C-LEARN: Learning Geometric Constraints from Demonstrations for Multi-Step Manipulation in Shared Autonomy

Claudia Pérez-D'Arpino[1] and Julie A. Shah[2]

Abstract— **Learning from demonstrations has been shown to be a successful method for non-experts to teach manipulation tasks to robots. These methods typically build generative models from demonstrations and then use regression to reproduce skills. However, this approach has limitations to capture hard geometric constraints imposed by the task. On the other hand, while sampling and optimization-based motion planners exist that reason about geometric constraints, these are typically carefully hand-crafted by an expert. To address this technical gap, we contribute with C-LEARN, a method that learns multi-step manipulation tasks from demonstrations as a sequence of keyframes and a set of geometric constraints. The system builds a knowledge base for reaching and grasping objects, which is then leveraged to learn multi-step tasks from a single demonstration. C-LEARN supports multi-step tasks with multiple end effectors; reasons about SE(3) volumetric and CAD constraints, such as the need for two axes to be parallel; and offers a principled way to transfer skills between robots with different kinematics. We embed the execution of the learned tasks within a shared autonomy framework, and evaluate our approach by analyzing the success rate when performing physical tasks with a dual-arm *Optimus* robot, comparing the contribution of different constraints models, and demonstrating the ability of C-LEARN to transfer learned tasks by performing them with a legged dual-arm *Atlas* robot in simulation.**

I. Introduction

Planning and executing multi-step dexterous manipulation tasks remains challenging in the field of robotics, particularly with regard to robots with a high number of degrees of freedom (DoF). We are interested in planning for functional tasks, or manipulation tasks requiring multiple end effectors to manipulate objects within an environment through a series of defined steps that accomplish a goal within a specific domain, such as the tasks illustrated in Fig.1. We are driven by real-world applications in which dexterous manipulation can play a key role in improving safety and efficiency during time-critical operations, such as render-safe procedures in explosive ordnance disposal (EOD); and disaster response situations [1], such as the tasks designed for the DARPA Robotics Challenge (DRC). There is a need within these settings for better assisted planning and interaction techniques to enable operators to efficiently and remotely control high-DoF robots – offering rich situational awareness and capable planning assistance while ensuring the manageability of the mental and physical demands placed on the operator.

[1]Claudia Pérez-D'Arpino is with Department of Electrical Engineering and Computer Science, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology `cdarpino@mit.edu`

[2]Julie Shah with the Department of Aeronautics and Astronautics, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology `julie_a_shah@csail.mit.edu`

The shared autonomy framework has been explored previously in the robotics community as a potential solution for exploiting the symbiosis between human skills (such as high-level cognitive understanding for planning) and robot skills (such as rapid computation) for motion planning and perception [2][3]. Results from prior work involving the deployment of remote robots operating under this framework have indicated the potential of this technique for assisting in the execution of functional manipulation tasks [4][5]. However, the proposed solutions require a highly skilled programmer with robotics experience to encode the sequence of motions that the robot should execute for a given task [6][7].

Techniques involving learning from demonstrations [8] have proven to be a successful way for non-experts to teach manipulation tasks to robots. However, this approach has limitations with regard to learning tasks that require the use of hard geometric constraints typically found in functional tasks, such as extracting a cylinder from a larger coaxial cylinder. In these approaches, robots typically learn from variance within the demonstrations and then apply regression over a generative model to compute new motion. One limitation inherent to this approach, however, is its inability to represent and guarantee satisfaction of explicit geometric constraints.

On the other hand, sophisticated motion planning techniques based on optimization exist [4][9] that make use of geometric constraints for planning. Given a set of carefully

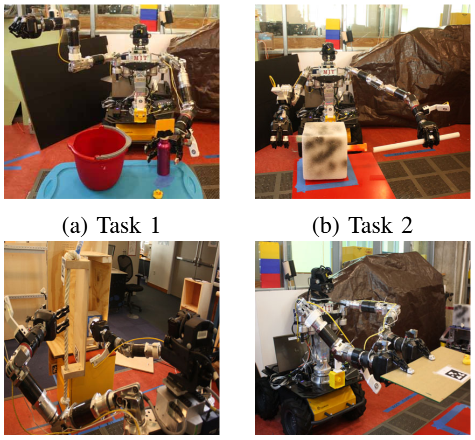
(a) Task 1 (b) Task 2 (c) Task 3 (d) Task 4

Fig. 1: *Optimus* robot executing the test tasks: (a) Pick and transport a cylinder. (b) Secure a box and extract a cylinder. (c) Open a door and push a button in a confined space. (d) Transport a tray.

crafted set of constraints, these methods are capable of whole-body planning for functional tasks involving multiple manipulators with complex kinematics. There is currently a technical gap between the benefits of learning from demonstrations for non-experts to teach robots and the capabilities of motion planners able to reason about explicit geometric constraints. We are interested in augmenting learning from demonstrations with support for use of hard geometric constraints.

