Bootstrapping Trajectory Transfer from Multiple Demonstrations with Applications to Deformable Object Manipulation

Ankush Gupta, Dylan Hadfield-Menell, Robbie Gleichman, and Pieter Abbeel

***Abstract*— Deformable object manipulation remains a challenging task in robotics. Continuous, high dimensional, state and action spaces make standard model-based approaches to manipulation planning intractable. Recent work has shown progress on this task by learning from demonstrations through *trajectory transfer* [21], [20]. This approach avoids planning in the state space of a deformable object by finding a spatial warping that can be used to apply an expert demonstration to a new scene.**

We propose a method for improving trajectory transfer that makes use of bootstrapping examples through simulation. Given a simulator and a way to detect success, we augment our trajectory library with example states and transferred trajectories that have succeeded in simulation. We apply this approach to a simulated overhand knot-tying task. The approach described in Schulman et al. [21] achieves a success rate of 59%. We demonstrate performance of up to 85%.

I. INTRODUCTION

A large challenge in applying standard manipulation and planning techniques to deformable object manipulation is that of tractable modeling. Deformable object are often characterized by high-dimensional, continuous state-action spaces. Model-based planning has yet to scale up to the task of efficient planning in this setting.

Recent work have gained traction on this problem through the technique of learning from demonstrations [21], [20]. These results are achieved through *trajectory transfer*, where a demonstration trajectory is generalized to fit to a new scenario. Trajectory transfer finds a non-rigid registration between an example scene and the current scene that trades off between goodness-of-fit and the curvature of the registration. This method of transfer is model-free and obviates the need to plan in complicated and intractable models of deformable objects. Trajectory transfer has demonstrated state-of-the-art performance for knot-tying and suturing.

An important aspect of these strategies is incorporation of multiple demonstrations. By increasing the number of demonstrations, it becomes possible to do more tasks. Additionally, demonstrations can take the form of steps in a task and can be ordered and combined to further increase the set of possible successful manipulations.

However, a key problem remains: how should we pick a trajectory to transfer from a library of example trajectories given an input scene? Incorrect selection may lead us to fail at a task which would otherwise be possible for the correct selection of trajectories. Furthermore, how can we use experience to improve trajectory transfer both at the level of selecting a trajectory to transfer and at the level of transferring a single trajectory.

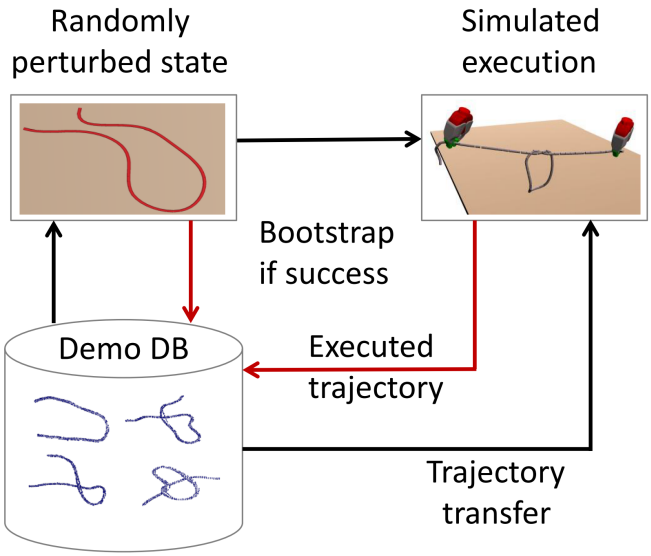


Fig. 1. Diagram of the bootstrapping approach we employ. Our system uses simulation of perturbed states from our demonstration set to augment a library of expert demonstrations. We find that this approach can leverage synthetic data to improve on the applicable of expert demonstrations by better modelling of states that trajectories can transfer to and allowing our approach to penalize less for transformations where transfer succeeds.

In this paper we present a method for improving the performance associated with a library of demonstrated trajectories through simulated attempts at trajectory transfer and feedback about the success of those transferred trajectory. Our approach applies bootstrapping to improve on standard trajectory transfer by by treating successful transfers as new demonstrations. This enables us improve performance without requiring new human supervision. Treating successful simulations as new demonstrations builds up a higher granularity representation of states where we expect transfer to succeed and enables better selection of a trajectory to generalize at test time. In addition, the feedback recieved can improve our ability to transfer a single trajectory by building an implicit model of non-rigid deformations such that transfer succeeds. We demonstrate the effectiveness of these improvements in knot-tying task and find improvements of up to 25% over the transfer method described in Schulman et al. [21].

Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA, USA. {gupta, dhm, rgleichman, pabbeel} @berkeley.edu

II. Related Work

Our approach to learning from demonstrations avoids building explicit representations and models of objects and states spaces. Thus, it is well suited to deformable object manipulations. It is challenging to manipulate deformable objects due to their nonlinearity and because the configuration spaces of such objects may be infinite-dimensional [9].

In previous work, Wada et al. model textile fabric and sponge blocks coarsely and then apply a control method that is robust to discrepancies between the coarse model and the object [23]. Howard et al. present a more general approach for grasping 3D deformable objects that does not assume prior knowledge of the object. They model particle motion of the object using nonlinear partial differential equations, and train a neural network for determining the minimum force required for manipulating the object [7].

We validate our approach and present results for knot tying. There is a rich literature of knot-specific approaches to this problem. For instance, in knot planning from observation (KPO), knot theory is used to recognize rope configurations and define movement primitives from visual observations of humans tying knots [13], [22]. Existing motion planning approaches for knot tying use topological representations of rope states (i.e. sequences of rope crossings and their properties) and define a model for transitioning between topological states [12], [18], [25]. Robust open loop execution of knot tying has also been explored [2]

The problem of learning from demonstrations (LfD) deals with the generalization of expert demonstrations to new scenarios [1], [19]. Behavioral cloning is an approach to LfD that directly learns a policy to mimic an expert's behavior.

One of the first successful applications of this strategy is the ALVINN system [15], which utilizes a neural network to learn a steering policy that enables an autonomous car to follow a road. [14] use a convolutional network to learn a steering policy for off-road driving. [16] uses multi-class classification to learn a function that scores actions to predict good foot steps for robot locomotion and good grasps for robot manipulation. [17] propose a method to directly control a Micro UAV from RGB camera input.

Miyamoto et al. describe an approach for learning to play Kendama [11] and hit a tennis ball [10] from demonstrated actions. Their method is successful at generalizing human trajectories and incorporates sequential information from multiple demonstrations. However, this approach requires hand tuning of waypoints and does not generalize to new scenes.

Isaac et al. [8] use behavioral cloning to learn to fly an airplane, by making use of an abstract, goal-directed, layer which sits on top of a low-level PID controller. This goal-directed learning is similar in spirit to ours, although it makes use of a different formalism and uses simpler low level controllers.

Calinon et al. learn a mixture of Gaussians to represent the joint trajectory of the robot and environment state across multiple demonstrations, and infer the trajectory for a new environment state by conditioning on that state [4], [3]. Their approach assumes access to a feature representation of the environment, so it cannot directly be applied to tasks in environments without fixed feature representations — such as our application of knot tying.

III. Trajectory Transfer with Thin Plate Splines

Trajectory transfer is an approach to learning from expert demonstrations [21]. The trajectory transfer algorithm is given a current scene, s_{test}, demonstration scene, s_{demo} and a demonstration trajectory, t_{demo}, as input. We assume that the scenes are made up of matched points in $\mathbb{R}^3$. The first step is to find a function, $f^* : \mathbb{R}^3 \to \mathbb{R}^3$, as the solution to the following optimization problem:

$$\min_f \sum_i ||s_{test}^{(i)} - f(s_{demo}^{(i)})||^2 + C \int dx ||D^2(f)||_{Frob}^2. \quad (1)$$

The minimizing f will be a Thin Plate Spline, and can be expressed as a linear combination of basis functions about the correspondence points [24]. C is a hyper-parameter that trades off between goodness of fit and the curvature of the function. The solution to this optimization can be computed as the solution to a linear system of equations.

Given a warping, f^*, between the demo and test scenes, we take each pose from the demo trajectory and pass it through f^*. Poses are transferred by mapping coordinate frames through the jacobian of f^*. The trajectory that results from this is used to guide a motion planner that finds a similar feasible trajectory. This trajectory is executed in the test scene. In the case where correspondences are not known initially, one can use TPS-RPM, an approach that jointly finds correspondences and a mapping between them by alternating between estimating correspondences and solving for a thin plate spline[5].

