

# Speeding up Reinforcement Learning using Manifold Representations: Preliminary Results

Robert Glaubius, Motoi Namihira, and William D. Smart

Department of Computer Science and Engineering  
Washington University in St. Louis  
St. Louis, MO 63130  
United States

rlg1@cse.wustl.edu, mjn1@cec.wustl.edu, wds@cse.wustl.edu

## Abstract

Reinforcement Learning (RL) has proven to be a useful set of techniques for planning under uncertainty in robot systems. Effective RL algorithms for this domain need to be able to deal with large, continuous state spaces, and must make efficient use of experience. In this paper, we present methods to better leverage observed experience by reusing experience across parts of the problem state space that are known to be similar. We present experimental results in a navigational, goal-based domain. We develop an approach to identifying portions of the world that appear similar based on observed transition samples.

## 1 Introduction

Reinforcement Learning (RL) has proven to be a useful set of techniques for planning under uncertainty in robot systems. Effective RL algorithms for this domain need to be able to deal with large, continuous state spaces, and must make efficient use of experience. In previous work [5, 4] we have shown that a value-function representation (VFA) scheme based on manifolds can effectively deal with continuous state problems. In this paper, we briefly summarize this approach, and show how it can be used to make efficient use of experience in large domains of the type typically encountered in robotics applications.

One particular hurdle in real-world domains in general, and robotics in particular, is the expense of acquiring experience. Physical agents are subject to the cost of moving about in a real environment. These experiences are also constrained to occur around the agent’s trajectory through the state space – the agent may not arbitrarily sample the space without intervention.

However, one observation is that, particularly in navigation domains, much of the world behaves similarly. Traveling down one corridor is much the same as traveling down another, and hitting a wall should generalize between most walls. Continuous-state reinforcement learning does not handle these similarities between topologically similar states, and the agent must learn a local policy from scratch for each.

In this paper, we use local features of the manifold representation to allow the reuse across many states of experiences observed elsewhere during training. This extends our work on manifold-based value-function approximation by incorporating synthesized experience into the training data. By “replaying” experiences gained in one part of the world in other parts of the world that are similar, we expect to approach a good policy more quickly, and to require fewer actual experiences.

## 2 Background

In the RL framework, an agent interacts with the environment by observing the current environment state  $s$ , taking an action  $a$ , and observing the new state of the environment  $s'$  and a real-valued reward signal  $r$ . The evolution of states is governed by the transition function  $T(s, a)$ , which maps a state and action to a distribution over possible next states.

One popular approach to solving RL problems is to estimate the  $Q$ -value of each state-action  $(s, a)$  pair. The optimal state-action value function  $Q^*(s, a)$  is the expected sum of rewards observed when taking action  $a$  from state  $s$  and behaving optimally from then on. To obtain an optimal policy, the agent simply chooses the action that maximizes  $Q^*$  for its current state.

The update equation shown in equation 1 is the Q-learning rule [17]. Iteratively applying this rule to an initial estimate of the state action value function is guaranteed to converge to the optimal value function. Each update is based on a sample  $(s, a, r, s')$ , where  $s$  is some initial state,  $a$  is the action taken, and  $r$  and  $s'$  are the observed reward and resulting state, respectively. The learning rate  $\alpha$  controls the contribution of new experiences to the current estimate. The discount factor  $\gamma \in [0, 1]$  weights the effect of future rewards. One interpretation of  $\gamma$  is that it is the prior probability of surviving to the next time step.

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \max_{a \in A} Q(s', a)] \quad (1)$$

## 2.1 Related Work

$Q$ -learning is guaranteed to converge only in the case of an exhaustive tabular representation of the  $Q$  function, making the application problems with continuous state a non-trivial problem. Many real-world problems are most naturally modeled using a continuous state space; in this work we assume that the state space  $S$  is a connected subset of  $\mathbb{R}^d$ .

