Knowledge-Enabled Motion Generation for Complex Manipulation Tasks

Dibyendu Das*†, Aditya Patankar§, Fumi Honda†, Dasharadhan Mahalingam§, Nilanjan Chakraborty§, C. R. Ramakrishnan†, and IV Ramakrishnan†

***Abstract*— We present a knowledge-enabled approach to manipulation planning for complex manipulation tasks with objects having similar function, but different geometry. Complex manipulation tasks are characterized by the presence of constraints on the end effector motion of the robot, which may be difficult to specify explicitly. Exploiting our previous work on extracting implicit motion constraints from a kinesthetic demonstration as a sequence of constant screw motions, we present a motion planning algorithm for performing the same task with various objects of similar functionality. Since the extracted screw geometry of motion is a coordinate-invariant representation of the motion, it allows us to transfer the constraints and plan motion across different variations of both the poses and the geometry of the objects, which are different from that in the demonstration. A key aspect of our approach is the design and use of a knowledge representation emphasizing the constraints characterizing the tasks. We present experimental results illustrating the ability of our approach to use one demonstration to plan paths for task instances with (functionally similar) different objects placed in different poses. A video supplement is available at:**
`https://youtu.be/UdnkbstVGrc`

I. Introduction

Complex manipulation tasks are characterized by the presence of constraints on the motion of the end-effector (i.e., a robot gripper or an object rigidly held by the gripper) during the execution of the task. For example, consider scooping and pouring cereal from one container to another with a spoon. There are constraints on the motion of the spoon so that the robot can scoop the contents successfully, move them without dropping, and pour them at the appropriate destination. These constraints are dependent on the task, as well as the properties of the objects being manipulated, including their poses (positions and orientations), sizes, and shapes. Furthermore, different constraints may be in effect at different points during the task execution. Although the constraints that characterize complex manipulation tasks may not always be describable easily (think about the motion for scooping), such constraints are implicit in any kinesthetic demonstration of the task execution. With modern robots, for many complex manipulation tasks, acquiring kinesthetic demonstrations by moving the robot in a zero-gravity mode by holding its hand, is straightforward. Thus, the goal of this paper is to develop a motion planning approach that will use kinesthetic demonstrations on one instance of a task, and use it for planning motion on different instances of the same task with different objects (assuming that the task is achievable by the new object).

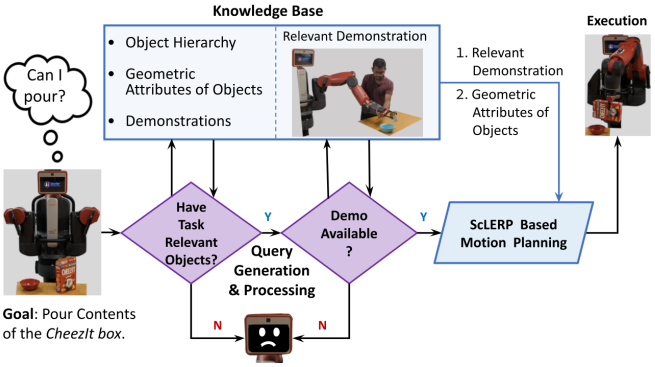


Fig. 1: An overview of our system consisting of a knowledge base, a query generation and processing module, and a motion generation module. The knowledge base consists of an object hierarchy that encodes the primary functions of the objects, the relevant geometric information about the objects along with the demonstrations for a fixed set of tasks.

More specifically, *given a set of complex manipulation tasks, a set of functionally similar objects on which these tasks may be performed, and a small set of demonstrations (assuming at least one for each task), compute a manipulation plan for any task, specified by the initial and final pose of all the task-relevant objects, which ensures satisfaction of the task constraints.* A key distinction of our problem formulation from much of the extant literature [1]–[15] is that, in the literature, it is implicitly assumed that the tasks are performed on the same objects on which the demonstrations are provided. Instead, we present a motion planning algorithm for performing the same task on various objects with similar functionality, by exploiting our work on *extracting motion constraints from a demonstration as a sequence of constant screw motions* (or motions in a one-parameter subgroup of $SE(3)$) [15]. Although there is some work on transferring manipulation plans across objects [16]–[19], they are either (a) for pick-and-place tasks [17]–[19], or (b) do not use any geometric structure of motion, and therefore need multiple examples [16] or (c) for very small amount of motion as is typical in assembly tasks (less than an inch) [18]. In contrast, our approach works for general complex manipulation tasks, using just a single demonstration example, and across wide range of object motion.

Our approach consists of two steps, which are the primary

*Corresponding author. Email: dibyendu.das@stonybrook.edu
†Affiliation: Department of Computer Science at Stony Brook University, Stony Brook, NY-11794, USA.
§Affiliation: Department of Mechanical Engineering at Stony Brook University, Stony Brook, NY-11794, USA.
This work was supported in part by NSF award CMMI 1853454, and a Stony Brook OVPR Seed Grant.

contributions of this work. First, we design and use a knowledge representation (KR) for manipulation tasks that allows us to efficiently store and query the constraints characterizing the tasks. Thus, it allows us to find task constraints from demonstration on one object and transfer them to another object. Although there has been several efforts [20]–[28] on representing knowledge for robots, including knowledge for manipulation, no existing KR facilitate task constraint representation.

