Learning Object Orientation Constraints and Guiding Constraints for Narrow Passages from One Demonstration

Changshuo Li[1] and Dmitry Berenson[2]
[1]Worcester Polytechnic Institute, [2]University of Michigan

Abstract. Narrow passages and orientation constraints are very common in manipulation tasks and sampling-based planning methods can be quite time-consuming in such scenarios. We propose a method that can learn object orientation constraints and guiding constraints, represented as Task Space Regions, from a single human demonstrations by analyzing the geometry around the demonstrated trajectory. The key idea of our method is to explore the area around the demonstration trajectory through sampling in task space, and to learn constraints by segmenting and analyzing the feasible samples. Our method is tested on a tire-changing scenario which includes four sub-tasks and on a cup-retrieving task. Our results show that our method can produce plans for all these tasks in less than 3 minutes with 50/50 successful trials for all tasks, while baseline methods only succeed 1 out of 50 times in 30 minutes for one of the tasks. The results also show that our method can perform similar tasks with additional obstacles, transfer to similar tasks with different start and/or goal poses, and be used for real-world tasks with a PR2 robot.

Keywords: Learning from Demonstration, Constraints Learning, Manipulation Planning

1 Introduction

Many manipulation tasks, such as changing the tire on a car, require several operations where the robot must navigate an object through a narrow passage (e.g. removing a nut from a stud or removing the tire from the hub). These narrow passages are induced by the geometries of the objects. There are also some manipulation tasks where pose constraints must be obeyed, such as not tilting a cup of water. These requirements further constrain the motion of the object. While motion planning algorithms capable of performing such tasks exist, they cannot easily be biased to search the relevant parts of the space when the constrains come from narrow passages or require manual input of pose constraints. These methods are either time-consuming or require significant domain knowledge on the part of the user.

In this paper we propose to learn the relevant area of the C-space to search from human demonstration of the task. Our framework is able to do this, not by attempting to follow the demonstration, but instead by inferring constraints from it; exploring regions around it by sampling, analyzing the geometric properties of the samples, and then extracting a relevant feasible area represented as a set of Task Space Regions (TSRs) [4]. The TSRs derived from this analysis allow

This work was supported in part by the ONR grant N00014-13-1-0735.

us to generate plans for similar tasks much faster than planning without such constraints in the case of narrow passages and allow performing the task without violating pose constraints on the object.

Our key contribution is that instead of requiring multiple demonstrations to form statistical models of how the object should move [2, 15] we learn pose constraints (specifically constraints on the object's orientation necessary for the task) and guiding constraints (i.e. constraints that limit our search space to only the relevant parts of the C-space) from a *single* demonstration. The results show that the constraints we learn allow planning for tire-changing and cup-retrieval tasks which outperforms other methods in terms of computation time and success rate. These constraints can also transfer to similar tasks, and can be used for real-world tasks with a PR2 robot.

The remainder of this paper is arranged as follows: Section 2 reviews related work in this area. We describe the problem we address in Section 3. Our method is then explained in detail in Section 4. Results of experiments in simulation and real world mock-ups are presented in Section 5. Finally, we discuss some drawbacks of our method, and conclude the paper in Section 6.

2 Related Work

Previous methods for learning constraints from demonstration can be divided into two classes according to the type of input: 1) Kinesthetic demonstrations, such as in [1, 11, 15]. The advantage of this kind of input is that the task is demonstrated in the robot's C-space directly, so there is no retargeting problem; but this kind of input requires the demonstrator to have a good understanding of the robot's kinematics, especially in high-DOF cases, otherwise the demonstration can be noisy and redundant. 2) Natural human motion, such as in [3, 9] with some also gathering verbal comments, e.g. [10]. In this case, we only need the demonstrator to act naturally, but retargeting the demonstrated motion is a challenge.

Most methods that learn constraints from demonstration require multiple demonstrations, which are often used to compute the variance along the trajectory. Some additional pre-processing, such as data alignment [1, 15], is also needed. Given the aligned data, represent the multiple demonstration trajectories [5] or key frames [1] as Gaussian Mixture Models (GMMs), and then a solution for the task is given by Gaussian Mixture Regression (GMR). The drawback of these methods is that they do not generalize to new environments where the task is similar but new obstacles are present and/or the start/goal are moved. To overcome this limitation, [15] learns a cost function from multiple demonstrations, and uses a sampling-based planner to find a feasible path with low cost.

