Learning Subjective Representations for Planning

Dana Wilkinson
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
d3wilkinson@uwaterloo.ca

Michael Bowling
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
bowling@cs.ualberta.ca

Ali Ghodsi
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
aghodsib@uwaterloo.ca

Abstract

Planning involves using a model of an agent's actions to find a sequence of decisions which achieve a desired goal. It is usually assumed that the models are given, and such models often require expert knowledge of the domain. This paper explores subjective representations for planning that are learned directly from agent observations and actions (requiring no initial domain knowledge). A non-linear embedding technique called Action Respecting Embedding is used to construct such a representation. It is then shown how to extract the effects of the agent's actions as operators in this learned representation. Finally, the learned representation and operators are combined with search to find sequences of actions that achieve given goals. The efficacy of this technique is demonstrated in a challenging robot-vision-inspired image domain.

1 Introduction

Planning, at its essence, involves searching an appropriately defined state space. This state space is a consequence of the agent's model of the effects of its actions (*e.g.*, STRIPS [Fikes and Nilsson, 1971], Markov Decision Processes [Puterman, 1994]). It is assumed that these models can be defined from domain experts and planning uses these models to find sequences of actions to achieve given goals. The problem is that models are not always known or easily built for a domain of interest. Others have studied methods for learning or agumenting models, such as STRIPS operators [Wang, 1995] or MDP transitions [Peng and Williams, 1993]. All of these techniques still require expert intuition about the domain to provide, at the least, an appropriate state representation.

This paper focuses on learning an appropriate representation for planning, using only an agent's observations and actions. We call this a *subjective representation* as the learned representation is extracted based only on the agent's experience, and requires no expert knowledge of the domain. The approach solves two important problems: (i) learning an appropriate state-space representation, and (ii) learning the effects of the agent's actions in this representation.[1] The required input is a sequence of actions and observations from the agent's subjective experience, but no semantic meaning (domain-specific or otherwise) is required. This input is used in a three step process.

First, Section 2 reviews Action Respecting Embedding (ARE). ARE is a technique for dimensionality reduction that specifically makes use of a temporal sequence of observations and actions. ARE learns manifolds that capture the important underlying dynamics of the high-dimensional data in much fewer dimensions (addressing the first problem). Next, Section 3 describes a method for learning operators for each action that can be applied to any point in the learned representation (addressing the second problem). Examples of learned operators are provided along with a discussion of how the semantic meaning of the associated actions can sometimes be extracted. Finally, Section 4 shows the results of planning in the resulting representation using the learned operators. Both the resulting plan in the learned representation, and the result of applying the plans in the original high-dimensional domain are compared.

2 Action Respecting Embedding

High-dimensional data sets, such as sequences of images, can often be characterized by a low-dimensional representation that is related to the process generating the data. For example, a low-dimensional representation for image data may correspond to the degrees of freedom of a platform moving a camera. Such a representation is ideal for planning as it directly captures the actions' effects on the world. The goal here is to take a temporal sequence of data points, $z_1, \ldots, z_n$, and associated actions, $a_1, \ldots, a_{n-1}$, and find a low-dimensional representation for z_i that is appropriate for planning.

Recently, *nonlinear manifold learning* techniques have been used to map a high-dimensional dataset into a smaller dimensional space. Semidefinite Embedding (SDE) [Weinberger and Saul, 2004] is one such technique. SDE learns a kernel matrix, which represents a non-linear projection of

[1] The approach of learning a subjective representation more closely resembles recent work on learning predictive representations [James and Singh, 2004; Rosencrantz *et al.*, 2004; Jaeger, 2000] than previous work on augmenting operators or transition probabilities. This approach, though, is specifically designed for very high-dimensional observation spaces as it implicitly involves a dimensionality reduction component.

Algorithm: SDE($\|\cdot\|, (z_1, \ldots, z_n)$)
Construct neighbors, N**, using** k**-NN with** $\|\cdot\|$**.**
Maximize $\mathrm{Tr}(K)$ **subject to** $K \succeq 0$, $\sum_{ij} K_{ij} = 0$, **and** $\forall ij \quad N_{ij} > 0 \vee [N^T N]_{ij} > 0 \Rightarrow K_{ii} - 2K_{ij} + K_{jj} = \|z_i - z_j\|^2$
Run Kernel PCA with learned kernel, K**.**

Table 1: Algorithm: Semidefinite Embedding (SDE).

