2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Dimensionality Reduction for Trajectory Learning from Demonstration

Nik A. Melchior and Reid Simmons

Abstract— **Programming by demonstration is an attractive model for allowing both experts and non-experts to command robots' actions. In this work, we contribute an approach for learning precise reaching trajectories for robotic manipulators. We use dimensionality reduction to smooth the example trajectories and transform their representation to a space more amenable to planning. Key to this approach is the careful selection of neighboring points within and between trajectories. This algorithm is capable of creating efficient, collision-free plans even under typical real-world training conditions such as incomplete sensor coverage and lack of an environment model, without imposing additional requirements upon the user such as constraining the types of example trajectories provided. Experimental results are presented to validate this approach.**

I. Introduction

Precise reaching and manipulation are important skills for robots that operate in real-world environments. Assembly-line robots align pieces of hardware, attach bolts, and weld seams. Humanoids need to precisely grasp and place objects. The motions performed by these robots must often be tediously scripted by a programmer or technician familiar with the capabilities and limitations of the particular robot in use.

This work seeks to make it easier for both experienced and novice robot users to command precise motions by providing examples. The motions are demonstrated by moving a robotic arm in passive mode or through a teleoperation interface. This kinesthetic form of training is intuitive for users, and it avoids the problem of mismatched models when learning from methods such as human motion capture.

Demonstration is also attractive because it allows the human to convey implicitly the location of obstacles (by avoiding them) and non-geometric constraints. Since these robots typically operate in sensor-poor environments, it would be difficult to place a ladar or stereo camera pair in a position capable of observing the entire workspace of a high degree of freedom (DOF) dexterous arm, particularly when accounting for occlusion by the arm itself. Building a precise model of the workspace by hand is a time-consuming and error-prone task. However, knowledge of free space can be inferred from the space occupied by the robot during kinesthetic demonstrations.

Non-geometric constraints are more subtle, but are also conveyed by the demonstrated trajectories. For example, the task may require the robot's end effector to remain in a certain orientation throughout execution, or to avoid a portion of the workspace shared with another robot or human, even if that area is not currently occupied. Both of these types of constraints are conveyed implicitly by the human teacher through demonstrations, and may be incorporated into learned trajectories.

Robotics Institute of Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, USA {nmelchio,reids}@cs.cmu.edu

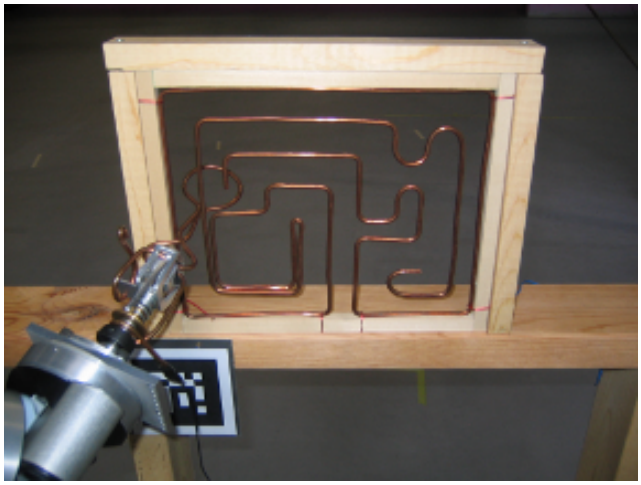


Fig. 1. The experimental platform: a Barrett WAM 7-DOF arm navigating a wire maze.

In this work, we present a method for learning robot manipulator trajectories from expert or novice demonstrations. We contribute a neighbor-selection technique for finding similar sections of demonstrated trajectories without the parameter selection required for previous methods. Using dimensionality reduction, we create a task-specific planning space in which collision-free trajectories may be created, even without a model of the environment in which the robot operates. Figure 1 shows the wire maze used in one set of experiments. Participants were asked to manually move the robot manipulator to navigate the wire maze, producing trajectories such as those in figure 5(a). The robot learner used these examples to produce novel trajectories capable of navigating the maze from a variety of initial conditions.