Others have explored using a linear-chain Conditional Random Field paired with motion-based features to detect and extract the rigid constraints which arise between pairs of objects [3, 9]. After that they use an interactive GUI to refine the learned constraints. In our approach we wish to have the user provide the minimum information possible, so we limit our input to the demonstration alone.

What distinguishes our work from those above is 1) We learn from only a single demonstration; 2) Our method does not require any input beyond the demonstration; 3) Our method does not require transferring a human motion to the robot (i.e. solving the retargeting problem); 4) The constraints we learn can transfer to similar tasks (i.e. a different start/goal pose or additional obstacles; and 5) Our method scales to tasks in $SE(3)$, such as changing a tire.

3 Problem Statement

A task is defined by a moving object, a reference object, a start pose and a goal pose of the moving object w.r.t. the reference object. The information we extract from the human demonstration is a trajectory of the moving object in the task space, which is $SE(3)$. This trajectory should be feasible and should connect the start pose and the goal pose of the task. So the input of our algorithm is a trajectory of poses of the moving object, the id of the moving object, the id of the reference object and the geometric model of the demonstration environment. The output of our algorithm is a pose constraints represented as Task Space Region (TSR)[1] [4] in the world frame and a series of guiding constraints represented as TSRs in the frame of the reference object.

The goal of our algorithm is to learn pose constraints and guiding constraints from the input, which will allow us to achieve fast planning across similar tasks. A similar task is defined as a feasible task which has the same moving object and reference object as the demonstrated task and either the start transform or the goal transform of the moving object w.r.t. the reference object is the same as the demonstrated task. The environment may or may not have a different arrangement of obstacles.

4 Technical Approach

Our algorithm is described in Fig. 1. First, we capture the demonstration motion using a motion capture system. From the demonstrated series of poses, we learn the pose constraints, which represent the range of orientations the moving object can have in this task. We then seek to learn the guiding constraints. As part of this process we calculate the ratio of feasible object poses samples vs. total samples around the demonstrated trajectory by rejection-sampling poses around every demonstrated pose of the moving object. The demonstration trajectory is then segmented based on this ratio. After segmentation, a guiding constraint is learned from the samples of each segment. These constraints are represent by TSRs. Finally the learned guiding constraints (TSRs) are input into the sampling-based planner CBiRRT [4] to generate a path for a robot to perform the task. We describe each step in detail below.

4.1 Learning Pose Constraints from Demonstrated Poses

Unlike geometric constraints, which are induced by the geometries of the objects in the environment, pose constraints are often induced by additional requirements of the task, such as not tilting a cup of water. In many of these cases, the

[1] A TSR, defined by a reference transform, an offset transform, and a matrix of bounds on each dimenions of $SE(3)$ represents a volume in $SE(3)$.

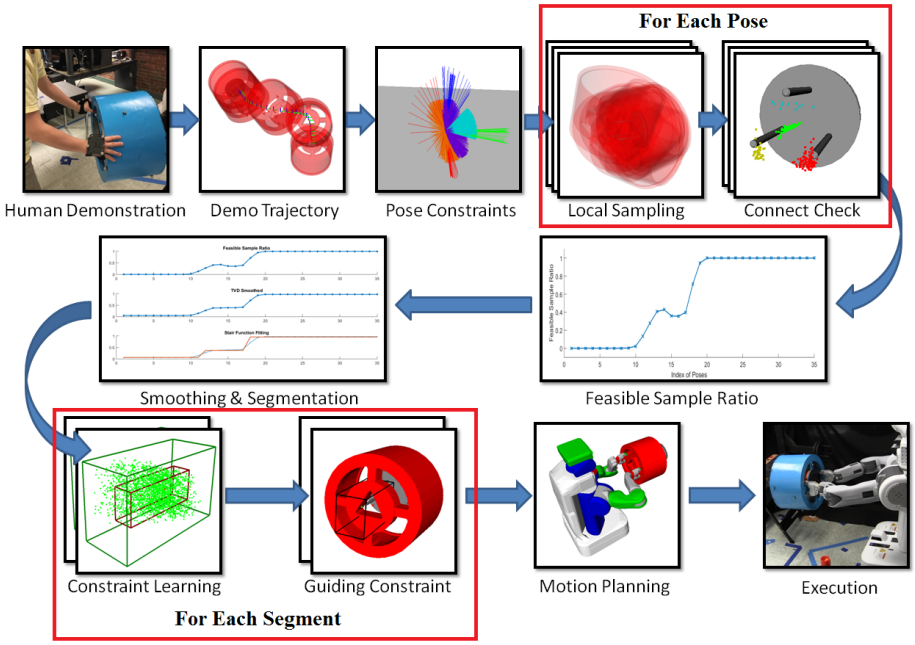

Fig. 1. A diagram of our constraint learning algorithm

pose constraints are only related to the orientation of the moving object, such as keeping the cup upright regardless of its position. Thus our pose constraints are the allowable range of orientations of the moving object.

