transformed to accommodate robot motion in human-centered environments and to ensure safety [2–4]. Yet, despite such advances, industry still largely relies on preprogrammed robots [4]. The challenge to improve autonomy is even more critical when it comes to service robots and scenarios where dual-arm constrained planning is considered [5]. Nonetheless, full autonomy in human-centered, unstructured environments will remain out of reach for the foreseeable future [4].

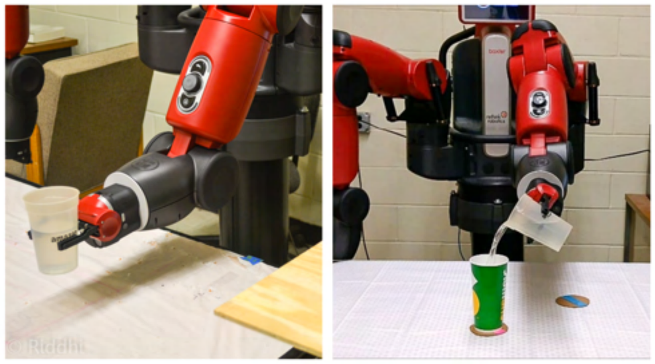


FIGURE 1: TASKS SUCH AS POURING, WHERE THERE THE ROBOT HAS TO TRANSFER A GLASS OF FLUID TO A NEW TABLE OF DIFFERENT HEIGHT, REQUIRE SATISFACTION OF TRANSLATION (WHILE POURING) AS WELL AS ORIENTATION (WHILE MOVING TO THE NEW LOCATION) CONSTRAINTS.

Due to the aforementioned limitations, most cobots are designed with easy access tools to allow a human co-worker to train different tasks—e.g., humans can engage in motion activities with the Baxter (Rethink Robotics), Panda (Franka Emika) and UR-3, UR-5, UR-10 (Universal Robots) by exploring their gravity compensation mode which is activated by a sensor at their arms' cuff or a tablet (UR). Currently, these accessible tools have only been applied to firm point-to-point motion generation or repetition ei-

*Corresponding author: riddhiman.laha@tum.de

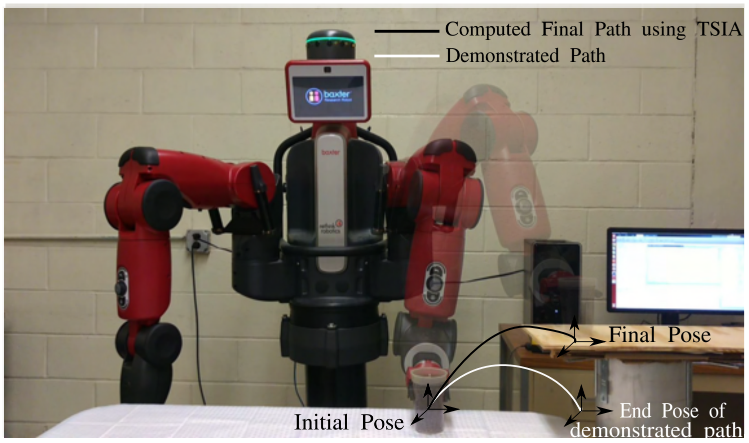


FIGURE 2: SCHEMATIC SKETCH FOR A FLUID TRANSFER TASK DENOTING THE PATHS. A KINESTHETIC DEMONSTRATION IS PROVIDED ALONG THE WHITE LINE. THE FINAL PATH ALONG THE BLACK LINE IS COMPUTED BY OUR *TSIA* USING THE DEMONSTRATED PATH AS A GUIDE.

ther in joint-space or Cartesian space. In other words, any slight modification in the point-to-point trajectory would lead to execution failure and retraining would be required. Take for instance, the task shown in Fig. 1. A simple kinesthetic demonstration or teaching can successfully teach a robot to take a glass from one table to another and pour a glass of water (see Fig. 2). Yet, if we move the receiver cup from one place to another, then we need to retrain the system.

It is a well known fact that generalization to different scenarios (instances) is extremely hard. Usually researchers investigating on "one-shot learning" or "learning from a single example" [6–8] assume the existence of an optimization criterion that defines the task [9] or make strong distributional assumptions about how the task motions are generated [8]. Different strategies, in the context of learning by demonstration (LbD) or imitation learning, rely on the use of machine learning techniques seeking to describe the observed motion—often through methods such as hidden Markov models [10], dynamic movement primitives [11], and Gaussian Mixture models [12]. LbD methods relying solely on human demonstration are however hard to generalize to scenarios only feasible to the cobot (lifting heavy objects). Also, detecting the human pose and its mapping to the cobot configuration remains challenging [1]. Even LbD scenarios using kinesthetic teaching requires a fair amount of user demonstration, particularly in highly constrained manipulation scenarios—which is considerably more cumbersome than simple demonstration. The motion generation problem can also be formulated in the joint or configuration space [13, 14] —which is not suitable for exploring task-space constraints due to the fact that sampling over the constrained manifold is a non-trivial problem. Finally, the probabilistic approach from the aforementioned methods may not be able to address hard constraints, e.g., the cup is not allowed to tilt in the first transfer in Fig. 1.

In contrast, in this paper, we adopt an approach fundamentally different from most works in the literature. Herein, we explore the (one-time) kinesthetic demonstration not to learn a trajectory or imitate the human motion but rather to follow the implicit geometric features underlying the trajectory. This information is observed by the screw transformation throughout the trajectory and is deterministic, hence only a single demonstration is needed. Such constraints are embedded in subgroups of the rigid body motion group, $SE(3)$ [15, 16]. In this work, the geometric constraints embedded in the user demonstrations will guide a basic yet real-time point-to-point[1] kinematic task-space planner based on screw interpolation that integrates the geometric constraints into the new planned path. In this regard, we propose our Task-Space Imitation Algorithm (TSIA) that allows for generalization of initial and final configurations, disturbances along the path (i.e., it can be considered a reactive planner), and it is even agnostic to the task or robot being used. A good example is the fulfilling of the task defined in Fig. 1 with different instances of the problem (different initial pose, robot or table size, for instance). This is mainly due to the fact that the path is fully designed in task-space which is later amalgamated with a resolved motion rate control to compute the corresponding joint space path which can include joint limit avoidance together with additional secondary tasks. When changing robots, one just needs to change the motion rate controller kinematics.

The literature on task-space control with kinematic redundancy resolution offers a wide variety of techniques. A quintessential approach in this context uses the pseudo-inverse of the Jacobian in order to determine the inverse kinematics transformation (resolved motion rate control scheme (RMRC))[17]. The motion planner that we use resolves kinematic redundancy at the velocity level, although using the general framework it is possible to resolve redundancy at acceleration [18] and torque levels [19]. Our algorithm also works with other variants of Jacobian-based redundancy resolution at the velocity level, namely the extended Jacobian technique, kinematic optimization, and augmented task space [20, 21]. The only change that is required to employ one of the variants is an alteration of the equations at the redundancy resolution step. Despite the abundance of literature in operational space planning/control, the combination of human demonstration with redundancy resolution at the velocity level has not been explored so far and is a novel aspect of this paper.

