Beyond Lowest-Warping Cost Action Selection in Trajectory Transfer

Dylan Hadfield-Menell, Alex X. Lee, Chelsea Finn, Eric Tzeng, Sandy Huang, and Pieter Abbeel

***Abstract*— We consider the problem of learning from demonstrations to manipulate deformable objects. Recent work [1], [2], [3] has shown promising results that enable robotic manipulation of deformable objects through learning from demonstrations. Their approach is able to generalize from a single demonstration to new test situations, and suggests a nearest neighbor approach to select a demonstration to adapt to a given test situation. Such a nearest neighbor approach, however, ignores important aspects of the problem: brittleness (versus robustness) of demonstrations when generalized through this process, and the extent to which a demonstration makes progress towards a goal.**

In this paper, we frame the problem of selecting which demonstration to transfer as an options Markov decision process (MDP). We present max-margin Q-function estimation: an approach to learn a Q-function from expert demonstrations. Our learned policies account for variability in robustness of demonstrations and the sequential nature of our tasks. We developed two knot-tying benchmarks to experimentally validate the effectiveness of our proposed approach. The selection strategy described in [2] achieves success rates of 70% and 54%, respectively. Our approach performs significantly better, with success rates of 88% and 76%, respectively.

I. Introduction

Robotic manipulation of deformable objects tends to be challenging due to high-dimensional, continuous state-action spaces and due to the complicated dynamics of deformable objects. Despite these challenges, recent work [1], [2], [3] has shown promising results that enable robotic manipulation of deformable objects through learning from demonstrations. This work uses non-rigid registration to register a training scene onto the current scene, and then extrapolates from this registration to perform *trajectory transfer* of the robot end-effector trajectory. The effectiveness of this approach was validated in knot-tying and suturing experiments.

For complex tasks, demonstrations often correspond to steps in the task, rather than the entire task itself. Figure 1 shows an example of the steps involved in tying an overhand knot. In general, a single demonstration for a step in the task cannot be expected to cover all possible scenarios that arise during execution. The natural solution to this is to use a library of demonstrations with multiple demonstrations for each step.

Realizing these benefits requires a robust technique to select a good trajectory to transfer. Certain trajectories will generalize better than others, and particular sequences of demonstrations may perform tasks more efficiently than others. The original paper presenting the approach of trajectory transfer prescribes choosing the trajectory segment from the demonstrations library with the lowest warping cost onto the current scene [2]. This approach does not account for the inherent generalizability of a particular demonstration: some demonstrations may be better adapted to transfer than others. For brittle demonstrations (e.g. grabbing near the edge of a rope), a small change in the state can have low registration cost, but the transferred trajectory will fail. As a result, such an approach often fails to accomplish tasks that can be solved with the existing library of demonstrations. In addition to ignoring potential brittleness in demonstrations, registration cost does not consider any notion of a goal or task.

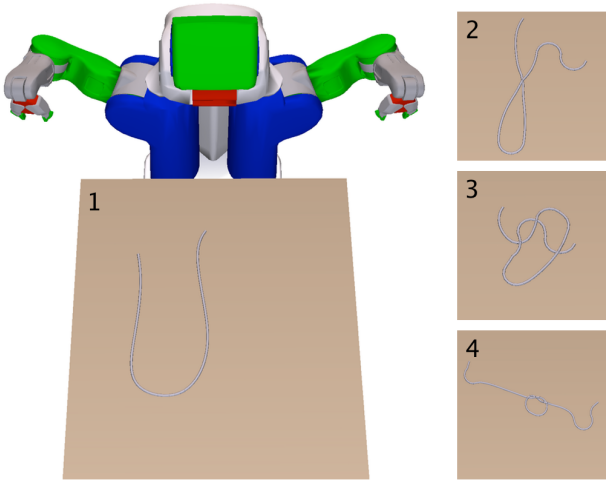

Fig. 1: The overhand knot manipulation task in our benchmark. A standard knot tie takes three steps, as shown in this particular execution from our benchmark.

Our approach to these issues is to use a demonstration library to define a related, simpler, Markov decision process where each action corresponds to performing trajectory transfer from a particular demonstration. We learn a policy for this representation in the form of a Q-function: a measure of cost-to-go for a state and action pair. Given an (approximate) Q-function there are several policies one can define. The simplest is to select the action that maximizes the value of the Q-function for the current state. A more computationally intensive approach uses a simulator to search over action sequences and selects the action that begins the 'best' action sequence, where best is defined as the sum of the cost encountered during the action sequence plus cost-to-go at the final state (as encoded by the Q-function).

In this paper, we present a solution to the demonstration selection problem that can account for the variability in

Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA, USA. {dhm, alexlee_gk, cbfinn, etzeng, shhuang, pabbeel}@cs.berkeley.edu

robustness of demonstrations and incorporates the sequential nature of our tasks. Our contributions are as follows: (i) We formulate the demonstration selection problem as an options Markov decision process (MDP); (ii) We present max-margin Q-function estimation (MMQE), a method for approximating Q-functions from expert-guided task executions; and (iii) We describe task-independent features that are rich enough to allow learning but make no additional assumptions beyond those of trajectory transfer.