Algorithm: ARE($\|\cdot\|, (z_1, \ldots, z_n), (a_1, \ldots, a_{n-1})$)
Construct neighbors, N**, as in [Bowling *et al.*, 2005].**
Maximize $\mathrm{Tr}(K)$ **subject to** $K \succeq 0$, $\sum_{ij} K_{ij} = 0$, $\forall ij \quad N_{ij} > 0 \vee [N^T N]_{ij} > 0 \Rightarrow K_{ii} - 2K_{ij} + K_{jj} \leq \|z_i - z_j\|^2$ **, and** $\forall ij \quad a_i = a_j \Rightarrow K_{(i+1)(i+1)} - 2K_{(i+1)(j+1)} + K_{(j+1)(j+1)} = K_{ii} - 2K_{ij} + K_{jj}$
Run Kernel PCA with learned kernel, K**.**

Table 2: Algorithm: Action Respecting Embedding (ARE).

the input data into a more linear representation. It then uses Kernel PCA [Scholkopf and Smola, 2002], a generalization of principle components analysis using feature spaces represented by kernels, to extract out a low-dimensional representation of the data. The kernel matrix K is learned in SDE by solving a semidefinite program with a simple set of constraints. The most important constraints encode the common requirement in dimensionality reduction that the non-linear embedding should preserve local distances. In other words, nearby points in the original input space should remain nearby in the resulting feature representation. Therefore SDE requires a distance metric $\|\cdot\|$ on the original input space, and uses this metric to construct a k-nearest neighbors graph. It then adds constraints into the semidefinite program to ensure that the distance between neighbors is preserved. The optimization maximizes $\mathrm{Tr}(K)$, *i.e.*, the variance of the learned feature representation, which should minimize its dimensionality. The SDE Algorithm is shown in Table 1.

SDE does not take into account two important pieces of knowledge about the data: the temporal ordering of the input vectors, z_i, and the action labels, a_i. Therefore, SDE doesn't guarantee that temporally-nearby input points will be spatially nearby in the feature representation. Also, SDE won't necessarily result in a space where actions have a simple interpretation. The recent Action Respecting Embedding (ARE) algorithm [Bowling *et al.*, 2005] extends SDE to make use of exactly this type of knowledge about the data.

Formally, ARE takes a set of D-dimensional input vectors, $z_1, \ldots, z_n$ (*e.g.*, images), in temporal order, along with associated discrete actions, $a_1, \ldots, a_{n-1}$, where action a_i was executed between input z_i and input z_{i+1}. ARE then computes a set of d-dimensional output vectors $x_1, \ldots, x_n$ in one-to-one correspondence with the input vectors. This provides a meaningful embedding in $d < D$ dimensions. ARE modifies SDE in two key ways. First, it exploits the knowledge that the images are given in a temporal sequence. It uses this knowledge to build an improved neighborhood graph based on each input's distances to its temporal neighbors using the provided local distance metric[2]. Second, it constrains the embedding to respect the action labels that are associated with adjacent pairs of observations. This ensures that the actions have a simple interpretation in the resulting feature space.

It is this second enhancement of ARE that is the critical feature for subjective planning. ARE constrains the learned manifold to be in a space where the labeled actions correspond to distance-preserving transformations—those consisting only of rotation and translation[3]. Therefore, for any two inputs, z_i and z_j, the same action from these inputs must preserve their distance in the learned feature space. Letting $\Phi(z_i)$ denote input z_i's representation in the feature space, action a's transformation, f_a, must satisfy:

$$\forall i, j \qquad \|f_a(\Phi(z_i)) - f_a(\Phi(z_j))\| = \|\Phi(z_i) - \Phi(z_j)\|. \tag{1}$$

Now, let $a = a_i$ and consider the case where $a_j = a_i$. Then, $f_a(\Phi(z_i)) = \Phi(z_{i+1})$ and $f_a(\Phi(z_j)) = \Phi(z_{j+1})$, and Constraint 1 becomes:

$$\|\Phi(z_{i+1}) - \Phi(z_{j+1})\| = \|\Phi(z_i) - \Phi(z_j)\|. \tag{2}$$

In terms of the kernel matrix, this can be written as:

$$\forall i, j \quad a_i = a_j \Rightarrow K_{(i+1)(i+1)} - 2K_{(i+1)(j+1)} + K_{(j+1)(j+1)} = K_{ii} - 2K_{ij} + K_{jj} \tag{3}$$

ARE simply adds Constraint 3 into SDE's usual constraints to arrive at the optimization and algorithm shown in Table 2.