II. Related Work

The problem of learning trajectories from examples occupies an interesting niche between the traditional fields of motion planning and machine learning. Motion planning algorithms typically choose actions for the robot to execute based on knowledge about the environment gathered by sensors or by some other means. Grid-based planners such as A* and D* [1] or randomized sampling-based planners such as RRT [2] and probabilistic roadmaps [3] all depend on knowledge of free space that the robot is allowed to occupy. Whether they use a binary occupancy grid or a graduated costmap, traditional motion planning techniques

will not operate directly on the examples provided. Without an explicit model of the environment in which the robot operates, obstacles may be conservatively inferred to exist in any region of the workspace not visited by the robot during training.

Traditional machine learning techniques tend to be more appropriate to this domain, but learning can be difficult to generalize properly. A single trajectory of a 7-DOF robotic arm may contain hundreds of points in a configuration space including position, velocity, and perhaps a few other features of individual points. For example, velocities may need to be included in the configuration space if dynamic constraints are to be respected. Providing examples of every area of this configuration space would be too time-consuming, so the learning algorithm must generalize examples correctly. Without knowledge of obstacles, though, simple approaches such as k-nearest neighbor can easily generalize too broadly, and other techniques must be used to correct this [4]. Other work in Reinforcement Learning [5] seeks to correctly generalize training examples as much as possible. Rather than drawing generalizations from individual points, other strategies use short portions of trajectories as their learning primitives. Several approaches use an intermediate representation such as domain-specific primitives [6], basis functions [7], or various types of splines [8], [9], [10] to improve the fit of example trajectories. Recent work using Gaussian Mixture Models has focused on limiting the number of necessary training examples while ensuring confident execution of learned behaviors [11], or ensuring correct behaviors even when online adaptation is required [12]. Another recent work [13] attempts to learn the cost functions implicitly used by the human teacher in generating entire examples.

Unfortunately, these approaches typically lack the greatest strength of the previous category of algorithms: since obstacles are not modelled, no guarantees can be made as to the safety of a planned path. One method for dealing with this problem is to present the planned path to a user in a graphical interface, providing an opportunity to correct or reject the planned path [14], [10]. Unfortunately, this introduces the requirement that the environment be precisely modelled. Another promising approach, provided that collisions are not catastrophic or costly, allows the user to mark portions of generated or example trajectories as undesirable [15] after they are executed. Delson and West [16] introduced an algorithm that, with certain assumptions, including a limit of two dimensions, ensured that learned trajectories would be collision-free. However, their work did not account for redundant manipulators, and imposed the constraint that all example trajectories must be homotopically equivalent. That is, all examples must take the same route between obstacles. While this is not an arduous restriction in simple cases, it may be difficult to ensure homotopic equivalence in environments with more obstacles or in higher dimensional spaces.

Our approach to safety requires a model of the robot, but not the environment. This requirement should be easy to fulfill since robots change far less often than the environments in which they operate, and many identical robots are generally produced with the same hardware configuration. Given the example trajectories, it is straightforward (though computationally expensive) to determine all areas of the workspace that have been occupied by the robot. Next, every grid cell in a discretized representation of the configuration space may be marked as safe if the robot, in that configuration, occupies only areas of the workspace that were occupied during training. The user may also elect to provide a conservative buffer around the robot's position that is also considered safe to occupy.

III. Approach

In this work, we seek to develop an approach that combines the strengths of previous programming by demonstration and planning systems. This system will learn to perform a precise reaching task with a dexterous arm from a variety of initial conditions. The generated trajectories should be guaranteed collision-free in static environments despite limited sensing and no explicit model of the environment. Finally, the system should be intuitive and easy to use, even without training or extensive knowledge of the algorithm used.