Our approach is based on the observation of the $SE(3)$ pose along the one-time demonstrated trajectory with respect to the desired final pose. TSIA computes a feasible path in the task space based on the screw linear interpolation (ScLERP) of the relative rigid-body poses, which implicitly captures the semantic similarity of the demonstration and geometric constraints along the path. A differential kinematics control-based algorithm is used alongside to convert the task space poses to joint space configurations with nullspace optimization in the joint-level. We demonstrate through extensive experimentation that our novel planning scheme computes local plans that satisfy task-specific constraints using only a single demonstration. Since we use ScLERP, any task-space constraint in which motion within subgroups of $SE(3)$ is permitted, can be handled without explicitly considering [16].

The paper is organized as follows. Section 2 contains the mathematical background. In Section 3, we present the problem formulation for user-guided point-to-point planning. Section 4

[1]Notice that the points used in the planning stage are not necessarily the same points used in the one-time demonstration.

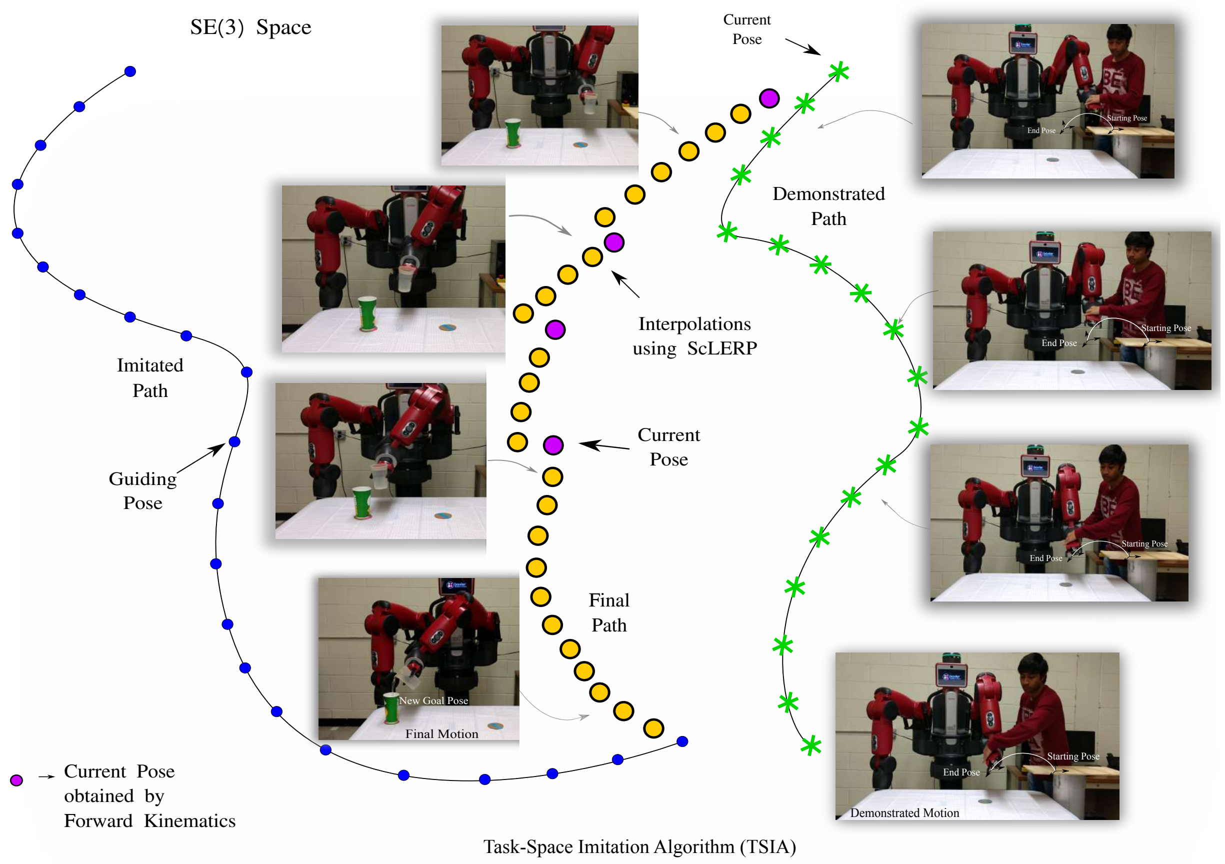


FIGURE 3: OVERVIEW OF THE PROPOSED APPROACH. THE DEMONSTRATED MOTION IS PROVIDED WHICH IS RECORDED AS POSITIONS IN JOINT SPACE (SNAPSHOTS ON THE RIGHT). THE *TSIA* CONVERTS THE DEMONSTRATED MOTION INTO THE IMITATED PATH (EXTREME LEFT) AND PLANS FOR A NEW PATH IN THE TASK-SPACE (CENTRE) GIVEN A NEW GOAL POSE.

describes our solution approach. The experimental setup and results are discussed in Sections 5. In Section 6, we conclude this approach and provide an outlook for future work.

Notations: Throughout this paper, we represent vectors with lowercase bold letters and matrices by uppercase bold letters. Quaternions are represented by lowercase bold letters, whereas dual quaternions are represented by underlined bold variables. The superscript '*' stands for quaternion and dual quaternion conjugate.

2. MATHEMATICAL PRELIMINARIES

Just as unit quaternions encode rigid body orientations, rigid body transformations can be elegantly described by unit dual quaternions as they encode both rotation and translation. A unit dual quaternion, $\underline{\boldsymbol{D}}$, is defined as $\underline{\boldsymbol{D}} = \boldsymbol{d}_r + \epsilon \boldsymbol{d}_d$, where $\epsilon \neq 0$, but $\epsilon^2 = 0$ and $<\boldsymbol{d}_r, \boldsymbol{d}_d> = 0$, considering $\boldsymbol{d}_r$ and $\boldsymbol{d}_d$ as elements of the quaternion group $\mathbb{H}$. Here $\boldsymbol{d}_r$ is a unit quaternion representing the rotation and $\boldsymbol{d}_d = \frac{1}{2}\boldsymbol{p}\boldsymbol{d}_r$ is a quaternion that combines translation and rotation. The translation vector $\boldsymbol{p}$, represented as a pure quaternion, can be computed by $\boldsymbol{p} = 2\boldsymbol{d}_d \otimes \boldsymbol{d}_r$, where the operator $\otimes$ denotes quaternion multiplication. The dual quaternion $\underline{\boldsymbol{D}}$ corresponding to a rigid body transformation can also be written as

$$\underline{\boldsymbol{D}} = \cos\left(\frac{\hat{\theta}}{2}\right) + \hat{\boldsymbol{t}}\sin\left(\frac{\hat{\theta}}{2}\right), \quad (1)$$