Second, by using the fact that the extracted screw geometry of motion is a coordinate-invariant representation of the motion, we develop an algorithm to transfer the constraints and plan motion across different variations of both the poses, and the geometry of the objects, which are different from that in the demonstration. Figure 1 gives an overview of our approach. To the best of our knowledge, this is the first paper that combines classical ideas in KR with classical ideas in screw geometry of motion along with data to plan motion for complex manipulation tasks across different functionally similar objects. We also present experimental results illustrating the ability of our approach to use one demonstration to plan paths for task instances with different objects placed in different poses.

II. Problem Formulation and Solution Overview

The objective of this work is to utilize demonstration of a manipulation task and use them to perform the same task on objects other than those used in the demonstration itself. In this section, we formalize the problem, while formalizing notions of tasks, their associated objects, and demonstrations.

Task-Relevant Objects: Objects whose poses affect manipulation plans for a task, or whose poses are changed by the task, are called *task-relevant objects*. Task-relevant objects are divided into three categories: **Primary** objects are directly manipulated by the robot's gripper. **Secondary** objects are not directly manipulated by the robot but their poses change during the task execution. **Passive** objects are not directly manipulated by the robot but their poses affect the manipulation plan for task execution.

In the example task of scooping cereal from Bowl A and pouring it into another Bowl B as shown in Fig 2, the objects relevant to the scooping task are the grains of cereal being transferred (**secondary** object), the spoon used to transfer the cereal (**primary** object), and the source and destination bowls (**passive** objects).

Fig. 2: Scooping and Pouring task. The task involves scooping the content from Bowl A and pouring it into Bowl B.

Task Instance: We define a task instance as a triple (T, O_{pr}, pa), where T is the task label, O_{pr} is the label of the primary object, pa is the set of passive object-poses, $\{(O_1, \mathbf{g}_1), (O_2, \mathbf{g}_2), \ldots, (O_k, \mathbf{g}_k)\}$ for $k \geq 0$, where pair $(O_i, \mathbf{g}_i)$ is the i-th passive object's label and pose respectively. The set of task instances is denoted by $\mathcal{T}$. For the scoop-and-pour task in our running example, a task instance can be $(\textit{scoop}, \textit{spoon}, (\textit{bowl}, \mathbf{g}_A), (\textit{bowl}, \mathbf{g}_B))$, where $\mathbf{g}_A$ and $\mathbf{g}_B$ are the poses of bowls A and B respectively.

Demonstration of a Task Instance: A kinesthetic demonstration is provided by holding the robot's gripper and moving it in a zero-gravity mode to perform the task. The joint encoder data is stored and converted to a sequence of task-space poses using the forward kinematics map, which represents a path in $SE(3)$. A task demonstration is a sub-sequence of this path, consisting of constant screw segments.

A path in $SE(3)$ can be approximated by a sequence of constant screw segments or 1-parameter subgroups of $SE(3)$. By using screw linear interpolation between each neighboring poses on the stored path, we can ensure that all poses generated the two poses belong to the subgroup , thus maintaining the screw invariant [29]. The set of kinesthetic demonstrations $\mathcal{D} \subset \mathcal{T} \times \mathcal{M}$ is the set of pairs (t, p) where $t \in \mathcal{T}$ is a task instance and $p \in \mathcal{M}$ is a sequence of constant screw segments.

Problem Statement: Given a set of complex manipulation tasks with labels $\mathcal{TL}$, a set of task-relevant objects with labels $\mathcal{OL}$, a set of demonstrations $\mathcal{D}$ with at least one demonstration for each task label, and a given task instance t, compute a manipulation plan, i.e., a sequence of joint angles $\mathbf{\Theta} = \langle \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_m \rangle$, where $\boldsymbol{\theta}_i \in \mathbb{R}^d$ is the vector of joint angles of a $d-$degree-of-freedom manipulator, such that the constraints that characterize t are satisfied.

Solution Overview: Given a task instance t, we first find a set of demonstrations $\{(t_1, p_1), \ldots, (t_m, p_m)\} \subseteq \mathcal{D}$, where the primary and passive objects of t_i are "functionally similar" (see Sec. III) to those of t. We then generate candidate manipulation plans for t by generalizing from the guiding poses in p_i (see Sec. IV).

III. Knowledge Representation of Objects and Tasks

We now describe the first step of our solution: given a task to perform, t, how do we identify relevant demonstrations that can be used to generate manipulation plans for t. We solve this problem by considering the functional similarity of objects: two objects are functionally similar if they can be used interchangeably in a task. We infer functional similarity from a knowledge base of objects and tasks. The key attributes of objects, tasks, and the relation between objects and tasks, which allows us to compute the functional similarity from the KB is described below.