We choose the Euler angles (Roll, Pitch and Yaw) as the parametrization of the orientations. Similar to Principle Component Analysis (PCA), we wish to find a reference frame in which the principal components or components with the largest weight are aligned with the axes. But the Euler angle space is not a linear space, so instead we try to find a reference frame in which the volume of the Axis-Aligned Bounding Box (AABB) of the demonstrated orientations is the smallest. A Random Volume Decent method is used to achieve this:

First, we start with an arbitrary reference frame, and calculate the volume of the AABB in this frame. Then, we apply a small random rotation to the reference frame and get the volume of the AABB in this new frame. If the volume decreases, we start from this new frame and try to find a better one; otherwise we go back to the old reference frame, generate a new small random rotation, and try again. The process terminates when we cannot find a better reference frame after T consecutive trials.

In the resulting reference frame, we calculate the range of the demonstrated orientations in all three dimensions (Roll, Pitch and Yaw). Then we say a dimension is unconstrained if the range of this dimension is greater than an angle α; otherwise this dimension is constrained and the orientation of the moving object should not go out of the range of this dimension. These rotation constraints are then input into a TSR which has unbounded position constraints to produce the pose constraint for the task.

4.2 Exploring Task Space Near the Demonstration Trajectory

We sample poses of the moving object around each of the poses in the demonstration trajectory in order to explore the feasibility of the local task space so that we can compute appropriate guiding constraints. The sampling can be considered as a random rotation and translation of the moving object in the demonstrated pose frame. The random translation is uniformly chosen from a cube centered at the origin of $\mathbb{R}^3$, and the size of the cube is decided by the size of the moving object. Here we use unit quaternions as the parametrization of the rotations. A random rotation is generated by a uniformly random spin around a uniformly random axis. We intentionally use this random axis-angle method instead of the more common uniformly-random quaternion method [13] because the axis-angle method's distribution is denser around the demonstrated pose than the the uniformly-random quaternion. This is useful in our application because the nearer a pose to the demonstrated pose, the higher chance it is feasible and informative. We then discard a pose sampled in this way if it is outside the learned pose constraint described in Section 4.1.

Next, we check collision for each sampled pose to evaluate feasibility. Both the sampled pose and its feasibility are recorded for later use. The sampling terminates when either the number of feasible samples or the number of total samples reaches a predetermined threshold.

After sampling, we need to examine the connectivity of the feasible samples (not all feasible poses are reachable from the demonstrated pose due to obstacles). This is illustrated in Fig. 2. The connectivity is examined by a local planner, which tries to connect samples by a straight line in $SE(3)$. This connectivity check can divide the feasible samples into several classes. Due to the inconsistency between the real world and the simulation environment, the demonstrated poses may not always be feasible. Thus if the demonstrated pose is feasible, then the set of samples that can be connected to the demonstrated pose by the local planner is chosen to be the feasible connected class; otherwise, the class nearest to the demonstrated pose is chosen.

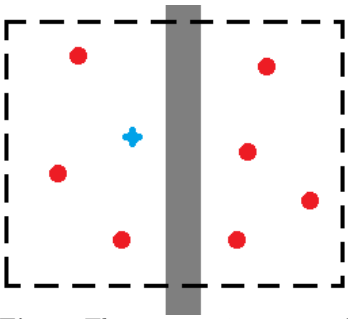

Fig. 2. The grey region is an obstacle, the blue cross is the demonstrated pose, the red circles are the feasible sampled poses, the black dashed lines show the sampling range. In this case, the samples are divided into two class, the left and the right. Only the left class can be reached from the demonstrated pose, so samples in the right class are also treated as infeasible.

Finally we calculate the *feasible sample ratio*, the ratio of the number of samples of the feasible connected class vs. the total number of samples. We compute this sample ratio for every demonstrated pose, thus obtaining a series of ratios for the entire trajectory.