where $\hat{\boldsymbol{t}}$ is the unit vector representing the axis of rotation and θ is the dual angle of rotation [22, 23]. The u-th power of a dual quaternion $\underline{\boldsymbol{D}}$ is defined as $\underline{\boldsymbol{D}}^u = cos(u\hat{\theta}) + \hat{\boldsymbol{t}}sin(u\hat{\theta})$. The conjugate of a dual quaternion is denoted by $\underline{\boldsymbol{D}}^* = \boldsymbol{d}_r^* - \epsilon \boldsymbol{d}_d^*$. To interpolate between two quaternions we generally use spherical linear interpolation (SLERP). An analogous scheme can be used to interpolate between two dual quaternions and is known as screw linear interpolation (ScLERP) which is based on the intuition of screw motion. For better understanding, let us consider two poses (frames) of the end effector : $\underline{\boldsymbol{F}}_0$ as the initial pose of the end effector and $\underline{\boldsymbol{F}}_f$ as the final pose of the end effector. The two frames can also be denoted by unit dual quaternions $\underline{\boldsymbol{D}}_0, \underline{\boldsymbol{D}}_f$. One might want to generate a smooth motion between the two poses. This leads to a screw motion which ensures consistent rotation and translation speeds at all points (in the case of linear interpolation). The screw linear motion can be elegantly described by the ScLERP function, defined as follows: ScLERP $(\tau; \underline{\boldsymbol{D}}_0, \underline{\boldsymbol{D}}_f) = \underline{\boldsymbol{D}}_0 \otimes (\underline{\boldsymbol{D}}_0^* \otimes \underline{\boldsymbol{D}}_f)^\tau$ with the time parameter $\tau \in [0, 1]$. This time parameter basically scales the rotation angle about the screw axis (dual angle) at each step of the interpolation. Therefore, any point on the path between $\underline{\boldsymbol{D}}_0$ and $\underline{\boldsymbol{D}}_f$ can be obtained according to the above formula, for a given value of τ. The interpolated poses are also independent of the choice of the coordinate frames. For a more detailed discussion on dual quaternions and their interpolation we refer readers to [24, 25].

3. PROBLEM FORMULATION

We consider motion tasks where there are implicit constraints on the end effector configuration during motion in order to perform the task successfully. Therefore, our goal is to develop a method that can use the demonstration of a single instance and can generate motion plans of other (*possibly nearby*) task instances. We will now introduce some notation to describe the problem formally.

Let FKM denote the forward kinematics map between joint-space ($\mathbb{R}^n$) and the unit dual quaternion $\underline{\boldsymbol{D}}$ depicting the end-effector pose, where N is the number of joints in the serial manipulator. Let the demonstrated (recorded) motion, which is a time sequence of joint angle vectors, be denoted by $\boldsymbol{q}_{\text{rec}}$, i.e.,

$$\boldsymbol{q}_{\text{rec}} = \{\boldsymbol{q}(1), \boldsymbol{q}(2), \cdots, \boldsymbol{q}(N)\},$$

where each element, $\boldsymbol{q}(i)$, is a vector of dimension equal to the degrees of freedom of the manipulator, i.e.,

$$\boldsymbol{q}(i) = \left[q_1(i)\ q_2(i)\ \ldots\ q_r(i)\right]^T, \quad i = 1, \ldots, N.$$

The index i above is not exactly the time index, but the order of i is the time sequence of the configurations reached during the motion. Let the final configuration to be reached by the manipulator be denoted by the unit dual quaternion $\underline{\boldsymbol{g}}_{\text{goal}}$. Let the final computed path be denoted by $\boldsymbol{q}_{\text{final}}$ where

$$\boldsymbol{q}_{\text{final}} = \{\boldsymbol{q}_f(1), \boldsymbol{q}_f(2), \cdots, \boldsymbol{q}_f(m)\}.$$

such that $\underline{\boldsymbol{g}}_{goal} = FKM(\boldsymbol{q}_f(m))$ and $\boldsymbol{q}_f(1)$ is the initial configuration of the manipulator.

The problem that we focus on solving can be stated as follows: *Given a user-demonstrated path recorded in the joint space,* $\boldsymbol{q}_{\text{rec}}$*, a goal configuration to reach in the task space,* $\underline{\boldsymbol{g}}_{\text{goal}}$*, and an initial configuration of the robot in the joint space,* $\boldsymbol{q}_f(1)$*, find a path in the joint space,* $\boldsymbol{q}_{\text{final}}$*, to move from the initial configuration to the goal configuration,* $\underline{\boldsymbol{g}}_{\text{goal}}$*, using the recorded motion,* $\boldsymbol{q}_{\text{rec}}$*, as a guide, while satisfying the task-relevant constraints implicit in the user demonstration (reflected as motion constraints of the end effector) as well as joint contraints.*

Note that in the above formulation we have specified some quantities in the joint space and some quantities in the task space. Without loss of generality, we have assumed that $\boldsymbol{q}_{\text{rec}}$ is known, because manipulators usually have joint encoders that can provide this information. The demonstrated motion in the end effector space can be obtained by using the forward kinematics map. The goal configuration $\underline{\boldsymbol{g}}_{\text{goal}}$ is specified in unit dual quaternions that double cover $SE(3)$ because it is natural to specify goals in task-space rather than joint trajectories which are known to be counter-intuitive for humans. TSIA can be employed in a standard inner/outer loop structure for robotic systems, i.e., for every time-step a reference for the joint velocities is calculated and integrated to the desired joint angles. This can in-turn be used as input to the dynamic controller that applies torque to the joint motors. In our experiments, we assume that the low-level controller accepts a sequence of joint angles that it can move through. Therefore, the final output path, $\boldsymbol{q}_{\text{final}}$, that we aim to compute is in the joint space of the robot.

To illustrate the above problem statement and the solution approach, we consider a planning problem for reaching to a pre-grasp configuration. The illustration in Figure 2 shows Baxter performing a fluid transfer task. The white solid line denotes the demonstrated path (after converting the recorded path in the joint space to task-space using the forward kinematics map) to move the end effector from the initial configuration to the end configuration. Now, given a new pre-grasp configuration, corresponding to a different configuration of the object, we want to compute the path from starting pose to the goal pose (black solid line in Figure 2) by imitating the demonstrated path (or using the demonstrated path as a guide). An example of a task-relevant constraint for the robot in this case would be to avoid collision with objects before reaching the pre-grasp configuration. Another constraint is the avoidance of joint limits during motion. The demonstrated motion satisfies these constraints.

In the next section we present our solution approach. Our solution approach is based on the following observation: *If we are free to choose the initial configuration of the end effector, then for any given goal configuration, by moving the end effector such that the relative motion of the end effector with respect to the goal configuration is identical to that of the demonstrated motion, we can ensure that the implicit task-related constraints present in the demonstration are satisfied.* However, we do not have the ability to start from anywhere, and the starting configuration is usually prescribed or given.

4. COMPUTING USER-GUIDED POINT-TO-POINT PLAN

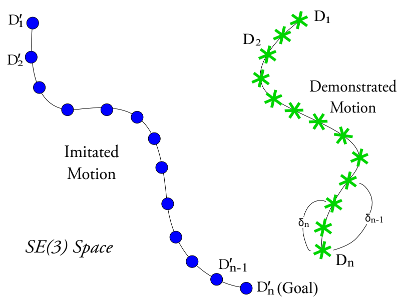
In this section, we discuss our approach for computing a path from the initial end effector configuration to the goal configuration $\underline{\boldsymbol{g}}_{\text{goal}}$ in the end effector space using the recorded motion $\boldsymbol{q_{rec}}$ as a guide. The recorded motion is in the joint space of the robot and consists of a sequence of joint angle vectors. We obtain the corresponding demonstrated motion in task-space using the forward kinematics map. We use a two-step approach for this problem. In the first step, we *replicate* the demonstrated motion in the task space to compute an *imitated path*. In the second, we use the imitated path as a guide to compute the planned path.

