Learning from Demonstrations Through the Use of Non-Rigid Registration

John Schulman and Jonathan Ho and Cameron Lee and Pieter Abbeel

Abstract We consider the problem of teaching robots by demonstration how to perform manipulation tasks, in which the geometry (including size, shape, and pose) of the relevant objects varies from trial to trial. We present a method, which we call *trajectory transfer*, for adapting a demonstrated trajectory from the geometry at training time to the geometry at test time. Trajectory transfer is based on non-rigid registration, which computes a smooth transformation from the training scene onto the testing scene. We then show how to perform a multi-step task by repeatedly looking up the nearest demonstration and then applying trajectory transfer. As our main experimental validation, we enable a PR2 robot to autonomously tie five different types of knots in rope.

1 Introduction

This paper is concerned with teaching robots to perform manipulation tasks by demonstration. In other words, a human performs the task one or more times (by teleoperation or directly guiding the robot's end-effector), and a learning algorithm extracts the essence of these demonstrations so the robot can perform the task autonomously under different starting conditions.

Our main running example is tying knots in rope. Knot tying is required in several tasks of practical importance, including tying fasteners around wire bundles (common in aerospace applications) and surgical suturing. While it is our running example, our method is not specific to knot tying and we also experimentally illustrate its capabilities in folding clothing and tasks involving household objects.

The procedure for tying a given type of knot can be broken into several segments based on grab and release events. For example, one segment might be to grab the end

J. Schulman, J. Ho, C. Lee, and P. Abbeel are in the Department of Electrical Engineering and Computer Sciences at UC Berkeley

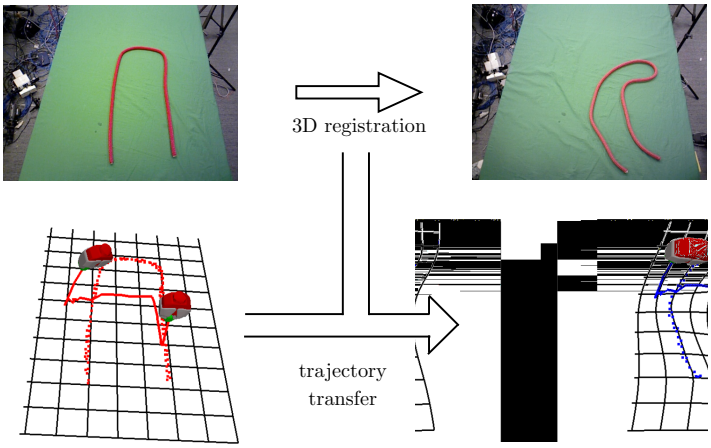

Fig. 1 Trajectory transfer as applied to the first stage (segment) of the overhand knot procedure. Top left: robot's first view of rope at training time. Top right: robot's first view of rope at testing time. Bottom left: point cloud of rope with demonstrated trajectory overlaid, along with x-y coordinate grid. Bottom right: point cloud of rope with warped trajectory generated by our algorithm, along with warped coordinate grid. Note that this trajectory is the first step of a multi-step procedure, and the warping will be performed at least two more times for this knot.

of the rope, move it to form a loop, and then release it. For each of these segments, the end-effector trajectory ought to depend on the geometry of the rope in some way that must be inferred by the algorithm. This paper presents a non-parametric way to learn from examples how the trajectory depends on the geometry of the initial state.

Our main contribution is to show how non-rigid registration can be used to adapt a demonstrated trajectory to a new situation. The problem of generalizing a single demonstration can be described as follows. Suppose that at training time, $\text{STATE}_{\text{train}}$ is the initial state of the rope (represented as an unorganized point cloud) and $\text{TRAJ}_{\text{train}}$ is the trajectory applied to that state by the demonstrator. At test time, the robot is presented with a new state $\text{STATE}_{\text{test}}$ and must generate a new trajectory $\text{TRAJ}_{\text{test}}$ based on the triple $(\text{STATE}_{\text{train}}, \text{TRAJ}_{\text{train}}, \text{STATE}_{\text{test}})$.

Our procedure, which we call *trajectory transfer*, is summarized as follows. We register $\text{STATE}_{\text{train}}$ to $\text{STATE}_{\text{test}}$, obtaining a nonrigid transformation $\mathbf{f} : \mathbb{R}^3 \to \mathbb{R}^3$, and then we apply $\mathbf{f}$ to $\text{TRAJ}_{\text{train}}$ to obtain the new trajectory $\text{TRAJ}_{\text{test}}$. See Figure 1 for illustration. Trajectory transfer enables the robot to generalize from a single demonstration to a much larger part of the state space. If multiple demonstrations are available, one can look up the demonstration with the closest initial state and then use trajectory transfer.

Our second contribution is a method for completing multi-step tasks robustly using a large number of demonstrations of the task. The method is simple: we repeatedly look up the nearest neighbor, using the registration cost as the measure of distance, and then apply trajectory transfer. This simple procedure can be used as a complete policy that enables the robot to complete a multi-step task, eliminating the

need to program a custom state machine. This policy recovers from failures, assuming that the demonstrator has provided examples of the failure states along with the corrective motions that get out of them.