4.3 Trajectory Segmentation

We wish to segment the trajectory using the feasible sample ratio because different regions of a task have significantly different constraints, and they should be

represented by different TSRs. For example, when removing a nut from a stud, the nut is highly constrained when it is on the stud, and is free when it is off the stud. The key to segmenting the trajectory into regions where different guiding constraints are active is to identify points where there are significant changes in the feasible sample ratio. To do this, we represent the feasible sample ratio as a time-varying signal across the duration of the demonstrated trajectory. We then smooth this signal using Total Variation Denoising (TDV) [12]. TDV has the advantage that it smooths noise in relatively flat stages while not shifting step edges [14]. Each step change in the smoothed signal implies a significant change of the feasible sample ratio and represents the start of a new segment.

We then need to fit a series of step functions to the smoothed signal (i.e. a "staircase" function). Each flat region of the staircase is a segment. Given a number of stairs k, the fitting problem can be formulated as [8]:

$$Y_n = X_n\beta^n + \epsilon_n = X_n N_k \beta^k + \epsilon_n \ , \tag{1}$$

where n is the length of the signal, Y_n is a n-by-1 vector of the feasible sample ratios, X_n is a n-by-n lower triangular matrix with non-zero elements equal to one, ϵ_n is a n-by-1 vector of the residual error, and β^n is a n-by-1 vector having all its components equal to zero except those k numbers corresponding to the start of a new stair. We then rewrite β^n as $N_k\beta^k$, where β^k is a k-by-1 vector of all k non-zero elements, and N_k is a n-by-k matrix, each column of which is one of the trivial orthonormal basis of $\mathbb{R}^n$. Then the optimal k-stair function is the one has the minimum residual squared error over all possible N_k and β^k:

$$\hat{N_k} = \underset{N_k}{\operatorname{argmin}}\{\min_{\beta^k}\{(Y_n - X_n N_k \beta^k)^T (Y_n - X_n N_k \beta^k)\}\} \ . \tag{2}$$

For a given N_k, the inner optimization is a trivial linear fitting problem. A straightforward way to solve the above optimization problem is to run over all possible N_k, which means to try all possible k-combinations of the trivial orthonormal basis of $\mathbb{R}^n$. Note that there is always a stair at the first point, so $[1, 0, \cdots, 0]^T$ should always be included in the k-combinations. We set an upper limit on k to test, which is defined as K_{max}.

We then need to determine the optimal k, which we call k^*. Inspired by [7], we assume that as k grows larger, the improvement of total residual square error should be large if $k < k^*$; while the improvement should be small if $k > k^*$. Thus the optimal k^* is found by successively fitting with larger k values until a large drop in improvement is observed. To avoid over-fitting, we stop testing larger ks when the residual squared error is less than 1% of the total squared error. The residual square error at k is:

$$\sigma_k^2 = \min_{\beta^k}\{(Y_n - X_n \hat{N_k} \beta^k)^T (Y_n - X_n \hat{N_k} \beta^k)\} \ . \tag{3}$$

Then we define the improvement at k as:

$$I_k = \sigma_{k-1}^2 / \sigma_k^2 \ . \tag{4}$$

And the optimal k^* satisfies:

$$k^* = \min\{k | I_{k+1} < K_{max}\} \ . \tag{5}$$

Then the segmentation results can be extracted from the corresponding $\hat{N}_{k^*}$.

4.4 Constraint Learning

For each segment determined above we learn a guiding constraint from all the samples of this segment. As mentioned before, guiding constraints are represented as TSRs, so our goal is to find the limits of each dimension in task space. In order to make sure the TSR includes at least one solution, the demonstrated poses have to be included (or if a demonstrated pose is in collision, the nearest sample of the feasible connected class has to be included in the TSR). We do not want to set the bounds of the TSR to be too large in a narrow passage, as we would loose the power of the TSR to narrow our search space. On the other hand, the TSR should not be too small, as it will not generalize well when new obstacles are introduced or objects are moved (as this may require taking a somewhat different path than was demonstrated).

If the feasible sample ratio of the entire segment is close to 1, i.e. greater than $1 - \epsilon$, which means the moving object is almost free around this segment, then we say there is no constraint on this segment, or the guiding TSR is unbounded. Otherwise, we first calculate a *core TSR* which is the smallest axis-aligned box in $SE(3)$ that includes only the demonstrated poses (or nearest samples) of that segment. Then we put all the samples of this segment together, and calculate a TSR that tightly includes all the feasible connected class samples of this segment by setting the TSR bounds to be the bounds of this set. We call this the *feasible TSR*. We then iteratively delete the feasible connected class sample which is the farthest from the *core TSR* (the distance between a pose and a TSR is defined in [4]), and then recalculate the *feasible TSR*. Thus the *feasible TSR* iteratively shrinks. During the shrinking process, we keep tracking the ratio of feasible connected class samples over all samples inside the *feasible TSR*. The process terminates when this ratio exceeds 0.5, or the feasible TSR has become the same as the core TSR. The resulting *feasible TSR* is the guiding constraint for this segment.

