Learning Sequential Composition Plans Using Reduced-Dimensionality Examples

Nik A. Melchior and **Reid Simmons**
Robotics Institute of Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, USA
{nmelchio,reids}@cs.cmu.edu

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. Next, regions with simple control policies are formed in the embedded space. Sequential composition of these simple policies is sufficient to plan a path to the goal from any known area of the configuration space. 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. Preliminary results are presented to validate this approach.

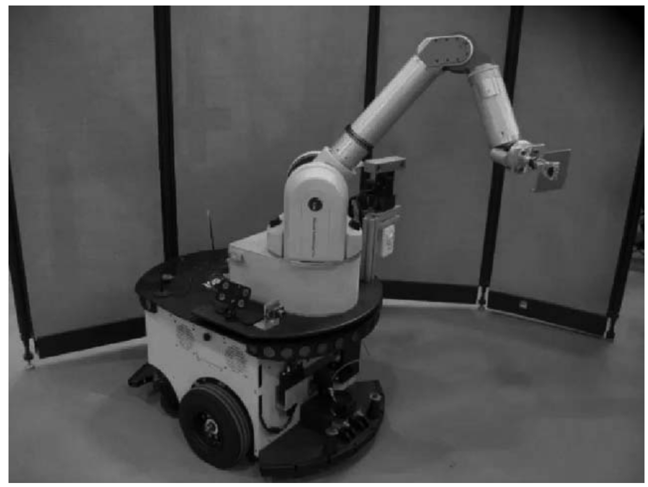


Figure 1: The experimental platform: a Mobile Robots Powerbot with a Barrett WAM 7-DOF arm.

Introduction

Some of the most iconic real-world uses for robots involve dexterous robotic arms performing precise reaching tasks. 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.

Previous work in our lab has addressed some of the issues inherent in robotic assembly tasks with tight tolerance (Sellner et al. 2006; 2008). Although these projects have used different robots with different controllers, much of the time allotted to final experimentation on the robots is spent tuning the manipulator trajectory used to assemble hardware. Our current approach relies on user-specified waypoints relative to the hardware attachment points, and it can be difficult for even the most experienced researchers to remember the coordinate frames associated with each manipulable object.

This work seeks to make it easier for both experienced and inexperienced robot users to command precise motions by providing examples. The motions should be demonstrated by moving the arm in passive mode or through a teleoperation interface. This kinesthetic form of training should be 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 both hard and soft constraints on the robot's motion. Hard constraints are obstacles within the robot's workspace. However, 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.

Soft 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 hard and soft constraints are conveyed implicitly by the human teacher through demonstrations, and may be incorporated into learned trajectories.

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* (Stentz 1994) or randomized sampling-based planners such as RRT (Lavalle and Kuffner 2001) and probabilistic roadmaps (Kavraki and Latombe 1994) 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. Since it is usually impractical to represent entire trajectories as points in spaces of thousands of dimensions, configurations spaces typically contain 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 like k-nearest neighbor can easily generalize too broadly, and other techniques must be used to correct this (Stolle and Atkeson 2006). Other work in Reinforcement Learning (Glaubius, Namihira, and Smart 2005) seeks to correctly generalize training examples as much as possible. This problem may also be ameliorated by drawing generalizations from portions of trajectories rather than single points. Several approaches use an intermediate representation such as domain-specific primitives (Bentivegna and Atkeson 2001) or basis functions (Ijspeert, Nakanishi, and Schaal 2001). Others have used various types of splines (Lee 2004), such as B-splines (Ude, Atkeson, and Riley 2000) or NURBS curves (Aleotti, Caselli, and Maccherozzi 2005) 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 (Chernova and Veloso 2007) of learned behaviors, or ensuring correct behaviors even when online adaptation is required (Hersch et al. 2008). Another recent work (Ratliff et al. 2007) 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 (Friedrich, Holle, and Dillmann 1998; Aleotti, Caselli, and Maccherozzi 2005). 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 (Argall, Browning, and Veloso 2007) after they are executed. Delson and West (Delson and West 1996) 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.

Approach

In this work, we seek to develop an approach that combines the strengths of previous programming by demonstration 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 has two major components. First, the examples are transformed to a lower-dimensional space. 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 reason spatially. Second, the portions of this space where the robot may safely move are divided into regions in which a single control law may be specified. By combining the example trajectories with a model of the manipulator, the free area in the workspace may be determined. By pre-computing the policy for all safe regions of the configuration space, execution can be more efficient.