Although VFA has is not guaranteed to converge in the general case [3], there have been several successful applications reported, using a variety of function approximators. Tesauro used artificial neural networks to represent the state value function for the game of backgammon, resulting in an expert level of play [15]. Smart and Kaelbling used instance-based learning to represent the state-action value function for robot control tasks [11]. Munos and Moore used an informed decomposition of the state space to approximate the state value function with a combination of simple function approximators [10]. Sutton has successfully used tile coding approximators (often called CMACs [1]) in a variety of applications [12, 13].

Our VFA representation is most closely related to the tile coding approaches, in that it covers the domain of the value function with overlapping patches, and learns a partial value function on each of them. However, our approach takes the system dynamics into account when performing this covering, where tile coding typically does not. The work by Munos and Moore is also similar, in that it divides up the domain based on the system dynamics. They use non-overlapping states, however, and cannot make the strong continuity guarantees that our manifold-based approach can make.

In previous work we have argued that a manifold representation is sufficiently general to describe much of the existing corpus of methods for continuous-state value-function approximation in reinforcement learning. In this paper we present a method for reusing experience in portions of the state space that are similar, in the context of a manifold representation of the state space.

One way to view experience reuse is as a means to directly update the value of multiple states from one experience. Boutilier and Dearden [2] aggregate states during the construction of decision tree representations of the value function. In this sense, two states are equivalent if they have the same, or nearly the same, value. Our work differs in that it allows generalization between states where the system dynamics are the same, but values may be different.

Other researchers' work on exploiting the structure of the world has primarily focused on learning temporally extended actions to speed up learning [16, 8]. Lane and Wilson [7] derived conditions under which policies could be relocated in navigation tasks with relational domains. Our work differs in that we exploit knowledge about parts of the world that behave in a similar fashion to more efficiently use the experiences we have obtained.

## 2.2 Manifold Representations

In this section, we define precisely what we mean by "manifold". Before giving the formal definition of a manifold, we provide an intuition about the structure, and how it can be used constructively. Consider an atlas of the world. Each page in the atlas has a partial map on it, usually a single country. Each of these single-page maps can have a different scale, depending on the size of the country. This allows more

detail to be shown where it is necessary, without committing to representing the entire atlas at a particular resolution.

The pages in the atlas also overlap with each other at the edges. For example, the map for France has part of northern Spain on it. This provides a well-defined way of moving from one map page to another. Notice that the maps may be at different scales, and may not line up perfectly. However, we can still establish a correspondence between points in the overlap regions.

This method of using overlapping partial maps allows us to cover a complex surface (the surface of the Earth) with a set of simpler surfaces (the individual maps). Each local map can be appropriate to the local features, such as scale, and be topologically simpler than the global map. In the case of an atlas, we are covering a sphere with a set of (topological) disks. We can define global features, such as distance, by composing their local versions on each page, and dealing appropriately with the overlap regions.

We now make some definitions. Each map page in the above example is a *chart*. The collection of all charts is the *atlas*. The area shared by two adjacent pages is the *overlap region*. The function on each chart (for example, the elevation, or value function) is the *embedding function*.

**Chart** A homeomorphism  $\varphi$  from  $U \subseteq \mathcal{S}$  to a disk in  $\mathbb{R}^d$ . A chart can be thought of as a local coordinate system on  $U$ .

**Atlas** A set  $\Phi$  of charts. The manifold is defined as the union of the chart domains,  $\mathcal{M} = \bigcup_{\varphi \in \Phi} \text{dom}(\varphi)$ . In this work,  $\Phi$  will always be finite.

A manifold  $\mathcal{M}$  is constructed by covering the state space  $\mathcal{S}$  with an atlas. Assuming that the chart domains are selected appropriately,  $\mathcal{M}$  will represent the manifold structure of the state space. We refer the reader to our previous work on methods used to construct a manifold representation given an RL problem instance [5].

### 2.3 Manifolds for Value Function Approximation

The basic idea of using a manifold representation for value-function approximation (VFA) is to model small, local value functions on each of the charts, then combine these local models into a larger global model. This has a number of significant benefits, including the ability to explicitly model the topology of the problem domain. Again, since VFA is not the focus of this paper, we will forego a detailed explanation here, and refer the interested reader to our previous work [5] for more details.