Knowledge About Objects: Our knowledge base defines attributes relevant for the manipulation of rigid body objects. Instances of objects which fully define the value of

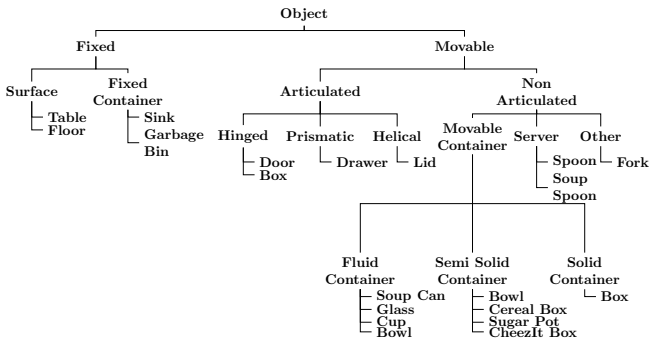


Fig. 3: Fragment of Functionality-based Object Hierarchy in the Knowledge Base

these attributes are kept in a separate database. Thus the knowledge base can be considered as defining the schema for representing object instances. The entire list of attributes is described in detail in Appendix B. The structure of object classes is specified in the knowledge base. An example of the class structure corresponding to a selected set of objects in a kitchen environment is shown in Fig 3. Abstract classes such as *fixed*, *movable*, and *articulated* can easily be extended to objects found in other environments such as warehouse.

Knowledge About Tasks: Complex activities such as "making breakfast", can be decomposed into such primitive tasks and they are not explicitly considered in this paper. Each task in the knowledge base has attributes specifying its task label, label of primary object, and a set of labels corresponding to its passive objects. Note that pose information is not stored with tasks in the knowledge base; hence our representation can be seen as defining schema for task instances.

Relating Objects to Tasks: We represent the relations between the labels of object and task, as relational ontology triples of the form $subject \xrightarrow{\text{relational\_predicate}} object$. For example, a *scoop* task which uses objects in the class *server* as its primary object; is represented by the triple: $scoop \xrightarrow{\text{has\_primary\_objects}} server$. The relationship that *spoon* is a subclass of *server* in the object class structure is given by the triple $spoon \xrightarrow{\text{is\_a}} server$.

Task Similarity: Two object classes O and O' are similar, denoted by $O \sim O'$, if and only if O and O' share the same parent in the object hierarchy. Due to multiple inheritance (*bowl* has two distinct parents), similarity relation "$\sim$" is reflexive and symmetric, but not transitive. Similarity between tasks can be defined using a task hierarchy analogously. Two task instances $t = (T, O, pa)$, and $t' = (T', O', pa')$, are similar (denoted by $t \sim t'$) iff $T \sim T'$ and $O \sim O'$.

IV. Knowledge-enabled Motion Generation

We now describe the manipulation plan for a task t generation based on a selected demonstration. Let the selected demonstration be $D = (t', M')$, where task $t' = \left(T', O'_{pr}, \{(O'_p, \mathbf{g}'_p)\}\right)$, and a feasible $SE(3)$ path M'. Recall that path $M' = \langle \mathbf{g}'_1, \ldots, \mathbf{g}'_m \rangle$ is a sequence of poses of the primary object. The motion corresponding to M' is the sequence of constant screw segments $\left\langle (\mathbf{g}'_i, \mathbf{g}'_{i+1}) \mid i \in [1, m-1] \right\rangle$ and implicitly encodes the task-related motion constraints. The problem then is to generate a new sequence of *guiding poses* $\langle \mathbf{g}_1, \ldots \mathbf{g}_m \rangle$ [15] conforming to the motion constraints, to be fed into our ScLERP based motion planner to generate a motion plan for task t.

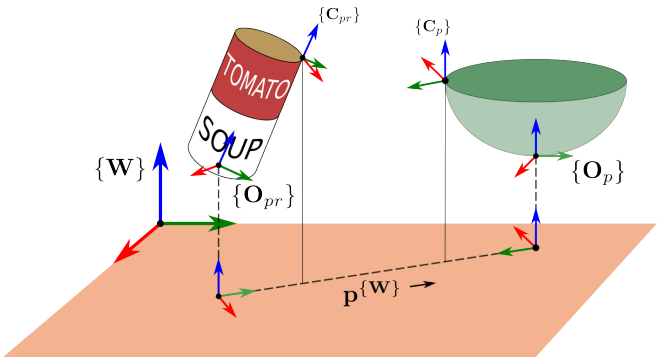


Fig. 4: Schematic sketch showing the computation of the **Transfer Frame**, **C**, for the task of pouring using the pose and geometric information of the associated objects.