We will first examine the strategy and motivation for region-based planning in two dimensions. Next, we present the dimensionality reduction technique that permits application of this approach to trajectories of higher dimensionality. Finally, an implementation of the complete system is tested on a 7-DOF robotic arm, and preliminary results are discussed.

Region Decomposition and Planning

To plan in two dimensions, we wish to identify contiguous regions in the embedding space in which a single, simple control law is able to move the robot to the next region, and eventually to the goal. Using this strategy of sequential composition (Burridge, Rizzi, and Koditschek 1999), each region has a range that is either the goal, or is in the domain of another region nearer to the goal.

Our approach to safety requires a model of the robot, but not the environment. This requirement should be easy

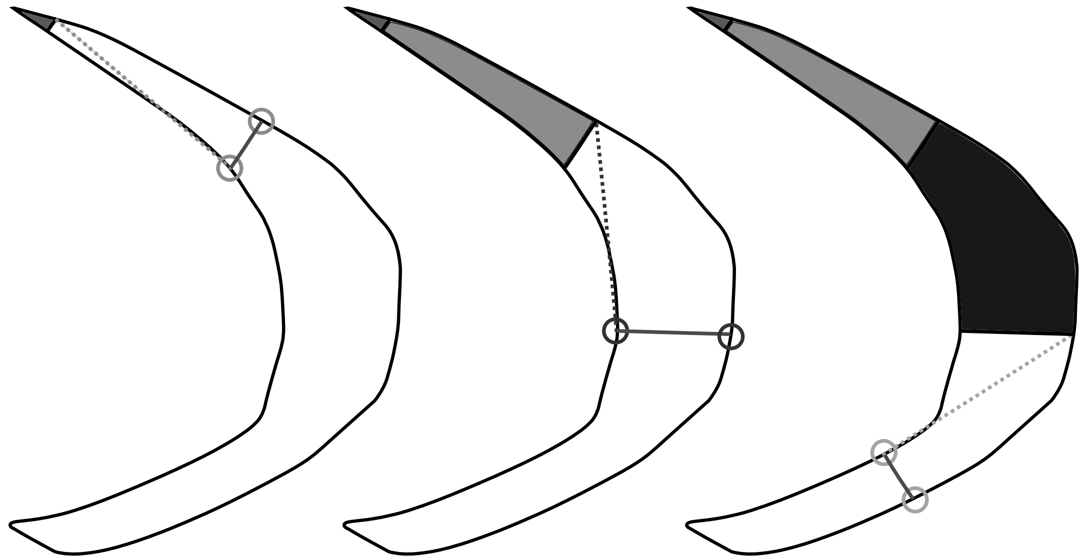


Figure 2: The formation of three regions. The top (red) region is the goal, the gray line represents the planning frontier, and the dashed line represents the test for visibility of the next region closer to the goal. When the visibility test fails for any point on the planning frontier, the next region is started. At this point, the gray line is the entry boundary for the previous region and the exit boundary for the next.

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. This collision-free, safe area may be constructed directly in the workspace if the manipulator is not redundant, or in a lower dimensional planning space.

The algorithm for forming regions is based on the simple control law used within regions; from any point within the region, there should be a straight-line path to the goal. The procedure is illustrated in figure 2. Starting at the goal (the region containing the endpoints of all example trajectories) we sweep a line backwards through the free space in the two-dimensional planning space. This line, the planning frontier, is initially oriented perpendicular to its direction of motion. When it is no longer possible to draw a straight line from the range (or *exit boundary*) of the region to every point on the frontier without crossing outside the free space, the swept line becomes the boundary of the current region and the range of a new region. This process iterates backwards until all safe points are contained within a region.

Figure 3 shows two examples of regions learned using different styles of examples in a simple two-dimensional task. Starting from a random position near the bottom of the image, the user had to drag the mouse cursor into the slot near the top of the image. The demonstrated motions could be executed by a planar RR robot arm, but these examples will ignore the arm's redundancy and represent the policy in the workspace for ease of understanding. Each colored area is a separate region, and the union of the regions is the portion of the workspace that was marked safe by the procedure described above. Although the task was the same in both cases, the policies learned are visibly distinct due to differences in the training examples.