In this paper, we present **C-LEARN**: a **C**onstraints **LEARN**ing from demonstrations method for multi-step manipulation tasks with *n* effectors. It builds a knowledge base (KB) of geometric constraints from multiple demonstrations provided through teleoperation in a 3D interactive user interface. Multi-step manipulation tasks are described as a sequence of steps, or keyframes, and a set of constraints per keyframe, which are inferred from a single demonstration by matching it with entries in the knowledge base. In addition to learning constraints as volumes in SE(3), we also introduce the use of constraints that define the relations between objects and end effectors in terms of perpendicular, parallel and coincident axes, together with motion along an axis or within a plane. We refer to these models as *CAD constraints*, due to their similarity with the geometric and distance constraints common to computer-aided design (CAD) software. Finally, we embed the execution of the learned tasks into a two-step workflow of planning and execution in which the system suggests motion plans through visualization to the operator, who can decide to execute the plans or perform modifications using end effector teleoperation.

We argue that ordered sets of explicit geometric constraints are a canonical form for representing the information necessary to reproduce a manipulation task in quasi-static settings – we do not consider dynamic manipulation. This paper contributes a method that (1) learns this representation from demonstrations; (2) supports multi-step manipulation tasks with *n* end effectors, and reasons about models of volumetric and CAD constraints, and (3) offers a principled way to transfer skills learned by one robot to other robot with different kinematics.

Section II contains a summary of related work, and Section III describes the learning and planning framework in C-LEARN. In Sections IV and V, we elaborate upon the algorithmic details of the learning components: building a knowledge base of reaching and grasping affordances (IV), and learning multi-step manipulation tasks from a single demonstration by retrieving information from the knowledge base (V). We embed our core technical contribution for learning constraints into an end-to-end system for planning and execution using a shared autonomy approach, evaluate our system by comparing planning performance across different functional tasks using varied sets of constraints models, and show the ability to transfer knowledge between robots with different kinematics and balancing constraints. Our results are presented in Section VI. Section VII presents a summary and discussion.

II. Related Work

Some techniques for learning manipulation tasks involve learning trajectory segments in which constraints are likely to exist by identifying low variance across demonstrations [10][11][12]. One common approach is to model demonstrations using Gaussian mixture model (GMM) and then generate new motion using Gaussian mixture regression (GMR). This technique results in motion that is expected to remain within the boundaries of the variance with some confidence, but cannot guarantee satisfaction of a specific behavior, such as strictly remaining parallel to an axis or moving within a plane. Under this method, generalization to variants of tasks with different initial or final positions of relevant objects typically employs some geometric transformation or similarity metric; however, it is unclear what guarantees of satisfaction of constraints exist under such transformations. Recognition of these segments with low variance often depends upon the performance of preprocessing steps, such as dynamic time warping (DTW), to align the time series associated with the demonstrations [13][14], and the selection of a number *k* of components for the GMM, which can yield either under- or over-constrained results. Task-parameterized variants of the GMM (e.g. TP-GMM [15]) have been applied to encode manipulator position and orientation requirements for bimanual tasks [16]. Other methods encode task constraints on a cost metric to be minimized [12]; in this case, the model is based on a Hidden Markov Model (HMM) and the planner is able to avoid new obstacles by implementing an asymptotically optimal, sampling-based motion planner. Here, we use the variance in features across demonstrations to select from a number of parametrized constraints models.

The use of hard geometric constraints has been explored previously using optimization-based [4][9] and sampling-based motion planners [17]. For sampling-based techniques, Berenson et al. introduced a representation of geometric constraints, named "task space regions" (TSR), along with a planning algorithm (CBiRRT2) with sampling strategies to satisfy the TSR constraints [17]. In their work, TSRs are defined as a volume in SE(3) with a target frame (target homogeneous transformation) and a tolerance in SE(3), using the roll, pitch, yaw representation for orientation. More recently, Li and Berenson presented a method for learning TSRs in object space in order to plan in narrow passages using a single human demonstration [18]. The authors used sampling to analyze the space surrounding the demonstrated trajectory of an object and reject samples where the object would have been in collision. Since TSRs are a sufficiently general representation for geometric constraints in SE(3), we use this model to represent volumetric constraints in SE(3) such that the end effector should live in this volume for a given keyframe. TSRs can be handled not only by a sampling-based planner [17], but also by optimization-based planners, such as the one used in our work [4]. However, unlike the work by Li and Berenson, in which TSRs were filled with collision-free samples from the vicinity of the continuous trajectory of the object during a single demonstration [18], our method to build TSRs is based on clustering samples from multiple demonstrated keyframes.

The concept of keyframes was introduced by Akgun et al. as a sparse set of poses that accomplishes a learned task if executed in sequence [19]. In that work, keyframes are learned as the points in C-Space needed to recover the trajectory in the demonstrations when using a spline method to generate the new trajectory. Under this definition, keyframes provide a convenient geometric feature for task reproduction but do not necessarily encode regions where task constraints are present. In more recent work, Kurenkov et al. extended the concept to introduce constrained-keyframes as a sequence of position and orientation constraints [20]. In that work, keyframes are clustered using k-means from kinesthetic demonstrations, or explicitly specified by the operator on a GUI. In our work, we similarly incorporated the concept of keyframes by defining manipulation tasks as a sequence of ordered keyframes, wherein each keyframe has a set of simultaneous geometric constraints that should be satisfied by the planner. The sequence of keyframes and the set of constraints for each keyframe are stored in a knowledge base for known affordances [21] for reaching and grasping.