Schulman et al. [21] provide some intuition for scenarios where this approach is likely to succeed. They assume a cost function, L, on states and trajectories, a reasonable option might be 0-1 loss, depending on whether the trajectory successfully executes a desired manipulation in a given state. Then we can justify warping the state s and the trajectory t in the case where $L(s,t) = L(f(s), f(t))$. Essentially, manipulation is preserved under a class of transformations, thus, we can successfully transform a state and trajectory and maintain the relation that the manipulation succeeds. The set of functions that have this property define a set of states that a particular demonstration trajectory can transfer to.

A final aspect of this approach is incorporation of multiple trajectories. Given a library of trajectories, one can increase the number of states that can be generalized to. This allows an expert to demonstrate steps of a complex task which can be sequenced at test time. This can make trajectory transfer more robust and reliable, as an expert can also include demonstrations to recover from common failures. Current approaches use the nearest-neighbor with respect to registration cost (the value of the optimization problem defined in (1)). This corresponds to modeling the set of states a demonstration can generalize to—that is states for which

L is invariant to the TPS warping found by TPS-RPM—as a hyper-sphere in a high-dimensional space where this registration cost is a distance function.

IV. Bootstrapping New Demonstrations from Experience

In this section, we present our algorithm for bootstrapping new examples from expert demonstrations. At its core, the idea is simple: if we get examples of new states that are able to successfully transfer a trajectory to, we get a new example of a successful manipulation. We can use these examples to transfer trajectories better in new settings.

Formally, we assume access to an environment and a reward signal. In our work, this environment is simulated although the approach can apply to real settings. Our reward signal is a 1-0 response which tells us if a manipulation succeeds. For tasks involving several steps, we associate success with a manipulation if a success signal is received before a fixed time horizon is reached.

To ease discussion, we will use the concept of a *transfer set*. Given a method for transfering a demonstration trajectory to a new state, the associated transfer set is the set of states such that the transferred trajectory will successfully execute a demonstrated manipulation. The assumption behind the nearest-neighbor selection method from Schulman et al. [21] can be stated that states which have a low registration cost to the demonstration scene are likely to be in the transfer set for that demonstration. This assumes that the state space is locally smooth with respect to registration cost. In practice, this assumption has been borne out in the success of this approach.

By attempting to transfer trajectories in simulation, we can get additional feedback and examples of states that are in the transfer set for our demonstrations. Continuing with this line of reasoning, a natural next step is to hope that that states that are close (with respect to registration cost) to states in the transfer set are likely to be that transfer set. This suggest a simple way to improve performance through experimentation: given a set of states that a trajectory, t, has been successfully transferred to, S_T, we select a trajectory to transfer according to the following rule:

$$\arg\min_{t} \min_{s \in S_t} \text{registration cost}(s). \tag{2}$$

We can take this idea a step further. When we succeed in completing a task in a new scenario, we get a new set of states and trajectories that perform our desired manipulation. These are new examples of successful manipulations in their own right. Instead of simply storing the states we successfully transfer to, we can add those examples to our trajectory library and consider transferring the derived trajectories to new scenes. Alg. 1 shows an exploration strategy to apply bootstrapping to a trajectory library. The process repeatedly selects the nearest-neighbour with respect to registration cost, and adds it to the trajectory library if successful.

One possible objection to this method is as follows: given that these derived examples are simply deformations of an original, why would we expect this to be better than simply transferring the original? The answer to this question is based in different aspects of the TPS approach to trajectory transfer.

input : trajLib $= [(s_1, t_1), (s_2, t_2), \ldots]$
output: bootstrapLib, bootstrapped trajectory library
bootstrapLib $\leftarrow$ trajLib;
for $i \leftarrow 0$ **to** N **do**
  $s_{test} \leftarrow$ sampleNewInitialState();
  $(s_p, t_p) \leftarrow \arg\min_{(s,t)} registration\_cost(s, s_{test})$;
  $t_{warped} \leftarrow fit\_TPS(s_p, s_{test}, t_{parent})$;
  if *successful_trajectory_execution* **then**
    bootstrapLib $\leftarrow (s_p, t_{warped}) \cup$ bootstrapLib ;
  end
end

Algorithm 1: Bootstrapping a Trajectory Library

The first is that, in addition to finding a transfer function that minimizes curvature, we are also finding correspondences between points in the different scenes. Finding correspondences is a difficult and well-studied problem in computer vision and the best approaches are subject to local optima. The TPS-RPM algorithm is no exception.