It is important to note that our use of the term “manifold” is somewhat different from the one currently in vogue in the machine learning community. Commonly in this literature, a “manifold” is the intrinsic space that a collection of data points are drawn from. Usually this manifold has a lower dimension than the space it’s embedded in. Estimating the intrinsic dimension of the state space is not the focus of our manifold representations. Our usage is similar to the application of manifolds to surface modeling in computer graphics [6].

## 3 Speeding up Reinforcement Learning

The most relevant aspect of our value-function approximation scheme is the construction of a manifold representation. This models the problem state space as a set of smaller, overlapping chart domains. A local model, i.e., a function approximator, is embedded on each chart, and these local models are combined to give a global representation of the value function.

We can use this manifold structure to make more efficient use of the experiences gathered during training. Each of the charts covers an area of the state space that is locally self-similar. For many problem domains, these charts will form equivalence classes, where members of the class cover parts of the state space that are similar. For example, consider an empty room, with the robot state represented as its  $(x, y)$  position. If we cover this domain with relatively small (compared to the size of the room) charts, there will be intuitively three equivalence classes: “open space”, “wall”, and “corner”. Each chart is a member of exactly one of these classes.

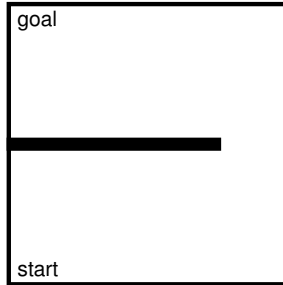


Figure 1: The experimental navigation domain.

We can define these equivalence classes according to the RL transition function over the charts. Two charts  $\varphi_i$  and  $\varphi_j$  are equivalent if the RL transition function, in their local coordinate frames, is “the same” in both. What do we mean when we say that the transition function is the same across two charts? We mean that the result of taking any action in  $\text{dom}(\varphi_i)$  moves the agent in the same direction, and the same distance, as that same action in  $\varphi_j$ . If we define equivalence in this way, we can reuse experience gathered in one chart in any chart that is equivalent to it.

Chart equivalence with respect to the transition function alone does ignore one important detail. In goal-based reinforcement learning, the reward function is tied to a region, or a set of regions. Any chart domain intersecting one of these regions can not be equivalent to one that is not, since generalizing a reward-bearing experience to a part of the domain that does not supply such a reward alters the problem that we are trying to solve. In the rest of this work, it is assumed that each chart intersecting a goal region is in its own singleton equivalence class.

Given this definition of equivalence does mean we have to revise our empty room example to include more equivalence classes. Charts that do not overlap a wall may still be close enough to a wall that an agent taking an action from inside that neighborhood may still strike a wall. This differentiates that particular chart from, say, an “open space” chart that is farther from walls. For now, we can work around this issue by restricting reuse only to experiences  $(s, a, r, s')$ , where  $s$  and  $s'$  have at least one chart domain in common, i.e., if there exists a chart  $\varphi$  such that  $s$  and  $s'$  are both in  $\text{dom}(\varphi)$ , then that experience generalizes to charts equivalent to  $\varphi$ .

If we assume that all charts are the same size, and that there is a single source of non-default reward, experience reuse is straightforward. Suppose we observe the experience  $(s, a, r, s')$ , where  $s, s' \in \text{dom}(\varphi_i)$ . Suppose that we also have a function  $\psi_{i,j}$  that maps experiences from  $\text{dom}(\varphi_i)$  to  $\text{dom}(\varphi_j)$  (in the case of our uniform-sized charts, this is just a translation). Then, for each chart  $\varphi_j$  equivalent to  $\varphi_i$ , we synthesize the experience  $\psi_{i,j}(s, a, r, s')$ , and update the local approximator on  $\varphi_j$  according to the synthesized experience. If there are many charts in a particular equivalence class, this will lead to a dramatic increase in learning speed.

## 4 Results

In this section, we present some experimental results to show the effectiveness of the techniques described above. Our experiments were performed in a simple navigation domain. The agent starts in the lower right corner of the world, and gets a reward of +10 for reaching the upper right corner, as shown in Figure 4. The agent can move in the four cardinal directions, with a reward of -1 on every step that does not end at the goal state. The state space of the problem is continuous, with two dimensions corresponding to the agent’s  $(x, y)$  position.