Experiments. Here we define ImageBot, a synthetic image interaction domain used for all experiments. Given an image, imagine a virtual robot that can observe a small patch on that image and also take actions to move the patch around the larger image. This "image robot" provides an excellent domain in which subjective planning can be demonstrated.

For these experiments, ImageBot will always be viewing a 200 by 200 patch of a 2048 by 1536 image displayed Figure 1. ImageBot has eight distinct actions: four translations, two zoom actions, and two rotation actions. The allowed translations are forward (F), back (B), left (L) and right (R) by 25 pixels The zoom changes the scale of the underlying image by a factor of $2^{1/8}$ (i) or $2^{-1/8}$ (o). The rotation rotates the square left (l) or right (r) by $\frac{\pi}{8}$ radians.

There are three distinct experimental data sets that are looked at in this paper.

[2] We have found that ARE is fairly robust to the choice of distance metrics, and use simple Euclidean distance for all of the experiments in this paper.

[3] Notice this is not requiring the actions in the objective space to be rotations and translations, since ARE is learning a non-linear feature representation.

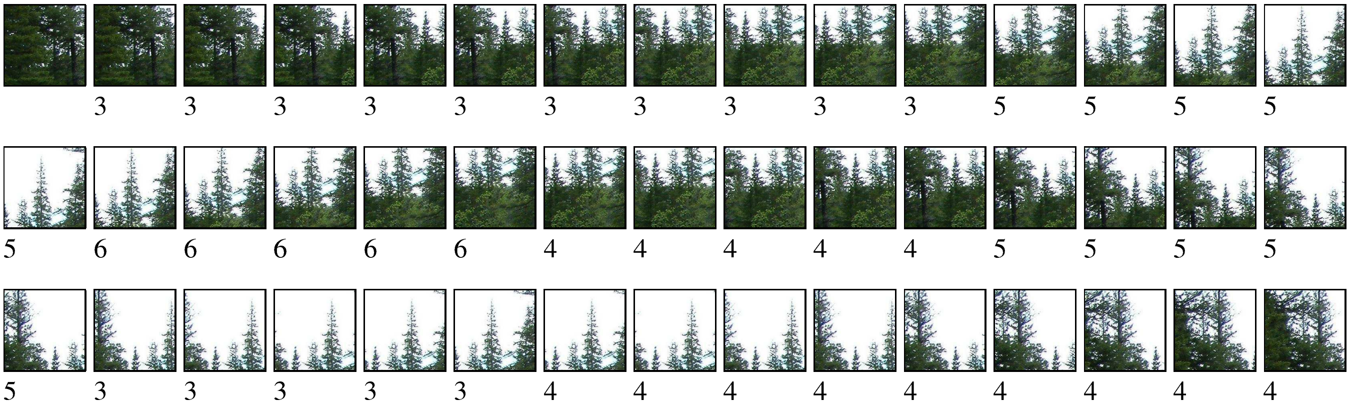

Figure 2: IMAGEBOT's output (and ARE's input) for the A_T data set.

Figure 1: IMAGEBOT's world.

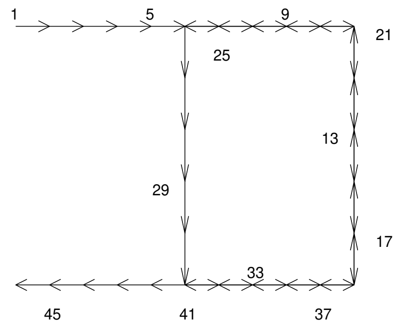

Figure 3: The path IMAGEBOT follows to generate A_T.

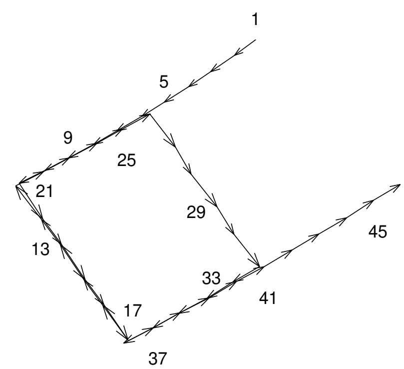

Figure 4: The representation learned by ARE for A_T.