While generating guiding poses for t, if the objects are different from those in the demonstration D', we account for the differences in their geometries by introducing an additional reference frame referred to as the **Transfer Frame** for each object. Hereafter the **Transfer Frame** is denoted by $\mathbf{C}$ and its pose along with the poses of all task-related objects are elements of $SE(3)$ unless stated otherwise. The pose of $\mathbf{C}$ depends on the geometry of the object. For instance, when used in a pouring task, the $\mathbf{C}$ frames are chosen such that their poses $\mathbf{g}^C_{pr}$ and $\mathbf{g}^C_p$ lie on the lip of the respective objects and their projections on the world $\mathbf{XY}$ plane lie on the line joining the projections of $\mathbf{g}^O_{pr}$ and $\mathbf{g}^O_p$ on that plane, see Fig. 4. Using the poses and geometries of their objects, we can easily compute ${}_{O_{pr}}\mathbf{g}^{C_{pr}}$ and ${}_{O_p}\mathbf{g}^{C_p}$, their poses relative to their object's reference frame. Combining the geometric information/attributes of all the objects associated with t and t', the following algorithm computes motion, M, from the demonstrated motion, M'.

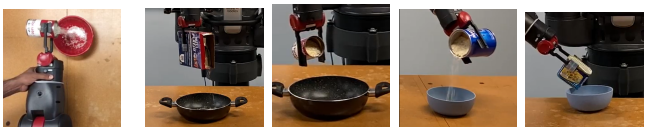
The first step is to determine C_p and C_{pr} in the demonstration task, as described in Fig. 4. The second step is to represent the poses in M' relative to $\mathbf{g}'_p$, the pose of the passive object in t'. This sequence, denoted by ${}_pM^{'} = \langle \mathbf{D}^{p*} \otimes \mathbf{g}'_i \otimes \mathbf{D}^p \mid 1 \leq i \leq m \rangle$ where $\mathbf{D}^p$ is the unit dual quaternion of $\mathbf{g}'_p$ and $\mathbf{D}^{p*}$ is its conjugate[1]. We refer to elements of this sequence as ${}_p\mathbf{g}^{'}_i$.

The next step is to transform poses in ${}_pM^{'}$ such that they represent the motion of $\mathbf{C}_{pr}$ with respect to $\mathbf{C}_p$ expressed relative to $\mathbf{C}_p$. This sequence is computed as

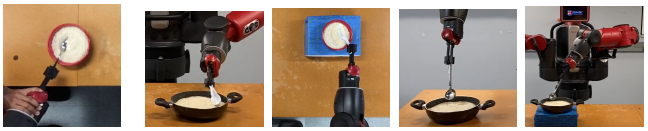
$$M_C = \left\langle {}_{O_p}\mathbf{D}^{C_p*} \otimes {}_p\mathbf{g}^{'}_i \otimes {}_{O_{pr}}\mathbf{D}^{C_{pr}} \mid 1 \leq i \leq m \right\rangle \quad (1)$$

In equation (1) above, ${}_{O_p}\mathbf{D}^{C_p*}$ is the conjugate of the unit dual quaternion of ${}_{O_p}\mathbf{g}^{C_p}$ and ${}_{O_{pr}}\mathbf{D}^{C_{pr}}$ is the unit dual quaternion of ${}_{O_{pr}}\mathbf{g}^{C_{pr}}$ and $\otimes$ denotes dual quaternion product.

[1]We denote pose $\mathbf{g}$ (and sequence of poses M) relative to a another pose b as ${}_b\mathbf{g}$ (resp. ${}_gM$). Poses relative to the world frame are naturally denoted by dropping the reference pose b.



(a) Demo #1 (b) Execution of the pour task



(c) Demo #3 (d) Execution of the scoop task

Fig. 5: (a) and (c) are the demonstrations for *pour* and *scoop* tasks respectively. Trials are executed with different combinations of primary and(or) passive objects with varying poses (even with raised height).

Note that M_C is derived from M' based on the poses and geometries of objects in t'. We also compute ${}_{O_{pr}}\mathbf{g}^{C_{pr}}$ and ${}_{O_p}\mathbf{g}^{C_p}$ for the objects in the new task instance t. We now complete the construction of M, the sequence of guiding poses of t as:

$$M = \left\langle {}_{O_p}\mathbf{D}^{C_p} \otimes \mathbf{g}_i^{''} \otimes {}_{O_{pr}}\mathbf{D}^{C_{pr}{}^*} \mid 1 \leq i \leq m \right\rangle \quad (2)$$

where $\mathbf{g_i}^{''}$ denotes the $i-th$ component of M_C, and ${}_{O_p}\mathbf{D}^{C_p}$ and ${}_{O_{pr}}\mathbf{D}^{C_{pr}}$ refer to the primary and passive objects in the new task instance, t.

V. Implementation and Results

We developed the knowledge-base as a storage of relational ontology triples of the form $subject \xrightarrow{\text{relational\_predicate}} object$. These triples, that bridge the semantic connections between object-labels and task-labels using their attributes form the basis of our knowledge base and are represented using the Web Ontology Language (OWL). To retrieve information from the knowledge base and to make inference on them, we used Prolog as our query processing engine. All the physical experiments were conducted using the Baxter robot from Rethink Robotics [30].