The method of learning from a single-shot demonstration or single sample has been studied previously in the context of robotics [18][22][23][24], as well as in machine learning research [25]. Alexandrova et al. proposed learning multi-step manipulation tasks from a single demonstration by learning target frames (target position and orientation in Cartesian space) with respect to relevant objects in the demonstration, called "landmarks" [24]. Using this approach, the operator has the opportunity to modify learned target frames and landmarks a posteriori via an interactive GUI. Motion during new scenarios of the same task is generated by accordingly transforming the learned target frames and planning by interpolating between poses with a velocity profile using a PR2 robot. If the new transformed frames are infeasible, the task is declared unexecutable.

In a similar spirit, our method of learning a multi-step task also incorporates a single demonstration, but keyframes for reaching and grasping motions leverage information from a pre-learned knowledge base that incorporates information about geometric constraints. The rest of the manipulation steps are learned in a similar fashion, by learning target frames with respect to landmarks, but with the added feature of identifying constraints that remain active so that motions such as "move in a line" are possible.

Multi-step manipulation has also been learned in prior work from a set of kinesthetic demonstrations on a PR2 robot using beta process autoregressive hidden Markov models and dynamic movement primitives [26]. This approach does not handle constraints, but focuses instead on the segmentation of the demonstrations in time, which is a challenge that our method avoids by collecting keyframe demonstrations, which are discrete in nature.

Learning constraints is an active research topic. In work by Phillips et al., articulated constraints (such as a door-opening task), are learned and re-used as previous experiences during planning through an "experience graph" – a collection of previously planned paths [27]. In our work, previous experience is encoded in a knowledge base that specifies affordances, keyframes, and geometric constraints, whereas Phillips et al. encoded experience as previously planned paths. Articulated constraints were also explored by Pillai et al. [28]; in their work, a model of the articulated motion is learned from visual demonstrations.

In a manufacturing context, parts assembly requires hard, tight constraints, such as aligning objects or matching surfaces. With this problem in mind, Somani et al. developed a method for planning robot motion given a set of specified (not learned) constraints using CAD semantics, such as how to assemble two parts given a specification of the CAD relation between the two objects [29]. In our work, we incorporate a parametric model of common CAD constraints with an algorithm to decide when these constraints are active in a given keyframe that has been identified in an unsupervised manner. We demonstrate here how the use of CAD constraints enables the planner to handle real-world functional tasks with skills required in domains such as manufacturing and EOD.

III. Method Overview

A. *Learning Framework*

The learning component consists of two phases. The first phase incorporates a number of demonstrations to learning constraints for approaching and grasping objects in different modes. Modes are defined as different approach angles and grasping positions that the operator might make use of for various purposes, such as grasping a cylinder from the side or from the top. These grasping and approaching constraints are used to build a knowledge base (KB), which is indexed by object type and mode. This KB is kept in memory as the knowledge of the robot. For each $(object, mode)$ pair, motion is described as a ordered sequence of keyframes and a set of geometric constraints for each keyframe. The data collected from the demonstration in Cartesian space is used to select constraints from a catalog of constraints models, including a volume in SE(3) to stay within, motion parallel or perpendicular to an axis, and motion within a plane. We present this learning phase in detail in Section IV.

The second phase has the goal of learning a multi-step manipulation task using a single demonstration. This demonstration is matched with entries in the KB via a distance metric, which enables retrieval of the appropriate set of keyframes and constraints learned from prior demonstrations of approaching and grasping motions. The remaining manipulation steps involved in a task – transportation, location, or release, for example – are learned from the single multi-step demonstration while attempting to identify constraints that might remain active during these steps. We present this learning phase in detail in Section V. Once a multi-step manipulation task has been learned, the sequence of keyframes and set of constraints is used to reproduce the task by invoking an optimization-based motion planner for each keyframe in sequence.

B. Demonstrations Method

Previous work in learning from demonstrations has explored different methods of providing demonstrations [8]. Kinesthetic demonstrations, for example, are intuitive for a novel user but generally require a robot with gravity compensation that can be freely manipulated while the sensing system is on. This capability is often unavailable in robots designed for use in specialized domains and applications that do not traditionally consider learning from demonstrations in the design process, or in scenarios where a high number of DoF and the use of multiple end effectors would complicate the teaching process.

Another possible demonstration source is human motion, which requires a motion capture system for tracking the human body and objects in the environment. A third method is based on graphical user interfaces (GUI) for either providing keyframes [20] or making modifications over the learned model from kinesthetic demonstrations [20] [24].

Unlike previous works, we explored the idea of providing demonstration through a GUI via end effector teleoperation. In this method, the operator guides the end effector of the robot while immersed in a 3D environment that represents the task, either in simulation with virtual objects or while operating the real robot with live sensing data. On one hand, this method enables the operator to provide demonstrations using any potentially relevant robot, without need for a back-drivable robot. On the other, the controls for the interface used to guide the end effector and manipulate the 3D view while providing demonstrations causes the data to be very noisy and non-uniform compared with data collected through human motion or kinesthetic demonstrations. For this reason, the application of time-based techniques intended to align demonstrations in time to later identify constraints in space proves to be unviable in this setting. Fig. 2 shows the process for providing keyframe demonstrations on the interface.

In this work, we implement our system for providing demonstrations, visualizing the results and operating the robot in order to execute the learned tasks in *Director*, a 3D user interface designed for operation of remote robots in shared autonomy, originally developed at MIT for the DARPA Robotics Challenge [6]. Inverse kinematics (IK) is computed using Drake [30].