The output of the above algorithm depends on the reference frame we choose. Since $SE(3)$ is not linear, we cannot use methods like PCA to choose the frame. Again, we use a Random Volume Descent method instead. First, given an arbitrary reference frame, we can calculate the volume of the *core TSR*. Then, we apply a small random rotation on the reference frame, and calculate the volume of the new *core TSR*. If the volume decreases, we set this new frame as the reference frame. We repeat this process until no improvement is made in T consecutive iterations.

5 Results

Our method is tested in a retrieve-cup-from-shelf task and in a tire-changing scenario, which includes four important sub-tasks: remove nut from stud, insert nut onto stud, unhang tire from hub and hang tire on hub, as shown in Fig. 3. All of these tasks require traversing at least one narrow passage. In the experiment, we simplify the tasks by removing the threads from the nut and stud. First, some intermediate results of the unhang tire tasks are presented. Then the results of our method on the five tasks are compared with those of baseline methods. After that the generalization ability of our method is shown. Finally we present

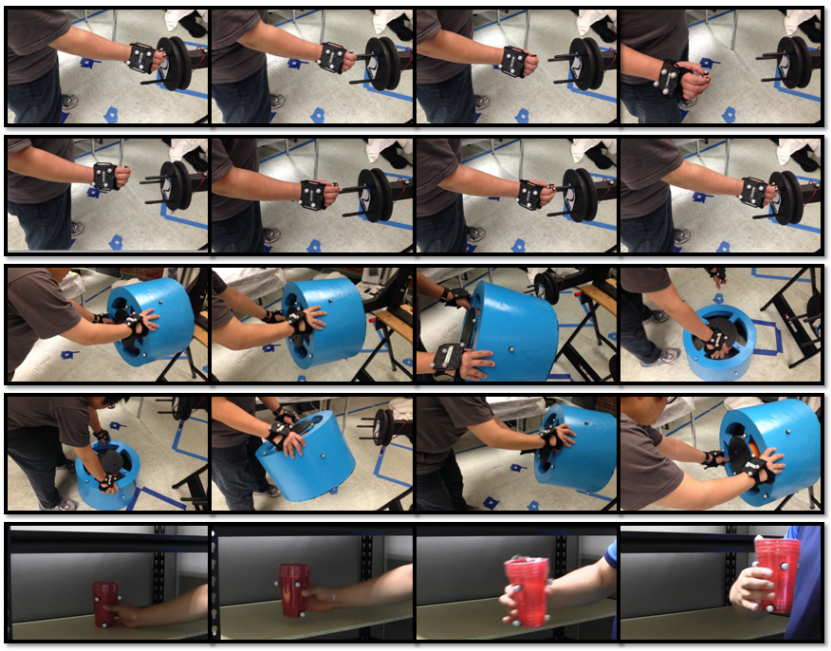

Fig. 3. Snapshots from humans demonstrating the five tasks. We only use the poses of the moving object and the reference object in our method. From top to bottom are remove nut, insert nut, unhang tire, hang tire and retrieve cup, respectively.

snapshots from the execution of plans generated by this approach on a physical PR2 robot. We used the following parameter values: $K_{max} = 5$, $T = 500$, $\alpha = \pi/4$, $\epsilon = 0.05$.

5.1 Intermediate Results of Unhang Tire Task

In this section, we show some intermediate results of the unhang tire task to illustrate how our method works. Fig. 4(a) shows the feasible samples at one demonstrated pose. After the connection check, the feasible samples are divided into 4 classes, shown in different colors. Fig. 4(b-e) shows a sample of each class. We can see that due to the complex geometry of the tire, there can be several classes around the demonstrated pose, and many feasible samples around cannot be connected to the demonstrated pose by the local planner.