For the experiments, we considered the set of tasks $\mathcal{TL} = \{pour, scoop\}$ with the set of objects $\mathcal{OL} = \{soup\_can,\ bowl,\ glass,\ cup,\ CheezIt\_box,\ spoon,\ soup\_spoon,\ ladle\}$. For the *pour* task, the **primary** and **passive** objects (as described in section III) are part of the the same set i.e. $\{soup\_can,\ bowl,\ glass,\ cup,\ CheezIt\_box\}$. For the *scoop* task, these sets are $\{spoon,\ soup\_spoon,\ ladle\}$ and $\{soup\_can,\ cup,\ bowl\}$ respectively. Not all task instances that can be realized with $\mathcal{OL}$ and $\mathcal{TL}$, are practical; such examples are:

- $\big(pour, (glass, \mathbf{g}_g, -), (CheezIt\_box, \mathbf{g}_{cb})\big)$: Pour from a glass into a CheezIt_box,
- $\big(scoop, (glass, \mathbf{g}_g, -), (soup\_can, \mathbf{g}_{sc})\big)$: Scoop using glass from a soup_can

etc. As explained in section III, the knowledge-base will immediately rule out these impractical ones. For the rest of the task instances we provided 2 demonstrations, each for the *pour* and *scoop* tasks (Fig 5) as follows:

Demo #1 $\big(pour, (soup\_can, \mathbf{g}_{sc}, -), (bowl, \mathbf{g}_b)\big)$
Demo #2 $\big(pour, (CheezIt\_box, \mathbf{g}_{cb}, -), (bowl, \mathbf{g}_b)\big)$
Demo #3 $\big(scoop, (spoon, \mathbf{g}_s, -), (bowl, \mathbf{g}_b)\big)$
Demo #4 $\big(scoop, (soup\_spoon, \mathbf{g}_{ss}, -), (bowl, \mathbf{g}_b)\big)$

For each demonstration, we conducted 15 trials by varying the objects used and their poses – totaling to 60 trials for the two tasks as summarized in Table I. While all 4 demonstrations were provided using the left arm of the robot, the number of trials conducted on the left and right arms respectively are 8 and 7 for Demo #1, 7 and 8 for Demo #2, 8 and 7 for Demo #3, 7 and 8 for Demo #4. Fig 5 shows a subset of our trials:

Pouring from a CheezIt box to a bowl i.e. $\big(pour, (CheezIt\_box, \mathbf{g}_{cb}, -), (bowl, \mathbf{g}_b)\big)$.

Scooping with a soup spoon from a bowl placed at a height i.e. $\big(scoop, (soup\_spoon, \mathbf{g}_{ss}, -), (bowl, \mathbf{g}_b)\big)$.

TABLE I: Conducted Experimental Trials

Demo No.	No. of Trials	Successful	Failed
Demo #1	15	12	3
Demo #2	15	14	1
Demo #3	15	12	3
Demo #4	15	11	4

The overall success rate is about 80%. Since the motion planner does not take collision and joint limit avoidance into consideration, we observed some failures either due to contact with the objects in the surroundings or as a result of the robot arm getting stuck after hitting its joint limit(s). Additionally, for the *scoop* task, since we didn't consider explicit *force control*, the pose of the primary object in the gripper changed while trying to scoop granular materials. These results nevertheless reinforce our overall claim of generating robust manipulation plan for a task without requiring a direct demonstration on its participating objects.

VI. Conclusion

This paper presents a novel approach of generating motion plans for complex manipulation tasks assisted by a knowledge-base that allows efficient storage and retrieval of task constraints. The task constraints are extracted from kinesthetic demonstrations as a sequence of constant screw motions. We present an algorithm for using these constant screws along with the geometric knowledge of all the task related objects to generate motion plans for manipulation of functionally similar objects of different shapes and sizes. We presented results from multiple experimental trials to validate our planning approach.

In this work we have focused on representing motion for "primitive tasks" in the KR as a sequence of constant screws, In future work, we want to pursue hierarchical representation of tasks that allows us to go from an activity to the motion required for performing the whole activity. Furthermore, we would like to explore inclusion of collision avoidance within the motion planning algorithm [31] to make our approach more robust.

REFERENCES

[1] Aude G Billard, Sylvain Calinon, and Rüdiger Dillmann. Learning from humans. *Springer handbook of robotics*, pages 1995–2014, 2016.

[2] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

[3] Sylvain Calinon, Florent D'halluin, Eric L Sauser, Darwin G Caldwell, and Aude G Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics & Automation Magazine*, 17(2):44–54, 2010.

[4] Claudia Pérez-D'Arpino and Julie A Shah. C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4058–4065. IEEE, 2017.

[5] Sylvain Calinon, Antonio Pistillo, and Darwin G Caldwell. Encoding the time and space constraints of a task in explicit-duration hidden markov model. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3413–3418. IEEE, 2011.

[6] Scott Niekum, Sachin Chitta, Andrew G Barto, Bhaskara Marthi, and Sarah Osentoski. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, volume 9. Berlin, Germany, 2013.

[7] Elena Galbally Herrero, Jonathan Ho, and Oussama Khatib. Understanding and segmenting human demonstrations into reusable compliant primitives. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9437–9444. IEEE, 2021.

[8] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Hierarchical neural dynamic policies. *arXiv preprint arXiv:2107.05627*, 2021.