Planning and executing novel trajectories is straightforward in this two-dimensional case. We simply determine the robot's position in the planning space. If it is inside a region, a waypoint (for position control) or direction of travel (for velocity control) may be chosen by searching for a visible point on the exit boundary of the current region. Typically, a segment of this boundary will be visible, so one point on the segment must be chosen. The center of the boundary is the safest choice, since it maximizes the distance from unknown space where obstacles may lie. However, computing the nearest point on the boundary to the robot's current location will provide the shortest path. We recommend a compromise between these two strategies, paying particular attention to the relative length of the boundary of the current region. Narrow boundaries indicate narrow gaps, regions of especially tight tolerance, or other situations in which all examples were more consistent than usual. In these cases, it may be advantageous to plan paths that remain close to the center of the demonstrated regions. When more variation is apparent in demonstrated paths, a strategy resulting in shorter path lengths may be favored.

If the robot is asked to plan from a position that is not inside a region, it must be outside the area covered by train-

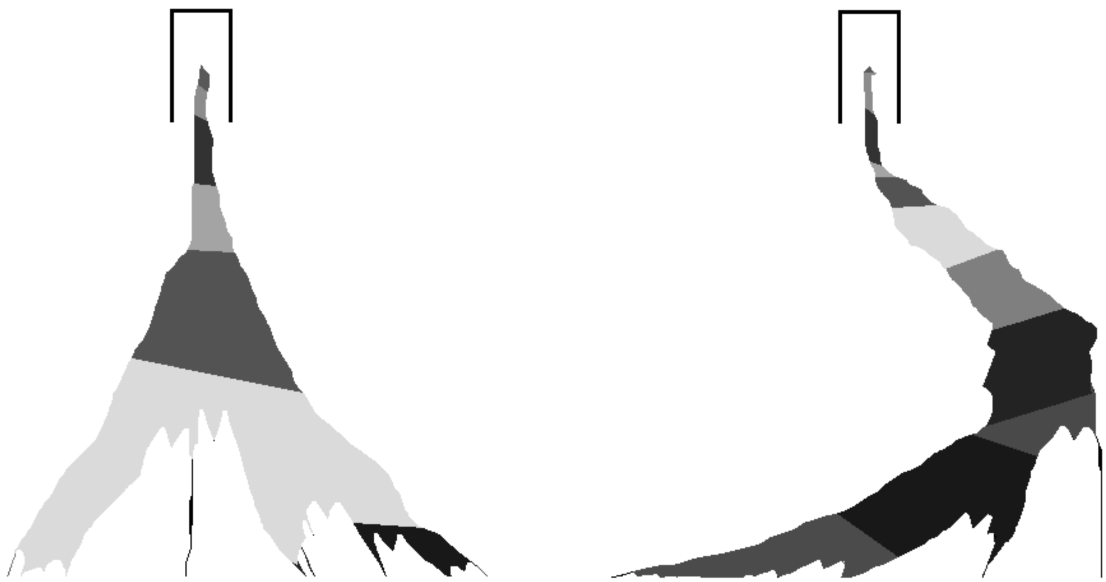


Figure 3: Regions learned for two styles of example trajectories. In each case, the task was to move from a random position near the bottom of the image into the slot near the top, and 10 example trajectories were collected.

ing examples. If safety is critical to the system, the robot must request help from a human operator, because it has no means to determine a safe action to execute. If unsafe operation is allowed, several options are available. To favor safety, the nearest region to the current position may be computed, and an action moving into that region may be executed. Alternately, we may compute the action that would be executed at the nearest known-safe point. By executing that action, or modifying it slightly to incrementally move the robot towards known-safe space, the robot's actions may still be made to resemble those provided during demonstration.

Dimensionality Reduction

The procedure described so far, and the region decomposition in particular, operates only in two dimensions. In order to learn trajectories in higher dimensional spaces, we must make a choice. One option is to extend the region decomposition and planning to higher dimensional spaces. The planning frontier would become a hyperplane, and it would follow the boundary of the collision-free area determined in an arbitrary-dimensional space. The other option is to perform all learning in two dimensions. This requires a mapping from the natural representation of arbitrary-dimensional trajectories to a two-dimensional space for learning and back again for execution. We choose to pursue the latter option.