We developed two knot-tying benchmarks for evaluating the effectiveness of our proposed approach, overhand knots and figure-eight knots. These benchmarks are available at sites.google.com/site/icra015mmqe). The nearest neighbor approach described in [2] achieves a success rate of 70% and 54% on these two benchmarks. The reactive policy with respect to our learned approximate Q-function achieves success rates of 82% and 63% respectively. Augmenting our policy with a simulator and beam search raises the success rate to 88% and 76%.

While our primary running example and experiments deal with knot tying, our policy learning and trajectory transfer methods make few assumptions about the specifics of the task except for the availability of point clouds for the scene; it is applicable in a wide class of manipulation problems.

II. Related Work

Related work for our contribution stems from three areas of research: learning from demonstrations, deformable object manipulation (in particular knot tying), and hierarchical reinforcement learning.

The problem of *learning from demonstrations* (LfD) deals with the generalization of expert demonstrations to new scenarios [4], [5]. 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 behavioral cloning is the ALVINN system, which utilizes a neural network to learn a steering policy that enables an autonomous car to follow a road [6]. Muller et al. use a convolutional network to learn a steering policy for off-road driving [7]. Ratliff et al. use multi-class classification to learn a function that scores actions to predict good foot steps for robot locomotion and good grasps for robot manipulation [8]. Ross et al. propose a method to directly control a Micro UAV from RGB camera input [9].

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 [10], [11]. Their approach assumes access to a fixed representation of the environment in terms of object frames, so it cannot be applied to tasks in environments without fixed feature representations — such as our application of knot tying.

Ratliff et al. [12] use max-margin planning to do *inverse reinforcement learning* [13], [14]. They obtain trajectories from expert human demonstrators for motion planning problems. They formulate a max-margin problem to infer the cost function the expert is optimizing. Our approach shares similarity with this in that both learn from demonstrations with a max-margin method. However, they learn a cost function whereas our approach learns a Q-function. In addition, their approach requires access to a dynamics model.

Dvijotham and Todorov directly learn a value function or Q-function for an MDP, given sample transitions from an optimal control policy [15]. They learn the expert's reward function but are limited to models with tractable discrete representations or linear dynamics models. In contrast, our approach fixes a cost function and uses learning to account for prohibitive state-action space size and complex dynamics.

Because our approach makes limited assumptions about the state space and dynamics of the model, it can be applied towards a variety of tasks in robotics, including the manipulation of deformable objects. It is challenging to manipulate deformable objects because of their nonlinearity and because the configuration spaces of such objects may be infinite-dimensional [16].

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 [17]. 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 to determine the minimum force required for manipulating the object [18]. In contrast, our approach enables manipulation of deformable objects without directly modeling the object, and makes no assumptions beyond those of trajectory transfer. When a model of state-transitions is available, we can use lookahead to reduce uncertainty in execution through the simulation of transferred trajectories.

We validate our approach in a knot tying domain, a commonly studied deformable object manipulation task in robotics. Previous approaches usually depend on rope-specific knowledge and assumptions. For instance, in knot planning from observation, knot theory is used to recognize rope configurations and define movement primitives from visual observations of humans tying knots [19], [20]. 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 [21], [22], [23]. Robust open loop execution of knot tying has also been explored [24]. In contrast to these methods, our approach does not explicitly make use of underlying rope-specific knowledge or directly model the rope; instead, it infers this knowledge by robustly applying human-guided demonstrations to new scenes through a learned selection criterion and trajectory transfer.

Konidaris et al. use LfD to initialize a skill chaining reinforcement learning algorithm [25]. However, they focus on taking a demonstration and decomposing it into explicit local policies. We simply make use of the associated trajectory transfer and leverage it to solve tasks beyond the reach

of current reinforcement learning approaches. Neumann et al. and Stulp et al. both explore using optimizing motion primitive parameters as another way to elicit this behavior in reinforcement learning [26], [27]. When they learn the primitives, they must find an appropriate policy for setting these parameters, which is intractable in our setting.

III. Technical Background

A. Markov decision processes

Markov decision processes (MDP) provide a mathematical model for sequential decision making problems. In this work, we consider *stochastic shortest path* (SSP) formulations of MDPs. An (undiscounted) SSP, M, is a tuple: $M = \langle \mathcal{S}, \mathcal{G}, \mathcal{A}, T, C \rangle$ [28]. $\mathcal{S}$ is a set of states, which represent different configurations of our world. $\mathcal{G} \subset \mathcal{S}$ is a set of terminal states. $\mathcal{A}$ is a set of available actions. $T : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{S}}$ is a function that maps a state and an action to a probability distribution over next states. $C : \mathcal{S} \rightarrow \mathbb{R}^+$ is a function that specifies the cost associated with states. In an SSP, costs are positive everywhere except for terminal states. A solution to an MDP is a *policy* that maps each state to an action.