The main contributions of this work can be summarized as follows:

This is the first work to use non-rigid registration to adapt end-effector trajectories to a new scene; we use a non-rigid registration formulation that is well-suited to this task.
We describe how to use optimization-based motion planning to optimally execute the transferred gripper trajectories.
We present experiments on knot tying and other multi-step tasks, where we repeatedly use the registration cost to choose the nearest demonstration.

2 Related work

Nonrigid registration has been used in other fields to transfer information from one geometric entity to another. In medical image analysis, a patient's image is commonly registered to an atlas to locate anatomical structures [1]. In 3D modeling, it has been used to fill in missing parts of scans [2]. In computer vision, nonrigid registration has been used for object recognition and handwriting recognition [3]. Also in vision, using nearest-neighbor based techniques (not involving registration) to transfer various sorts of metadata from one image onto another has had some recent success [4, 5, 6].

Learning from demonstrations, also known as programming by demonstration, has always been a topic of interest in robotics; see [7] for a review. Calinon, Billard, and collaborators [8, 9] have advanced one line of work that is similar in motivation to ours, in that they develop a learning method to perform manipulation tasks under varying initial conditions. Their approach is based on learning a mixture of Gaussians to encode the joint distribution of the robot trajectory and the environment state variables, so that by conditioning on the environment state, they can infer the appropriate trajectory. Ye and Alterovitz have augmented this approach with a repair stage that does motion planning around the learned trajectory to avoid obstacles [10].

The approach of Calinon and Billard assumes access to a featurization of the environment, since regression requires an input vector of fixed dimensionality. Their approach is most applicable in situations when the learned trajectory depends on a few landmarks in the environment that can be reliably detected. Their approach is not applicable to knot tying, where there is no fixed-length vector of landmarks that can be extracted from every rope configuration. In contrast, our approach operates directly on point clouds—the outputs of our sensor hardware—so it is suitable for this task, and it can be applied to a new task without developing a new vision system.

Rope manipulation and knot tying have been a subject of robotics research for almost three decades [11]. Researchers have addressed the problem with motion planning [12, 13], learning from observation (using knot theoretic representations) [14],

robotic hands with tactile feedback [15], and fixtures that enable robust open-loop execution [16]. Note that our work is targeted at general manipulation tasks and is not specialized for the knot tying problem.

3 Generalizing from one example: trajectory transfer

This section describes our method for generalizing from a single demonstrated trajectory. The problem is follows: at training time, the initial state is $\text{STATE}_{\text{train}}$, and the demonstrator provides a trajectory $\text{TRAJ}_{\text{train}}$. At testing time, the robot is presented with a new state $\text{STATE}_{\text{test}}$ and must generate an appropriate trajectory $\text{TRAJ}_{\text{test}}$.

First we'll review non-rigid registration, which lies at the core of our approach. Non-rigid registration finds a mapping from a "source" geometry to a "target" geometry. In our application, the source geometry is $\text{STATE}_{\text{train}}$, and the target geometry is $\text{STATE}_{\text{test}}$.

3.1 Non-rigid registration

3.1.1 Registration with known correspondences

First let us suppose that the source geometry consists of K landmark points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_K$ and the target geometry consists of K corresponding landmark points $\mathbf{p}'_1, \mathbf{p}'_2, \ldots, \mathbf{p}'_K$. Then the registration problem is to compute a function $\mathbf{f} : \mathbb{R}^3 \; ! \; \mathbb{R}^3$ that maps each source point to its corresponding target point, which can be quantified as the optimization problem

$$\underset{\mathbf{f}}{\text{minimize}} \left\{ \text{REGULARIZER}(\mathbf{f}) + \sum_k \left\| \mathbf{f}(\mathbf{p}_k) \quad \mathbf{p}'_k \right\|^2 \right\}. \tag{1}$$

The regularizer encourages the function to be smooth, at the expense of increasing the norm of the residuals $\left\| \mathbf{p}'_k \quad \mathbf{f}(\mathbf{p}_k) \right\|$.

We'll use a particular regularizer: the thin plate spline functional,

$$\text{REGULARIZER}(\mathbf{f}) = \int d\mathbf{x} \sum_{i2\{x,y,z\}} \left\| \mathrm{D}^2 f_i(\mathbf{x}) \right\|^2_{\text{Frob}} \tag{2}$$

where $k\,k_{\text{Frob}}$ refers to the Frobenius norm, and i indexes over the spatial dimensions of the range of $\mathbf{f}$. The thin plate spline regularizer (2) encourages $\mathbf{f}$ to be globally smooth and assigns zero cost to affine functions. It has been observed in other fields (e.g., [17]) that thin plate splines extrapolate well on spatial data; extrapolation is

important for our application. As shown in [18], Equation (1) can be analytically solved efficiently; details are provided in Appendix 9.1.