C. Trajectory Motion Planning

We use an optimization-based motion planner available in Drake [30] that uses an efficient SQP solver available in SNOPT [31]. The motion planning problem is posed as a trajectory optimization problem defined as follows [4] :

$$\underset{q_1,\dots,q_k}{\arg\min} \sum_{j=1}^{k} (q_{nom,j} - q_j)^T W (q_{nom,j} - q_j) + \dot{q}_j^T W_v \dot{q}_j + \ddot{q}_j^T W_a \ddot{q}_j$$
$$subject \quad to \qquad f_i(q_1,\dots,q_k) \le b_i, \quad i = 1,\dots,m, \qquad (1)$$

The cost function drives the set of joint angles $q \in \mathbb{R}^n$ to deviate as little as possible from a nominal configuration $q_{nom} \in \mathbb{R}^n$. Some costs can be added optionally over velocity and acceleration in order to promote smooth trajectories [4].

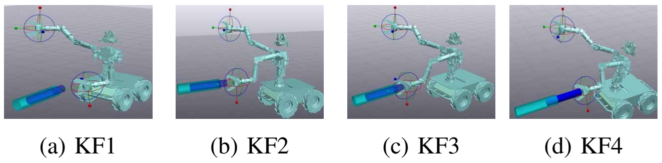
(a) KF1 (b) KF2 (c) KF3 (d) KF4

Fig. 2: User providing keyframe (KF) demonstrations for a multi-step manipulation task. The tasks consists of extracting a cylinder (dark blue) from a concentric external cylinder (light blue).

Throughout this work, we use a nominal configuration and a seed configuration for the optimization equal to the initial position of each segment of the demonstration (the last configuration of the previous keyframe). The set of constraints includes the joint limits, center of mass position and quasi-static constraints (in the case of legged robots), and position and orientation constraints for any link of the robot with a user-specified tolerance.

A solution for this problem returns a trajectory from the initial configuration keyframe to the desired final configuration keyframe that satisfies the constraints. We refer to this solution as a *motion plan*, which can either be feasible and satisfy all constraints or return the best solution identified by the solver while violating a subset of constraints (which are explicitly listed by the solver).

In this paper, we generate a motion plan for each pair of sequential keyframes learned through the demonstration. The learned geometric constraints associated with each keyframe are incorporated into the constraint set of Eq. 1 and used by the Drake planner to compute robot motion.

D. Execution

The final learned task can be executed in shared autonomy mode by suggesting a sequence of motion plans to the operator, who can decide to either execute the plans or make adjustments using teleoperation. The process begins with a preview of a motion plan between two consecutive keyframes. Similarly to [6], the operator is able to review this plan and approve it for execution. The process repeats for each step in the learned sequence of keyframes.

For the execution of the test tasks, we use the *Optimus* robot, a 16-DOF dual-arm highly dexterous mobile manipulator that integrates a Highly Dexterous Manipulation System (HDMS) by RE2, a Husky UGV by Clearpath, 3-finger grippers by Robotiq and a Multisense SL by Carnegie Robotics. We also experiment in simulation with an *Atlas* robot, a 28-DoF humanoid robot by Boston Dynamics.

IV. Learning a Knowledge Base of Constraints from Demonstrations

C-LEARN starts by learning a knowledge base (KB) containing information for reaching and grasping objects. Fig.3 depicts an illustration of the content of a KB. For each object, a set of demonstrations is provided for reaching and grasping the object in different modes. The KB stores the list of modes, as defined in Sec.III.

The operator provides a number of demonstrations for each pair $(object_i, mode_j)$, sometimes referred to as an affordance. The position and orientation of the target object are randomized across demonstrations to elicit variability in features that are not constrained by the task. Each demonstration D_d for reaching and grasping requires the operator to teleoperate the robot though a set of demonstrated keyframes $\{dKF\}$ to collect the following set:

$$D_d = \{dKF_k\}_{k=1:K_d} \tag{2}$$

where d is the demonstration index and k is the demonstrated keyframe index. During the demonstration, poses are marked as keyframes by the user. Each demonstration D_d might have a different number of total demonstrated keyframes K_d. The demonstrations collector records the data associated with each demonstrated keyframe. In this work, we use the positions of the n end effectors with respect to the object as features. The set of all demonstrations provided for a mode is expressed as $\{D_d\}_{d=1:N_{dem}}$.

An entry in the knowledge base $KB_{i,j}$ for $(object_i, mode_j)$ consists of a set of learned keyframes $\{lKF\}$ for that mode, obtained by using k-means clustering over $\{D_d\}$ in the space of selected features, as summarized in Eq. 3. The number of clusters used by the k-means algorithm, $N_{clusters}$, is the rounded average number of keyframes per demonstration in $\{D_d\}$.

$$KB_{i,j} = \{lKF_l\}_{l=1:N_{clusters}} = \{clusters(\{D_d\}_{d=1:N_{dem}})\} \tag{3}$$

Each learned keyframe consists of the set of demonstrated keyframes grouped into the same cluster by k-means. For each learned keyframe, the algorithm identifies the geometric constraints that describe it in a manner such that the task is reproducible by planning motions that meet these constraints. For this purpose, C-LEARN relies on a catalog of parametric constraint models to compare against the data in each cluster and select them based on predefined criteria, such as detecting that the variance in orientation of the end effector with respect to a principal axis of the manipulated object is smaller than a given threshold. The selected constraints are then listed in the KB under the corresponding learned keyframe, as illustrated in Fig.3. Note that this method could produce representations that are either over or under-constrained.