This solution is found by finding a value function, V^*, that satisfies the Bellman equations:

$$V(s) = \begin{cases} \max_a \sum_{s'} T(s,a,s')[V(s') - C(s)] & s \notin \mathcal{G} \\ 0 & s \in \mathcal{G} \end{cases}$$

It is sometimes easier to work with a Q-function, given by the expression inside the $\max$ in the above equation. Thus, $V^*(s) = \max_a Q^*(s,a)$. There are many approaches to finding such a value function, but they require storing a vector that is $O(|\mathcal{S}|)$, which is prohibitive in many applications.

A common (approximate) approach to solving an MDP is linear value function approximation [29]. Given a M-dimensional feature map, $\phi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^M$, we restrict ourselves to value functions that are linear combinations features and minimize the error associated with the Bellman equations for a set of sampled state transitions.

An *options* MDP extends MDPs to allow temporal abstraction [30]. An option, o, is a combination of a policy, an instantiation set and a termination set. When an agent is in a state within o's instantiation set they can select o like any other action. Actions are then selected according to o's policy until the termination set is reached. This enables simple policies over options to specify complex behavior for large MDPs.

B. Trajectory transfer through non-rigid registration

In complex, high-dimensional and continuous MDPs, such as those that arise in deformable object manipulation, linear value function approximation often fails to compute a reasonable policy—it is prohibitively expensive to sample enough state transitions to effectively represent the dynamics of the problem. A promising alternative leverages demonstrations from successful task executions. This approach, *learning from demonstrations* (LfD), has been successfully used in a wide variety of applications [31], [10]. In this work, we build on a recent LfD method that uses non-rigid registration to perform *trajectory transfer* from demonstrations.

Non-rigid registration is a method to compute mappings between two scenes based on correspondences between landmark points. A commonly-used, effective method for registering spatial data is the Thin Plate Spline (TPS) regularizer [32], [33]. Given a set of correspondence points $(\mathbf{x}_i, \mathbf{y}_i)$, the corresponding (regularized) TPS is a function, $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, that minimizes the following energy functional:

$$E_{bend}(\mathbf{f}; \mathbf{X}, \mathbf{Y}, C) = \sum_i ||\mathbf{f}(\mathbf{x_i}) - \mathbf{y}_i||^2 + C \int dx ||\mathrm{D}^2(\mathbf{f})||^2_{\text{Frob}}.$$

This functional is called bending energy because the second term measures the curvature (bending) of f as the norm of the Hessian of f. $\operatorname{argmin}_f E_{bend}(f; X, Y, C)$ admits a solution of the form:

$$\mathbf{f}(\mathbf{x}'; \mathbf{A}, \mathbf{B}, \mathbf{c}, \mathbf{X}) = \mathbf{A}K(\mathbf{X}, \mathbf{x}') + \mathbf{B}\mathbf{x} + \mathbf{c}.$$

Where K is the 3D TPS kernel $K(\mathbf{x}, \mathbf{x}') = -||\mathbf{x} - \mathbf{x}'||$, $\mathbf{A} \in \mathbb{R}^{3 \times N}$, $\mathbf{B} \in \mathbb{R}^{3\text{x}3}$, and $\mathbf{c} \in \mathbb{R}^3$. Given a point cloud and correspondences, we solve for the coefficients A, B, and c with a straighforward application of least squares.

In the case where correspondences are unknown, Thin Plate Spline Robust Point Matching (TPS-RPM) algorithm provides a method to jointly find correspondences and mappings between them [34]. TPS-RPM is an expectation maximization style algorithm that alternates between (1) estimating correspondences between the point clouds of two scenes based on a current mapping and (2) fitting the optimal TPS transformation based on these estimated scene correspondences.

Recent approaches leverage TPS to perform *trajectory transfer* [2]. They represent a demonstration, d, as a paired point cloud and trajectory of end effector poses: $d = (P_d, \tau_d)$. In the current (or test) scene, P', they compute the TPS to the demonstration point cloud $f^*_{P_d, P'}$ with TPS-RPM. Then we execute the *transferred trajectory*: $\tau' = f^*_{P_d, P'}(\tau_d)$. In the case where this transfer is infeasible, trajectory optimization is used to find a nearby feasible trajectory. We use the pose transfer approach from Lee et al. [3] to use a 3D mapping to transfer end effector configurations.

Schulman et al. [2] extend trajectory transfer to demonstration libraries by first selecting

$$d^*_{P'} = \operatorname*{argmin}_{d \in \mathcal{D}} E_{bend}(f^*_{P_D, P'}) \tag{1}$$

and applying trajectory transfer from $d^*_{P'}$. While it was designed for deformable object manipulation, it is straightforward to apply trajectory transfer in any scenario where point clouds are available. As long as a successful trajectory can be determined from geometric information, then trajectory transfer can be expected to perform well.