We could appeal to local features to improve this difficulty, but finding feature descriptors that capture important aspects of general manipulation problems is a difficult task. The states we add to our trajectory library are examples of states and correspondences that successfully transferred a demonstration trajectory. By transferring directly from those states, as opposed to the original demonstration state, we are providing a better initialization to the TPS-RPM algorithm and we should be able to find better correspondences between points.

The second reason we would expect this to be successful is that in transferring derived states and trajectories, we enable the use of a broader class of functions for transferring trajectories. In transferring a trajectory, t, from state s_1 through state s_2 to s_3, we compute a thin plate spline from s_1 to s_2 ($f_{1\to2}$) then from s_2 to s_3 ($f_{2\to3}$). The trajectory we execute is then $f_{1\to2}(f_{2\to3}(t)) \neq f_{1\to3}(t)$. Instead of using a thin plate spline, we are using a form of iterated thin plate spline.

The intuition behind this is that a thin plate spline represents an encoding of a preference for non-rigid functions to transfer a state. For a general approach, this is a good preference to have. However, for a particular manipulation task, not all deformations will have the same effect on transfer success.

As an example, consider a robot transferring trajectories for opening a drawer. In transferring the first portion of a demonstration, almost any deformation is OK: all that needs to happen is that the robot grabs the drawer handle. However, for the second part—actually opening the drawer—almost any non-rigid deformation will result in a failed transfer.

In fitting a thin plate spline to derived trajectories, we gain the ability to learn these transfer properties for the manipulation we are exploring. The non-rigid deformations

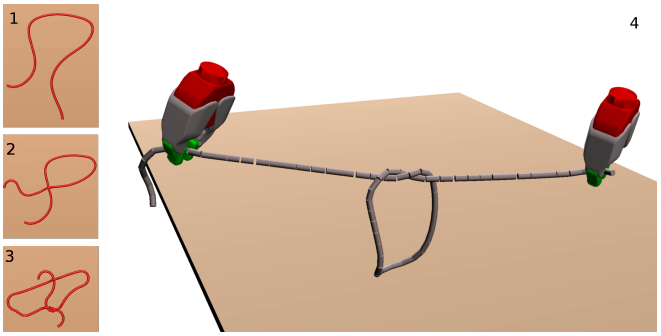


Fig. 2. Example of the steps involved in an overhand knot-tying task in our simulated environment. The standard demonstrations in our trajectory tie a knot as a sequence of 3 steps.

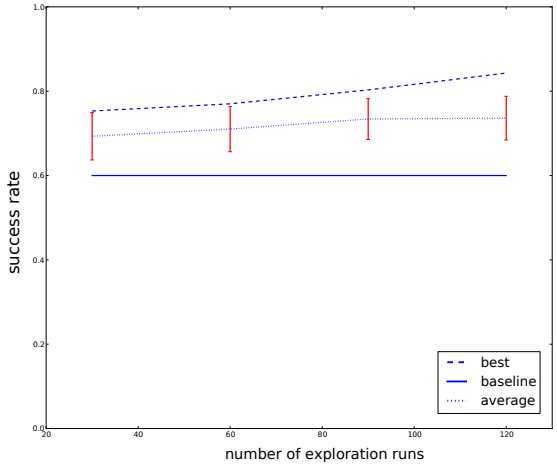


Fig. 3. Success rate of tying a an overhand knot in simulation with bootstrapped examples compared with the baseline approach from Schulman et al. [21]. The problems for this scenario were generated by selecting a random initial state from a demonstration library and perturbed randomly by dragging random points on the rope. Directly transferring from the demonstrations achieves a success rate of 59%. After doing 170 round of bootstrapping, we are able to improve on to an average of 74% success. Our top performing trained set achieved 84% success, an improvement of 25% over the baseline. This success can be attributed to several factors, key among them are better modelling of states a trajectory can transfer to and ability to penalize less for deformations that have allowed successful transfer in the past.

that resulted in successful transfers are no longer penalized in fitting the thin plate spline. For our drawer example, after enough examples of successful transfers this technique would effectively learn to allow certain types of deformations (e.g. those that allow us to grab the drawer) but still maintain the ability to penalize for others (e.g. deformations that do not allow the robot to open the drawer).