Our primary reason for dimensionality reduction is to support the application of the region-based planning algorithm described above. 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 (Dautenhahn and Nehaniv 2002) 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 single trajectories 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. Finally, two dimensions appear to be sufficient to account for almost all of the variance in the data from the experiments described below using the 7-DOF arm, so dimensionality reduction is unlikely to cause problems later in the planning process.

One promising technique for dimensionality reduction is Spatio-Temporal-Isomap (Jenkins and Matarić 2004), an extension of the original Isomap algorithm (Tenenbaum, de Silva, and Langford 2000) for handling time-series data. The original Isomap algorithm finds a non-linear embedding for data using geodesic distances between nearby points. Instead of simply treating the input data as a collection of

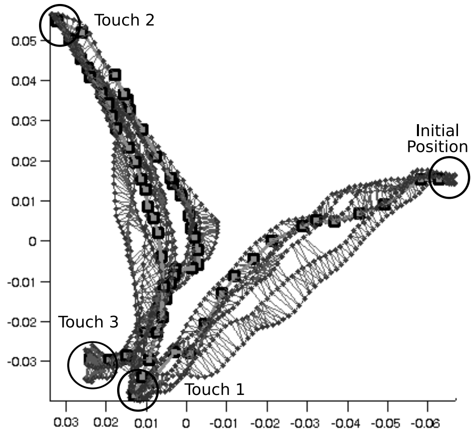


Figure 4: The 2D SVD projection of sample trajectories represented by thick blue lines. The thin red lines represent neighbor links determined by ST-Isomap. A planned path is shown by the larger, green squares.

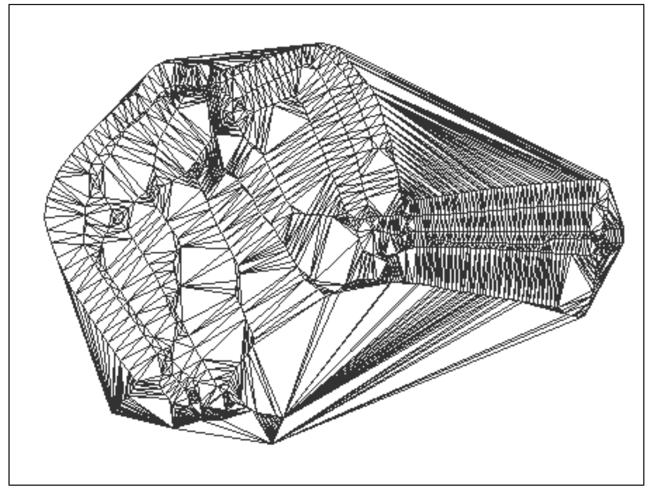


Figure 6: The Delaunay triangulation of approximately 1300 points in 6 example trajectories from a 7 dimensional configuration space. The ST-Isomap embedding is shown in figure 5(b).

points, ST-Isomap attempts to account for relationships between subsequent points in a single trajectory (temporal neighbors), and similar points in different trajectories that correspond to one another (spatio-temporal neighbors) by decreasing the perceived distance between these types of neighbor points.

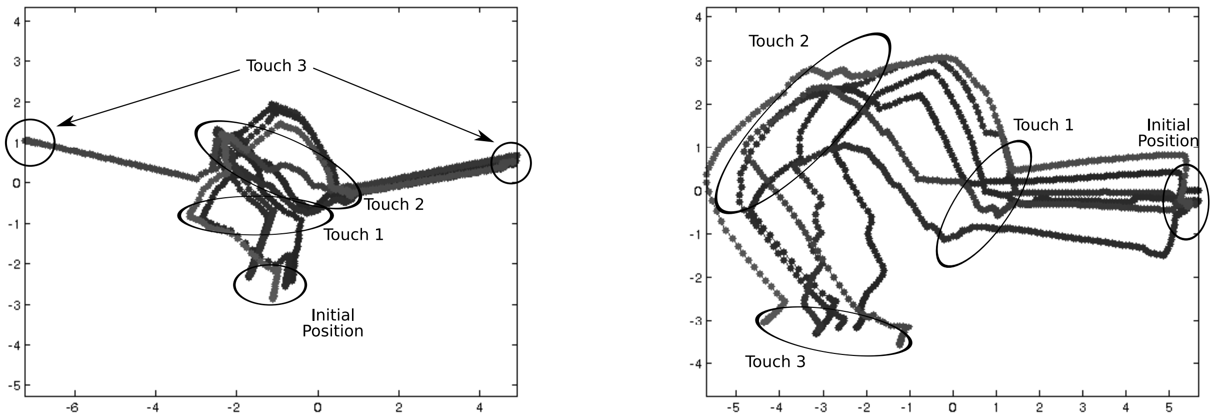
Figure 4 shows the two-dimensional singular value decomposition (SVD) projection of several example trajectories demonstrated by a human operator on the Barrett WAM 7-DOF dexterous arm shown in figure 1. When the arm is operated in gravity-compensation mode, it can be easily moved by hand. We recorded the position of all seven joints during a contrived task, in which the user was asked to stretch out the arm to touch 3 nearby points, with the end-effector in a different orientation each time. The blue traces in the plot show the first two dimensions of the example trajectories after SVD transformation. We operate in joint space because the arm is redundant, and we should be able to distinguish between different configurations that result in the same end-effector pose.