C. Structured max margin

In this paper, we explore the problem of learning a (task-specific) ranking function to select $d^*_{P'}$. We approach this task in a max-margin framework. The simplest (although somewhat naïve) approach in this framework is to train an SVM to recognize the expert's state-action pairs. We assume a feature representation of state-action pairs and a set of optimal state-action pairs, $\mathcal{L} = \{(a_i, s_i)\}$. Then we find a hyperplane in feature space that maximizes the margin from optimal state-action pairs to suboptimal state-action pairs. This is formalized as the following convex optimization, ϕ is a feature map for state-action pairs:

$$\begin{aligned} \underset{\mathbf{w},\xi\geq 0}{\text{minimize}} \quad & ||\mathbf{w}||^2 + C\sum \xi_i \\ \text{subject to} \quad & \mathbf{w}^\top \phi(s_i, a_i) \geq \mathbf{w}^\top \phi(s_i, a') + 1 - \xi_i \\ & \forall (s_i, a_i) \in \mathcal{L}, \forall a' \in \mathcal{A} \setminus a_i. \end{aligned} \tag{2}$$

The ξ_i are slack variables that ensure the optimization is always feasible. This frequently runs into issues when there are actions which are suboptimal, but only barely. For example, if there are two identical actions, of which the expert only selected one, we will be unable to find a reasonable separating hyperplane. In a structured max-margin method, we account for this issue with a similarity measure m between state-action pairs. The corresponding optimization is as follows:

$$\begin{aligned} \underset{\mathbf{w},\xi\geq 0}{\text{minimize}} \quad & ||\mathbf{w}||^2 + C\sum \xi_i \\ \text{subject to} \quad & \mathbf{w}^\top \phi(s_i, a_i) \geq \mathbf{w}^\top \phi(s_i, a') + m(a_i, s_i, a') - \xi_i \\ & \forall (s_i, a_i) \in \mathcal{L}, \forall a' \in \mathcal{A} \setminus a_i. \end{aligned} \tag{3}$$

IV. Learning a policy to select demonstrations

A. From expert demonstrations to options

In this section, we present the key conceptual contribution of our work: an interpretation of trajectory transfer from a demonstration library as options within an MDP. This allows us to apply MDP learning techniques to this problem and enables substantial performance improvements.

To motivate trajectory transfer, consider an overhand knot-tying task. The state space is the joint space of rope configurations and velocities as well as robot configurations. The action space is the continuous set of motor torques that can be commanded to the robot's joints. The goal set is states where the rope configuration satisfies a complex topological constraint. The state transition function is defined by physical properties and include both contact and tension forces.

Direct solution is impractical—even representing the state compactly is a challenge. An additional challenge arises from the length of plans to take required to solve this problem. If we discretize actions by specifying voltage commands for some fixed amount of time, it is entirely reasonable to suspect that reaching a goal state will take hundreds of individual actions.

With the knot-tying task in mind, let us consider the properties of trajectory transfer. We obtain a demonstration that maps a particular rope configuration (represented as a discrete set of points) to a series of end effector poses. When we transfer this demonstration to a new state, s, we obtain a new trajectory. This mapping is deterministic for a fixed transfer method and a fixed demonstration. When we have multiple demonstrations, we select one of these trajectories to execute.

What relation do these *transferred* trajectories have to the original problem? Trajectory execution runs a controller to take the current end effector pose to the next one in the trajectory, so we can conceptualize trajectory transfer as a mapping from states to policies the original (intractable) problem description. Thus, a demonstration library, a transfer technique, and an MDP, combine to create an options MDP that we call a *demonstration MDP*. Figure 2 illustrates this observation with a system diagram.

The demonstration MDP for a task will typically be far simpler. Consider our overhand knot task with the addition of a demonstration library. While the state space is still unconscionably large, the action space and effective horizon are much shorter. The demonstration library used in [2] has 148 demonstrations and splits a knot-tying task into four steps. The action space has been reduced from a subset of $\mathbb{R}^{14}$ to a finite set with 148 members. The planning horizon has been reduced from hundreds to approximately four.

In this light, the decision rule in Equation 1 is a hand-coded policy for a demonstration MDP that leverages knowledge about the transfer process. However, the problem is now of a form such that it is reasonable to *learn* a policy. We also observe that search algorithms have running times that are $O(|\mathcal{A}|^H)$. Thus, the reduction in complexity obtained through a demonstration MDP enables us to apply search techniques with learned value functions to solve this problem.

B. Learning a policy in the demonstration MDP

Much of policy learning is viewed from the perspective of learning a Q-function. This provides a measure of the quality of executing a state-action pair. Replacing the bending cost in Equation 1 with the Q-function would provide the optimal selection rule for a demonstration library.

Computing an optimal Q-function for challenging tasks such as deformable object manipulation is likely intractable, as the state space is prohibitively large, but approximate methods are applicable. We use a feature map to embed state-action pairs in a common space. We will use linear combinations in this space to define our approximate Q-function:

$$\tilde{Q}(s, a; w_0, w) = w_0 + w^\intercal \phi(s, a). \tag{4}$$

A way to learn a policy for this problem uses the structured max margin approach described in Section III-C. We collect examples of optimal state-action pairs, where the actions indicate which demonstration to transfer and require that $\tilde{Q}$ rank expert selection higher than the others.