In these experiments, the world is a  $(0, 1)^2$  room with a wall with vertical width 0.05 dividing the center.  $0.05 \times 0.05$  charts were placed uniformly across the environment, with adjacent charts overlapping by 0.02. The local model on each chart is a single scalar value for each action. This chart allocation is similar to a tabular discretization, except that adjacent cells overlap.

The chart equivalence classes are precomputed; in the next section we will discuss methods for computing equivalence classes online. The only class of charts we consider equivalent in these experiments are

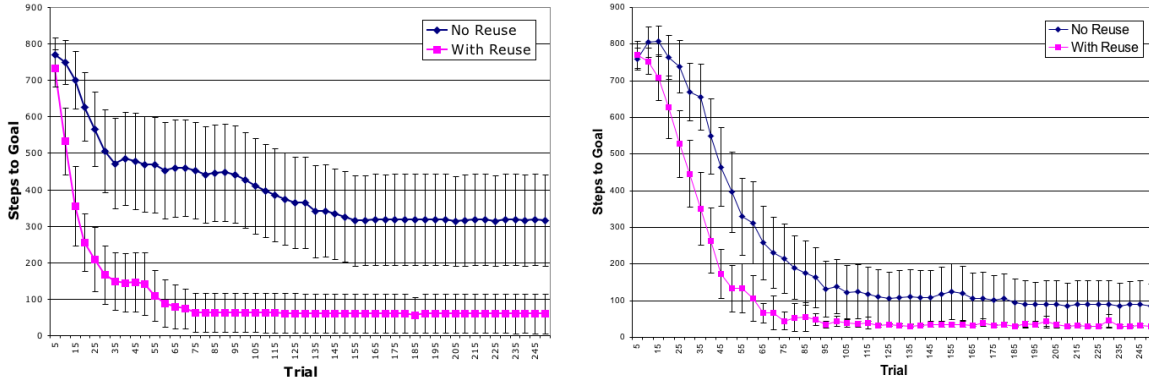


Figure 2: Comparison of performance with and without experience reuse on 2-d (left) and 3-d (right) navigation domains. Results are the mean of fifty experiments; 95% confidence intervals are shown.

the open space charts, i.e., charts that do not intersect any walls. There were 1225 total charts in the atlas, 979 of which were “open space” charts.

Experiments consist of a series of 250 trials. Each trial is terminated once the agent reached the goal or after 1000 steps. The agent starts each trial from a random point in a small disc, as shown in Figure 4. We compare the performance of an agent using experience reuse to one which does not. Value backups for experiences observed along the agent’s trajectory were performed according to the SARSA update rule [14], while synthesized experiences were backed up using the Q-learning update from Equation 1.

Figure 4 provides the results of our experiments. Experience reuse results in significantly better performance than is observed when experiences are not reused. Two of the experiments with experience reuse never found the goal; with these two outliers removed the performance in later trials improves from about 60 steps to to about 20 steps to reach the goal. In the no-reuse case, performance reached an expected 316 steps to goal about trial 250; 15 experiments without experience reuse failed to learn a good policy.

In our next set of experiments we increase the complexity of the navigational domain by incorporating the robot pose into the world state. The agent can drive forward or backward, or turn left or right  $\frac{1}{4}\pi$  radians then drive forward. The robot moves a maximum distance of 0.03 on each time step. The goal and start positions are changed in order to increase the chance that the agent will find the goal. The goal region is centered about  $x = 0.5$ ,  $y = 0.8$ , and starting points are selected from a disc about  $(0.2, 0.2, \frac{\pi}{2})$ .

The world is covered with  $0.08 \times 0.08 \times \frac{5\pi}{18}$  charts. Adjacent charts overlap by an extent of 0.04 in the  $x$  and  $y$  directions, and  $\frac{\pi}{10}$  radians along the  $\theta$ -axis. The total number of charts in the covering is 5013, 3437 of which do not overlap walls.