3.1.2 Registration without known correspondences

In many problems of interest we are not given corresponding landmarks, rather we have two unorganized point clouds $\mathbf{P} = f\mathbf{p}_1, \ldots, \mathbf{p}_M g$ and $\mathbf{P}^0 = f\mathbf{p}'_1, \ldots \mathbf{p}'_N g$ and we want to find a transformation $\mathbf{f}$ so that $f(\mathbf{P}) = f\mathbf{f}(\mathbf{p}_1), \ldots, \mathbf{f}(\mathbf{p}_M) g$ is similar to $\mathbf{P}^0$ in some sense. We use a modification of the TPS-RPM algorithm of Chui and Ragnarajan [19], which alternates between the following steps (1) estimate correspondences between source and target point clouds clouds, and (2) fit a thin plate spline transformation based on these correspondences. Details are provided in Appendix 9.2.

3.2 Trajectory transfer procedure

Our trajectory transfer procedure consists of three steps:

Step 1: Find a transformation f from the training scene to the test scene. Suppose $\mathbf{P}_{\text{train}}$ and $\mathbf{P}_{\text{test}}$ are the point clouds of the manipulated object in the training and test scene, respectively. We perform non-rigid registration as described in Section 3.1.2 to obtain the transformation $\mathbf{f}$ that maps $\mathbf{P}_{\text{train}}$ onto $\mathbf{P}_{\text{test}}$.

Step 2: Apply transformation f to the demonstrated end-effector trajectory. Suppose the demonstrated end-effector trajectory is given by a series of poses $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_T$, where each pose $\mathbf{T}_t$ consists of a position $\mathbf{p}_t$ and an orientation $\mathbf{R}_t$.

We transform the positions and orientations as follows, to adapt the trajectory to the test situation:

$$\mathbf{p}_t \;!\; \mathbf{f}(\mathbf{p}_t) \tag{3}$$

$$\mathbf{R}_t \;!\; \text{orth}\left(\mathbf{J}_{\mathbf{f}}(\mathbf{p}_t)\mathbf{R}_t\right). \tag{4}$$

Here, $\mathbf{J}_{\mathbf{f}}(\mathbf{p})$ is the 3 3 Jacobian matrix of $\mathbf{f}$ evaluated at $\mathbf{p}$,

$$\mathbf{J}_{\mathbf{f}} = \begin{pmatrix} \partial f_x/\partial x & \partial f_x/\partial y & \partial f_x/\partial z \\ \partial f_y/\partial x & \partial f_y/\partial y & \partial f_y/\partial z \\ \partial f_z/\partial x & \partial f_z/\partial y & \partial f_z/\partial z \end{pmatrix}, \tag{5}$$

and orth( ) is a function that orthonormalizes a 3 3 matrix (e.g. using the SVD).

Equation (3) says that we apply the warping function $\mathbf{f}$ to all of the positions. As for rotations, the natural way to transform a vector $\mathbf{v}$ at a point $\mathbf{p}$ through a function $\mathbf{f}$ is to multiply it by $\mathbf{J}_{\mathbf{f}}(\mathbf{p})$, the Jacobian[2]. Equation (4) applies this transformation

[2] In differential geometry, given a mapping $\mathbf{f}$ between two manifolds M and N, the so-called pushforward maps the tangent space $T_{\mathbf{p}}$ at $\mathbf{p}$ 2 M to the tangent space $T_{\mathbf{f}(\mathbf{p})}$ at $\mathbf{f}(\mathbf{p})$ 2 N. In terms of coordinates, it multiplies a vector $\mathbf{v}$ 2 $T_{\mathbf{p}}$ by the Jacobian of the transformation [20].

to the x, y, and z axes of the gripper, and then orthogonalizes the resulting basis so it corresponds to a gripper pose.

Note that if $\mathbf{f}$ happens to be a rigid transformation $\mathbf{T_f}$, then our method simply left-multiplies each end-effector pose $\mathbf{T}_t$ by it

$$\mathbf{T}_t \;!\; \mathbf{T_f}\mathbf{T}_t. \tag{6}$$

Step 3: Convert the end-effector trajectory into a joint trajectory. The simplest approach would be to use inverse kinematics. We found that this approach was not sufficient because there is often no continuous and collision-free trajectory that achieves the desired end-effector trajectory, so we need to compromise. To enable the robot to follow the trajectory as closely as possible while satisfying constraints, we formulate the following optimization problem on the joint trajectory $\boldsymbol{\theta}_{1:T}$:

$$\underset{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_T}{\text{minimize}} \left[\sum_{t=1}^{T-1} k\boldsymbol{\theta}_{t+1} \quad \boldsymbol{\theta}_t k^2 + \mu \sum_{t=1}^{T} \left\| \text{err}\left(\tilde{\mathbf{T}}_t^{\;1} \quad \text{fk}(\boldsymbol{\theta}_t)\right) \right\|_{\ell_1} \right]$$

subject to

No collisions, with safety margin d_{safe}

$\boldsymbol{\theta}_{\min} \quad \boldsymbol{\theta}_{1:T} \quad \boldsymbol{\theta}_{\max}$ (Joint limits)

Here, $\tilde{\mathbf{T}}_t$ is the desired end-effector pose at time t, fk( ) indicates the robot's forward kinematics function applied to $\boldsymbol{\theta}_t$, and μ is a scalar parameter. err( ) is an error function that maps a pose in $SE(3)$ to an error vector in $\mathbb{R}^6$. In particular, after decomposing a pose $\mathbf{T}$ into translation $\mathbf{p}$ and quaternion rotation $\mathbf{q}$, the error vector is simply given by $(p_x, p_y, p_z, q_x, q_y, q_z)$.