V. Experiments and Results

A. Experimental Setup

We evaluate our learning method in a simulation environment on an overhand knot-tying task. We use floating grippers to study the effects of trajectory transfer without the complications of altering derived trajectories to incorporate joint constraints.

1) Demonstrations: The demonstrations we use to initialize the trajectory library are those used by Schulman et al. [21] for their experiments. The demonstrations split the task of tying an overhand knot into 3 steps. Demonstrations were collected by physically guiding a Willow Garage PR2 through these steps and opening or closing the grippers at the appropriate points. There are 36 demonstrations of full knot ties in the data set in addition to several demonstrations that correct for common failures. Point clouds were collected with an Asus Xtion Pro RGBD camera and filtered by color to extract rope points.

2) Simulation Environment and Task Distribution: The tasks we consider are simulations of an overhand knot tying tasks. We simulate a rope as a chain of cylinders linked by bending and torsional constraints. Simulation is done through the use of Bullet Physics engine [6].

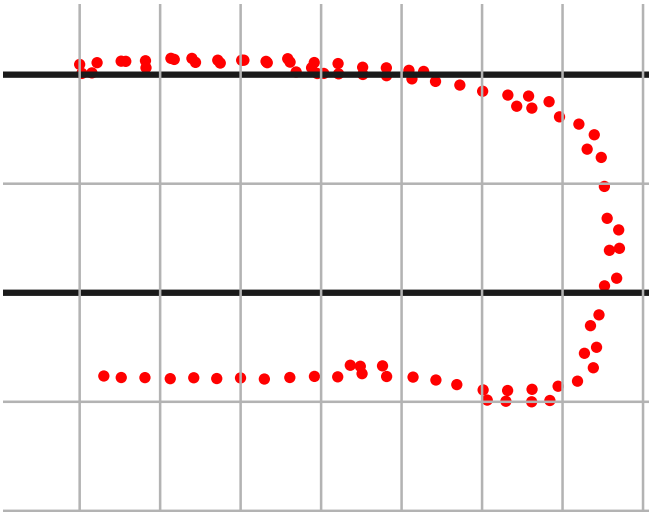
Our distribution over initial states is defined procedurally. We begin by uniformly selecting an initial state from the demonstration. Then 7 points along the rope are drawn and subjected to 10cm of perturbation in a random direction. Finally a random rotation between 0 and $\frac{\pi}{4}$ is applied to the perturbed rope. We ran our bootstrapping algorithms on initial states from this distribution and tested on a separate evaluation set.

3) Training and Evaluation: We generated 10 sets of 170 states each drawn IID from our initial state distribution. We trained a trajectory library for each of these 10 sets of initial states. Training was accomplished by an initial exploration phase of 50 attempts where only the initial (expert/ human) demonstrations were used. Then there were 120 knot-tying attempts that chose and transferred trajectories using the new techniques. We evaluated our bootstrapped libraries on an evaluation set that consisted of 300 initial states that were held out from training.

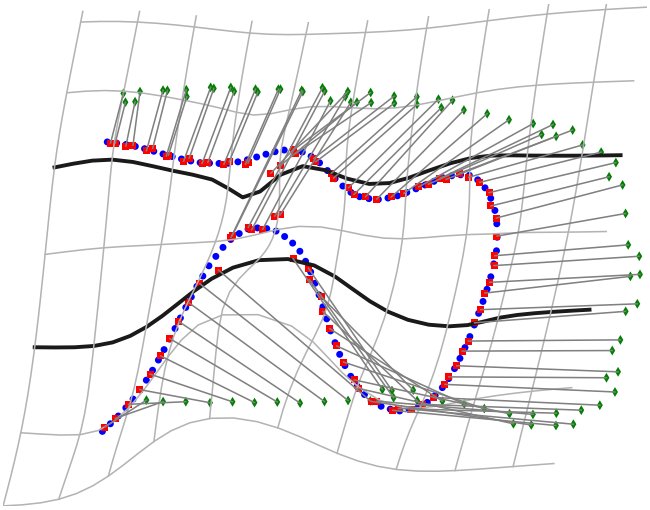
B. Results & Analysis

On our test set, across 10 different training sequences, our bootstrapped trajectories yielded an average success rate of 74%. This represents a 15% increase over the baseline approach which only used the initial human demonstrated trajectories. Our best performing run gained an additional 10% to reach an 84% success rate overall. Fig. 3 illustrates these results.