4.1 Computing the Imitated Path

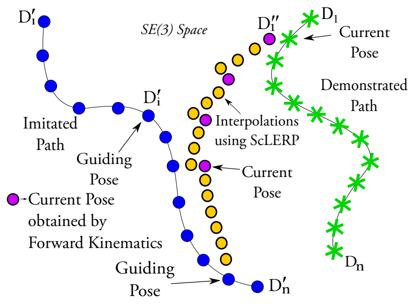
The demonstrated end-effector motion is *replicated* from the goal object configuration ($\underline{\boldsymbol{g}}_{\text{goal}}$) and is defined as the *imitated path*. We first use the forward kinematics map to convert the recorded path into a path in the task space. We use the dual-quaternion parameterization to represent a configuration in the task space. For any configuration, $\underline{\boldsymbol{D}}$, we use the bold face font, $\underline{\boldsymbol{D}}$, to represent the configuration in dual quaternion form. Thus, the demonstrated path in the task space is a sequence of dual quaternions: $DP = \{\underline{\boldsymbol{D}}_1, \underline{\boldsymbol{D}}_2, \cdots, \underline{\boldsymbol{D}}_n\}$.

From the demonstrated motion, we compute the relative motion with respect to the final end effector configuration (in the relative end-effector path space). The transformation, $\underline{\boldsymbol{\delta}}_i$, between the last pose on the demonstrated motion (denoted by $\underline{\boldsymbol{D}}_n$) and every other pose on the demonstrated motion is

$$\underline{\boldsymbol{\delta}}_i = \underline{\boldsymbol{D}}_{i-1}^* \otimes \underline{\boldsymbol{D}}_n, \; i = 2, \ldots, n. \tag{2}$$



(a) Imitation in $SE(3)$



(b) p2p planning in $SE(3)$

FIGURE 4: SCHEMATIC SKETCH OF (A) THE DEMONSTRATED MOTION (GREEN POINTS) AND IMITATED PATH (BLUE POINTS) COMPUTED USING OUR DUAL QUATERNION TRANSFORMATION TECHNIQUE. EACH POINT $\underline{\boldsymbol{D}}'_i$ ON THE IMITATED CURVE HAS A CORRESPONDING POINT $\underline{\boldsymbol{D}}_i$ ON THE DEMONSTRATED CURVE. (B) POINT-TO-POINT PLANNING IN THE TASK-SPACE USING USER GUIDANCE AND IMITATION. PLEASE NOTE HOW THE FINAL PATH BLENDS INTO THE IMITATED PATH SO THAT PART OF THE DEMONSTRATED TRAJECTORY (CONTAINING THE TASK-SPECIFIC CONSTRAINTS) IS REPLICATED.

Let the unit dual quaternion representation of the goal object configuration in task-space be $\underline{\boldsymbol{D}}'_n$. The values of $\underline{\boldsymbol{\delta}}_1 \cdots \underline{\boldsymbol{\delta}}_n$ are used to compute the imitated path using

$$\underline{\boldsymbol{D}}'_{i-1} = \underline{\boldsymbol{D}}'_n \otimes \underline{\boldsymbol{\delta}}_i^*, \; i = 2, \ldots, n. \tag{3}$$

The set of dual quaternions, $IP = \{\underline{\boldsymbol{D}}'_1, \underline{\boldsymbol{D}}'_2, \cdots, \underline{\boldsymbol{D}}'_n\}$, represent the imitated path, which is the demonstrated motion replicated from the goal object configuration.

Figure 4a gives a schematic sketch of the mathematical description given above. The demonstrated motion is depicted by the path with green asterisks and imitated path is the path with blue dots. By replication, we mean that for every point $\underline{\boldsymbol{D}}_i$ on the demonstrated path, there is a point $\underline{\boldsymbol{D}}'_i$ on the imitated path such that the relative transformations between the demonstrated goal $\underline{\boldsymbol{D}}_n$ and $\underline{\boldsymbol{D}}_i$ is the same as the relative transformation between the new goal $\underline{\boldsymbol{D}}'_n$ and $\underline{\boldsymbol{D}}'_i$. Thus, we construct the imitated path starting at the new goal configuration ($\underline{\boldsymbol{D}}'_n$) and ending at some configuration ($\underline{\boldsymbol{D}}'_1$).

4.2 Computing the final motion using the imitated path as guide

The new initial configuration of the end effector $\boldsymbol{q}_f(1)$ is given in the joint space. Using forward kinematics we compute the initial configuration of the end effector in the task space, i.e., $\underline{\boldsymbol{D}}''_1$ (the purple dot at the top in Figure 4b). The goal configuration of the end effector is $\underline{\boldsymbol{D}}'_n$. Path planning from $\underline{\boldsymbol{D}}''_1$ to $\underline{\boldsymbol{D}}'_n$ has two parts:

1. Generation of intermediate target task space configurations using unit dual quaternion interpolation.
2. Computation of the joint configuration to reach the target task space configuration using a linear approximation of the velocity kinematics equations.

Generating Intermediate Targets in Task Space: In Figure 4b, if the manipulator were to start at pose $\underline{\boldsymbol{D}}'_1$, then by imitating the demonstrated motion, it would reach the goal with a motion that would look exactly same in task-space as the demonstrated motion. However, the robot starts at some $\underline{\boldsymbol{D}}''_1$, which is different from $\underline{\boldsymbol{D}}'_1$. Our goal is to move to $\underline{\boldsymbol{D}}'_n$, which is the end point of the imitated path. So, intuitively speaking, our goal is to find a path that blends into the imitated path (the path with blue dots). Note that once we blend into the imitated path, we can follow the imitated path thereafter and have the same motion as the demonstrated motion. Since for many tasks (e.g. moving to a pre-grasp configuration for grasping an object with a parallel jaw gripper for stacking) imitating the later part of the motion is more important than imitating the beginning of the motion, it is not necessary to imitate the whole trajectory to capture the task-relevant constraints. Furthermore, if we consider relative transformations between new initial and final configurations that are *in a neighborhood* of the relative transformations between initial and goal configuration of demonstrated path, we expect the blending to be early enough to capture the relevant portion of the constraints in the demonstration.

Let $\underline{\boldsymbol{D}}_c$ be the current task space pose of the manipulator (a purple point in Figure 4b). Let $\underline{\boldsymbol{D}}'_i$ be a *guiding pose* on the imitated path (we will elaborate later on methodical ways of choosing the guidance pose). We compute a target pose $\underline{\boldsymbol{D}}_t$ using screw linear interpolation in dual quaternion space. The dual quaternion corresponding to $\underline{\boldsymbol{D}}_t$ is computed as

$$\underline{\boldsymbol{D}}_t(\tau) = \underline{\boldsymbol{D}}_c \otimes \left(\underline{\boldsymbol{D}}_c^* \otimes \underline{\boldsymbol{D}}'_i\right)^{\tau}, \tag{4}$$

where $\tau \in [0, 1]$ is the time primitive. Different choices of the parameter τ give different target poses in task space. Note that $\tau = 0$ corresponds to $\underline{\boldsymbol{D}}_c$ and $\tau = 1$ corresponds to $\underline{\boldsymbol{D}}'_i$. The yellow dots in Figure 4b show different target poses.

Intuitively speaking, the dual quaternion interpolation presented here is the analogue of the straight line interpolation in Euclidean space and thus has some properties which are useful in our context. Suppose we want to interpolate between two configurations with the same orientation (i.e. the real part of the dual quaternions are identical). The intermediate points produced with ScLERP have the same orientation. This is helpful in transfer of fluid without spilling, where we want to keep the orientation of the wrist same while changing the position. Similarly, if we interpolate between two poses with the same position but different orientation, ScLERP will produce intermediate poses where the position is kept constant. This is helpful in situations like pouring fluids where we may want to keep the position of the wrist constant while changing the orientation.