We solve this problem using the trajectory optimization method from [21]. We initialize with the joint trajectory from the demonstration, which is in roughly the right part of configuration space. In our experiments, this initialization strategy led to finding good locally-optimal trajectories.

We will illustrate the trajectory transfer procedure with a two-dimensional toy example, where the task is to draw a two-dimensional curve through four guide-points. Note that this example merely illustrates the trajectory of positions, not orientations. The left image of Figure 2 shows the training situation: environment shown in solid lines, gripper tip trajectory shown as a dotted line, coordinate grid lines shown as thin solid lines. The right image shows the test situation for which we want to predict a good gripper trajectory. The registered points are the four corners. First, we use the method of thin plate splines to find a function that maps the four corners of the square in the training situation to the four vertices of the new quadrilateral. Then we apply the nonlinear transformation $\mathbf{f}$ to the demonstrated path to obtain a new path (dotted line), which has the same topological characteristics. The warped grid lines are shown.

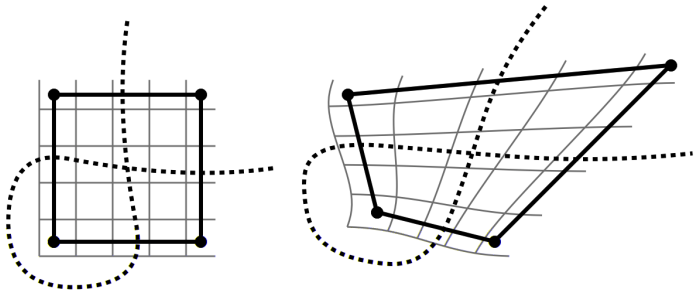

Fig. 2 Illustration of trajectory transfer procedure on a cartoon 2-D example. Left: training situation. Right: testing situation.

3.3 Some intuition on conditions under which trajectory transfer is likely to succeed

3.3.1 Cost function invariance

Our trajectory transfer procedure can be justified by an invariance assumption, which relates it to cost function learning. In inverse optimal control, one learns a cost function $L(\text{STATE}, \text{TRAJ})$ assuming that TRAJ is generated according to $\text{TRAJ} = \arg\min_{\text{TRAJ}} L(\text{STATE}, \text{TRAJ})$. Probability density function estimation is closely related—here, L is the negative log-likelihood. Our trajectory transfer procedure can be justified by assuming that there is a class of smooth transformations $\mathbf{f}$ with the following property:

$$L(\text{STATE}, \text{TRAJ}) = L(\mathbf{f}(\text{STATE}), \mathbf{f}(\text{TRAJ})) \tag{7}$$

Here, $\mathbf{f}(\text{TRAJ})$ means transforming the trajectory as described in Section 3.2.

Given that the demonstration trajectory has low cost, and $\mathbf{f}$ transforms $\text{STATE}_{\text{train}}$ into $\text{STATE}_{\text{test}}$, it follows that trajectory transfer produces a low-cost trajectory:

$$\begin{aligned} L(\text{STATE}_{\text{test}}, \mathbf{f}(\text{TRAJ}_{\text{train}})) \quad & L(\mathbf{f}(\text{STATE}_{\text{train}}), \mathbf{f}(\text{TRAJ}_{\text{train}})) \\ & = L(\text{STATE}_{\text{train}}, \text{TRAJ}_{\text{train}}). \end{aligned} \tag{8}$$

Equation (7) is a strong assumption and defines L on a large part of the state space using a single $(\text{STATE}, \text{TRAJ})$ pair. That said, one can imagine situations where Equation (7) does not hold, even for a rigid transformation $\mathbf{f}$—for example, when absolute orientation of the robot's end-effector matters.

3.3.2 Dynamics invariance

An alternative perspective is that trajectory transfer works in scenarios where the dynamics of the system are approximately invariant—more properly, *covariant*—under sufficiently smooth coordinate transformations. Suppose that the effect of applying the trajectory TRAJ is given by the propagator Π_{TRAJ}, so that

$$\Pi_{\text{TRAJ}}(\text{STATE}^t) = \text{STATE}^{t+1} \tag{9}$$

The dynamics are defined to be covariant under the transformation **f** if and only if

$$\mathbf{f}(\Pi_{\text{TRAJ}}(\text{STATE}^t)) = \Pi_{\mathbf{f}(\text{TRAJ})}(f(\text{STATE}^t)) \tag{10}$$

which is illustrated by the following commutative diagram

$$\begin{array}{ccc} f\text{STATE}^t_{\text{train}}g & \overset{\Pi_{\text{TRAJ}}}{!} & \{\text{STATE}^{t+1}_{\text{train}}\} \\ \mathbf{f}\downarrow & & \downarrow\mathbf{f} \\ f\text{STATE}^t_{\text{test}}g & \overset{\Pi_{\mathbf{f}(\text{TRAJ})}}{!} & \{\text{STATE}^{t+1}_{\text{test}}\} \end{array} \tag{11}$$