A_T: IMAGEBOT follows a path which looks like an "A" using only the translation actions:
$F \times 10, L \times 5, R \times 5, B \times 5, L \times 5, F \times 5, B \times 10$

A_Z: IMAGEBOT follows the same A pattern but substituting zoom in for left actions and zoom out for right actions:
$F \times 10, i \times 8, o \times 8, B \times 5, i \times 8, F \times 10, B \times 20$

Fr: IMAGEBOT moves back and forth in a line, but only using F and r actions:
$F \times 10, r \times 8, F \times 10, r \times 8, F \times 5, r \times 16, F \times 5$

Note that in A_Z the F and B actions only move half as much when zoomed in as when zoomed out. Note that in Fr there are no opposites for the two actions used. An example of the output of IMAGEBOT, and correspondingly an input for ARE, is shown in Figure 2. These are the images seen in the A_T data set. Note that while *we* know that, for example, action label 3 corresponds to action F, ARE gets no such semantic information—it gets as input only the images and the labels associated with them.

The effectiveness of ARE has been demonstrated previously [Bowling *et al.*, 2005]. Here evidence is shown of its power in capturing useful representations for planning. Figure 3 shows the actual manifold underlying the A_T test set (*i.e.*, IMAGEBOT's path). Figure 4 shows the manifold learned by ARE on that test set. Clearly the structure has been captured.

Figure 5 shows a "top" view and a "side" view of the 3-dimensional manifold learned for the Fr data set. here the portion of the manifold corresponding to rotating to the right (r) is a black line while the portion corresponding to moving forward (F) is a light-gray line. The point corresponding to the first image is circled. Although not as clear as in the previous case, this manifold is clearly and distinctly capturing the structure of the original path, with two dimensions capturing the r action and a third capturing the F action. In Figure 3, the original domain was one in which the actions were distance-preserving and this structure was, indeed, extracted. In Figure 5 it is not immediately obvious what man-

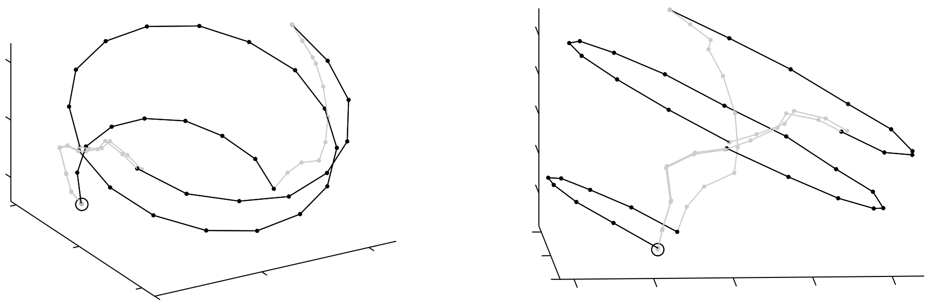

Figure 5: Two different views of the three-dimensional cylindrical manifold learned for the Fr data set.

ifold will be learned which makes the resulting one all the more impressive—not just as a representation appropriate for planning but as an aid to intuitive understanding of the original underlying structure.

3 Distance Preserving Operators

ARE learns a representation with explicit constraints that the actions correspond to distance-preserving transformations in that representation. Before one can plan, though, one needs to discover these transformations. For each unique action a there is a collection of data point pairs (x_t, x_{t+1}) which are connected by that action. Another way of thinking of this is that there is a function f_a where $f_a(x_t) = x_{t+1}$, and such a function needs to be learned for each action. Because of the distance-preserving constraints, f_a can be represented as:

$$f_a(x_t) = A_a x_t + b_a = x_{t+1}$$

Recall that transformations of the above form encode translation in the b_a vector, and rotation and scaling in the A_a matrix. A_a and b_a could be learned using simple linear regression but scaling is not distance preserving so there is the additional constraint that A_a does not scale, *i.e.*, $A_a^T A_a = I$. It turns out that this is similar to the extended orthonormal Procrustes problem [Schoenemann and Carroll, 1970], but without allowing for a global scaling constant. Here the solution to the regression problem is derived.

Let X_a be the d by n matrix whose columns are x_t for all t such that $a_t = a$, and let Y_a be the d by n matrix whose columns are x_{t+1} for the same t. The goal is to learn a rotation matrix A_a and a translation b_a which maps X_a to Y_a. Formally, the following optimization problem needs to be solved.

$$\begin{aligned} &\textbf{minimize:} \quad ||A_a X_a + b_a e^T - Y_a|| \\ &\textbf{subject to:} \quad A_a^T A_a = I \end{aligned}$$

where e is a column vector with n ones.