Our approach uses dimensionality reduction to transform the example trajectories to an intrinsic embedding suitable for learning the particular task at hand. This retains explicit representation of each of the examples provided while smoothing some noise and jitter, and providing a more convenient domain in which to plan. By combining the example trajectories with a model of the manipulator, the free area in the workspace may be determined. A conservative planner should assume that any space not occupied by the manipulator during training may contain an obstacle. Once the intrinsic task embedding is discovered, a novel plan may be created in the low-dimensional space and transferred, or *lifted*, to the original control space.

We will first examine the strategy and motivation for planning in a reduced-dimensionality space. Next, we present our extension to Isomap [17] for time-series data. Finally, we present an implementation of the complete system on a 7-DOF robotic arm and experimental results are discussed.

A. Dimensionality Reduction

Our primary motivation for dimensionality reduction is to ease the task of planning by discovering an intrinsic embedding for examples of the task. That is, rather than planning arbitrary trajectories through the robot's workspace or configuration space, we create a *task space* in which the desired trajectory is as simple as possible. In our work to date, a two-dimensional task space has been used to represent full 6-DOF trajectories using a redundant 7-DOF manipulator. One dimension represents time, or progress through the task, while the other dimension represents variations between distinct examples.

However, finding an intrinsic low-dimensional representation of the data has other benefits. Dimensionality reduction is typically used in Programming by Demonstration to

deal with the correspondence problem [18] between high-dimensional training data (such as human motion capture) and low-dimensional controls (such as dexterous arm joint angles). When kinesthetic demonstrations are used, the training data is collected in the robot's control space. However, dimensionality reduction is still useful as it accentuates commonality among multiple training examples, and eliminates much of the noise (due to imperfect sensors) and jitter (due to imperfect human motion) that needlessly distinguishes them. Smoothing is achieved because the lower dimensionality space lacks the degrees of freedom to precisely represent every aspect of the example trajectories. Thus, the high frequency noise is eliminated while the features common to all examples are emphasized. This strategy is to be favored over other techniques that smooth trajectories one at a time by removing high-frequency or low magnitude variations. Although neither approach requires domain knowledge for its application, dimensionality reduction is able to preserve features common to many trajectories, even at the same magnitude as the noise, because it operates on all the trajectories at once. For instance, dimensionality reduction smooths out random jitters, but can maintain small "bumps" that are common across trajectories.

One promising technique for dimensionality reduction is Isomap. This algorithm finds a non-linear embedding for data using geodesic distances between nearby points. In essence, it discovers a lower-dimensional manifold embedded in the original space. The input data points lie on (or near) this manifold, so they can be represented in fewer dimensions. Isomap operates by first constructing a neighborhood graph connecting all of the points. A matrix of all-pairs shortest path distances are computed over this neighborhood graph. Finally, the low-dimensional embedding is constructed by applying Multi-Dimensional Scaling (MDS) to the distance matrix. This creates a space in which geodesic (graph-based) distances are preserved.

The original Isomap algorithm was not specificly tailored for time-series data, but it may be applied to points sampled from trajectories. One extension to Isomap, Spatio-Temporal Isomap [19], attempts to exploit the inherent relationships between these points to improve the embedding. ST-Isomap introduces changes to the first two steps of Isomap: construction of the neighborhood graph and computation of the distance matrix. First, it exploits the obvious neighbor relationships inherent in time-series data. Adjacent samples from a single trajectory, which we call *temporal neighbors*, are clearly related, and are thus linked in the neighbor graph. ST-Isomap also attempts to discover what we call *spatio-temporal neighbors*, or neighbors in different trajectories that occur at the same time in the task. For each of these types of neighbors, a tunable parameter is used to reduce the perceived distance between linked points. Although ST-Isomap is able to produce reasonable embeddings for many tasks, we have found that the tunable parameters must be chosen accurately for the nature and scale of the task at hand. In this work, we attempt to refine the neighbor selection mechanism in order to obviate the need for parameter selection and produce a better embedding.