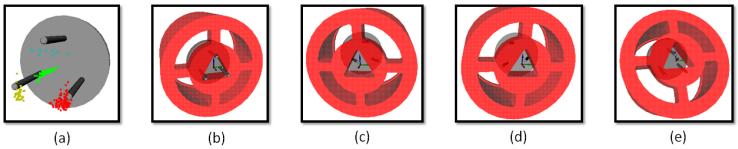

Fig. 4. Results of the connection check of feasible samples at one demonstrated pose. (a) 4 different connected classes shown in different colors, green is the feasible connected class; (b-e) samples from the green, yellow, red, and blue classes, respectively.

In Fig. 5(a), we can see that the feasible sample ratio curve clearly has a 3-step shape. This shows that, during the demonstration, the tire goes from a region with hardly any free space, to a region with some freedom, and finally

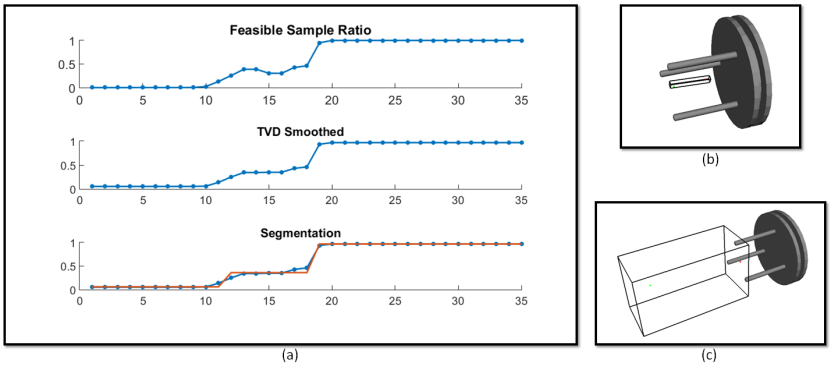

Fig. 5. (a) Results of smoothing and staircase fitting. The horizontal axis is the index of the demonstrated trajectory. Top: feasible sample ratio; Middle: TVD smoothed; Bottom: the optimal staircase function in red. (b) Learned TSR of segment 1 and (c) Learned TSR of segment 2. Only the position boundaries are shown.

to a region with almost no constraints. After smoothing and staircase fitting, the demonstration trajectory is divided into 3 segments. Fig. 5(b-c) shows the position boundaries of the learned guiding constraints of segment 1 and 2. Since the feasible sample ratio of segment 3 is nearly 1 it has no translation constraints.

5.2 Comparison to Planning with Learned Constraints

We compare three planning approaches to evaluate the contribution of our method. All approaches use the CBiRRT sampling-based motion planner [4] with different inputs or sampling strategies: 1) CBiRRT using ordinary sampling and no guiding constraints, 2) CBiRRT using bridge sampling [6] (a method that samples narrow passages); 3) CBiRRT using ordinary sampling and guiding constraints learned from a single demonstration; and 4) CBiRRT using ordinary sampling and both guiding and pose constraints learned from a single demonstration. Tests are performed in the OpenRAVE virtual environment but demonstrations are gathered from live human demonstrations using motion capture. 50 trials are run for each planner on each primitive task. A trial that cannot find a solution within 3 minutes is considered a failure.

Table 1 shows the success rate of the four methods for the tested tasks. The first two variants of CBiRRT were not able to solve any of tasks in any of the trials. Table. 2 shows the averages and standard deviations of the planning time of successful trials using each planner (recall that there were no successful trials for the first two planners). The results show that only CBiRRT with guiding constraints and CBiRRT with both guiding and pose constraints can solve any

Table 1. Success Rate of the Planners

Planner	Unhang Tire	Hang Tire	Remove Nut	Insert Nut	Retrieve Cup
CBiRRT	0/50	0/50	0/50	0/50	N/A
CBiRRT with bridge sampling	0/50	0/50	0/50	0/50	N/A
CBiRRT using guiding constraints	41/50	43/50	50/50	50/50	N/A
CBiRRT using guiding & pose constraints	50/50	50/50	50/50	50/50	50/50

Table 2. Planning Time of the Planners (avg.±std.(sec))

Planner	Unhang Tire	Hang Tire	Remove Nut	Insert Nut	Retrieve Cup
CBiRRT	-	-	-	-	N/A
CBiRRT with bridge sampling	-	-	-	-	N/A
CBiRRT using guiding constraints	46.4 ± 38.0	84.9 ± 51.7	2.16 ± 1.15	2.52 ± 1.31	N/A
CBiRRT using guiding & pose constraints	18.4 ± 9.61	34.6 ± 27.7	0.345 ± 0.233	0.748 ± 0.263	2.46 ± 1.67

of the tasks in under 3 minutes. For the cup-retrieval task, since we should always keep the cup upright and only the planner with pose constraints can guarantee that, all other planners are not applicable for this task.