In these experiments we restrict ourselves again to experience reuse by translation of sample endpoints only. Since two charts open space with disjoint  $\theta$  intervals will not observe transitions with the same orientation between initial and end point, we limited experience reuse to open space charts with the same  $\theta$  interval. We could clearly enhance the amount of reuse by taking into account that some charts many charts in this domain are related by rotation as well as translation.

As the results in Figure 4 show, even this relatively naive implementation of reuse continues to significantly outperform the no-reuse case. The performance of each is much closer in the later trials in these experiments. It is likely that admitting more extensive reuse would increase this margin.

## 5 Detecting similar charts

The results with experience reuse shown in the previous section illustrates the merit of this approach. However, in practice it is unreasonable to expect that the agent has access to enough information about the world to determine equivalence classes *a priori*. The desired solution is to determine these equivalence classes based on the agent’s interaction with the environment. In order to achieve this goal, we present a chart similarity measure based on comparison of vector fields. In deterministic domains with continuous

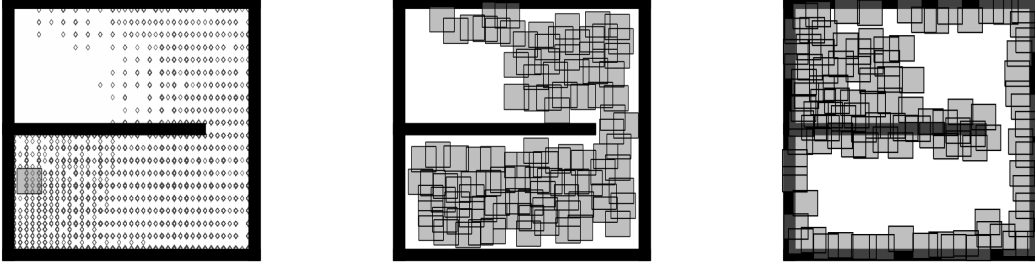


Figure 3: Automatically detected charts similar to a selected chart. Left: the collection of stored samples' initial points, and the selected chart. Center: Charts similar to the selected chart. Right: Charts dissimilar to the selected chart.

dynamics, the transition function  $T(s, a)$  for any choice of  $a$  induces a vector field on  $S$ .

Recall that the divergence  $\nabla F(\mathbf{x})$  for vector field  $F(\mathbf{x})$  is  $\sum_{i=1}^d \frac{\partial F(\mathbf{x})}{\partial x_i}$ . Based on this notion, one measure of difference between charts  $\varphi_i$  and  $\varphi_j$  related by  $\psi_{ij}$  is the squared difference in divergences, where  $A$  is the set of actions:

$$\int_{a \in A} \int_{s \in \text{dom}(\varphi_i)} [\nabla T(s, a) - \nabla T(\psi_{ij}(s), a)]^2 ds da \quad (2)$$

This gives us a measure that is rotation invariant. If we were to consider a more realistic version of the navigation environment in the previous section, the robot might have available actions “turn right”, “turn left”, and “go forward”, and state will be the pose  $\rho = (x, y, \theta)$ . In this case, two open space charts may be related by a rotation as well as translation. For now we restrict ourselves to similar charts that are related by translation, so we must amend our similarity measure to avoid rotational invariance.

$$\int_{a \in A} \int_{s \in \text{dom}(\varphi_i)} (\nabla[T(s, a) - T(\psi_{ij}(s), a)])^2 ds da \quad (3)$$

In practice we do not have access to the system dynamics. We do have a set of transition samples  $\Xi_\varphi = \{\xi = (s_\xi, a_\xi, r_\xi, s'_\xi) : s \in \text{dom}(\varphi)\}$ . At each sample transition  $\xi = (s_\xi, a_\xi, r_\xi, s'_\xi)$ , we can compute the divergence  $T(s_\xi, a_\xi)$  directly in terms of  $s'_\xi - s_\xi$ . Recasting Equation 3 for the sampled case, we get

$$D(\varphi_i, \varphi_j) = \sum_{\xi \in \Xi_{\varphi_i}} (\nabla[T(s_\xi, a_\xi) - T(\psi_{ij}(s_\xi), a_\xi)])^2 + \sum_{\xi \in \Xi_{\varphi_j}} (\nabla[T(\psi_{ij}^{-1}(s_\xi), a_\xi) - T(s_\xi, a_\xi)])^2 \quad (4)$$