In order to obtain the least squares estimation of A_a and b_a, write the Lagrangian function L:

$$\begin{aligned} &L(A_a, b_a, \Lambda) = \\ &\quad \mathrm{Tr}((A_a X_a + b_a e^T - Y_a)^T (A_a X_a + b_a e^T - Y_a)) + \\ &\quad \mathrm{Tr}(\Lambda(A_a^T A_a - I)) \end{aligned}$$

where Λ is a matrix of lagrangian multipliers, and Tr(.) stands for the trace of a matrix.

$$\begin{aligned} &L(A_a, b_a, \Lambda) = \\ &\quad \mathrm{Tr}(Y_a^T Y_a) + \mathrm{Tr}(X_a^T A_a^T A_a X_a) + n b_a^T b_a \\ &\quad -2\mathrm{Tr}(Y_a^T A_a X_a) - 2\mathrm{Tr}(e b_a^T Y_a) \\ &\quad +2\mathrm{Tr}(e b_a^T A_a X_a) + \mathrm{Tr}(\Lambda(A_a^T A_a - I)) \end{aligned}$$

Now take the derivative of the Lagrangian function with respect to the unknowns and set to zero:

$$\begin{aligned} \frac{\partial L}{\partial A_a} &= 2A_a X_a^T X_a - 2Y_a X_a^T + \\ &\quad +2b_a e^T X_a^T + A_a(\Lambda + \Lambda^T) = 0 \qquad (4) \\ \frac{\partial L}{\partial b_a} &= 2nb_a - 2Y_a e + 2A_a X_a e = 0 \qquad (5) \end{aligned}$$

The translation vector from Equation 5 gives:

$$b_a = \frac{(Y_a - A_a X_a)e}{n} \qquad (6)$$

Multiplying Equation 4 by $A_a^T/2$ on the right:

$$\begin{aligned} A_a X_a^T X_a A_A^T + A_a(\Lambda + \Lambda^T)A_a^T/2 = \\ Y_a X_a^T A_a^T - b_a e^T X_a^T A_a^T \end{aligned}$$

Since the left hand side is symmetric, the right hand side must also be symmetric. Substituting Equation 6, the right hand side can be written as:

$$Y_a X_a^T A_a^T - Y_a \left(\frac{ee^T}{n}\right) X_a^T A_a^T + A_a X_a \left(\frac{ee^T}{n}\right) X_a^T A_a^T$$

where the last term is symmetric. Thus the rest of the expression must also be symmetric. This can be simplified as:

$$\left(Y_a \left(I - \frac{ee^T}{n}\right) X_a^T\right) A_a^T \qquad (7)$$

Since 7 is symmetric, it should be equivalent to its transpose:

$$\left(Y_a \left(I - \frac{ee^T}{n}\right) X_a^T\right) A_a^T = A_a \left(Y_a \left(I - \frac{ee^T}{n}\right) X_a^T\right)^T \qquad (8)$$

One can easily verify that the following satisfies Equation 8:

$$\begin{aligned} VSW^T &= \mathrm{svd}\left(Y_a \left(I - \frac{ee^T}{n}\right) X_a^T\right) \\ A_a &= VW^T \qquad (9) \end{aligned}$$

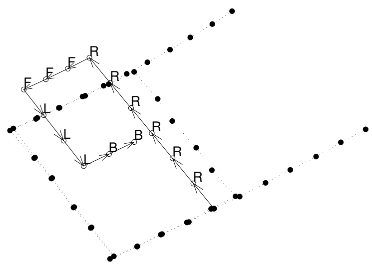

Figure 6: Demonstrating distance-preserving operators in the representation learned for A_T.

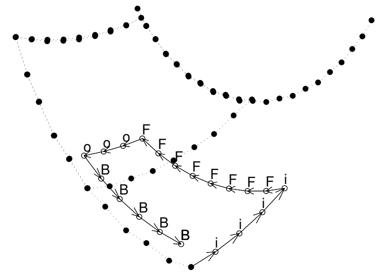

Figure 7: Demonstrating distance-preserving operators in the representation learned for A_Z.

where svd$(\cdot)$ is the singular value decomposition, so,

$$VSW^TWV^T = VW^TWS^TV^T$$

because $W^TW = I$ (since W is orthonormal) and $S = S^T$ (since S is diagonal). Thus Equations 6 and 9 are a solution to the dual function, $D = \min_\Lambda L(A_a, b_a, \Lambda)$. Since this solution is also feasible with regards to the primal problem, strong duality holds even though the original problem is non-convex.