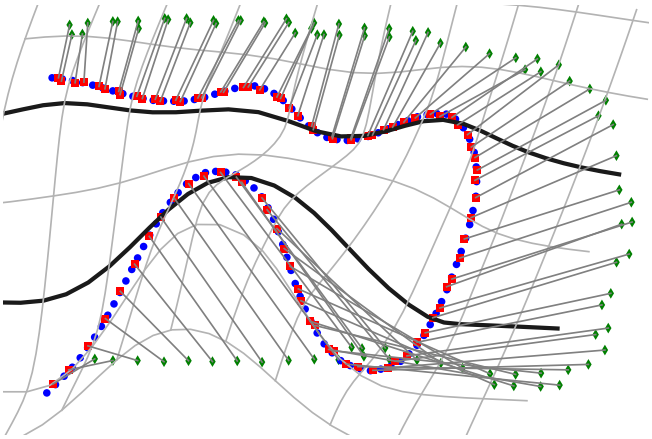
We believe that these results indicate the utility of this approach to improving trajectory transfer with non-rigid registration. By successively warping from states that we have transferred trajectories successfully, we enable transfer that penalizes less for deformations that preserve important aspects of the manipulation task without hand-coding prior knowledge. For example, in our knot-tying task, trajectory will transfer robustly for deformations in the X-Y plane, but deformations in the Z dimension will often cause unsuccessful transfers. Fig. 4 illustrates this for one example from our evaluation set. Directly warping with TPS-RPM fails because correspondences are hard to find. Even with correct correspondences, standard trajectory transfer is prone



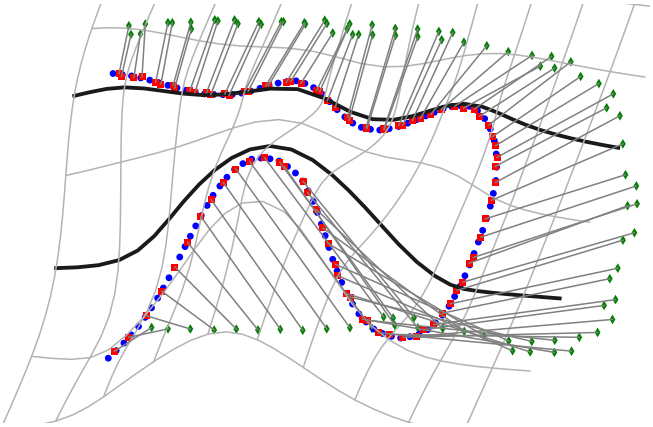
(a) A demonstration state



(b) The result from TPS-RPM



(c) TPS with Known Correspondences



(d) TPS from a Bootstrapped State

Fig. 4. Illustrations of various methods of Thin Plate Spline mappings. (a) shows the state from the demonstration being warped. (b) shows the results of directly warping from this state with TPS-RPM. The differences between the demonstration state and current state cause errors errors in the correspondences. (c) shows the results of directly fitting a thin plate spline with known correspondences. The intersection of the bold line and the rope (which are seperate in the initial scene) illustrates a large amount of non-rigidity in the Z-dimension to reduce the warping the X-Y plane. This can cause trajectory transfer to fail for this task. (d) shows the results of warping from a nearby bootstrapped state. The overall structure of the demonstration scene is better mapped into this scene because the iterated thin plate spline does not penalize for deformations that led to successful trajectory transfers during training.

to failure because the thin plate spline fit will penalize equally for all deformations. By contrast, bootstrapping from successful trajectories discovers this structure and is able to leverage it to succeed.

We believe that these results indicate the utility of this approach. We have demonstrated that given access to feedback about successful transfers can have large payoffs. However, the particular algorithm we use to actually perform this bootstrapping exhibits large variance. We believe that this can be combated through more explicit and careful trade off between exploration and exploitation. After the initial exploration phase, the focus on exploitation can miss the potential to better transfer new trajectories. A more sophisticated treatment of the exploration exploitation trade off in this setting is an important direction for future work.