Let G denote a goal set, i.e., a set of desirable final states. Suppose that the assumptions hold:

1. $\mathbf{f}(\text{STATE}^t_{\text{train}}) = \text{STATE}^t_{\text{test}}$, i.e., **f** transforms $\text{STATE}_{\text{train}}$ into $\text{STATE}_{\text{test}}$.
2. $\text{STATE}^{t+1}_{\text{train}}$ *2* G, i.e., the demonstration ends up in the goal state.
3. $\mathbf{f}(G) = G$, i.e., the goal set is preserved by **f**
4. Equation (10) holds, i.e., the dynamics are covariant under **f**.

Then by applying the trajectory $\mathbf{f}(\text{TRAJ})$ to $\text{STATE}^t_{\text{test}}$, we obtain $\mathbf{f}(\text{STATE}^{t+1}_{\text{train}})$ *2* G, i.e., the final state is in the goal set.

In the knot tying domain, the goal set is defined topologically, and a transformation **f** will preserve it, provided that it is a homeomorphism (i.e., a continuous function whose inverse is continuous), justifying assumption 3. Assumption 4 holds approximately in this domain; we have checked this qualitatively by comparing the final states of the rope at training and test time.

4 Generalizing from many examples: nearest neighbor method

Intuitively, if $\text{STATE}_{\text{train}}$ is very close to $\text{STATE}_{\text{test}}$, then trajectory transfer is likely to work. Given a collection of demonstrations, with initial states $\text{STATE}_1, \text{STATE}_2, \ldots, \text{STATE}_K$, we can select the one that is closest to $\text{STATE}_{\text{test}}$, and use that one for trajectory transfer:

$$\text{STATE}_i = \arg \min_{i2\{1,2,\ldots,K\}} \text{DISTANCE}(\text{STATE}_i, \text{STATE}_{\text{test}}). \tag{12}$$

One crucial question is how to measure distance. One natural distance measure of geometric similarity is the bidirectional fitting cost used in our registration procedure.

$$\text{REGULARIZER}(\mathbf{f}) + \sum_k \left\| \mathbf{f}(\mathbf{p}_k) \quad \mathbf{p}_k^0 \right\|^2$$
$$+\text{REGULARIZER}(\mathbf{g}) + \sum_k \left\| \mathbf{g}(\mathbf{p}_k^0) \quad \mathbf{p}_k \right\|^2$$

We found that this distance function agreed very well with our intuition about which states are closer and appropriate for trajectory transfer.

We can use this distance function to define a simple policy for completing multi-step tasks. The policy loops through the following steps until task completion:

1. Acquire a new observation $\text{STATE}_{\text{test}}$.
2. Choose the nearest demonstration using the registration cost, as in Equation (12).
3. Apply trajectory transfer to obtain a new trajectory $\text{TRAJ}_{\text{test}}$.
4. Execute $\text{TRAJ}_{\text{test}}$ on the robot.

This policy enables the robot to recover from errors, provided that the demonstrator has provided a demonstration starting from the failure state.

5 Experiments with rope

Fig. 3 Robot executing knot ties. Top row to bottom row: figure-eight knot, double-overhand knot, square knot, and clove hitch. Figure 1 already illustrated the overhand knot.

We enabled the PR2 to autonomously tie five different types of knot: overhand, figure-eight, double-overhand, square knot, and clove hitch (around a pole). The

latter four knots are shown at several stages of execution in Figure 3. Videos are available at `http://rll.berkeley.edu/isrr2013lfd`.

Teaching was performed kinesthetically, by guiding the robot's arms through the motion with its controllers off. The demonstrator noted *look*, *start*, and *stop* times, which indicated when to acquire a point cloud for the initial configuration and when the motion begins and ends. Point clouds were acquired from an RGBD camera mounted on the PR2 head. We extracted the rope points using color filtering (the rope was red or white, and the tablecloth was green).

We first performed an exploratory experiment with one demonstration per knot tie, where we measured how robust the procedure was under perturbations of the rope state of the demonstrations. We randomly perturbed the rope as follows. First the rope was laid out in the initial configuration from the demonstration, Then each of five points, uniformly spaced along the length of the rope, was manually dragged by a distance d_{pert} in a random direction. Then, we executed the knot-tying procedure and judged its success in tying the knot.

We performed five trials for each type of knot. The rates of success of these knot ties are shown in Table 1. As expected, the failure rate increases at higher d_{pert}, where the test situation is further away from the training situation. Two of the common failure modes were (1) the rope would end up in a previously unseen crossing state, often due to a small difference in its position; (2) the robot would grab the wrong piece of rope, or two pieces instead of one.

Knot type	Segments	Success ($d_{pert} = 3\,\text{cm}$)	($d_{pert} = 10\,\text{cm}$)
Overhand	3	5/5	4/5
Figure-eight	4	5/5	1/5
Dbl-overhand	5	3/5	3/5
Square	6	5/5	3/5
Clove hitch	4	1/5	0/5

Table 1 Results for generalizing from a single demonstration: knot-tie success vs. size of random perturbations.