Results. Figure 6 shows the two-dimensional representation that ARE generated for A_T. The solid arrows show a path which consists of new points resulting from the application of operators learned for each action (F, B, L and R) as described above. Clearly, the operators are intuitively capturing the essence of the actions used to generate the data.

Note that while there is no semantic meaning given with the input actions, such meaning can now be derived. It can easily be tested whether a pair of actions are opposites, such as (F, B) or (R, L). Also two actions can be tested for orthogonality or independence, such as F and R. If the learned representation captures some underlying structure within the data, the learned operators will maintain that structure and, from them, relationships can be successfully hypothesized.

Figure 7 is similar to Figure 6, except the underlying representation is from the A_Z data set. Here, note that while some actions are again opposite to each other, no actions are orthoganal—the F and B actions are not independent of the i and o actions. This critical facet of the original data set has been successfully captured in the representation learned by ARE and consequently in the action functions learned in that manifold. Note, in particular, that when zoomed in all the way ($i \times 8$) 10 F actions are equivalent to 5 F actions when zoomed out all the way. Although F action when zoomed in half way was never observed, the learned operators capture the fact that between 7 and 8 F actions at this scale are equivalent to 5 and 10 actions at the other scales.

4 Planning

Now that low-dimensional representations and operators in those representations can be learned, all the pieces are in place to perform planning. The points in the learned representation are states, and the operators learned in Section 3 define transitions between the states. This new domain has two advantages over the original data set. First, the dimensionality has been reduced drastically (from 40,000 to 2 or 3). Second, from the action labels functions have been learned for each unique action that explicitly give the resulting state when that action is applied from any state. Learning such a function in the orignal space is not only intractable, but would require application of knowledge specific to IMAGEBOT (or even the underlying image) in order to attain any success.

Given any two images, one can find a shortest path between them, even if it traverses unobserved parts of the space. First, find the corresponding points in the low-dimensional representation, then find the shortest path between them using traditional search methods and the set of learned operators. Since each operator in the low-dimensional space corresponds to an action label in the orginal space the list of action labels that indicate the desired path can be returned. For the following results, iterative-deepening depth-first search was used and the path whose final point was closest to the desired goal was returned. The quality of a path is demonstrated by starting IMAGEBOT at the initial state, applying the sequence of actions and showing the resulting image.

Results. For each of the three test sets, two sub-figures will be shown. The first shows the representation learned for that data set and the shortest path between a chosen initial state (labelled with a triangle pointing right) and a chosen goal state (labelled with a triangle pointing left). The second figure contains two images. The left image shows the image at the initial state, the right image contains two highlighted boxes. The light-gray dotted box shows the image at the goal state, the darker-gray solid box highlights the image obtained after executing the resulting sequence of actions.

Figures 4(a) and 4(b) show the results for A_T—the goal state image and the image corresponding to the final state in our path in Figure 4(b) are the same. Note that the shortest path was successfully found, even though it involves moving through portions of the space that we have never seen.

Figures 4(c) and 4(d) show the results for A_Z—the goal state image and the image corresponding to the final state in the path in Figure 4(d) are the same. Note, the path found successfully identifies that action B must occur before action o—if taken after then the B action would jump over the desired end state to a state halfway between it and the next state.

Figures 4(e) and 4(f) show the results for Fr—the goal state image and the image corresponding to the final state in the path in Figure 4(f) are very close to each other. Recall that for this data set, unlike the others, the only actions were