Comparing the results of the last two rows, we can see that CBiRRT with both guiding and pose constraints has better performance. This is because pose constraints consider how the moving object is manipulated by the demonstrator, and rule out many poses that the robot's end effector can not reach, even though they are collision-free. So it is easier for CBiRRT to find a solution.

We also ran the first two planners on each tire-changing sub-task with a 30-minute time limit, 10 trials for each setup. Out of all the tasks and trials, only one trial of the remove-nut task using planner (1) succeeds with a planning time of 685.01s.

5.3 Performance on Similar Tasks

To show the generalization capabilities of our method we first test our method on a similar remove nut task. This task has the same start pose as the demonstrated one, but has a different goal pose and an additional obstacle, as shown in Fig. 6(a-c). Our method can solve this task with a success rate of 50/50 and the planning time is 5.01 ± 2.07s. We then test our method on a similar insert nut task. This time both the start and the goal poses are different. The demonstration is to insert the nut on the top stud but in this test the robot is required to insert the nut on the lower right stud (see Fig. 6(d-f)). Our method can solve this task with a success rate of 50/50 and the planning time is 3.29 ± 2.31s. We also test on a similar retrieve cup task. Both the start and the goal poses are different, and there is an additional obstacle. As shown in Fig. 6(g-j). Our method can solve this task with a success rate of 50/50 and the planning time is 4.20 ± 1.80s.

5.4 Real Robot Experiment

Using the pose and guiding constraints we learn from the demonstrations, we are able to plan motions for PR2 robot to achieve those five tasks in real world mock-ups. Snapshots of the execution of the most constrained parts of those tasks are shown in Fig. 7. For more information of our real robot experiment, please see the video accompanying the paper.

6 Discussion and Conclusion

A disadvantage of our method is that it cannot guarantee completeness. Two reasons account for the loss of completeness: First, we only use a single demonstration, so we may easily detect pose constraints in cases that do not have such

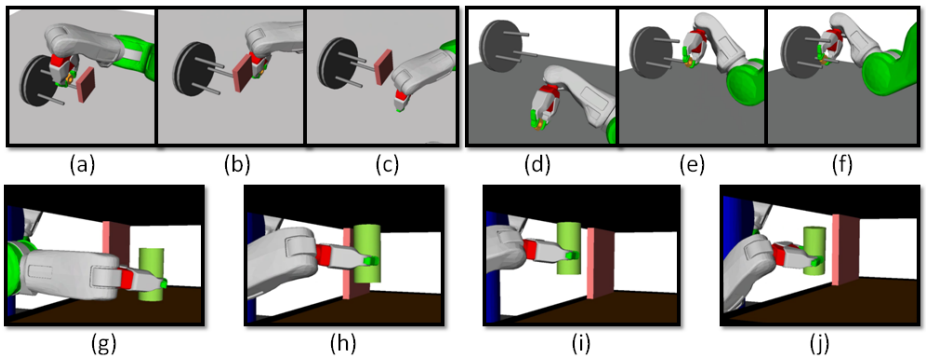

Fig. 6. PR2 performing similar tasks to those demonstrated. (a-c) robot removing nut while avoiding an obstacle; (d-f) robot inserting nut on the lower right stud; (g-j) robot retrieving cup while avoiding an obstacle.

constraints, for example, we learned pose constraints in the unhang tire task. While applying this pose constraint reduces the planning time by 59%, we may loose some generalization ability to similar tasks where extremely different rotation of the tire are necessary. Second, we only explore a limited region around the demonstration trajectory, so solutions that outside of this region are not considered. However, a conventional sampling-based planner can be run in parallel to the planner that uses the learned constraints to maintain probabilistic completeness.

This paper proposed an algorithm that can learn pose and guiding constraints from a single demonstration. The key idea of our algorithm is to explore the area around the demonstration trajectory by sampling poses of the object, and to learn constraints by segmenting and analyzing the feasible samples. We tested our method in a cup-retrieval task and a tire-changing scenario, and verified that this method can allow us to achieve fast planning across similar tasks. We also showed that the trajectories planned by our approach can be used for real-world tasks with the PR2 robot.