[9] Micha Hersch, Florent Guenter, Sylvain Calinon, and Aude Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*, 24(6):1463–1467, 2008.

[10] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

[11] Matteo Saveriano, Felix Franzel, and Dongheui Lee. Merging position and orientation motion primitives. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7041–7047. IEEE, 2019.

[12] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[13] Riddhiman Laha, Ruiai Sun, Wenxi Wu, Dasharadhan Mahalingam, Nilanjan Chakraborty, Luis FC Figueredo, and Sami Haddadin. Coordinate invariant user-guided constrained path planning with reactive rapidly expanding plane-oriented escaping trees. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 977–984. IEEE, 2022.

[14] Riddhiman Laha, Anjali Rao, Luis FC Figueredo, Qing Chang, Sami Haddadin, and Nilanjan Chakraborty. Point-to-point path planning based on user guidance and screw linear interpolation. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 85451. American Society of Mechanical Engineers, 2021.

[15] Dasharadhan Mahalingam and Nilanjan Chakraborty. Human-guided planning for complex manipulation tasks using the screw geometry of motion. In *2023 International Conference on Robotics and Automation (ICRA)*, 2023.

[16] Skye Thompson, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Shape-based transfer of generic skills. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5996–6002. IEEE, 2021.

[17] Wei Gao and Russ Tedrake. kpam 2.0: Feedback control for category-level robotic manipulation. *IEEE Robotics and Automation Letters*, 6(2):2962–2969, 2021.

[18] Bowen Wen, Wenzhao Lian, Kostas Bekris, and Stefan Schaal. You only demonstrate once: Category-level manipulation from single visual demonstration. *Robotics: Science and Systems*, 2022.

[19] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *Robotics Research: The 19th International Symposium ISRR*, pages 132–157. Springer, 2022.

[20] Moritz Tenorth and Michael Beetz. Knowrob – knowledge processing for autonomous personal robots. In *2009 IEEE/RSJ international conference on intelligent robots and systems*, pages 4261–4266. IEEE, 2009.

[21] Moritz Tenorth and Michael Beetz. Knowrob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research*, 32(5):566–590, 2013.

[22] Jan Winkler, Georg Bartels, Lorenz Mösenlechner, and Michael Beetz. Knowledge enabled high-level task abstraction and execution. In *First Annual Conference on Advances in Cognitive Systems*, volume 2, pages 131–148. Citeseer, 2012.

[23] David Paulius, Yongqiang Huang, Roger Milton, William D Buchanan, Jeanine Sam, and Yu Sun. Functional object-oriented network for manipulation learning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2655–2662. IEEE, 2016.

[24] Moritz Tenorth and Michael Beetz. Representations for robot knowledge in the knowrob framework. *Artificial Intelligence*, 247:151–169, 2017.

[25] Michael Beetz, Daniel Beßler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoğlu, and Georg Bartels. Knowrob 2.0 – a 2^{nd} generation knowledge processing framework for cognition-enabled robotic agents. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2018.

[26] Paola Ardón, Èric Pairet, Ronald P.A. Petrick, Subramanian Ramamoorthy, and Katrin S Lohan. Learning grasp affordance reasoning through semantic relations. *IEEE Robotics and Automation Letters*, 4(4):4571–4578, 2019.

[27] Alex Mitrevsk, Paul G Plöger, and Gerhard Lakemeyer. Ontology-assisted generalisation of robot action execution knowledge. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6763–6770. IEEE, 2021.

[28] Qiang Zhang, Yunzhu Li, Yiyue Luo, Wan Shou, Michael Foshey, Junchi Yan, Joshua B Tenenbaum, Wojciech Matusik, and Antonio Torralba. Dynamic modeling of hand-object interactions via tactile sensing. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2874–2881. IEEE, 2021.

[29] Anik Sarker, Anirban Sinha, and Nilanjan Chakraborty. On screw linear interpolation for point-to-point path planning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9480–9487. IEEE, 2020.

[30] Rethink Robotics. Baxter hardware specifications. `https://sdk.rethinkrobotics.com/wiki/Hardware_Specifications`.

[31] Anirban Sinha, Anik Sarker, and Nilanjan Chakraborty. Task space planning with complementarity constraint-based obstacle avoidance. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 85451. American Society of Mechanical Engineers, 2021.