Examining the SVD projection in figure 4, we see that the trajectory traces are grouped together well, but different portions of the trajectories overlap. Since touch points 1 and 3 are close in the workspace, they also appear close in joint space, even though to points are always visited at different times during the execution of the task. This can complicate planning since the joint angles alone do not provide enough information to determine the desired motion to execute. The two-dimensional embedding created by conventional Isomap (figure 5(a)) stretches most of the examples so that they do not overlap, but still does not provide a suitable space for planning. Notice that the final position of one of the trajectories is far removed from the others, and the touch points are not well-distinguished.

The neighbor assignment mechanism described above (and represented by the red lines in figure 4) helps to overcome the problems of these techniques. The ST-Isomap embedding in figure 5(b) is able to distinguish between points in different parts of the task, and the initial and final positions of all examples are reasonably well grouped. This provides a more useful representation for planning. All trajectories follow similar courses through the embedding space, and the separation between them is a represenation of the inherent variations between examples.

Trajectory plans may be created using this two-dimensional embedding and the region decomposition procedure described in the previous section. Trajectories created in this two-dimensional planning space must be transformed to the seven-dimensional configuration space before they can be executed on the robot, but there is no global linear transformation between these spaces. Instead, individual points in the planned path must be *lifted* to the original high-dimensional space. This is accomplished using the Delaunay triangulation (Bern and Eppstein 1992) (see figure 6 of the points in two-dimensions. For any query point in the plane, the 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 range of the query 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 must be outside the planned regions, and thus



(a) (b)

Figure 5: The two-dimensional embeddings created by (a) Isomap and (b) ST-Isomap.

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. Again, the query point should lie within the collision-free area of the space. Using the correspondences already known between points in both spaces, we again interpolate between neighbors found in the configuration space.

Preliminary System Implementation

Finally, the region-based planner was applied to the reduced-dimensionality WAM data to produce novel, open-loop plans. While this preliminary implementation still suffers from a number of shortcomings to be addressed in the proposed work, it allows us to examine 7-DOF plans created using these algorithms in the two-dimensional planning space created by ST-Isomap. These plans were transformed to the seven-dimensional configuration space and executed on the WAM arm.

Figure 7 shows the regions learned from the ST-Isomap-transformed trajectories shown in figure 5(b). Due to the preliminary nature of the region-decomposition software, some simplifications had to be applied to the data in order to produce this result. First, the region-decomposition method does not yet handle *holes* in the safe area of the planning space. Holes can be formed when clustered trajectories briefly diverge from one another during the course of demonstration, then rejoin. For example, two sets of non-homotopic example trajectories will diverge when passing on opposite sides of an obstacle. This will produce an island of unknown, undemonstrated area in the planning space. As we have previously argued, users of a Programming by Demonstration system may not be able to ensure that all examples are homotopic, so it is important that this approach properly deal with these cases. For now, however, we have elided a small portion of one demonstration which created a hole in this planning space.