References

1. Akgun, B., Cakmak, M., Jiang, K., Thomaz, A.L.: Keyframe-based learning from demonstration. International Journal of Social Robotics 4(4), 343–355 (2012)
2. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and autonomous systems 57(5), 469–483 (2009)
3. Baisero, A., Mollard, Y., Lopes, M., Toussaint, M., Lutkebohle, I.: Temporal segmentation of pair-wise interaction phases in sequential manipulation demonstrations. In: IROS (2015)
4. Berenson, D., Srinivasa, S.S., Kuffner, J.: Task space regions: A framework for pose-constrained manipulation planning. IJRR (2011)
5. Calinon, S., Guenter, F., Billard, A.: 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)
6. Hsu, D., Jiang, T., Reif, J., Sun, Z.: The bridge test for sampling narrow passages with probabilistic roadmap planners. In: ICRA (2003)

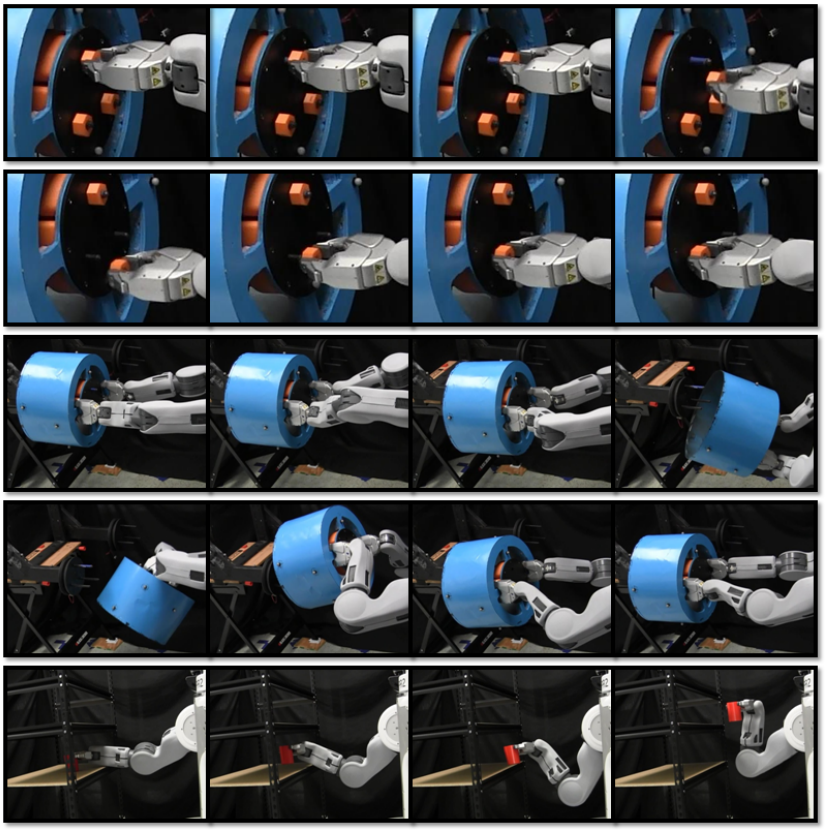

Fig. 7. Snapshots of PR2 executing the five tasks. From top to bottom are remove nut, insert nut, unhang tire, hang tire and retrieve cup, respectively.

7. Krzanowski, W.J., Lai, Y.: A criterion for determining the number of groups in a data set using sum-of-squares clustering. Biometrics pp. 23–34 (1988)
8. Levy-leduc, C., Harchaoui, Z.: Catching change-points with lasso. In: Advances in Neural Information Processing Systems. pp. 617–624 (2008)
9. Mollard, Y., Munzer, T., Baisero, A., Toussaint, M., Lopes, M.: Robot programming from demonstration, feedback and transfer. In: IROS (2015)
10. Pardowitz, M., Knoop, S., Dillmann, R., Zollner, R.D.: Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37(2), 322–332 (2007)
11. Phillips, M., Hwang, V., Chitta, S., Likhachev, M.: Learning to plan for constrained manipulation from demonstrations. Autonomous Robots 40(1), 109–124 (2016)
12. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena 60(1), 259–268 (1992)
13. Shoemake, K.: Uniform random rotations. In: Graphics Gems III. pp. 124–132. Academic Press Professional, Inc. (1992)
14. Strong, D., Chan, T.: Edge-preserving and scale-dependent properties of total variation regularization. Inverse problems 19(6), S165 (2003)
15. Ye, G., Alterovitz, R.: Demonstration-guided motion planning. In: ISRR (2011)