APPENDIX

A. Mathematical Preliminaries

In this section, we present the mathematical concepts of rigid body motion, that we have used throughout the paper. The set of all rigid body rotations is known as the Special Orthogonal Group of dimension 3 and is denoted as $SO(3)$. Mathematically, $SO(3) = \left\{ \boldsymbol{R} \in \mathbb{R}^{3\times 3} \,\middle|\, \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^{\boldsymbol{T}} = \boldsymbol{I}_3, \left| \boldsymbol{R} \right| = 1 \right\}$, where $\boldsymbol{I}_3$ is a 3×3 identity matrix and $|\cdot|$ is the determinant operator. The configuration space of a rigid body is the set of all positions and orientations of the rigid body, known the Special Euclidean Group of dimension 3, and is denoted as $SE(3) = \mathbb{R}^3 \times SO(3)$, $SE(3) = \left\{ (\boldsymbol{R}, \boldsymbol{p}) \mid \boldsymbol{R} \in SO(3), \boldsymbol{p} \in \mathbb{R}^3 \right\}$. We will use the term "*pose*", for a rigid body configuration $\mathbf{g} \in SE(3)$, and reserve the term "*configuration*" for a kinematic chain of rigid bodies, i.e., an articulated object or a robot. The pose of a rigid body can be expressed by a 4×4 homogeneous transformation matrix as $\boldsymbol{T} = \left[\begin{smallmatrix} \boldsymbol{R} & \boldsymbol{p} \\ \mathbf{0} & 1 \end{smallmatrix} \right]$ where $\mathbf{0}$ is a 1×3 zero vector. The set of all end-effector configurations of a robot is called the *task space* of the robot and it is a subset of $SE(3)$.

Quaternions and Rotations: The quaternions are the set of hypercomplex numbers, $\mathbb{H}$. A quaternion $Q \in \mathbb{H}$ can be represented as a 4-tuple $Q = (q_0, \boldsymbol{q}_r) = (q_0, q_1, q_2, q_3)$, $q_0 \in \mathbb{R}$ is the real scalar part, $\boldsymbol{q}_r = (q_1, q_2, q_3) \in \mathbb{R}^3$ corresponds to the imaginary part. The conjugate, norm, and inverse of a quaternion Q is given by $Q^* = (q_0, -\boldsymbol{q}_r)$, $\|Q\| = \sqrt{QQ^*} = \sqrt{Q^*Q}$, and $Q^{-1} = Q^*/\|Q\|^2$, respectively. Addition and multiplication of two quaternions $P = (p_0, \boldsymbol{p}_r)$ and $Q = (q_0, \boldsymbol{q}_r)$ are performed as $P + Q = (p_0 + q_0, \boldsymbol{p}_r + \boldsymbol{q}_r)$ and $PQ = (p_0 q_0 - \boldsymbol{p}_r \cdot \boldsymbol{q}_r, p_0 \boldsymbol{q}_r + q_0 \boldsymbol{p}_r + \boldsymbol{p}_r \times \boldsymbol{q}_r)$. The quaternion Q is a *unit quaternion* if $\|Q\| = 1$, and consequently, $Q^{-1} = Q^*$. Unit quaternions are used to represent the set of all rigid body rotations, $SO(3)$. The unit quaternion corresponding to a rotation is $Q_R = (\cos\frac{\theta}{2}, \boldsymbol{l}\sin\frac{\theta}{2})$, where $\theta \in [0, \pi]$ is the angle of rotation about a unit axis $\boldsymbol{l} \in \mathbb{R}^3$.

Dual Quaternions and Rigid Displacements: In general, dual numbers are defined as $d = a + \epsilon b$ where a and b are elements of an algebraic field, and ϵ is a *dual unit* with $\epsilon^2 = 0, \epsilon \neq 0$. Similarly, a dual quaternion D is defined as $D = P + \epsilon Q$ where $P, Q \in \mathbb{H}$. The conjugate, norm, and inverse of the dual quaternion D is represented as $D^* = P^* + \epsilon Q^*$, $\|D\| = \sqrt{DD^*} = \sqrt{PP^* + \epsilon(PQ^* + QP^*)}$, and $D^{-1} = D^*/\|D\|^2$, respectively. Another definition for the conjugate of D is represented as $D^\dagger = P^* - \epsilon Q^*$. Addition and multiplication of two dual quaternions $D_1 = P_1 + \epsilon Q_1$ and $D_2 = P_2 + \epsilon Q_2$ are performed as $D_1 + D_2 = (P_1 + P_2) + \epsilon(Q_1 + Q_2)$ and $D_1 \otimes D_2 = (P_1 P_2) + \epsilon(P_1 Q_2 + Q_1 P_2)$, where $\otimes$ on the left denotes dual quaternion product. The dual quaternion D is a *unit dual quaternion* if $\|D\| = 1$, i.e., $\|P\| = 1$ and $PQ^* + QP^* = 0$, and consequently, $D^{-1} = D^*$. Unit dual quaternions can be used to represent the group of rigid body displacements, $SE(3)$. A rigid body displacement (or transformation) is represented by a unit dual quaternion $D_T = Q_R + \frac{\epsilon}{2} Q_p Q_R$ where Q_R is the unit quaternion corresponding to rotation and $Q_p = (0, \boldsymbol{p}) \in \mathbb{H}$ corresponds to the translation.