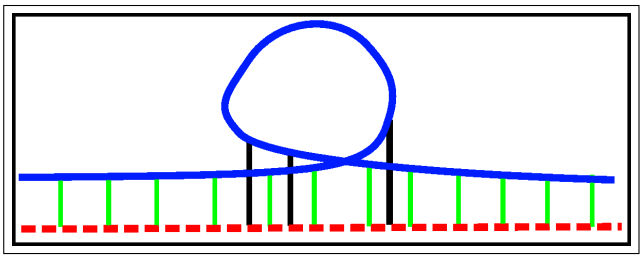


Fig. 2. Regularly spaced neighbor links from the solid (blue) trajectory to the dashed (red) trajectory result in some undesirable links (black).

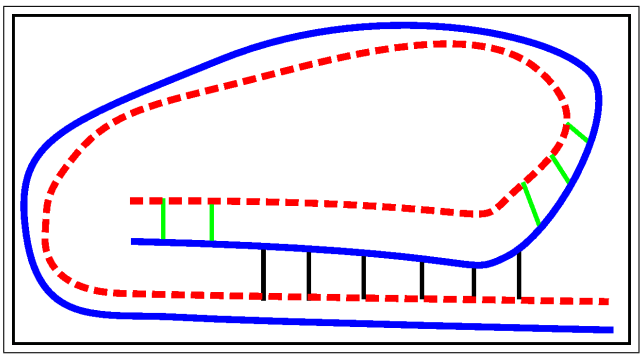


Fig. 3. The solid (blue) trajectory drifts closer to distant sections of the dashed (red) trajectory. Pointwise nearest-neighbor alone is not sufficient to correct this problem.

B. Neighbor Selection

Neighbor selection is the key to discovering an intrinsic task embedding for time-series data. Although humans can typically discern global structure even in noisy collections of points, it is a challenging task for a robot learner. This problem, known as graph or manifold denoising, has been studied extensively for application to Isomap [20] and other graph-based learning algorithms[21], [22], [23]. Time-series data presents specific challenges to this effort, but it also has the advantage that part of the structure of the data is known. Specifically, we retain the temporal links between adjacent points on the same trajectory as discussed above. In this section, we examine the selection of spatio-temporal neighbors between trajectories.

In keeping with the framework established by Isomap, all neighbor links are bidirectional. The typical Euclidean metric is used to calculate the distance between neighboring points, and the geodesic distance between all other pairs is the shortest distance along links in the graph.

When searching for a point's neighbors, individual points are not considered in isolation. Instead, we consider each pair of trajectories separately, and search for subsequences of those trajectories that contain points that match in a roughly pairwise manner. That is, the indices in both subsequences increase in the same direction, and both subsequences contain approximately the same number of points. Since all trajectories are initially subsampled at a uniform distance between adjacent points, this mean that, informally, matching subsequences travel in the same direction.

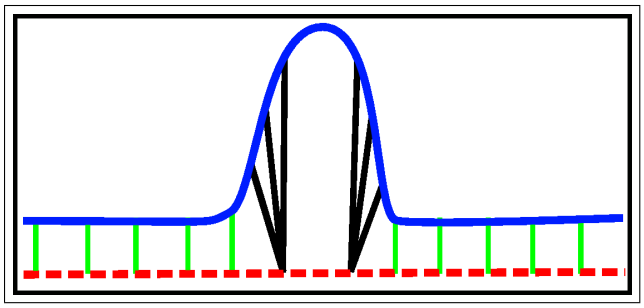
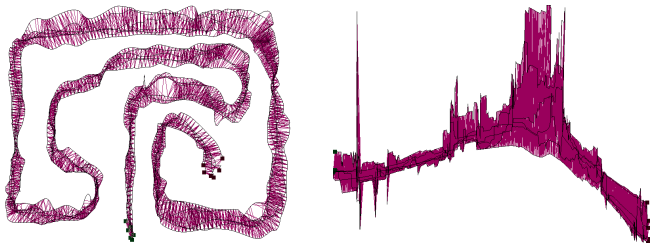


Fig. 4. Many-to-one neighbor links result in a graph with too many links, and thus geodesic distances that are too small. Deviations such as that shown in the solid (blue) trajectory should appear distant in the neighbor graph.