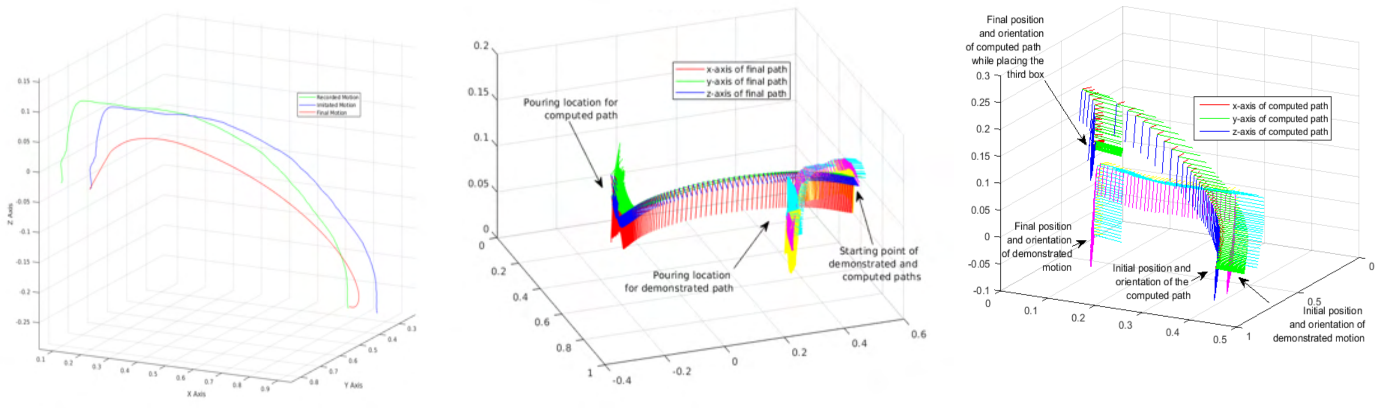


FIGURE 5: ***LEFT.*** **THE RECORDED MOTION (GREEN), IMITATED MOTION (BLUE), AND THE FINAL MOTION (RED) FOR THE REACHING TASK.*CENTRE.* PATHS FOR THE POURING TASK. THE IMPLICIT CONSTRAINT (OF THE END EFFECTOR ORIENTATION NOT CHANGING) THAT IS PRESENT IN THE DEMONSTRATED MOTION IS ALSO REFLECTED IN THE FINAL PATH. *RIGHT.* THE DEMONSTRATED MOTION (THE PATH AT THE BOTTOM) AND THE FINAL MOTION (THE PATH AT THE TOP) COMPUTED FOR PLACING THE THIRD BOX IN THE STACKING TASK. THE READER SHOULD NOTE THE SIMILARITY OF THE PRE-GRASP PORTION OF BOTH THE MOTIONS.**

Computing Joint Space Configuration for a Target Task Space Pose: Consider two consecutive configurations of the end effector at times t and $t+h$. The method to map the end effector configuration from end effector space to joint space is explained below. The end effector configuration is represented by a combination of $\mathbf{p}$ for position and quaternion $\mathbf{r}$ for the rotation. The relationship between rate of change of quaternion and angular velocity is

$$\dot{\boldsymbol{r}} = \frac{1}{2}\omega^s \boldsymbol{r} \quad \text{or} \quad \omega^s = 2\dot{\boldsymbol{r}}\boldsymbol{r}^*, \tag{5}$$

where ω^s is the spatial angular velocity of the rigid body.
Let $\mathbf{p}$ denote the position of the end effector. The spatial velocity $\mathbf{v}^s$ is given by

$$\mathbf{v}^s = \dot{\mathbf{p}} - \hat{\omega}\mathbf{p} = \dot{\mathbf{p}} + 2\hat{\mathbf{p}}\mathbf{J}_1\dot{\boldsymbol{r}}, \tag{6}$$

where $\boldsymbol{J}_1$ is the matrix transformation of $\boldsymbol{\omega}^s$. Therefore, in the matrix form,

$$\begin{bmatrix}\mathbf{v}^s\\ \omega^s\end{bmatrix} = \begin{bmatrix}\mathbf{I}_{3\times3} & 2\hat{\mathbf{p}}\mathbf{J}_1\\ \mathbf{0}_{3\times3} & 2\mathbf{J}_1\end{bmatrix}\begin{bmatrix}\dot{\mathbf{p}}\\ \dot{\boldsymbol{r}}\end{bmatrix} = \mathbf{J}_2\begin{bmatrix}\dot{\mathbf{p}}\\ \dot{\boldsymbol{r}}\end{bmatrix}, \quad \text{where } \mathbf{J}_2 = \begin{bmatrix}\mathbf{I}_{3\times3} & 2\hat{\mathbf{p}}\mathbf{J}_1\\ \mathbf{0}_{3\times3} & 2\mathbf{J}_1\end{bmatrix}. \tag{7}$$

Since $\begin{bmatrix}\mathbf{v}^s\\ \omega^s\end{bmatrix} = \mathbf{J}^s\dot{\boldsymbol{q}}$, where $\dot{\boldsymbol{q}}$ is the joint rate vector and $\mathbf{J}^s$ is the spatial Jacobian,

$$\dot{\boldsymbol{q}} = (\mathbf{J}^s)^T(\mathbf{J}^s(\mathbf{J}^s)^T)^{-1}\begin{bmatrix}\mathbf{v}^s\\ \omega^s\end{bmatrix} = (\mathbf{J}^s)^T(\mathbf{J}^s(\mathbf{J}^s)^T)^{-1}\mathbf{J}_2\begin{bmatrix}\dot{\mathbf{p}}\\ \dot{\boldsymbol{r}}\end{bmatrix}. \tag{8}$$

Let $[\dot{\mathbf{p}}^T \ \dot{\mathbf{q}}^T]^T = \dot{\boldsymbol{\gamma}}$ and $(\mathbf{J}^s)^T(\mathbf{J}^s(\mathbf{J}^s)^T)^{-1}\mathbf{J}_2 = \mathbf{B}$. Then

$$\dot{\boldsymbol{q}} = \mathbf{B}\dot{\boldsymbol{\gamma}}. \tag{9}$$

Let h be a small time-step. Using Euler time-step to discretize (9), we get

$$\frac{\boldsymbol{q}(t+h) - \boldsymbol{q}(t)}{h} = \mathbf{B}(\boldsymbol{q}(t))\frac{\boldsymbol{\gamma}(t+h) - \boldsymbol{\gamma}(t)}{h}, \tag{10}$$

$$\text{or} \qquad \delta\boldsymbol{q} = \mathbf{B}(\boldsymbol{q}(t))\boldsymbol{\gamma}(t+h) - \boldsymbol{\gamma}(t), \tag{11}$$

where $\delta\boldsymbol{q} = \boldsymbol{q}(t+h) - \boldsymbol{q}(t)$. Equation (11) is used to compute the sequence of joint angles of the planned path in the joint space. We summarize these steps in algorithmic format (1) for ease of implementation.