The above results suggest that when starting from a state that is very close to a demonstration, the success rate is high, at least for the easier knots. We hypothesized that if we collected enough demonstrations, every rope state of interested would be near some demonstration's initial state, so we would achieve that high level of performance over the entire state space by doing nearest neighbor lookups.

To test this hypothesis and explore how our algorithm performs in the many-demonstrations regime, we collected a large number of demonstrations of the overhand knot. The dataset contains a total of 148 trajectory segments, which include 36 demonstrations of the standard 3-step sequence and 40 additional segments starting from failure states—states that prevent the standard knot tying procedure from proceeding and require corrective actions.

Our initial (qualitative) results are promising: we find that the nearest-neighbor policy from Section 4 is able to successfully tie knots from a much larger set of initial configurations than was possible with a single demonstration, and it is also able

to choose corrective movements to recover from the failure states. Two executions are shown in Figure 4.

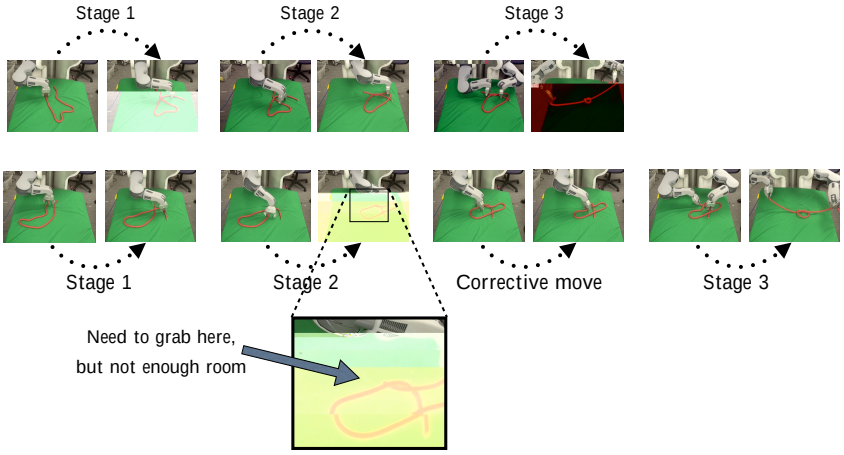

Fig. 4 Two successful executions of an overhand knot, based on our dataset of 148 demonstration segments. The top row shows the usual three-stage procedure. The bottom row shows a situation where an extra step was needed. After stage two, the rope ended up in a difficult state—the end that it needs to grab was too short. However, the nearest neighbor lookup found a demonstration with a very similar starting state and a corrective movement, which (after trajectory transfer) made the end graspable.

6 Experiments on other manipulation tasks

We performed some other experiments to validate that the proposed method can be applied to manipulation tasks other than knot tying. The three tasks considered were folding a T-shirt, picking up a plate using a non-trivial two-arm motion, and opening up a bottle. The former task was executed using three segments, whereas the latter two consisted of one open-loop trajectory segment. We performed these tasks with the same algorithm and code that was used for the knot tying. See Figure 5.

7 Conclusion

We have presented a method for adapting trajectories to new situations, based on non-rigid registration between the geometry of the training scene and the testing scene. This enables us to successfully perform a task under various initial conditions

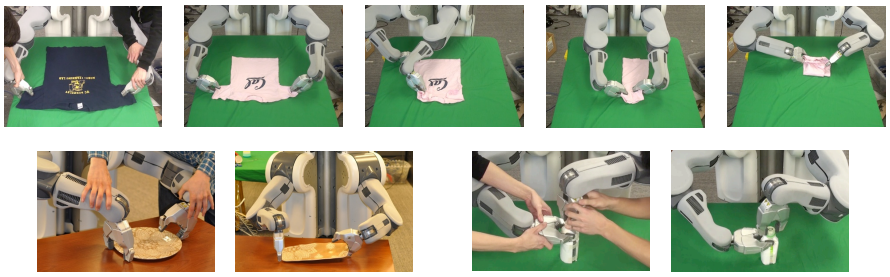

Fig. 5 Top row, left image: recording demonstration on a large T-shirt. Top row, right images: executing shirt folding procedure autonomously on small T-shirt, based on single demonstration on large T-shirt. Bottom row, left: demonstration and autonomous execution of plate pickup. Round plate from training was registered to rectangular plate at test time. Bottom row, right: demonstration and autonomous execution of bottle opening. The bottle used at test time had a different size and shape from the one used for training.

based on a single demonstration. Our method is justified by invariance assumptions as discussed in Section 3.3.

The non-rigid registration metric can also be used to find the nearest demonstration, when multiple demonstrations have been provided. This simple scheme enables our algorithm to successfully perform challenging multi-step manipulation tasks.

8 Source code

Complete source code and tutorials for our software are available at `http://rll.berkeley.edu/rapprentice`. Other supplementary material for this paper is available at `http://rll.berkeley.edu/isrr2013lfd`.