(a) (b)

Fig. 5. A two-dimensional projection of the workspace trajectories for the wire maze task (left) and the two-dimensional embedding of its 7-DOF configuration space (right). Purple lines represent neighbor links.

Noisy links are detected and removed by searching for sequences of neighbor links in one trajectory whose indices in the matching trajectory do not monotonically increase. This ensures that adjacent subsequences in one trajectory are not connected to distant portions of the other trajectory, as may occur when a trajectory contains a loop (see figure 2) or other artifact (see figure 3). Finally, one-to-many neighbor links are not allowed. Only the one-to-one link with the shortest distance is permitted in the final neighbor graph. This restriction ensures that the geodesic distances increase quickly when trajectories deviate from one another (see figure 4). This strategy is illustrated further in the experimental results in section IV.

Additional checks may be incorporated into this strategy to restrict the types of neighbor links formed. For example, given an explicit or implicit environment model, as discussed above, we might check that a motion between every pair of neighbors is safe to execute. If the robot would impact an obstacle, that link may be removed from the graph. This helps ensure that, in the embedding, movement between nearby points is safe.

C. Planning

A robot is clearly capable of performing a demonstrated task by simply repeating example trajectories. However, this is undesirable for several reasons. Even expert robot operators are unlikely to produce perfect examples. Accidental movements, unavoidable jitter, detours, and sensor errors can all contribute to variations between demonstrations. In addition, the robot should have a safe strategy for operation if its position deviates from demonstrations due to sensing or actuation noise, or even due to the variety of initial conditions.

Planning requires a generalization of the provided example trajectories. Extrapolating cannot be safe without additional information about obstacles in the environment, but interpolation is possible if we know which portions of the demonstrated trajectories occur at the same point in the task. We argued in the previous section for the use of global information about trajectories to make this determination. Figure 5(a) shows a neighbor graph for the maze of figure 1. The graph alone is not sufficient to determine an action policy for undemonstrated points in the configuration space, though. Instead, we use the neighborhood information in the graph to create a simple low-dimensional space that facilitates interpolation of demonstrated actions.

MDS, the final step of Isomap, is used to create a two dimensional embedding such as the one shown in figure 5(b). This embedding preserves (as much as possible) the geodesic distances between all pairs of points. Although figure 5(a) illustrates a two-dimensional projection of the Cartesian workspace trajectories of the end effector, planning must occur in the seven-dimensional configuration space. Because the manipulator is redundant, it is necessary to be able to distinguish between different configurations that result in the same end-effector pose. In the embedding of figure 5(b), the starting points of the trajectories (the dark lines) are clustered along the left edge of the image. The trajectories end near the bottom-right of the image. The thinner, purple lines are neighbor links between trajectory points. Although nothing in the algorithm explicitly forces it to be so, the horizontal axis is roughly equivalent to time. This is because MDS selects the dimension with the greatest variance as the first dimension in the embedding. The vertical axis separates trajectories from one another. Vertical spikes represent portions of example trajectories that deviated from neighbor trajectories, similar to the examples in figures 2 and 4. Since the problematic spatio-temporal links pictured there were eliminated, the geodesic distances between trajectories increases significantly, and the trajectories become distant in the embedding.