ACKNOWLEDGMENT

We would like to acknowledge John Schulman, Jonathan Ho, Sachin Patil, Sandy Huang, and Alex Lee for helpful discussions, and writing help with this paper.

REFERENCES

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, May 2009.

[2] Matthew Bell. *Flexible Object Manipulation*. PhD thesis, Dartmouth College, Hanover, NH, USA, 2010.

[3] S. Calinon, F. D'halluin, D.G. Caldwell, and A.G. Billard. Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 582–588, Dec 2009.

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

[5] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, 2003.

[6] Erwin Coumans. Bullet physics, Jan 2014.

[7] Ayanna M. Howard and George A. Bekey. Intelligent learning for deformable object manipulation. *Autonomous Robots*, 10(1):51–58, 2000.

[8] Andrew Isaac and Claude Sammut. Goal-directed learning to fly. In *In Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, pages 258–265. AAAI Press, 2003.

[9] Florent Lamiraux and Lydia E. Kavraki. Planning paths for elastic objects under manipulation constraints. *International Journal of Robotics Research*, 20:188–208, 2001.

[10] Hiroyuki Miyamoto and Mitsuo Kawato. A tennis serve and upswing learning robot based on bi-directional theory. *Neural Networks*, 11(7-8):1331–1344, 1998.

[11] Hiroyuki Miyamoto, Stefan Schaal, Francesca Gandolfo, Hiroaki Gomi, Yasuharu Koike, Rieko Osu, Eri Nakano, Yasuhiro Wada, and Mitsuo Kawato. A Kendama learning robot based on bi-directional theory. *Neural Networks*, 9(8):1281–1302, November 1996.

[12] Mark Moll and Lydia E. Kavraki. Path planning for deformable linear objects. *IEEE Transactions on Robotics*, 22:625–636, 2006.

[13] Takuma Morita, Jun Takamatsu, Koichi Ogawara, Hiroshi Kimura, and Katsushi Ikeuchi. Knot planning from observation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 3887–3892. IEEE, 2003.

[14] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann LeCun. Off-road obstacle avoidance through end-to-end learning. In *Advances in Neural Information Processing Systems*, pages 739–746, 2005.

[15] Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1989.

[16] Nathan Ratliff, J. Andrew Bagnell, and Siddhartha S. Srinivasa. Imitation learning for locomotion and manipulation. In *IEEE-RAS International Conference on Humanoid Robots*, 2007.

[17] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew (Drew) Bagnell, and Martial Hebert. Learning monocular reactive uav control in cluttered natural environments. In *IEEE International Conference on Robotics and Automation*. IEEE, March 2013.

[18] Mitul Saha, Pekka Isto, and Jean-Claude Latombe. Motion planning for robotic manipulation of deformable linear objects. In Oussama Khatib, Vijay Kumar, and Daniela Rus, editors, *Experimental Robotics*, volume 39 of *Springer Tracts in Advanced Robotics*, pages 23–32. Springer Berlin Heidelberg, 2008.

[19] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[20] John Schulman, Ankush Gupta, Sibi Venkatesan, Mallory Tayson-Frederick, and Pieter Abbeel. A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario. In *Proceedings of the 26th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[21] John Schulman, Jonathan Ho, Cameron Lee, and Pieter Abbeel. Learning from demonstrations through the use of non-rigid registration. In *Proceedings of the 16th International Symposium on Robotics Research (ISRR)*, 2013.

[22] J. Takamatsu, T. Morita, K. Ogawara, H. Kimura, and K. Ikeuchi. Representation for knot-tying tasks. *Trans. Rob.*, 22(1):65–78, November 2006.

[23] T. Wada, S. Hirai, H. Mori, and S. Kawamura. Robust manipulation of deformable objects using model based technique. In Hans-Hellmut Nagel and FranciscoJ. Perales Lpez, editors, *Articulated Motion and Deformable Objects*, volume 1899 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin Heidelberg, 2000.

[24] G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

[25] Hidefumi Wakamatsu, Eiji Arai, and Shinichi Hirai. Knotting/unknotting manipulation of deformable linear objects. *The International Journal of Robotics Research*, 25(4):371–395, 2006.