We let $\mathcal{L} = \{(s_i, a_i)\}$ be a set of expert state-action pairs. We can solve the optimization in Equation 3 to obtain a (state-dependent) ranking function for demonstrations. This

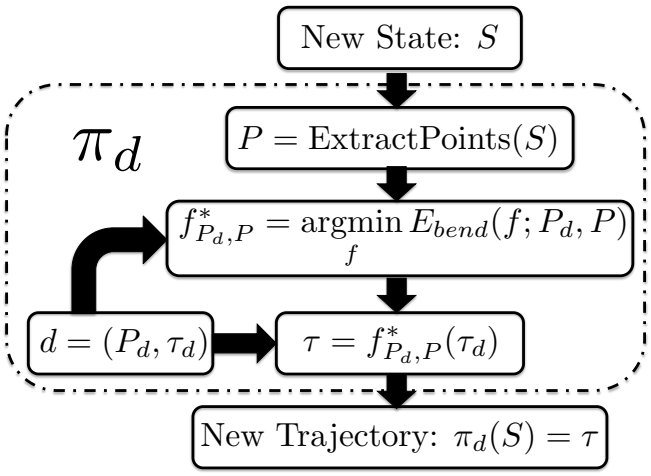

Fig. 2: System diagram indicating the trajectory transfer process. This illustrates that, for a fixed transfer process and demonstration, trajectory transfer specifies a policy mapping states to trajectories. Each trajectory is a policy when viewed from the point of view of primitive actions. Thus, we can think of a demonstration library as providing the structure of an options MDP for a given problem. This options MPD will typically be much simpler to solve.

is easily turned into a policy by selecting the demonstration that maximizes this ranking function for the current state.

This will learn a policy that approximates the expert's and is easy to control for overfitting. However, it would be a mistake to call this an approximate Q-function. In addition to ranking optimal actions higher, the Q-function also satisfies the Bellman equations and provides a measure of cost-to-go. There are two reasons we would like a learned ranking to model this behavior.

The first is that, in the case that $\mathcal{L}$ is composed of sequences of state action pairs, we discard information by ignoring Bellman constraints. It may be possible to obtain a ranking function that will generalize better by including information about Bellman equations in the learning process.

The second drawback is related to our observation that demonstration MDPs have actions sets and effective horizons that are small enough to make search a viable option. Putting this into practice corresponds to a multistage lookahead policy [28]. These policies expand a local region of the state space as a tree (using a chosen search method). Then, the agent acts optimally under the assumption that leaves have value $\tilde{Q}$. This strategy does not make sense unless $\tilde{Q}$ can rank state-action pairs where the states are different. However, the max-margin solutions provide no guarantees that this will be the case—each state's evaluation is only loosely related to the others'.

C. Maximum margin Q-function estimation (MMQE)

To learn a Q-function in a max-margin framework, we propose *maximum margin Q-function estimation* (MMQE). The basic idea behind this approach is straightforward. In addition to maximizing the margin between optimal actions and suboptimal actions, we also minimize a measurement of the *Bellman error* associated with $\tilde{Q}(\cdot, \cdot; w_0, w)$

Formally, we assume that our labelled examples are broken up into sequences $l_j \in \mathcal{L}$, each of length n_j:

$$l_j = [(s_1^{(j)}, a_1^{(j)}), (s_2^{(j)}, a_2^{(j)}), \ldots, (s_{n_j}^{(j)}, a_{n_j}^{(j)})].$$

Each labeled sequence l_j corresponds to a single complete task execution and is ordered chronologically: $s_i^{(j)}$ is the result of taking action $a_{i-1}^{(j)}$ in state $s_{i-1}^{(j)}$, and $s_{n_j}^{(j)}$ is a goal state.

For a fixed $\tilde{Q}$, we can estimate the Bellman error associated with l_j as

$$\sum_{i=0}^{n_j} |\tilde{Q}(s_i, a_i; w_0, w) - \tilde{Q}(s_{i+1}, a_{i+1}; w_0, w) + C(s_i)|. \quad (5)$$

MMQE simply includes this error in the objective of a structured margin optimization:

$$\min_{w,\xi \geq 0,\nu} \quad ||w||^2 + C \sum_{j=1}^{|\mathcal{L}|} \sum_{i=1}^{n_j} \xi_i^{(j)} + D \sum_{j=1}^{|\mathcal{L}|} \sum_{i=1}^{n_j} |\nu_i^{(j)}| \quad (6)$$

$$\text{s.t.} \quad w^\intercal \phi(s_i^{(j)}, a_i^{(j)}) \geq w^\intercal \phi(s_i^{(j)}, a') + m(s_i^{(j)}, a_i, a') - \xi_i^{(j)} \quad (7)$$

$$\forall j = 1, \ldots, |\mathcal{L}|;\ \forall i = 1, \ldots, n_j;$$
$$\forall a' \in \mathcal{A} \setminus a_i$$

$$w^\intercal \phi(s_i^{(j)}, a_i^{(j)}) - w^\intercal \phi(s_{i+1}^{(j)}, a_{i+1}^{(j)}) + C(s_i^j) = \nu_i^{(j)} \quad (8)$$

$$\forall j = 1, \ldots, |\mathcal{L}|;\ \forall i = 1, \ldots, n_j - 1$$

This optimization will find an approximate Q-function that matches the expert's action ranking and the sampled Bellman conditions. Equation 6 maximizes the margin while minimizing the total constraint violation. Equation 7 expresses that expert selections should be preferred to other options (as in a structured margin optimization). Equation 8 expresses that the Q-function satisfy Bellman the equations for expert demonstrations.