This does present one problem, however; if we have a sample  $\xi \in \Xi_{\varphi_i}$ , it is likely that we do not have a sample rooted at  $\psi_{ij}(s_\xi)$ , much less with the same action  $a_\xi$ . In this case, some approximation is needed; for this paper we use the sample  $\psi^j \in \Xi_{\varphi_j}$  such that  $a_{\xi^j} = a_\xi$  that minimizes the distance  $d(s_\xi, \psi_{ij}(s_{\xi^j}))$ . More sophisticated approximations than this nearest neighbor approach are possible, but this straightforward approach appears to be sufficient in practice.

Figure 5 shows the set of charts similar to a selected chart in a manifold constructed from randomly-placed fixed-size charts. The sample sets used for each chart were obtained by executing 30 800-step random walks from a fixed starting point. In order to obtain an unbiased sample, the resulting sample sets were subsampled on each chart by generating a set of 30 points uniformly at random on each; the nearest sample to each point was retained. This was performed once for each action on each chart. Charts  $\varphi_i$  and  $\varphi_j$  were considered similar if  $D(\varphi_i, \varphi_j) < \frac{\sum_{\varphi_k \in \Phi} D(\varphi_i, \varphi_k)}{2|\Phi|}$ .

The figure demonstrates that, when the charts are well-sampled, the set of similar charts closely matches our intuition. However, errors may occur when charts are under-sampled. For instance, in Figure fig:equivs,

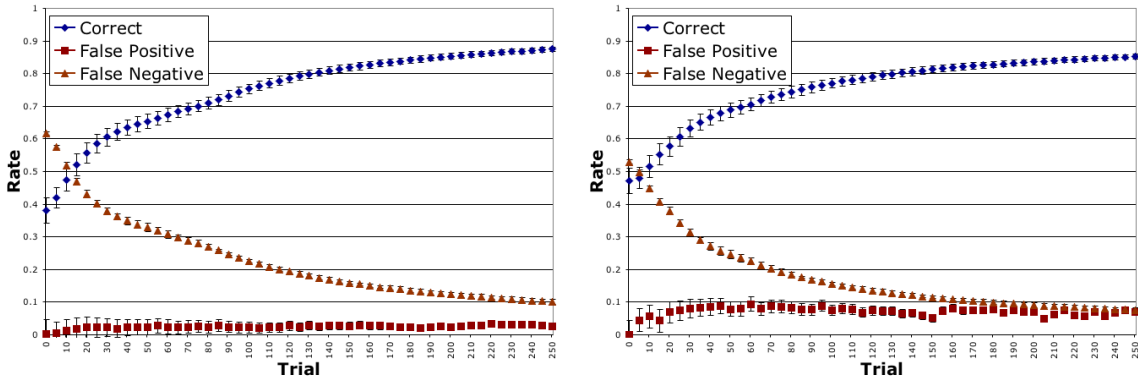


Figure 4: Classification performance on two example chart classes with respect to a hand-coded algorithm. Chart classes are open space (left) and bounded on the left by a wall (right).

we apply our similarity measure to an open space chart. However, some charts overlapping the upper wall are found to be similar. This may be due to the fact that any observed samples with action “up” were from an initial state near the bottom of the chart. Since such an action would not carry the agent all of the way to the wall, the chart appears to be open based on the observed samples. As more samples are collected, these misclassification errors should become less frequent.

Figure 5 compares the classification performance of this similarity metric against a hand-coded algorithm for determining equivalence classes. Performance is shown for two chart types in the 2-dimensional navigation domain – open space charts, and charts bounded on the left by a wall. Samples were collected by running a series of random walk trials. In each trial, the agent is initially placed in a small disc in the lower right-hand corner. The agent then performs a 1000-step random walk by selecting random actions. At the end of each trial, the sample set on each chart was subsampled as described above, so that as many as thirty samples per chart per action are retained.