Handling Joint Limits: In the literature, researchers usually deal with joint limits using two methods: (a) Optimization (b) utilizing the Nullspace of the Jacobian. In optimization terminology, the hard joint constraints are converted into soft ones by putting them in an objective with a penalty function. However, the satisfaction of joint limits is not always guaranteed as the Cartesian task always has the highest priority. With the nullspace based approaches, that we are currently using, since the Jacobian is based on the configuration, the nullspace is local, they cannot provide global guarantees for constraint satisfaction [26]. Joint Limits can also be handled robustly using [27], and we are currently exploring this option.

Readers should note that a key aspect of our approach is the selection of the guiding pose on the imitated path. There are various possibilities to explore. We make a simplistic choice where we start with the first point on the imitated path ($\underline{\boldsymbol{D}}_1{}'$) as the first guiding pose and at each iteration choose the next point on the imitated path. If we reach the imitated path then we have blended with the imitated path and we continue choosing the next pose as the guiding pose. Otherwise if all points on the imitated path are exhausted, the guiding pose is always the goal pose. All the experimental results in this paper are based on this choice. Exploration of different choices for the guiding pose including path segmentation for obtaining the task-relevant path fragment is deferred to future work.

4.3 Parameters for TSIA

The variables m and e are the error in orientation and position, respectively, between current pose and guiding pose on the imitated trajectory.

Algorithm 1 TASK-SPACE IMITATION ALGORITHM

```
1:  procedure INPUT:( Goal configuration $\underline{g}_{goal}$ and $q_{rec}$)
2:      From the one-shot user demonstration, record path $q_{rec}$
3:      Define position and orientation tolerance errors: $\text{tol}_p$ and $\text{tol}_o$
4:  end procedure
5:  procedure PATH PLANNING GENERATION:
6:      Convert $q_{rec}$ to a path in task-space using (3) and (4) to obtain the imitated path, $IP$ to new goal configuration.
7:      Compute position and orientation error's norm in the task space ($e$ and $m$ respectively) between the starting point of the imitated path ($\underline{D}'_1$) and the current task space pose ($\underline{D}_c$).
8:      while goal not reached do
9:          Obtain a desired configuration $\underline{D}'_i$ based on the conditions as described earlier for start guiding pose on the $IP$
10:         Find corresponding joint angles and the actual current configuration ($\underline{D}_c$) based on previous $\underline{D}_i$.
11:         Check whether solution is feasible by computing the distance of current configuration ($\underline{D}_c$) to the next guiding configuration ($\underline{D}'_{i+1}$) on the $IP$ and the distance of current configuration to goal.
12:         if $e > \text{tol}_p \parallel m > \text{tol}_o$ then
13:             Change the guide $\underline{D}'_{i+1}$ on $IP$ till goal ($\underline{D}'_n$) not reached and all points on the imitated path not exhausted .
14:             Use ScLERP between poses $\underline{D}'_{i+1}$ and $\underline{D}_c$ to get intermediate poses and select the next pose.
15:         end if
16:     end while
17: end procedure
```

The parameters on which the algorithmic computation directly depends on are the tolerance in position and orientation error between two configurations, the ScLERP interpolation parameter τ, which is set as 0.01, and $\delta\theta_{allowable}$, which denotes the maximum amount by which an angle is allowed to change in an iteration. These are further described in detail below:

- To compute the distance between the position of two poses $\mathbf{p}_1 = (x_1, y_1, z_1)$ and $\mathbf{p}_2 = (x_2, y_2, z_2)$, the Euclidean distance is used, given by

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \quad (12)$$

  To evaluate the orientation of two poses a distance metric for rotation is used. There are several ways which have been discussed in the literature [28]. The distance between two rotations can be defined as the Euclidean distance between two unit quaternions.The distance metric ϕ is a mapping from the unit quaternion space to $\mathfrak{R}$. It is computed as:

$$\phi(\boldsymbol{r}_1, \boldsymbol{r}_2) = \min\{\|\boldsymbol{r}_1 - \boldsymbol{r}_2\|, \|\boldsymbol{r}_1 + \boldsymbol{r}_2\|\}, \quad (13)$$

  where q_1 and q_2 are quaternion representations of the rotations, and $\|\cdot\|$ represents the Euclidean norm. Values of tol_p and tol_o are set to $1e-03$. This is because the manipulator has a positioning accuracy of 0.005 m and thus a tolerance of $1e-03$ is sufficient for the configuration error.

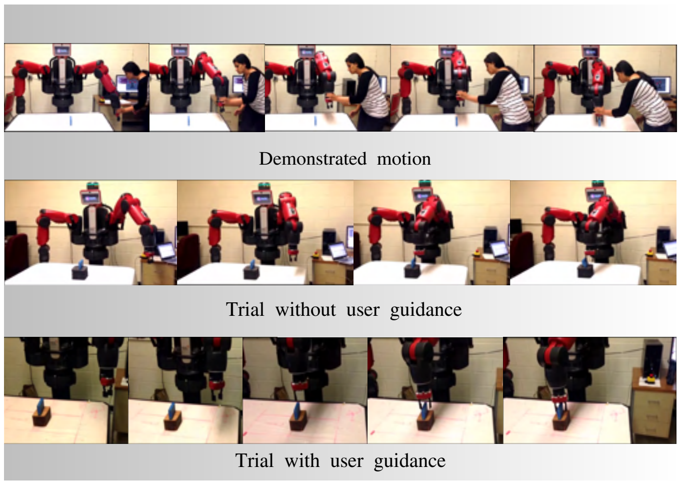


FIGURE 6: REACHING TASK. DEMONSTRATED MOTION (TOP), PLANNING FOR PATH TO NEW GOAL POSE AT A DIFFERENT HEIGHT WITHOUT USER GUIDANCE RESULTS IN FAILURE (MIDDLE), SUCCESSFUL COMPLETION USING THE DEMONSTRATION (BOTTOM).

- The maximum allowable change in an angle is represented by $\delta\theta_{allowable}$ and is set as 0.01 radians. Larger choices of $\delta\theta_{allowable}$ my make the linearization of the velocity kinematics equation invalid, whereas smaller choices of $\delta\theta_{allowable}$ would make the algorithm more computationally expensive. We believe that a choice of 0.01 is a suitable compromise, and it has worked adequately in our experiments.

5. EXPERIMENTAL SETUP AND RESULTS

The Baxter Research Robot is used as the experimental platform. The same parameters as mentioned in the previous section was used across all experiments. Although more performance could be gained by tuning the parameters for each problem, a reasonable default is desirable, especially from a naive user's perspective. All experiments were performed on a workstation with an Intel® Core™ i5 - 4460 processor @ 3.20 GHz. The basic IK-based planner does not consider collision avoidance constraints (like hitting the table or hitting objects while grasping or stacking), or any task-space constraints that are a feature of the task at hand. The experiments shown here are meant to both demonstrate the effectiveness of the point-to-point local planner (ability to generalize from a single demonstration without hitting joint limits and avoiding collisions) as well as preserve the task-specific constraints for the new computed path. For experimental validation of our algorithm we perform multiple experiments consisting of some common household tasks like (a) a reaching task where the end-effector moves to a pre-grasp configuration, (b) stacking objects, (c) transferring fluid-filled containers, and (d) a pouring task.