Our previous work[24] investigated a strategy for planning in embedded spaces such as these. However, with the improved neighbor-selection mechanism presented here, the embedding creates a space in which planning is straightforward. A roughly linear path through this space from left to right, remaining in the area between example trajectories, is one simple strategy for generating novel plans. Future work will focus on other strategies, and their relative merits. For example, planning a path which remains near the densest areas of example trajectories may produce a plan more qualitatively similar to that desired by the user.

Trajectories created in this two-dimensional planning space must be transformed to the configuration space of the robot before they can be executed, but there is no global linear transformation between these spaces. Instead,

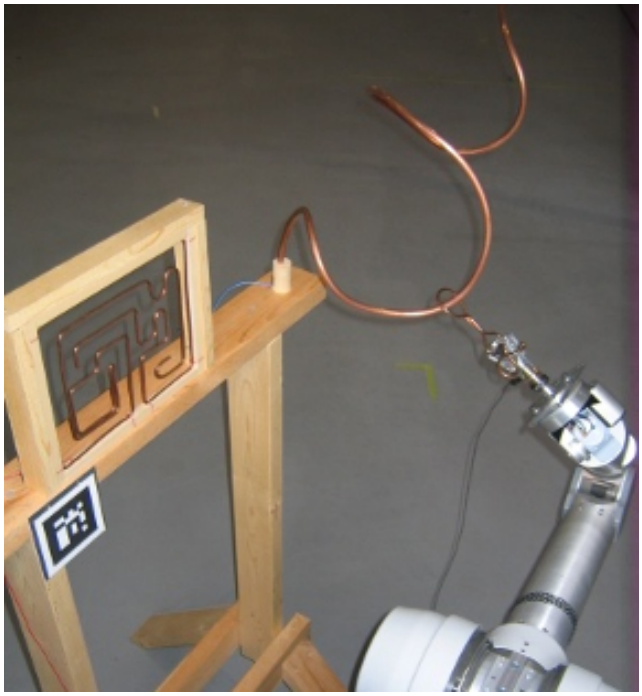


Fig. 6. The WAM manipulator, the wire maze (left) and the tube maze (right). The 2-D barcode in the lower-left is a visual fiducial used to detect the location of the rig relative to the robot. The rig and mazes are not modelled by the learning algorithm.

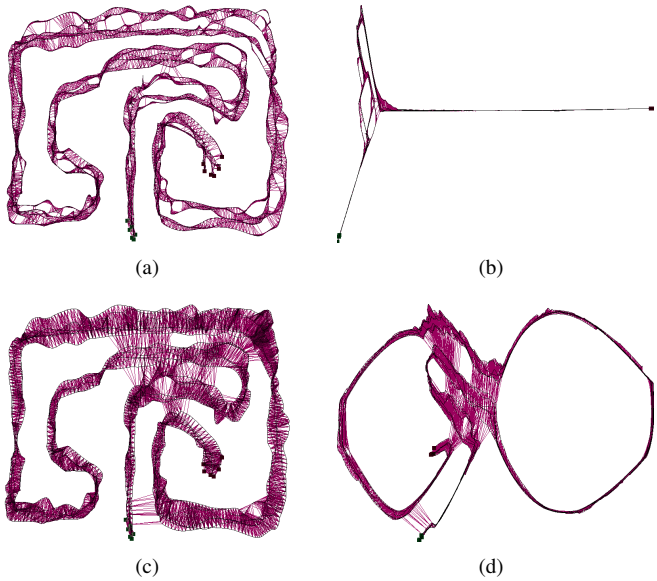


Fig. 7. Neighbors selected by (a) k-NN and (c) ST-Isomap, and their embeddings (b) and (d). Spurious short-circuit neighbor links produce embeddings unusable for planning.

individual points in the planned path must be *lifted* to the original high-dimensional space. This is accomplished using the Delaunay triangulation [25] of the trajectory points embedded in two-dimensions. For any query point in the plane, the unique enclosing triangle is found. The barycentric coordinates of the query point within the triangle are used as weights to interpolate between the points corresponding to the triangle vertices in the original space. It should be noted that the mapping from the planning space to the original space is naturally a one-to-many mapping. However, since the correspondence between points in the planning space and the original points in the higher dimensional space is known, interpolation ensures that the resulting point is in the correct region of the configuration space.