V. Feature Design

In the formulation above, we assumed the presence of a feature function $\phi(s, a)$ that produces a featurized representation of a state-action pair. We briefly outline a few basic features that are general enough to apply to any task in which trajectory transfer is applicable.

- **Action bias**: An $|\mathcal{A}|$-dimensional vector, with each component corresponding to a particular action in $\mathcal{A}$. Let i_a be an index associated uniquely with action a. The action bias vector is 0 at every component except i_a, where it has a value of 1. This enables learning whether actions generalize well or poorly and weights them accordingly.
- **Registration cost**: An $(|\mathcal{A}| + 1)$-dimensional vector based on the TPS registration cost r between s and a_{start}. The registration cost vector consists of a shared component, which is always r, and an action-specific component, in which component i_a is set to r and every

other component is left at 0. The shared component allows for a single penalty to be applied for any large registration cost. The individual components allow for additional adjustments in the cases where demonstrations are particularly sensitive to poor registrations.

- **End effector position**: A scalar feature that indicates the minimum distance between the transferred trajectory and the observed point cloud at the first time the gripper closes. This feature enables limited reasoning about failures for grasping tasks. For knot-tying it allows $\tilde{Q}$ to encode that demonstrations whose transferred trajectory fail to grasp the rope have low value.
- **Landmarks**: We randomly select a set of "landmark" states $\mathcal{K}$ from the set of expert demonstrations. The landmark feature is an $|\mathcal{K}|$-dimensional vector consisting of the TPS registration costs to each of these landmarks. We apply a Gaussian RBF kernel to these costs and normalize the vector to sum to 1. This serves to identify which portion of the state space we are in by comparing to known states, and enables preference for states that lie closer to the goal.

In our experiments, which we outline in Section VI, it suffices to simply concatenate the outputs of these feature functions into a single feature representation. For this work, we restrict ourselves to features that apply in any setting where trajectory transfer applies; it would be straightforward to incorporate domain-specific features if desired.

VI. Experiments and Results

A. Experimental Setup

In our experiments, we compare two methods for extracting a policy from an approximate Q-function:

- Reactive: simply selecting the action with the largest estimated Q-value
- Lookahead: using a simulator and beam search to select the action that yields the maximum approximate Q-value at a future time step.

We evaluate the reactive policy with two different feature sets and we evaluate two lookahead policies, all using the Willow Garage PR2 in a simulation environment on two knot tying tasks.

1) Demonstrations: For overhand knots, we use the set of demonstrations collected by [2] for their rope-tying experiments. These demonstrations were collected in the real world and consist of 148 pairs of point clouds and gripper trajectories. For figure-eight knots, we collected a new set of demonstrations, consisting of 88 cloud, trajectory pairs. In both cases, the point clouds were collected using an RGB-D camera and then filtered based on color to remove the green background of the table, leaving only the point cloud representation of the rope. The gripper trajectories were recorded kinesthetically, from human-guided demonstrations that move the robot's grippers to tie a knot. The overhand knot dataset contains demonstrations for a three-step method of tying an overhand knot and recovery demonstrations that enable recovery from common failures, whereas the figure-eight dataset contains four-step demonstrations without failure recovery demonstrations.

2) Simulation Environment for Knot Tying: The training, validation, and evaluation of our policy are run in simulation. The simulation environment uses Bullet Physics [35] as the physics engine. The rope is simulated with a linked chain of capsules with bending and torsional constraints.

3) Benchmark: For the labeled examples $\mathcal{L}$ used to train our MMQE model, we use human-labeled examples consisting of complete task executions: from a randomly drawn initial rope configuration to a successfully tied knot.

Initial rope states are randomly perturbed configurations of samples uniformly drawn from the initial robe states in the demonstration library. The process of perturbing a rope state consists of selecting five uniformly-spaced points along the rope and dragging each in a random direction for a radius of 10 cm. All initial rope configurations in our experiments are drawn from this distribution.

The goal of labeling is to specify the best demonstration to generalize from for many observed rope states. Our labeling interface shows the user simulations of applying demonstrations in decreasing order of registration cost, which is a rough measure of the quality of an action. When the human labeler sees an action that they deem optimal (or close enough to optimal) they indicate this, the simulated robot commits to that action, and the process proceeds until a knot is tied.

In addition, the benchmarks contain an evaluation set of 100 randomly drawn initial configurations. In the first benchmark, we define success to be tying an overhand knot in a sequence of 5 or fewer actions, and in the second benchmark, tying a figure-eight knot in 6 or fewer actions.