Screw Displacement: Chasles-Mozzi theorem states that the general Euclidean displacement/motion of a rigid body from the origin $\boldsymbol{I}$ to $\boldsymbol{T} = (\boldsymbol{R}, \boldsymbol{p}) \in SE(3)$ can be expressed as a rotation θ about a fixed axis $\mathcal{S}$, called the *screw axis*, and a translation d along that axis. Plücker coordinates can be used to represent the screw axis by $\boldsymbol{l}$ and $\boldsymbol{m}$, where $\boldsymbol{l} \in \mathbb{R}^3$ is a unit vector that represents the direction of the screw axis $\mathcal{S}$, $\boldsymbol{m} = \boldsymbol{r} \times \boldsymbol{l}$, and $\boldsymbol{r} \in \mathbb{R}^3$ is an arbitrary point on the axis. Thus, the screw parameters are defined as $\boldsymbol{l}, \boldsymbol{m}, \theta, d$. The screw displacements can be expressed by the dual quaternions as $D_T = Q_R + \frac{\epsilon}{2} Q_p Q_R = (\cos\frac{\Phi}{2}, L\sin\frac{\Phi}{2})$ where $\Phi = \theta + \epsilon d$ is a dual number and $L = \boldsymbol{l} + \epsilon \boldsymbol{m}$ is a dual vector. A power of the dual quaternion D_T is then defined as $D_T^\tau = (\cos\frac{\tau\Phi}{2}, L\sin\frac{\tau\Phi}{2})$, $\tau > 0$.

Screw Linear Interpolation (ScLERP): Using Chasles' theorem, it can be inferred that any path in $SE(3)$ can be approximated arbitrarily closely as a sequence of constant screw motions. To perform a one degree-of-freedom smooth screw motion (with a constant rotation and translation rate) between two object poses in $SE(3)$, the screw linear interpolation (ScLERP) can be used. The ScLERP provides a *straight line* in $SE(3)$ which is the closest path between two given poses in $SE(3)$. If the poses are represented by unit dual quaternions D_1 and D_2, the path provided by the ScLERP is derived by $D(\tau) = D_1 \otimes (D_1^{-1} \otimes D_2)^\tau$ where $\tau \in [0, 1]$ is a scalar path parameter. As τ increases from 0 to 1, the object moves between two poses along the path $D(\tau)$ by the rotation $\tau\theta$ and translation τd. Let $D_{12} = D_1^{-1} \otimes D_2$. To compute D_{12}^τ, the screw coordinates $\boldsymbol{l}, \boldsymbol{m}, \theta, d$ are first extracted from $D_{12} = P + \epsilon Q = (p_0, \boldsymbol{p}_r) + \epsilon(q_0, \boldsymbol{q}_r) = (\cos\frac{\theta}{2}, \boldsymbol{l}\sin\frac{\theta}{2}) + \epsilon Q$ by $\boldsymbol{l} = \boldsymbol{p}_r/\|\boldsymbol{p}_r\|$, $\theta = 2\,\mathrm{atan2}(\|\boldsymbol{p}_r\|, p_0)$, $d = \boldsymbol{p} \cdot \boldsymbol{l}$, and $\boldsymbol{m} = \frac{1}{2}(\boldsymbol{p} \times \boldsymbol{l} + (\boldsymbol{p} - d\boldsymbol{l})\cot\frac{\theta}{2})$ where $\boldsymbol{p}$ is derived from $2QP^* = (0, \boldsymbol{p})$ and $\mathrm{atan2}(\cdot)$ is the two-argument arctangent. Then, $D_{12}^\tau = (\cos\frac{\tau\Phi}{2}, L\sin\frac{\tau\Phi}{2})$ is directly derived from $\left(\cos\frac{\tau\theta}{2}, \sin\frac{\tau\theta}{2}\boldsymbol{l}\right) + \epsilon\left(-\frac{\tau d}{2}\sin\frac{\tau\theta}{2}, \frac{\tau d}{2}\cos\frac{\tau\theta}{2}\boldsymbol{l} + \sin\frac{\tau\theta}{2}\boldsymbol{m}\right)$. Note that $\theta = 0, \pi$ corresponds to pure translation between two poses and the screw axis is at infinity.

B. Object Attributes Stored in the Knowledge Base

Here we list all the attributes of the objects we are using in the knowledge base. We only store the attributes in the knowledge base, their actual values are stored separately in another database. These attributes include: (*a*) **Geometric Attributes** like pose, gross geometric aspects like height, length, width, radius of the opening of containers, and other geometric aspects like special graspable regions on the object. The values of some of the geometric attributes like height and radius of the opening are used in our work for computing the **Transfer Frame**, **C**. (*b*) **Inertial attributes** of the object like mass, mass moment of inertia, (*c*) **Visual attributes** like color, visual textures, which may be important for rendering or visual recognition, (*d*) **Contact attributes** like material, tactile texture, coefficient of friction, (*e*) **Mobility Attributes** that describe whether the object is movable or fixed, articulated or not, and, if articulated, then

the screw parameters that constrain their motion, (*f*) **Utility Attributes** that encode what kind of tasks the object can be used in. Utility attributes encode refinements that cannot be encoded in terms of the class hierarchy alone. For instance, the most common use of a spoon may be as the primary object in *scoop* task, but may also be used to mix liquids in a cup. While the common purpose can be inferred from the class structure, utility attributes let us encode secondary usages without complicating the structure.

Although, we're not making use of **Inertial attributes**, **Visual attributes**, and **Contact attributes** in the current scope of work, nevertheless they may come handy in future works when we take *force control* and *collision avoidance* into account.