All the executed tasks comprise of the following three steps: (a) Demonstration: The task to be performed is kinaesthetically demonstrated to the robot.(b) Goal object configuration: A goal object configuration that deviates from the initial configuration in either position, orientation, (or both) and height. (c) Final

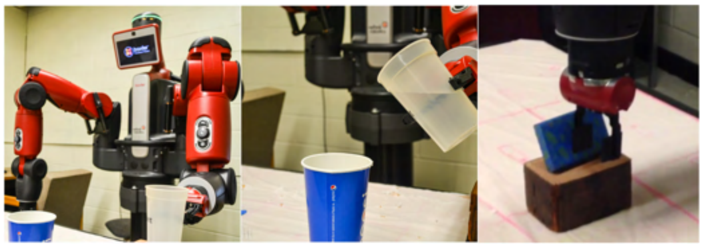


FIGURE 7: *LEFT*.THE END-EFFECTOR MOVES THE WHITE GLASS WHILE THERE IS A CONSTRAINT ON THE ORIENTATION IN THE POURING TASK.*CENTRE*. THE END-EFFECTOR POURS OUT THE FLUID AT A GOAL CONFIGURATION AS THE ORIENTATION CHANGES (TRANSLATION IS FIXED).*RIGHT*. THE END-EFFECTOR FAILS TO REACH THE PRE-GRASP CONFIGURATION AND DISPLACES THE OBJECT DURING A REACHING TASK.

motion: The algorithm discussed in section 4 is used to compute the final motion. The computed set of joint angles for the final motion is used as input to the position control scheme to move the Baxter arm.

A. Reaching Task: The first experiment that we conduct is reaching to a pre-grasp configuration for an object[2]. In this task, the user demonstration was meant to capture the approach to the pre-grasp configuration so that the gripper did not hit the object to be grasped (please see video 2). The video also shows that the basic IK-based algorithm can result in the end effector hitting the vertical block and toppling it (the right panel in Figure 8 shows a snapshot) when moving to the pre-grasp configuration.

We have experimented using a convex and a non-convex object with new object configurations varying in position, orientation, and height for each. For each object, we performed 7 experiments at 7 different poses where a demonstration was given. For each of these demonstrations, we used a neighborhood of 30×30 mm in position and ± 10 degrees in orientation to select 10 trials for checking whether using the demonstrated path results in a successful motion plan using our algorithm. We also did experiments with the height of the object different from the height for demonstration. All of these trials were successful. However going further beyond this region we could observe some failures. For one exemplar experiment, Figure 5 represents the recorded motion in the end effector space in green. The imitated motion is shown in blue. This is the motion computed from the end effector configuration for the goal object configuration. The final motion computed using the algorithm proposed in this study is represented in red. The initial end effector configuration is the same for the final and the demonstrated motions. Note that the final path can (red) reach the goal object configuration, which varies in height. Also, snapshots of the task are provided in Figure 6.

B. Stacking Objects: The next task consisted of stacking similar boxes (Figure 9) at a particular configuration in the workspace of the robot. This task can be thought of as a sequence of multiple reaching tasks. We were successful in accomplishing the task at various locations across the workspace of the robot with a success rate of about 70% in 20 trials[3].

[2]Video link for the reaching task: http://bit.ly/2uy8Gab

[3]Link to video for the stacking task: https://youtu.be/O7VSVY3FCZQ

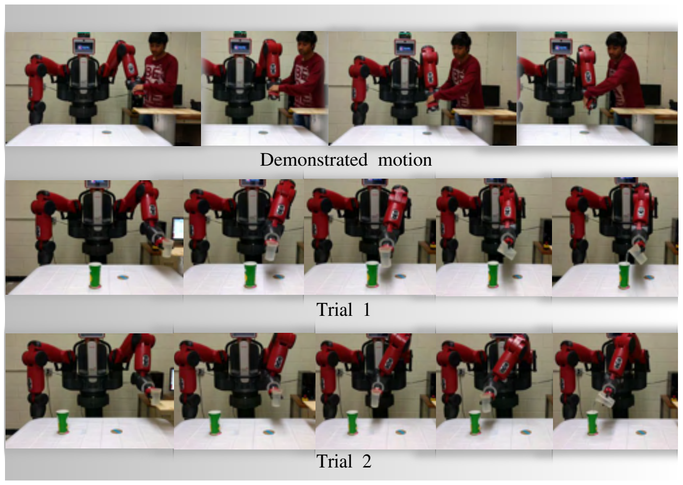


FIGURE 8: OVERVIEW OF THE POURING TASK. THE TOP ROW SHOWS THE KINAESTHETIC DEMONSTRATION PROVIDED TO THE ROBOT. THE LOWER TWO ROWS DEPICT TWO DIFFERENT TRIALS OF THE EXPERIMENT.

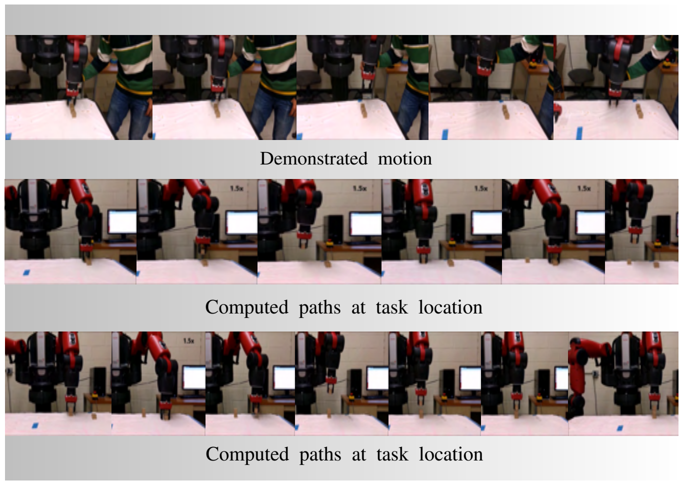

Demonstrated motion
Computed paths at task location
Computed paths at task location

FIGURE 9: STACKING TASK OUTLINE. THE TOP ROW DEPICTS THE KINAESTHETIC DEMONSTRATION PROVIDED TO THE ROBOT. THE MIDDLE AND BOTTOM ROW SHOW ONE SUCCESSFUL TRIAL.

This experiment shows that from a single demonstrated motion the algorithm is successful in planning multiple trajectories with new initial and final configurations for moving multiple boxes. Figure 5 shows the demonstrated motion and the final motion in the end effector space. A key point is that the part of the computed trajectory associated with grasping and release of the box is similar to the demonstrated trajectory, but the poses in between differ as they are not of much significance to the task.

C. Pouring Task: The final task that we considered is a pouring task (Figure 8) where the end-effector has to move a fluid filled cup from an initial position to a goal pouring location (while there's some bound on the orientation) and pour the contents by changing the orientation (while the translation is constant). The human demonstration and the computed path to a new position in the end-effector space are shown in Figure 5[4]. In this example also, we achieved full success in 20 trials across the whole workspace.

[4]Video link for the pouring task: https://youtu.be/r-UH_KZMusg

To summarize, the experimental results demonstrate that our proposed algorithm is fairly successful in computing paths using a single user demonstration as a guide, in a "neighborhood" of the demonstrated task instance. However this neighborhood cannot be characterized analytically. This calls for the development of an interactive framework which incrementally takes in human demonstrations and a self-evaluation scheme so that the robot *believes* it can (cannot) accomplish a task [29].