9 Acknowledgements

This research has been funded in part by the Intel Science and Technology Center on Embedded Computing, by an AFOSR YIP grant, by a Sloan Fellowship, by a SURF Rose Hills Fellowship, and by NSF under award 1227536. We thank Sachin Patil and Ken Goldberg for their inspiring ideas and valuable advice. We thank Henry Lu and Robbie Gleichman for substantial contributions to the experiments. We would like to acknowledge the following open-source software that we used: OpenRAVE [22], ROS [23], PCL [24], OpenCV [25].

References

1. J. V. Hajnal and D. L. Hill, *Medical image registration*. CRC press, 2010.
2. M. Pauly, N. J. Mitra, J. Giesen, M. H. Gross, and L. J. Guibas, "Example-based 3D scan completion.," in *Symposium on Geometry Processing*, pp. 23–32, 2005.
3. S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 4, pp. 509–522, 2002.
4. A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 341–346, ACM, 2001.
5. T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 89–96, IEEE, 2011.
6. C. Liu, J. Yuen, and A. Torralba, "Nonparametric scene parsing: Label transfer via dense scene alignment," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1972–1979, IEEE, 2009.
7. S. Calinon, "Robot programming by demonstration," in *Springer Handbook of Robotics*, pp. 1371–1394, Springer, 2008.
8. 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, 2007.
9. 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*, pp. 582–588, Ieee, 2009.
10. G. Ye and R. Alterovitz, "Demonstration-guided motion planning," in *International Symposium on Robotics Research*, 2011.
11. H. Inoue and M. Inaba, "Hand-eye coordination in rope handling," in *Robotics Research: The First International Symposium*, vol. 1, pp. 163–174, 1985.
12. 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.
13. M. Saha, P. Isto, and J. Latombe, "Motion planning for robotic manipulation of deformable linear objects," in *Int. Symp. On Experimental Robotics (ISER)*, 2006.
14. T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, "Knot planning from observation," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 3, pp. 3887–3892, IEEE, 2003.
15. Y. Yamakawa, Y. Namiki, M. Ishikawa, and M. Shimojo, "One-handed knotting of a flexible rope with a high-speed multifingered hand having tactile sensors," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 703–708, IEEE, 2007.
16. M. Bell, *Flexible object manipulation*. PhD thesis, Dartmouth College, Hanover, New Hampshire, 2010.
17. 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 *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 67–76, ACM, 2001.
18. G. Wahba, *Spline models for observational data*, vol. 59. Society for Industrial Mathematics, 1990.
19. H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding*, vol. 89, no. 2, pp. 114–141, 2003.
20. R. Abraham, J. Marsden, T. Ratiu, and R. Cushman, *Foundations of mechanics*. Benjamin/Cummings Publishing Company, 1978.
21. J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Proc. Robotics: Science and Systems*, 2013.

22. R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, p. 79, 2008.
23. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, 2009.
24. R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, IEEE, 2011.
25. G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

Appendix

This appendix describes our registration procedure, which is based of the TPS-RPM algorithm of Chui and Ragnarajan [19].

9.1 Thin plate splines

The classic method of smoothed thin plate splines [18] minimizes the following cost functional on $f : \mathbb{R}^d \; ! \; \mathbb{R}$:

$$J(f) = \sum_i (y_i \quad f(\mathbf{x}_i))^2 + \lambda \, k f k^2_{\text{tps}} \tag{13}$$

Here, $k f k_{\text{tps}}$ is a measure of curvature or distortion, and is defined as

$$k f k^2_{\text{tps}} = \int d\mathbf{x} \left\| \mathrm{D}^2 f(\mathbf{x}) \right\|^2_{\text{Frob}} \tag{14}$$

where $\mathrm{D}^2 f$ is the matrix of second partial derivatives of f, and $k \, k^2_{\text{Frob}}$ indicates its Frobenius norm, i.e., the sum of squares of its entries. λ is a parameter that controls the tradeoff between smoothness and goodness-of-fit.

Remarkably, the minimizer to the functional in Equation (13) is a finite-dimensional expansion in terms of basis functions, centered around the data points $\mathbf{x}_i$, plus an affine part:

$$f(\mathbf{x}) = \sum_i a_i K(\mathbf{x}_i, \mathbf{x}) + \mathbf{b}^T \mathbf{x} + c \tag{15}$$

where K is the kernel function, and in 3D, $K(r) = \quad r$ (after dropping the irrelevant constant factor.)