This method is reliable only for query points that lie within the triangulation of the points of the example trajectories. Fortunately, this is precisely the area in which we can be confident executing novel plans. Points outside the triangulation cannot be lifted by interpolating between example points, and thus represent extrapolations outside the demonstrated area of the configuration space. A similar method may be used to map points in the other direction, from the configuration space to the planning space. This mapping is required to query the plan for an action to perform at a given configuration.

IV. Experimental Evaluation

Experiments were conducted using the 7-DOF WAM manipulator shown in figure 6 on two similar tasks. When the arm is operated in gravity-compensation mode, it can be easily moved by hand. Participants were asked to perform kinesthetic demonstrations navigating the two mazes pictured. When the robot's copper end effector contacts the walls of either maze, a buzzer provides auditory feedback. The wire maze on the left is effectively two-dimensional, though some of the rotational axes are relatively unconstrained, allowing additional variation in the demonstrated trajectories. This rotational variation is not strictly necessary for navigating the maze and is unlikely to be correlated between example trajectories. The linear portion of the end effector is used to navigate the maze, and the proximal and distal loops keep the end effector within the plane of the maze. The tube maze on the right more fully explores the six degrees of freedom of the workspace. In this task, the loop at the tip of the end effector must be threaded over the copper tube to reach its base. Six demonstrations were performed on each maze by each participant.

Figure 7 shows the neighbor links chosen by other methods and the resulting embeddings. Figures (a) and (b) were produced using the k-nearest neighbor strategy of the original Isomap algorithm, with $k = 10$ chosen to ensure the number of neighbors per point is roughly equivalent to that of the other algorithms. The workspace plot of figure 7(a) appears sparser than the corresponding images for ST-Isomap (figure 7(c)) and our algorithm (figure 5(a)) because k-NN produces more temporal neighbors than spatio-temporal neighbors. The ST-Isomap results illustrate some of the most difficult situations for neighbor selection. In many cases, such as the spurious neighbor links near the bottom of figure 7(c), links are formed between trajectories that are relatively near to one another, and even travelling in parallel directions. Without considering global information about the trajectories, these incorrect links are difficult to detect.

The trajectories learned, even by our simple initial ap-

proach, appear qualitatively sound, and have been successfully executed on both mazes. In addition, the planned trajectories avoided joint limits more effectively than the example trajectories. During demonstration, users often rotated arm joints to their extremes even when this was not necessary to complete the task. Because the planning method relies on interpolation of demonstrations, planned trajectories remain away from joint limits whenever allowed by the demonstration data.

V. Conclusions and Future Work

In this paper, we presented a method for embedding robot trajectories in a low-dimensional space and a simple technique for creating novel plans in this space. Our work extends Isomap by introducing a neighbor-finding technique suited for time-series data such as configuration-space trajectories, and is free from the parameter selection required for the application of other approaches to multiple domains. This approach allows dimensionality reduction to produce a two-dimensional task-specific space in which safe planning is straightforward, even without a model of the environment in which the robot operates.

Future work will focus on development and evaluation of planning techniques in the embedded space, and possibly over the neighborhood graph itself. We also plan to address computational inefficiencies in the current approach. Since the embedding is not a globally linear transformation, straight lines are not preserved, and a large number of points must be lifted to faithfully transfer a plan from the embedded space to the original configuration space. By identifying regions of the demonstrations where less detail is required, such as areas of low diversity or low curvature, planning may be simplified in these areas.

Finally, we expect the improved neighbor-finding approach presented here to be useful in other applications. In addition to programming by demonstration, this technique may prove useful for activity recognition, robot fault detection, and other applications in which time-series trajectories are compared.