B. Experiments

We evaluate the success rate for the following policies and feature combinations:

(i) reactive policy: action bias, registration cost
(ii) reactive policy: (i) + end effector pose
(iii) two-step lookahead with width 5: (i) + landmark
(iv) three-step lookahead with width 3: (i) + landmark

For the overhand knot tying task, our training data is 130 expert-labeled task executions, consisting of 565 individual segments. Similarly, for the figure-eight task, 123 labeled executions of 559 segments are used. For each of the evaluated policies, the optimization hyperparameters C and D are first tuned via holdout validation, using a holdout set of 100 randomly drawn initial rope states (distinct from the evaluation set). For this case, the best-performing hyperparameters we found via cross-validation are $C = 100$, $D = 1$ for the reactive policies and $C = 100$, $D = 100$ for the lookahead policies. We applied MMQE with these hyper-parameter settings and ran each of the four policies on the evaluation set. As a baseline comparison, we also evaluate using the nearest neighbor policy presented in [2]. Their policy chooses the action associated with the state that has the smallest registration cost with respect to the current

Policy	OH Success Rate	F8 Success Rate
Nearest neighbor [2]	70%	54%
Reactive	82%	59%
Reactive, with EE feature	80%	63%
Lookahead (width 2, depth 5)	87%	61%
Lookahead (width 3, depth 3)	88%	76%

TABLE I: Success rate of tying overhand (OH) and figure-eight (F8) knots. The nearest-neighbor method selects the demonstration to transfer as in Equation 1. Other policies maximize the approximate Q-function we learn through MMQE. Reactive maximizes this value in the current state. The lookahead policies act to maximize a backed up value computed by beam search with the specified parameters. The reactive policy succeeds in an additional 18% and 23% of examples in each task when compared with the baseline. Lookahead policies achieve very high performance rates and approach the best possible with our demonstration library.

# of Training Examples	OH Success Rate	F8 Success Rate
Nearest neighbor [2]	70%	54%
40	77.0%	53.3%
80	79.3%	57.3%
All (130, 123)	82%	63%

TABLE II: We vary the number of examples used to generate constraints for the MMQE optimization problem and examine how average task performance is affected over 100 test trials using the resulting reactive policies. It turns out that the number of examples has a relatively small influence on the performance of the resulting policy for the overhand knot task and a larger influence for the figure-eight task.

state. The success rates obtained under these policies are summarized in Table I. Note that our best results surpass the baseline by 18% and 23% respectively. About half of this improvement results from the reactive approach of choosing the action that maximizes the Q-value, given the current state. The remaining improvement is gained through lookahead; the performance of depth 3, width 3 lookahead surpasses that of depth 2, width 5 lookahead (Table I).

We also examine the relation between the number of expert-labeled executions and the success rate, using the most successful reactive policy. For each task, we generate six randomly sampled subsets of the training data: three sets of size 40 and three of size 80. The average success rates for each training set size are shown in Table II. As expected, the success rate increases as we use more labeled examples. For the overhand knot task, the resulting policies remain surprisingly resilient to the number of training examples. Even with only 40 labeled task executions, the reactive policy is able to exceed the baseline by 7%. On the figure-eight knot, a more difficult task, the number of training examples is more important for learning a successful policy. The average performance differs by nearly 10% when changing the training set size from 40 task executions to 123.

VII. Conclusions and Future Work

In summary, we presented a method to improve the use of trajectory transfer in learning from demonstrations for deformable object manipulation. We formulated this task as a discrete-action options MDP. We presented a novel approach to learning from demonstrations for this MDP, which we call maximum margin Q-function estimation, that integrates behavior cloning and value function approximation. We provide features for learning in this scenario that make no additional assumptions beyond the assumptions required to apply trajectory transfer.

We validated our approach on two simulated knot-tying tasks and contribute two benchmark sets of demonstrations (both at the trajectory level and in the abstract MDP) and problems. We showed a significant improvement over a baseline method that uses a registration cost from trajectory transfer to select actions. Our experiments suggest that the quantity of training examples is important in the more difficult figure-eight task, whereas our method approaches peak performance on the overhand knot with a modest number of training examples.

A important avenue for future work is to investigate real world performance of these approaches. Our results illustrate a proof of concept and show the utility of modelling the cost-to-go for this task, but it is important to investigate the application of our methods and abstraction techniques to the real world. In particular, it is important to account for mismatch between the physics model in our simulator and that of the real world.

More broadly, our key insight is that we can formulate a temporally abstracted problem representation from a library of expert demonstrations. This abstraction greatly simplifies the complexity of the problem and, in turn, this allows us to develop and apply efficient learning and planning methods. This suggests a general approach to LfD that extracts computationally tractable problem representations from expert demonstrations.

Acknowledgements

This research was funded in part by AFOSR through a Young Investigator Program award. Dylan Hadfield-Menell was supported by a Berkeley Fellowship, Alex Lee by an NSF Graduate Research Fellowship, Chelsea Finn by a Berkeley EECS Fellowship, Sandy Huang by a Chancellor's Fellowship, and Eric Tzeng by a Berkeley EECS Fellowship.

References

[1] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. 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.