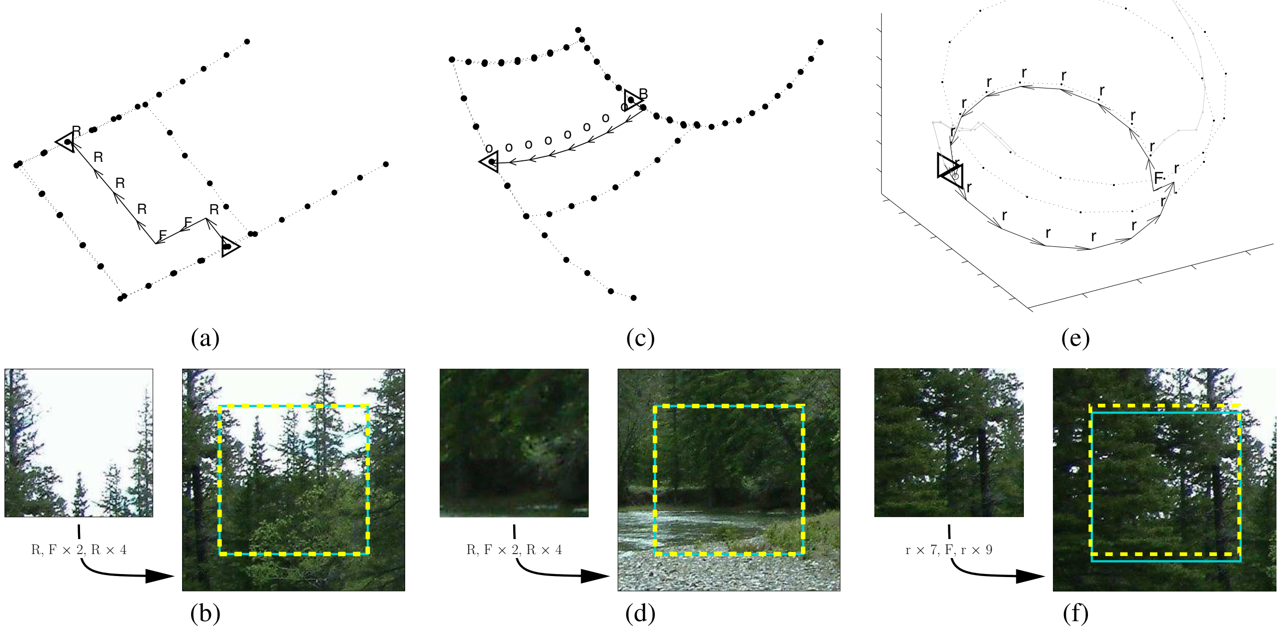

Figure 8: Results of planning in three examples. (a), (c), and (e) show the paths in the learned representation. (b), (d), and (f) show the starting image, the sequence of actions, and outlines the resulting and goal images.

F and r with no corresponding opposites. This means that such a path must rotate all the way around, step forward, then rotate all the way around again—a fairly complex path.

5 Conclusion

ARE can be used to learn a subjective representation appropriate for planning. The only input required is a sequence of observations and actions—no additional domain-specific knowledge is necessary. The output from ARE is a new low-dimensional space which captures the critical dynamics of the environment. Operators which reflect these dynamics can be recovered in the new space. Simple search procedures can then can be used to find sequences of operators which achieve goals in the learned representation. Since operators correspond to original actions, this sequence provides a plan in the original space. These plans are accurate, even though they can involve actions in unobserved parts of the space.

Acknowledgments

We thank Finnegan Southey, Dale Schuurmans, and Pascal Poupart for discussions and insight. We thank Wesley Loh for helping with implementation details. We acknowledge Alberta Ingenuity Fund for their support of this research through the Alberta Ingenuity Centre for Machine Learning.

References

[Bowling *et al.*, 2005] Michael Bowling, Ali Ghodsi, and Dana Wilkinson. Action respecting embedding. Technical Report TR05-09, University of Alberta, 2005.

[Fikes and Nilsson, 1971] Richard Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In *IJCAI*, pages 608–620, 1971.

[Jaeger, 2000] Herbert Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398, 2000.

[James and Singh, 2004] Michael R. James and Satinder Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *ICML*, 2004.

[Peng and Williams, 1993] J. Peng and R. J. Williams. Efficient learning and planning within the Dyna framework. *Adaptive Behavior*, 2:437–454, 1993.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[Rosencrantz *et al.*, 2004] Matthew Rosencrantz, Geoff Gordon, and Sebastian Thrun. Learning low dimensional predictive representations. In *ICML*, 2004.

[Schoenemann and Carroll, 1970] P. H. Schoenemann and R. Carroll. Fitting one matrix to another choice of a central dilation and a rigid motion. *Psychometrika*, 35(2), 1970.

[Scholkopf and Smola, 2002] B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[Wang, 1995] Xuemei Wang. Learning by observation and practice: an incremental approach for planning operator acquisition. In *ICML*, pages 549–557, 1995.

[Weinberger and Saul, 2004] K. Weinberger and L. Saul. Unsupervised learning of image manifolds by semidefinite programing. In *CVPR*, pages 988–995, 2004.