References

[1] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, May 1994, pp. 3310 – 3317.

[2] S. M. Lavalle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[3] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration space for fast path planning," in *Proceedings of the International Conference on Robotics and Automation*, San Diego, CA, 1994, pp. 2138–2145.

[4] M. Stolle and C. Atkeson, "Policies based on trajectory libraries," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 2006, pp. 3344–3349.

[5] R. Glaubius, M. Namihira, and W. D. Smart, "Speeding up reinforcement learning using manifold representations: Preliminary results," in *Proceedings of the IJCAI 2005 Workshop on Reasoning with Uncertainty in Robotics (RUR 05)*, Edinburgh, Scotland, July 2005.

[6] D. Bentivegna and C. Atkeson, "Learning from observation using primitives," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2, 2001, pp. 1988–1993 vol.2.

[7] A. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2, 2001, pp. 752–757 vol.2.

[8] C. Lee, "A phase space spline smoother for fitting trajectories," *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, vol. 34, pp. 346–356, 2004.

[9] A. Ude, C. Atkeson, and M. Riley, "Planning of joint trajectories for humanoid robots using b-spline wavelets," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 3, 2000, pp. 2223–2228 vol.3.

[10] J. Aleotti, S. Caselli, and G. Maccherozzi, "Trajectory reconstruction with nurbs curves for robot programming by demonstration," in *Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings. 2005 IEEE International Symposium on*, 2005, pp. 73–78.

[11] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using gaussian mixture models," in *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, May 2007.

[12] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Transactions on Robotics*, 2008.

[13] N. Ratliff, D. Bradley, J. Bagnell, and J. Chestnutt, "Boosting structured prediction for imitation learning," in *Advances in Neural Information Processing Systems 19*. Cambridge, MA: MIT Press, 2007.

[14] H. Friedrich, J. Holle, and R. Dillmann, "Interactive generation of flexible robot programs," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 1, 1998, pp. 538–543 vol.1.

[15] B. Argall, B. Browning, and M. Veloso, "Learning by demonstration with critique from a human teacher," in *ACM/IEEE international conference on Human-robot interaction*. Arlington, Virginia, USA: ACM Press, 2007, pp. 57–64.

[16] N. Delson and H. West, "Robot programming by human demonstration: adaptation and inconsistency in constrained motion," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 1, 1996, pp. 30–36 vol.1.

[17] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, Dec. 2000.

[18] K. Dautenhahn and C. L. Nehaniv, *Imitation in Animals and Artifacts*. MIT Press, 2002.

[19] O. C. Jenkins and M. J. Matarić, "A spatio-temporal extension to isomap nonlinear dimension reduction," in *The Twenty-first International Conference on Machine Learning*. Banff, Alberta, Canada: ACM Press, 2004, p. 56.

[20] A. Tsoli and O. C. Jenkins, "2d subspaces for user-driven robot grasping," in *Robotics, Science and Systems Conference: Workshop on Robot Manipulation*, 2007.

[21] M. Hein and M. Maier, "Manifold denoising as preprocessing for finding natural representations of data," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07)*. Menlo Park, CA: AAAI Press, Jul. 2007, pp. 1646–1649.

[22] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, B. Schlkopf, and B. S. Olkopf, "Learning with local and global consistency," *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 16*, vol. 16, pp. 321—328, 2003.

[23] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," in *Exploring artificial intelligence in the new millennium*. Morgan Kaufmann Publishers Inc., 2003, pp. 239–269.

[24] N. A. Melchior and R. Simmons, "Learning sequential composition plans using reduced-dimensionality examples," in *Papers from the 2009 AAAI Spring Symposium, Technical Report SS-09-01*. American Association for Artificial Intelligence, 2009.

[25] M. Bern and D. Eppstein, "Mesh generation and optimal triangulation," *Computing in Euclidean Geometry*, vol. 1, pp. 23–90, 1992.