[2] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from Demonstrations through the Use of Non-Rigid Registration," in *Proceedings of the 16th International Symposium on Robotics Research (ISRR)*, 2013.

[3] A. X. Lee, S. H. Huang, D. Hadfield-Menell, E. Tzeng, and P. Abbeel, "Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects," in *27th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A Survey of Robot Learning from Demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009.

[5] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[6] D. Pomerleau, "ALVINN: An Autonomous Land Vehicle In a Neural Network," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed. Morgan Kaufmann, 1989.

[7] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. LeCun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems*, 2005, pp. 739–746.

[8] N. Ratliff, J. A. Bagnell, and S. S. Srinivasa, "Imitation learning for locomotion and manipulation," in *IEEE-RAS International Conference on Humanoid Robots*, 2007.

[9] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. D. Bagnell, and M. Hebert, "Learning Monocular Reactive UAV Control in Cluttered Natural Environments," in *IEEE International Conference on Robotics and Automation*. IEEE, March 2013.

[10] 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*, vol. 37, no. 2, pp. 286–298, April 2007.

[11] S. Calinon, F. D'halluin, D. Caldwell, and A. 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*, Dec 2009, pp. 582–588.

[12] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum Margin Planning," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 729–736.

[13] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[14] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, 2000, pp. 663–670.

[15] K. Dvijotham and E. Todorov, "Inverse Optimal Control with Linearly-Solvable MDPs," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

[16] F. Lamiraux and L. E. Kavraki, "Planning Paths for Elastic Objects Under Manipulation Constraints," *International Journal of Robotics Research*, vol. 20, pp. 188–208, 2001.

[17] T. Wada, S. Hirai, H. Mori, and S. Kawamura, "Robust Manipulation of Deformable Objects Using Model Based Technique," in *Articulated Motion and Deformable Objects*, ser. Lecture Notes in Computer Science, H.-H. Nagel and F. Perales Lpez, Eds. Springer Berlin Heidelberg, 2000, vol. 1899, pp. 1–14.

[18] A. M. Howard and G. A. Bekey, "Intelligent Learning for Deformable Object Manipulation," *Autonomous Robots*, vol. 10, no. 1, pp. 51–58, 2000.

[19] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, "Knot planning from observation." in *Proceedings of the International Conference on Robotics and Automation (ICRA)*. IEEE, 2003, pp. 3887–3892.

[20] J. Takamatsu, T. Morita, K. Ogawara, H. Kimura, and K. Ikeuchi, "Representation for Knot-tying Tasks," *Trans. Rob.*, vol. 22, no. 1, pp. 65–78, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1109/TRO.2005.855988

[21] M. Moll and L. E. Kavraki, "Path planning for deformable linear objects," *IEEE Transactions on Robotics*, vol. 22, pp. 625–636, 2006.

[22] M. Saha, P. Isto, and J.-C. Latombe, "Motion Planning for Robotic Manipulation of Deformable Linear Objects," in *Experimental Robotics*, ser. Springer Tracts in Advanced Robotics, O. Khatib, V. Kumar, and D. Rus, Eds. Springer Berlin Heidelberg, 2008, vol. 39, pp. 23–32.

[23] H. Wakamatsu, E. Arai, and S. Hirai, "Knotting/Unknotting Manipulation of Deformable Linear Objects." *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 371–395, 2006.

[24] M. Bell, "Flexible Object Manipulation," Ph.D. dissertation, Dartmouth College, Hanover, NH, USA, 2010.

[25] G. Konidaris, S. Kuindersma, R. Grupen, and A. S. Barreto, "Constructing skill trees for reinforcement learning agents from demonstration trajectories," in *Advances in neural information processing systems*, 2010, pp. 1162–1170.

[26] G. Neumann and W. Maass, "Learning complex motions by sequencing simpler motion templates," in *In Proceedings of the 26th International Conference on Machine Learning*, 2009.

[27] F. Stulp, E. Theodorou, and S. Schaal, "Reinforcement Learning With Sequences of Motion Primitives for Robust Manipulation," *Robotics, IEEE Transactions on*, vol. 28, no. 6, pp. 1360–1370, Dec 2012.

[28] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, vol. 1. IEEE, 1995, pp. 560–564.

[29] P. J. Schweitzer and A. Seidmann, "Generalized polynomial approximations in Markovian decision processes," *Journal of mathematical analysis and applications*, vol. 110, no. 2, pp. 568–582, 1985.

[30] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[31] H. Miyamoto and M. Kawato, "A tennis serve and upswing learning robot based on bi-directional theory." *Neural Networks*, vol. 11, no. 7-8, pp. 1331–1344, 1998.

[32] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and Representation of 3D Objects with Radial Basis Functions," in *Computer Graphics (SIGGRAPH 01 Conf. Proc.), pages 6776. ACM SIGGRAPH*. Springer, 2001, pp. 67–76.

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

[34] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding*, vol. 89, no. 2-3, pp. 114–141, 2003.

[35] E. Coumans, "Bullet Physics," Jan 2014. [Online]. Available: http://code.google.com/p/bullet/