Another algorithmic simplification was used in determining the safe area of the planning space. Rather than produce a three-dimensional model of the WAM arm for determining the workspace occupied in each configuration, we represented the arm as a fixed-radius ball in the planning space. This simplification permits region decomposition and planning entirely within the reduced-dimensionality space without requiring reasoning in the workspace and configuration space as well. Normally, safe area determination would use configuration space trajectories along with a model of the manipulator to determine the free area of the workspace. Then, we would determine the safe points in the planning space by determining whether the corresponding configurations result in safe workspace poses. By modelling the manipulator as a ball in the planning space, all computation can be performed in the low-dimensional space.

With these simplifications, the WAM trajectories produced a single, contiguous area for planning, and region decomposition could be performed using our proof-of-concept implementation. Next, we chose starting points within the created regions, and allowed the planner to produce paths to the goal region, the final region near the bottom of figure 7. Regularly sampled points from these trajectories were lifted to the original configuration space using the Delaunay triangulation shown in figure 6. These lifted trajectories were then executed on the WAM arm.

The learned trajectories appeared qualitatively sound, visiting each of the “Touch” points labelled in previous figures. 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.

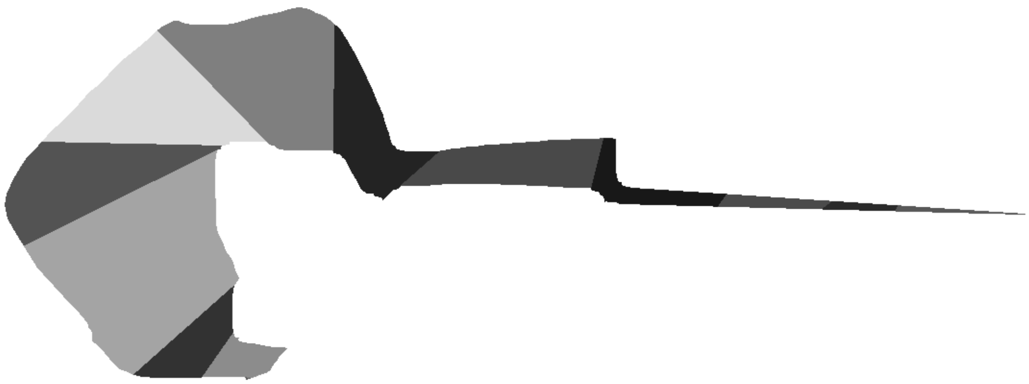


Figure 7: Regions learned for the ST-Isomap-transformed WAM trajectories depicted in figure 5(b).

Discussion

The region-based planning system depicted in figure 3 operates in the workspace of a planar RR robotic arm, and is able to produce plans that safely move the end-effector to the goal from any point that is known to be safe. Plans beginning in unvisited areas of the workspace are possible using a variety of strategies, but such plans are considered undesirable. Instead, the system should request assistance from a human operator.

Planning in the workspace is not ideal for redundant manipulators. We would like to plan in configuration space, but it is not clear that the region-based method can extend to higher-dimensional spaces. Dimensionality reduction is employed to make planning more tractable and to help smooth noise in the example trajectories.

Preliminary, automated region-based planning has been applied to embedded 7-DOF trajectories such as those shown in figure 5(b). After lifting these plans into the seven-dimensional configuration space, they may be executed on the WAM arm. These planned paths appear qualitatively correct, and smoothly follow the shape of the demonstrated trajectories.

Finally, we have examined the validity of the use of dimensionality reduction in terms of the residual variance of each of these approaches. Residual variance is an indication of the amount of information lost by reducing the dimensionality of data. For the Isomap variants, it measures the error introduced in the original dataset by comparing the geodesic distances between all pairs of points in the original space with the Euclidean distances in the reduced dimensionality space. SVD simply reprojects the new points into the original space and compares Euclidean pairwise distances with the original points. Error are normalized, and the result is presented as a fraction of the original distances.

Figure 8 quantifies the error in SVD, Isomap, and ST-Isomap for the task on the WAM arm. The data is further separated based on the level of experience that the user had in operating this device. We may draw a few conclusions about ST-Isomap from this plot. There is a distinct elbow in the traces for ST-Isomap for both experienced and inexperienced users. This suggests that two dimensions may be sufficient to represent at least the seven-dimensional trajectories from this scenario. In fact, in the experienced case, the residual has nearly reached its minimum at two dimensions, so additional dimensions would offer little improvement. The inexperienced case, though, retains far more variance initially, and seems to asymptote higher. Qualitatively, this seems unsurprising since the collection of inexperienced trajectories appears less consistent than those of the experienced users. This disparity may actually be useful in that it may allow the system to automatically distinguish the skill level of the person demonstrating the skill. For example, if a human is being trained in a skill (rather than the robot), this may allow a quantitative analysis to determine when the person has mastered the skill.