The described clustering approach is similar to that employed by Kurenkov et al. [20], but with the added capability to cluster keyframes involving multiple end effectors by clustering in a higher dimensional space that includes the features of all end effectors. This supports constraints learning for quasi-static manipulation with multiple end effectors that move in sequence or simultaneously, as showcased by Task 4, described in Sec.VI. The list of learned keyframes is ordered to obtain the sequence of execution. For each cluster, the system extracts the sequence number of each demonstrated keyframe included in the cluster, and assigns a position in the sequence according to the mode of that set.

The following is the catalog of parametric constraint models we use in C-LEARN.

A. *Posture Constraints*

Our approach enables a user to implicitly specify through demonstration the following constraints to characterize a learned keyframe:

Task space regions (TSR)[17]: TSRs are defined as a volume in SE(3), which is equivalent to a target frame (target position and orientation of the end effector) and a tolerance of position and orientation around that target frame. TSRs are convenient for representing keyframes for which demonstrations indicate the existence of a variance larger than a predefined threshold. (We refer to this constraint also as "volumetric constraint.") The criteria for characterizing a keyframe as a TSR constraint take into consideration the variance in Cartesian space of the data points associated with the learned keyframe. The learned keyframe is labeled as a TSR constraint if the variance is greater than a user-specified threshold T_{TSR} in any of the dimensions.

CAD Posture Constraints: We introduce the use of hard geometric orientation constraints that consider the cases in which an axis of the end effector should be parallel (or perpendicular) to an axis of an object within the environment. The criteria for characterizing a keyframe as a CAD constraint takes the variance in SO(3) into consideration. The three rotational DoF of the end effector are compared with those of the object, and an orientation constraint is created if the variation of one with respect to the other is smaller than a predefined threshold T_{CAD}. Intuitively, this can be referred to as "locking" the rotation of the end effector into alignment with an axis of the object if the variance of the relative rotation between them is smaller than a given threshold – similarly to user-assistance behavior common to CAD software.

B. *CAD Trajectory Constraints*

The following constraints can be used to describe the motion between keyframes:

Move-in-line Constraint: A move-in-line trajectory constraint is included in the motion plan between two consecutive keyframes (dKF_i, dKF_{i+1}) if a posture CAD constraint is active in both. This constraint connects the target frame of dKF_i with the target frame of dKF_{i+1} using a straight line in Cartesian space along which the end effector orientation should remain constant.

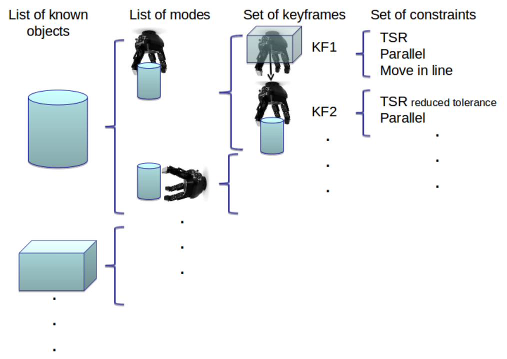

Fig. 3: Illustration of the content of the knowledge base for reaching and grasping motions.

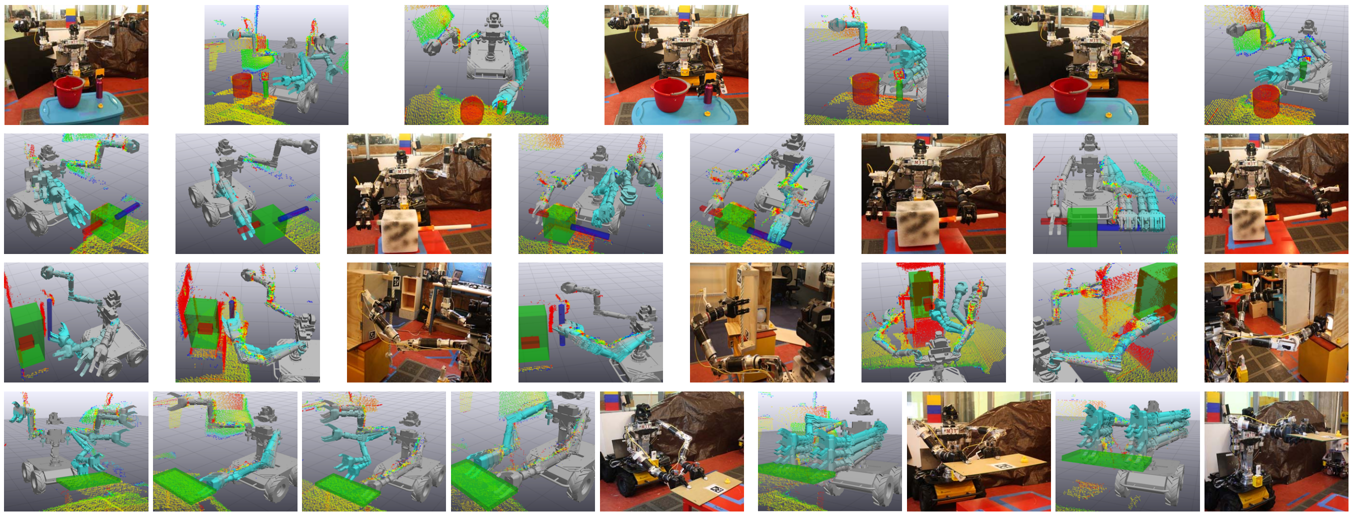
Fig. 4: *Optimus* robot performing the four test tasks autonomously. Motion plans for each keyframe are shown using still images of the trajectory, with color ranging from gray (initial position) to light blue (end position). Tasks 1 to 4 are presented in row 1 to 4, respectively.