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented a method for incorporating user demonstration in differential inverse kinematics-based local path planning with screw linear interpolation. Our task-space imitation algorithm generates a sequence of joint angles and is successful in planning for tasks like reaching to a pre-grasp configuration, stacking, fluid transfer, and pouring. Based on a single demonstration, we were able to achieve successful trials where the developed method was able to generalize for the goal object configuration, which varied in position, orientation, and height.

A feature of our approach is that the user guidance based planning is done in unit dual quaternions which are the universal cover of $SE(3)$, which in turn implies that the same user demonstration can be potentially used for multiple robots with different hardware architecture. In future work, we plan to explore this aspect further. One limitation of the presented work is that the local IK-based planner does not consider collision avoidance. In recent work, we have developed a variant of the IK-based planner with ScLERP that takes into consideration collision avoidance (by formulating the problem as a Linear Complementarity Problem (LCP)) [30]. In future work we plan to integrate the LCP-based planner with the imitation scheme described here. We also plan to develop an interactive scheme for the robots to ask for demonstrations, in regions of workspace where they are unable to plan successfully. Some initial work in this direction is reported in [31].

ACKNOWLEDGEMENTS

This work was supported in part by NSF CMMI 1853454, a SBU OVPR seed grant, and the KoBo34 (Verbundprojektnummer V5ARA202) project by the BMBF (grant 16SV7985).

REFERENCES

[1] El Zaatari, S., Marei, M., Li, W., and Usman, Z., 2019. "Cobot programming for collaborative industrial tasks: An overview". *Robotics and Autonomous Systems,* ***116***, pp. 162–180.

[2] Robla-Gomez, S., Becerra, V. M., Llata, J. R., Gonzalez-Sarabia, E., Torre-Ferrero, C., and Perez-Oria, J., 2017. "Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments". *IEEE Access,* **5**, pp. 26754–26773.

[3] Bütepage, J., and Kragic, D., 2017. "Human-Robot Collaboration: From Psychology to Social Robotics". *arXiv:1705.10146*.

[4] Kragic, D., Gustafson, J., Karaoguz, H., Jensfelt, P., and Krug, R., 2018. "Interactive, collaborative robots: Challenges and opportunities". In IJCAI International Joint Conference on Artificial Intelligence, pp. 18–25.

[5] Laha, R., Figueredo, L. F., Vrabel, J., Swikir, A., and Haddadin, S., 2021 (to appear). "Reactive Cooperative Manipulation based on Set Primitives and Circular Fields". In IEEE International Conference on Robotics and Automation.

[6] Wu, Y., and Demiris, Y., 2010. "Towards one shot learning by imitation for humanoid robots". In Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE, pp. 2889–2894.

[7] Zeestraten, M. J. A., Havoutis, I., Silvério, J., Calinon, S., and Caldwell, D. G., 2017. "An approach for imitation learning on Riemannian manifolds". *IEEE Robotics and Automation Letters (RA-L),* ***2***(3), June, pp. 1240–1247.

[8] Yu, T., Finn, C., Xie, A., Dasari, S., Zhang, T., Abbeel, P., and Levine, S., 2018. "One-shot imitation from observing hkumans via domain-adaptive meta-learning". In Robotics:Science and Systems.

[9] Atkeson, C. G., and Schaal, S., 1997. "Learning tasks from a single demonstration". In Proceedings of IEEE International Conference on Robotics and Automation, Vol. 2, IEEE, pp. 1706–1712.

[10] Inamura, T., Toshima, I., Tanie, H., and Nakamura, Y., 2004. "Embodied symbol emergence based on mimesis theory". *The International Journal of Robotics Research,* ***23***(4-5), pp. 363–377.

[11] Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S., 2013. "Dynamical movement primitives: Learning attractor models for motor behaviors". *Neural Computation,* ***25***(2), pp. 328–373. PMID: 23148415.

[12] Calinon, S., and Billard, A., 2007. "Learning of gestures by imitation in a humanoid robot". In *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*, K. Dautenhahn and C.L. Nehaniv ed. Cambridge University Press, pp. 153–177.

[13] LaValle, S. M., 2008. "Configuration space". In *Planning Algorithms*. Cambridge University Press,, pp. 127–183.

[14] Kuffner, J. J., and LaValle, S. M., 2000. "Rrt-connect: An efficient approach to single-query path planning". In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), Vol. 2, IEEE, pp. 995–1001.

[15] Murray, R. M., Li, Z., and Sastry, S. S., 1994. *An Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, FL.

[16] Sarker, A., Sinha, A., and Chakraborty, N., 2020. "On screw linear interpolation for point-to-point path planning". In Proceedings of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'20), IEEE.

[17] Whitney, D. E., 1969. "Resolved motion rate control of manipulators and human prostheses". *IEEE Transactions on man-machine systems,* ***10***(2), pp. 47–53.

[18] Hollerbach, J. M., and Suh, K., 1987. "Redundancy resolution of manipulators through torque optimization". *IEEE Journal on Robotics and Automation,* ***3***(4), pp. 308–316.

[19] Featherstone, R., and Khatib, O., 1997. "Load independence of the dynamically consistent inverse of the jacobian matrix". *The International Journal of Robotics Research,* ***16***(2), pp. 168–170.

[20] Baillieul, J., 1985. "Kinematic programming alternatives for redundant manipulators". In Robotics and Automation. Proceedings. 1985 IEEE International Conference on, Vol. 2, IEEE, pp. 722–728.

[21] Nakanishi, J., Cory, R., Mistry, M., Peters, J., and Schaal, S., 2008. "Operational space control: A theoretical and empirical comparison". *The International Journal of Robotics Research,* ***27***(6), pp. 737–757.

[22] Selig, J. M., 2005. *Geometric Fundamentals of Robotics*, 2nd ed. Springer-Verlag New York Inc.

[23] Figueredo, L. F. C., 2016. "Kinematic control based on dual quaternion algebra and its application to robot manipulators". PhD thesis, University of Brasilia, Brazil.

[24] Adorno, B. V., 2011. "Two-arm manipulation: From manipulators to enhanced human-robot collaboration". PhD thesis, Université Montpellier II-Sciences et Techniques du Languedoc.

[25] Kavan, L., Collins, S., O'Sullivan, C., and Zara, J. Dual quaternions for rigid transformation blending.

[26] Flacco, F., De Luca, A., and Khatib, O., 2015. "Control of redundant robots under hard joint constraints: Saturation in the null space". *IEEE Transactions on Robotics,* ***31***(3), pp. 637–654.

[27] Moe, S., Antonelli, G., Teel, A. R., Pettersen, K. Y., and Schrimpf, J., 2016. "Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results". *Frontiers in Robotics and AI,* ***3***, p. 16.

[28] Huynh, D. Q., 2009. "Metrics for 3d rotations: Comparison and analysis". *Journal of Mathematical Imaging and Vision,* ***35***(2), pp. 155–164.

[29] Laha, R., and Chakraborty, N., 2018. "Task-specific motion planning using user-guidance, imitation, and self-evaluation". In Robotics: Science and Systems Conference Workshop.

[30] Sinha, A., Sarker, A., and Chakraborty, N., 2021. "Task space planning with complementarity constraint-based obstacle avoidance". In Proceedings of ASME IDETC/CIE, 45th Mechanisms and Robotics (MR) Conference.

[31] Laha, R., 2018. "Task-specific motion planning using user-guidance, imitation, and self evaluation". Master's thesis, Stony Brook University, New York.