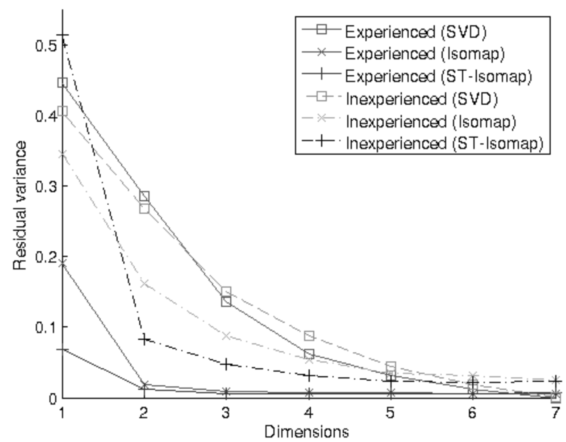


Figure 8: The residual variance for three different dimensionality reduction techniques. ST-Isomap performs best in both cases, but the examples provided by the inexperienced user retain more variance.

Future Work

The system described in this paper is a work in progress, in need of further algorithmic development and quantitative analysis. In particular, the ST-Isomap dimensionality reduction, while effective, has a few parameters which must be tuned for each problem domain. In addition, the algorithm for selection of spatio-temporal neighbors is frequently suboptimal and requires further investigation.

In the second portion of the algorithm, the method used to divide space into regions is somewhat simplistic. It should account not only for the presence of trajectories within a region, but also for the direction of motion of these trajectories. The region decomposition should also consider which parts of the space are considered safe to occupy. Plans cannot be created in areas not covered by regions, so this will lead to safer plans. This safety may also be obtained by ensuring that no example points are considered to be neighbors if the motion from one point to the other does not remain in the safe area.

Finally, the efficiency of the transformations between workspace, configuration space, and the planning space should be addressed.

References

Aleotti, J.; Caselli, S.; and Maccherozzi, G. 2005. 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*, 73–78.

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

Bentivegna, D., and Atkeson, C. 2001. Learning from observation using primitives. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, 1988– 1993 vol.2.

Bern, M., and Eppstein, D. 1992. Mesh generation and optimal triangulation. *Computing in Euclidean Geometry* 1:23–90.

Burridge, R. R.; Rizzi, A. A.; and Koditschek, D. E. 1999. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research* 18(6):534–555.

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

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

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

Friedrich, H.; Holle, J.; and Dillmann, R. 1998. Interactive generation of flexible robot programs. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 1, 538–543 vol.1.

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

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

Ijspeert, A.; Nakanishi, J.; and Schaal, S. 2001. Trajectory formation for imitation with nonlinear dynamical systems. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, 752–757 vol.2.

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

Kavraki, L., and Latombe, J.-C. 1994. Randomized preprocessing of configuration space for fast path planning. In *Proceedings of the International Conference on Robotics and Automation*, 2138–2145.

Lavalle, S. M., and Kuffner, J. J. 2001. Randomized kinodynamic planning. *International Journal of Robotics Research* 20(5):378–400.

Lee, C. 2004. A phase space spline smoother for fitting trajectories. *Systems, Man and Cybernetics, Part B, IEEE Transactions on* 34:346–356.

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

Sellner, B.; Heger, F.; Hiatt, L. M.; Simmons, R.; and Singh, S. 2006. Coordinated multi-agent teams and sliding autonomy for large-scale assembly. *Proceedings of the IEEE* 94(7).

Sellner, B.; Heger, F. W.; Hiatt, L. M.; Melchior, N. A.; Roderick, S.; Akin, D.; Simmons, R.; and Singh, S. 2008. Overcoming sensor noise for low-tolerance autonomous assembly. In *Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems*.

Stentz, A. 1994. Optimal and efficient path planning for partially-known environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, 3310 – 3317.

Stolle, M., and Atkeson, C. 2006. Policies based on trajectory libraries. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 3344–3349.

Tenenbaum, J. B.; de Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290:2319–2323.

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