V. Learning a Multi-step Manipulation Task from a Single Demonstration

In the second phase of our learning framework, C-LEARN leverages the KB to learn multi-step manipulation tasks from a single demonstration. The form of the demonstration consists of a single sequence of keyframes that accomplish a given task. For example, Fig.2 illustrates a series of demonstrated keyframes for extracting a cylinder from a concentric external cylinder.

For keyframes associated with reaching and grasping motions, the algorithm matches the keyframes with one of the $(object_i, mode_j)$ entries in the KB, by selecting the entry that minimizes a distance metric. In our work, we use a simple Cartesian distance metric between ordered pairs (as indicated by the sequence) of keyframes. The final keyframes are retrieved from the KB.

For subsequent keyframes after grasping the object, we rely only on the single demonstration to learn the rest of the task. The algorithm includes each demonstrated keyframe in the learned task sequence in coordinates relative to the last previous keyframe (so that the entire sequence is susceptible to displacements and rotations in a new environment with different objects positions). This is essentially equivalent to remember the keyframes after grasping the object in relative coordinates, but with the additional benefit of analyzing what constraints remain active between each pair of consecutive keyframes (dKF_i, dKF_{i+1}). If a constraint is identified to remain active according to the catalog criteria, the corresponding trajectory constraint is added to the planner. For example, in the case of the task illustrated in Fig.2, the extraction trajectory should satisfy that the cylinders remain concentric, which is expressed in our framework as a CAD trajectory constraint (for example: "move in line between two target frames with the same orientation").

It is impractical to build a knowledge base containing all tasks for all steps; instead, we restrict the KB to the segment of the manipulation concerned with reaching and grasping – which can be quickly demonstrated more than once. We hypothesize that, while all possible objects in the world cannot be individually considered in the KB, the majority of the objects can be manipulated in practice with a small variant of the basic forms known to the KB. This topic is outside the scope of this paper, but offers an empiric motivation for our selection of the KB representation.

VI. Evaluation

We assess the benefits of C-LEARN by learning and performing four functional tasks that incorporate geometric constraints on motion, similar to those present in EOD procedures or manufacturing. In this section, we present the success rate when performing the tasks using the dual-arm *Optimus* robot (VI-A); demonstrate in simulation the transfer of learned tasks from the Optimus robot to the legged dual-arm *Atlas* robot (VI-B); and analyze the relative contributions of CAD posture constraints, CAD trajectory constraints and TSRs towards task success (VI-C).

Manipulation Tasks: The four tasks are illustrated in Fig.1. Tasks 1 is a single-arm pick and place task where the goal is to pick up the cylinder and drop it inside the container. Tasks 2 is a dual-arm task that requires grasping the handle of the box with the right hand to secure the box, and then extracting the cylinder with the left hand. The goal of Tasks 3 is to open the door using the left hand and then push a button in the interior of the box with the right hand. Tasks 4 consists of grasping and lifting up a tray with two hands while keeping the tray horizontal. Task 4 is distinguished from the other tasks in that it requires simultaneous constrained motion of both end effectors.

A. *Performance*

Fig. 4 illustrates the *Optimus* robot performing the four test tasks. Objects were tracked using April Tags and an on-board camera (Multisense SL), and were rendered on the interface as virtual objects. The knowledge base was learned using seven (7) demonstrations of reaching and grasping

TABLE I: Number of successful trials of planning and execution in 10 trials of each task using the *Optimus* Robot

Task ID	Autonomous Behavior	Shared Autonomy # successful trials (# of interventions)
1	9	10 (1)
2	8	10 (3)
3	8	10 (2)
4	10	10 (0)

motions per object per mode with a catalog of constraints that included TSRs, CAD posture constraints and CAD trajectory constraints. Each multi-step task was learned from a single demonstration of the task. Table I presents the number of successful executions of the four tasks for two cases: one where the tasks were performed autonomously by executing the sequence of learned keyframes, and a second where the tasks were executed using the shared autonomy method as described in Sec.III-D. For each trial, the objects were located in a different initial position such that the task is feasible for the kinematics of the robot. Otherwise the task would require mobility, which is out of the scope of this paper. Images of autonomous task execution are shown in Fig. 4, including the motion plan for each keyframe and selected images of the real robot executing the task.

Autonomous execution produced a success rate of 87.5% on average across ten trials of each of the four tasks, while the shared autonomy method resulted in an overall success rate of 100%. C-LEARN produced the correct set of queries (keyframe information) for the planner in each trial. The success of the execution depends on the ability of the particular robot's controller to follow the motion plan. *Optimus'* position control accuracy varies in different regions of the workspace, causing the execution to fail in a number of trials. These cases can be handled using shared autonomy, where small adjustment using teleoperation can compensate for this limitation. The results are summarized in Table I.