In non-rigid registration, one needs to solve for a function $\mathbf{f} : \mathbb{R}^3 \; ! \; \mathbb{R}^3$, rather than a scalar-valued function. We can build a vector-valued function $\mathbf{f}$ by combining three scalar-valued components of the form (15), i.e., $\mathbf{f}(\mathbf{x}) = (f_0(\mathbf{x}), f_1(\mathbf{x}), f_2(\mathbf{x}))^T$. Thus $\mathbf{f}$ has the form

$$\mathbf{f}(\mathbf{x}) = \sum_i \mathbf{a}_i K(\mathbf{x}_i, \mathbf{x}) + \mathbf{B}\mathbf{x} + \mathbf{c} \tag{16}$$

for $\mathbf{a}_i \in \mathbb{R}^3$, $\mathbf{B} \in \mathbb{R}^{3\times 3}$, $\mathbf{c} \in \mathbb{R}^3$. Adding an additional regularization term $r(\mathbf{B})$ on the linear part of the transformation, we obtain the following optimization problem

$$\min_{\mathbf{A},\mathbf{B},\mathbf{c}} \left\| \mathbf{Y} \quad \mathbf{K}\mathbf{A} \quad \mathbf{X}\mathbf{B} \quad \mathbf{1}\mathbf{c}^T \right\|_{\text{Frob}}^2 + \text{trace}(\mathbf{A}^T \mathbf{K} \mathbf{A}) + r(\mathbf{B})$$
$$\text{subject to } \mathbf{X}^T\mathbf{A} = \mathbf{0}_{3\times 3} \text{ and } \mathbf{1}^T\mathbf{A} = \mathbf{0}_{1\times 3} \tag{17}$$

where $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots \end{pmatrix}^T$, $\mathbf{Y} = \begin{pmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots \end{pmatrix}^T$, and $\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots \end{pmatrix}^T$. For certain choices of $r(\mathbf{B})$ This problem completely decouples the three components of $\mathbf{f}$, i.e., we are separately fitting three functions for the separate output dimensions.

We found that the regularization $r(\mathbf{B})$ was necessary because sometimes the transformation is underdetermined in certain dimensions. For our experiments, we used $r(\mathbf{B}) = \text{trace}\left((\mathbf{B} \quad \mathbf{I})^T \mathbf{D} (\mathbf{B} \quad \mathbf{I})\right)$, where $\mathbf{D}$ is a diagonal matrix. With this quadratic regularization term, the optimization problem still can be solved analytically as a least squares problem.

9.2 Iterative Registration Algorithm

This section considers the problem raised in Section 3.1.2 of registering two point clouds where we are not given correspondences between their points. We use a modification of the TPS-RPM algorithm of Chui and Ragnarajan [19], where the main modification is to jointly fit a forward transformation $\mathbf{f}$ and inverse transformation $\mathbf{g}$. The algorithm iterates between two steps: soft assignment between the points, and fitting the pair of thin plate spline transformations $\mathbf{f}, \mathbf{g}$. First, let us assume that we have N source points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and M target points $\mathbf{y}_1, \ldots, \mathbf{y}_M$.

The full optimization problem that we solve is the following

$$\underset{\mathbf{f},\mathbf{g}}{\text{minimize}} \sum_{n=1}^{N} \sum_{m=1}^{M} C_{nm} \left(\|\mathbf{f}(\mathbf{x}_n) \quad \mathbf{y}_m\|^2 + \|\mathbf{g}(\mathbf{y}_m) \quad \mathbf{x}_n\|^2 \right) +$$
$$\text{REGULARIZER}(\mathbf{f}) + \text{REGULARIZER}(\mathbf{g}) \tag{18}$$

such that $\mathbf{C} \in \mathbb{R}^{N\times M}$ is is the unique solution to the following constraints:

$$C_{nm} = s_n t_m \exp\Big(\quad (k\mathbf{y}_m \quad \mathbf{f}(\mathbf{x}_n)k^2 + k\mathbf{x}_n \quad \mathbf{g}(\mathbf{y}_m)k^2))/2\sigma^2\Big),$$

$$n = 1,\ldots,N,\ m = 1,\ldots,M$$

$$\sum_{m=1}^{M} C_{nm} = 1, \quad n = 1,\ldots,N \tag{19}$$

$$\sum_{n=1}^{N} C_{nm} = N/M \quad m = 1,\ldots,M \tag{20}$$

where $\mathbf{s}\ 2\ \mathbb{R}_+^N$ and $\mathbf{t}\ 2\ \mathbb{R}_+^M$ are scaling factors (uniquely determined by constraints (19) and (20)), and σ is a parameter that controls the correspondence difference, which will by systematically varied in the optimization procedure below.

Alternating optimization procedure. Following the Chui and Ragnarajan [19], we alternate between fitting (solving for $\mathbf{f},\mathbf{g}$) and soft assignment (solving for $\mathbf{C}$). Meanwhile, we are exponentially decreasing two parameters: a scale parameter σ that controls the correspondence distance, and the regularization parameter λ for the thin plate spline fitting. These parameters will be decreased exponentially in the series $\sigma_1,\sigma_2,\ldots,\sigma_{\text{NUMITERATIONS}}$ and $\lambda_1,\lambda_2,\ldots,\lambda_{\text{NUMITERATIONS}}$. The algorithm is given below:

Initialize $\mathbf{f},\mathbf{g}$ to the identity
For i = 1 to NUMITERATIONS
  Compute correspondence matrix $\mathbf{C}$
  (using $\mathbf{X},\mathbf{Y},\mathbf{f},\mathbf{g},\sigma_i$)
  Fit forward and inverse transformations $\mathbf{f},\mathbf{g}$
  (using $\mathbf{X},\mathbf{Y},\mathbf{C},\lambda_i$)