The hand-coded algorithm for selecting equivalence classes groups charts that overlap a wall in the same way, i.e., do not intersect a wall, or all charts that are bounded on the left and above by walls, etc. This gives defines all of the obvious equivalence classes under translation. We do not expect the divergence-based similarity measure to achieve perfect accuracy with respect to the hand-coded equivalence measure, however. As was mentioned above, two charts that do not overlap a wall are still not necessarily equivalent with respect to the transition function, as it may be possible to take an action that results in the agent hitting a wall from one of the charts but not the other. The divergence measure is sensitive to this distinction, while the hand-coded policy is not.

## 6 Conclusions

The results shown in Section 4 demonstrate that a substantial increase in learning speed can be obtained from even a straightforward approach to experience reuse. Admitting a richer set of equivalence classes, such as “wall” and “corner” classes would present more opportunities for experience synthesis. Of course, as more charts are eligible for reuse, the cost of reuse increases. We have not discussed this cost up to this point, as we expect the largest expense to be acquisition of the training data from the world.

A potential improvement that we have not considered in this work is the order in which chart models are updated based on synthesized experiences. In the context of goal-based reinforcement learning, there is a natural extension to the manifold representation that will allow appropriate ordering. By constructing a graph with vertices corresponding to charts, with edges between overlapping charts, we can perform updates on synthesized data backwards from the goal in a breadth-first fashion. This should speed up the propagation of value back from charts containing the goal, and is similar to prioritized sweeping [9].

One limitation in our current experiments was the requirement that experience reuse only occurs for samples whose start and endpoint share a chart. In Section 5 we described one approach to determining

more robust chart similarity that will allow the reuse of samples that do not satisfy this constraint. By admitting a more complex relationship between samples on charts, we are likely to further increase the potential for experience reuse, but will likely require a more complex similarity measure.

Finally, we have alluded to the possibility of allowing more general sample transformations between charts. Thus far we have limited reuse to charts that are related by translation. We have seen a domain, the 3-dimensional navigation domain, charts are naturally related by rotation. Extending our methods to handle more general transforms will lead to more efficient reuse of experiences.

## References

- [1] J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement and Control*, pages 220–227, 1975.
- [2] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 2000.
- [3] J. A. Boyan and A. W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D. S. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 369–376, 1995.
- [4] R. Glaubius and W. D. Smart. Manifold representations for value-function approximation. In *Working Notes of the Workshop on Markov Decision Processes, AAAI 2004*, San Jose, California, USA, 2004.
- [5] R. Glaubius and W. D. Smart. Manifold representations for continuous-state reinforcement learning. Technical Report WUCSE-2005-19, Department of Computer Science and Engineering, Washington University in St. Louis, 2005.
- [6] C. M. Grimm and J. F. Hughes. Modeling surfaces of arbitrary topology using manifolds. *Computer Graphics*, 29(2), 1995. Proceedings of SIGGRAPH '95.
- [7] T. Lane and A. Wilson. Toward a topological theory of relational reinforcement learning for navigation tasks. In *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS-2005)*, 2005.
- [8] A. McGovern, D. Precup, B. Ravindran, S. Singh, and R. S. Sutton. Hierarchical optimal control of MDPs. In *Proceedings of the Tenth Yale Workshop on Adaptive and Learning Systems*, pages 186–191, 1998.
- [9] A. Moore and C. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.
- [10] R. Munos and A. Moore. Variable resolution discretization for high-accuracy solutions of optimal control problems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99)*, pages 1348–1355, 1999.
- [11] W. D. Smart and L. P. Kaelbling. Practical reinforcement learning in continuous spaces. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pages 903–910, 2000.
- [12] P. Stone and R. S. Sutton. Scaling reinforcement learning toward robocup soccer. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*, 2001.
- [13] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1038–1044, 1996.
- [14] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computations and Machine Learning. The MIT Press, Cambridge, MA, 1998.
- [15] G. J. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8(3/4):257–277, 1992.
- [16] S. Thrun and A. Schwartz. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*. MIT Press., 1995.
- [17] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.