B. *Demonstration of Transfer Across Robots*

Next we demonstrate the ability of C-LEARN to transfer tasks learned using one robot to another robot with different kinematics. This is achieved by invoking the planner in Eq.1 using the kinematics information of the target robot and using the task representation (i.e sequence of keyframes and set of constraints) learned for the source robot. Note that Eq.1 naturally allows one to incorporate the set of constraints previously learned with any new set of constraints required for motion planning with the target robot. For example, a legged robot requires additional constraints to specify that the center of mass must remain within the support polygon. We test this procedure by transferring the tasks learned using the stable *Optimus* robot to a balancing *Atlas* robot and performing the tasks in kinematic simulation.

Motion plans shown for Atlas are guaranteed to be stable and executable on the real hardware [6]. In simulation, objects were located in different initial positions and *Atlas* was able to perform the four test tasks in each trial out of 10 scenarios for each task. Fig. 5 shows snapshots of *Atlas* planning for Tasks 2 and 4.

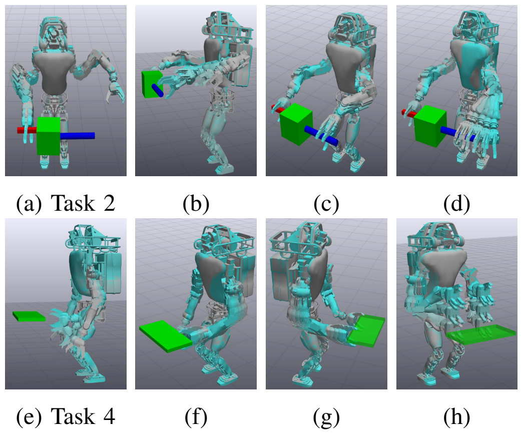

Fig. 5: *Atlas* planning in simulation for trasfered tasks. (a-d) Task 2; (e-h) Task 4.

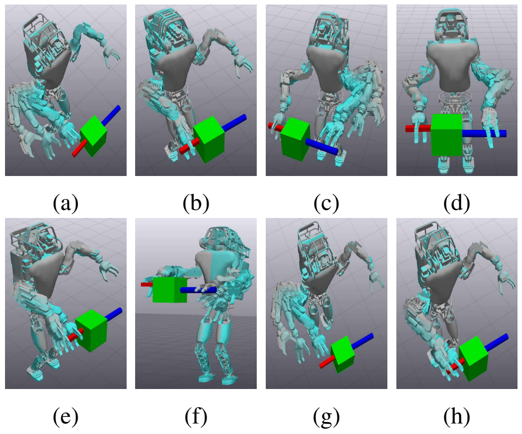

Fig. 6: Planning with different constraint types. (a,b,c,d) Planning without CAD posture constraints; (e,f) Planning without CAD trajectory constraints; (g,h) Planning using only TSR keyframes.

C. *Contribution of Different Constraint Types*

This section analyzes the utility of different constraint types (CAD posture constraint, CAD trajectory constraint, TSR) in the scenario where the *Atlas* robot performs a task using the knowledge base and task learned for *Optimus*.

The results for planning when all constraint types were available to the learner are presented in Fig.5. When performing the same test, but removing the CAD posture constraints from the knowledge base, we obtain the plans shown in Fig. 6 (a,b,c,d). Note that the orientation of the hands with respect to the objects in the reaching and grasping phases exhibits larger angular deviation with respect to the perpendicular to the cylinder principal axis when compared to Fig. 5. This variance is permitted by the TSR constraints, but results in the production of keyframes that do not lock into strict perpendicularity with respect to the objects.

The reaching phase and cylinder extraction show the utility of the CAD trajectory constraint for in-line motion. When CAD posture constraints are available to the learner but CAD trajectory constraints are not, the planner produces trajectories that fail to accomplish the task, as illustrated in Fig. 6 (e,f). Similarly, if only TSR keyframes are available, the planner fails to accomplish the task, as illustrated in Fig. 6 (g,h).

VII. Conclusions

This paper presents C-LEARN, a method of learning from demonstrations that supports the use of hard geometric constraints for planning multi-step functional manipulation tasks with multiple end effectors in quasi-static settings. By combining machine learning and motion planning techniques, this approach allows non-experts to teach robots a new class of manipulation tasks not covered by previous approaches. Specifically, this paper explores how to learn explicit geometric constraints that can be listed as the set of constraints in an optimization-based motion planning problem. In addition to SE(3) volumetric constraints (TSRs), we introduce the use of parametric models of CAD constraints, which enables the descriptions of parallel or perpendicular axes and moving in a line. We evaluated this method through execution of four multi-step tasks with multiple end-effectors using a shared autonomy framework using the *Optimus* robot. We demonstrated the capability of transferring the learned task to an *Atlas* robot with different kinematics without providing new demonstrations. Accompanying this paper, we present a video of the execution of the tasks, available online at `https://goo.gl/fAVm3z`.

Our work is limited in certain aspects, which represent potential avenues for future research. Functional tasks often involve articulated constraints, such as opening a door, but our method does not explicitly reason about articulation. The motion planner we used in this implementation does not handle collision avoidance. Exploring other optimization- and sampling-based planners is the next step in this line of research. Finally, we will conduct a user study to further evaluate the performance of C-LEARN with end users in shared autonomy mode.

References

[1] R. R. Murphy, "Human-robot interaction in rescue robotics," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 138–153, 2004.

[2] L. Conway, R. A. Volz, and M. W. Walker, "Teleautonomous systems: Projecting and coordinating intelligent action at a distance,"

[3] T. B. Sheridan, *Telerobotics, Automation, and Human Supervisory Control*. Cambridge, MA, USA: MIT Press, 1992.

[4] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. P. D'Arpino, R. Deits, M. DiCicco, D. Fourie, T. Koolen, P. Marion, M. Posa, A. Valenzuela, K.-T. Yu, J. Shah, K. Iagnemma, R. Tedrake, and S. Teller, "An architecture for online affordance-based perception and whole-body planning," *Journal of Field Robotics*, vol. 32, no. 2.

[5] M. DeDonato, V. Dimitrov, R. Du, R. Giovacchini, K. Knoedler, X. Long, F. Polido, M. A. Gennert, T. Padr, S. Feng, H. Moriguchi, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Human-in-the-loop control of a humanoid robot for disaster response: A report from the darpa robotics challenge trials," *Journal of Field Robotics*, vol. 32, no. 2, pp. 275–292, 2015.

[6] P. Marion, M. Fallon, R. Deits, A. Valenzuela, C. Pérez D'Arpino, G. Izatt, L. Manuelli, M. Antone, H. Dai, T. Koolen, J. Carter, S. Kuindersma, and R. Tedrake, "Director: A user interface designed for robot operation with shared autonomy," *Journal of Field Robotics*, vol. 34, no. 2, pp. 262–280, 2017.

[7] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, J. Carff, W. Rifenburgh, P. Kaveti, W. Straatman, J. Smith, M. Griffioen, B. Layton, T. de Boer, T. Koolen, P. Neuhaus, and J. Pratt, "Team IHMC's Lessons Learned from the DARPA Robotics Challenge Trials," *Journal of Field Robotics*, vol. 32, no. 2, pp. 192–208, 2015.

[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469 – 483, 2009.

[9] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization.," in *RSS 2013*, 2013.

[10] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 367–372, IEEE, 2008.

[11] L. Pais, K. Umezawa, Y. Nakamura, and A. Billard, "Learning robot skills through motion segmentation and constraints extraction," in *HRI Workshop on Collaborative Manipulation*, 2013.

[12] C. Bowen, G. Ye, and R. Alterovitz, "Asymptotically optimal motion planning for learned tasks using time-dependent cost maps," *IEEE Transactions on Automation Science and Engineering*, vol. 12, pp. 171–182, Jan 2015.

[13] C. Perez-D'Arpino and J. A. Shah, "Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification," in *IEEE ICRA 2015*, pp. 6175–6182, 2015.

[14] A. Vakanski, I. Mantegh, A. Irish, and F. Janabi-Sharifi, "Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1039–1052, 2012.

[15] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *2014 IEEE ICRA*, pp. 3339–3344, 2014.

[16] J. Silvrio, L. Rozo, S. Calinon, and D. G. Caldwell, "Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems," in *IEEE IROS 2015*, pp. 464–470, 2015.

[17] D. Berenson, S. S. Srinivasa, and J. Kuffner, "Task Space Regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, p. 14351460, 2011.

[18] C. Li and D. Berenson, "Learning object orientation constraints and guiding constraints for narrow passages from one demonstration," *ISER 2016*, 2016.

[19] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

[20] A. Kurenkov, B. Akgun, and A. L. Thomaz, "An evaluation of GUI and kinesthetic teaching methods for constrained-keyframe skills," in *IEEE IROS 2015*, pp. 3608–3613, 2015.

[21] J. J. Gibson, "The theory of affordances," in *Perceiving, Acting, and Knowing* (R. Shaw and J. Bransford, eds.), John Wiley and Sons, 1977.

[22] Y. Wu and Y. Demiris, "Towards one shot learning by imitation for humanoid robots," in *IEEE ICRA 2010*, pp. 2889–2894, IEEE, 2010.

[23] C. Groth and D. Henrich, "One-shot robot programming by demonstration by adapting motion segments," in *IEEE ROBIO 2014*, pp. 1068–1075, Dec 2014.

[24] S. Alexandrova, M. Cakmak, K. Hsiao, and L. Takayama, "Robot programming by demonstration with interactive action visualizations," *RSS 2014*, 2014.

[25] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 594–611, April 2006.

[26] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *IEEE IROS 2012*, pp. 5239–5246, Oct 2012.

[27] M. Phillips, V. Hwang, S. Chitta, and M. Likhachev, "Learning to plan for constrained manipulation from demonstrations," *Autonomous Robots*, vol. 40, no. 1, pp. 109–124, 2016.

[28] S. Pillai, M. R. Walter, and S. Teller, "Learning articulated motions from visual demonstrations," in *RSS 2014*, 2014.

[29] N. Somani, A. Gaschler, M. Rickert, A. Perzylo, and A. Knoll, "Constraint-based task programming with CAD semantics: From intuitive specification to real-time control," in *IEEE IROS 2015*, pp. 2854–2859, 2015.

[30] R. Tedrake, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2014. `http://drake.mit.edu`.

[31] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM journal on optimization*, vol. 12, no. 4, pp. 979–1006, 2002.