From Pixels to Policies: A Bootstrapping Agent

Jeremy Stober and Benjamin Kuipers
Department of Computer Sciences
The University of Texas at Austin
1 University Station, Austin, TX 78712, USA
Email: {stober, kuipers}@cs.utexas.edu

***Abstract*—An embodied agent senses the world at the *pixel level* through a large number of sense elements. In order to function intelligently, an agent needs high-level concepts, grounded in the pixel level. For human designers to program these concepts and their grounding explicitly is almost certainly intractable, so the agent must learn these foundational concepts autonomously.**

We describe an approach by which an autonomous learning agent can bootstrap its way from pixel-level interaction with the world, to individuating and tracking objects in the environment, to learning an effective policy for its behavior. We use methods drawn from computational scientific discovery to identify derived variables that support simplified models of the dynamics of the environment. These derived variables are abstracted to discrete qualitative variables, which serve as features for temporal difference learning. Our method bridges the gap between the continuous tracking of objects and the discrete state representation necessary for efficient and effective learning.

We demonstrate and evaluate this approach with an agent experiencing a simple simulated world, through a sensory interface consisting of 60,000 time-varying binary variables in a 200 x 300 array, plus a three-valued motor signal and a real-valued reward signal.

I. Introduction

A central goal of developmental robotics is to show how an autonomous robot can go from the "blooming, buzzing confusion" of raw sensory experience to useful high-level knowledge. For common sensor configurations, a situated robot's experience of the real world is continuous and high-dimensional. For many tasks, the robot only needs access to a small set of state variables derived from this sensory signal, and by extension from the environment. A key part of this process is learning to track and act on objects in the environment.

To accomplish these basic tasks requires that an agent first develop a primitive understanding of its own sensorimotor capabilities, including the geometry of its sensor space. Previous work on sensorimotor reconstruction [1], [2] provides methods for generating geometric models of sensor arrays based on data collected during undirected experience. Even with known sensor geometry the agent faces a considerable challenge in extracting a useful model of world interaction from uninterpreted changes in its sensor field.

Our learning agent attempts to meet this challenge through a sequence of progressively more informative models of sensor state (Figure 1). The agent does not have access to the laws governing state evolution. It only has access to a high-dimensional visual signal representing the current sensory image of the environment (Figure 2).

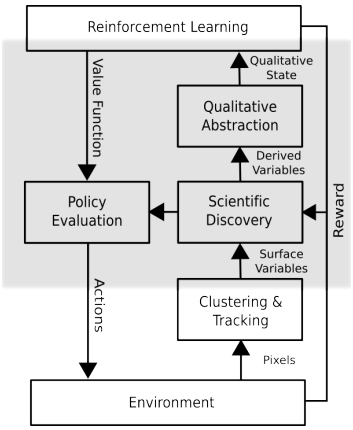

Fig. 1. Our agent generates a policy over high level states that are a result of bootstrapping from clusters in raw sensory space. Our primary contribution (boxed) is a method of qualitative abstraction over derived terms that bridges the gap between the continuous state representation of the underlying environment provided by trackers and discrete decisions that constitute a policy. We refine the state representation by evaluating the individual policy contributions of qualitative variables.

Inspired by the use of trackers to anchor perceptual symbols in the sensor stream [3], [4], we build and maintain a low-dimensional representation of the world state in terms of connected components in the visual image provided by the environment. To accomplish this, we adapt a method for tracking and categorization of objects used in [5]. The tracker model provides a low-dimensional description of the state of the environment in terms of the dynamic properties of objects that change over time, which we refer to as *surface variables*.

From a developmental robotics perspective, a low-dimensional, high-level representation of the world state provided by trackers and associated perceptual functions provides a non-task-specific *abstraction*. To test the efficacy of this abstract representation we consider the resulting worldstate representation in the context of a natural reward signal. The resulting domain is a reinforcement learning problem. Since each sensor image contains no velocity information, the raw state signal is not Markov.

We treat the dynamic properties (surface variables) of trackers in this domain as continuous. Reinforcement learning agents in continuous state spaces experience a limited subset of possible states during training. In order to extend a learning agent's policy to novel states, the agent must generalize from previous experience. Algorithm designers incorporate function

approximation, and its attendant state representation, into algorithms in order to accomplish this needed generalization. With existing methods of function approximation, a certain amount of design is required to marshall the state representation into a form that is both convenient and harmonious with the chosen method of function approximation.

Without a designer in the loop, a bootstrapping agent faces the difficult problem of autonomously determining the best state representation given the available methods of function approximation. In order to evaluate the efficacy of a representation, the agent must attempt policy search using the representation. The success of this policy search provides the feedback needed for representation search.

Given a set of continuous surface variables describing the behavior of the environment at a high level, we apply the classic methods of BACON [6] to generate new derived variables by simple algebraic transformations of the available surface variables. Then, in order to transform the continuous description into a discrete form convenient for policy learning, we apply methods from qualitative reasoning [7] to form qualitative variables using natural landmarks.

We demonstrate how the resulting state representation can be evaluated in terms of the policy contributions of the derived qualitative state variables, and how, if necessary, qualitative state variables can be pruned from the representation. With autonomous methods both for generating representations and for pruning those representations, our agent can couple its search for a good policy with its search for a good representation.

Methods for clustering and tracking are varied and well studied [8]. Solving reinforcement learning problems through temporal difference methods are also explored extensively in the literature [9], [10]. Our main contribution is providing a method of autonomously bridging the gap between knowledge of state grounded in object tracking and the state representation required for effective temporal difference learning. We divide this contribution into two components drawn from existing literature, computational scientific discovery [6] and qualitative abstraction [7], which generate new candidate state features, and a novel component, *representation refinement*, that seeks to refine the choice of state representation for the bootstrapping agent.

II. Agent Architecture

A. Formation of Trackers

In the simulated domain shown in Figure 2, the agent begins the bootstrapping process by learning a pixel-based static-world model. During the learning period, the agent determines the mode (most common value) $\bar{z}^i$ for each pixel z^i in the pixel vector z based on observational experience. Using this static-world model of the environment, the agent defines a stochastic model of its experience. The pixel vector at time t is given by

$$z_t = \bar{z} + \epsilon_1 \tag{1}$$

where ϵ_1 is a random variable describing the discrepancy between the prediction $\bar{z}$ and the observation z_t. An *active* pixel is any pixel which violates the static world model, i.e. $z_t^i \neq \bar{z}^i$. In this model, dynamic change is treated as noise, described only by the ϵ_1 term.

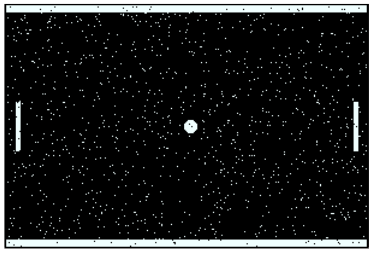

Fig. 2. We evaluate our approach using a simulated video game environment whose primary sensory output is a grid of 300x200 binary pixels. The agent must discover some low-dimensional representation of the environment state based on this visual image. The agent controls both paddles, which it can synchronously move up, down, or hold steady. The puck bounces off the top and bottom walls, and exits through the ends. The agent receives a reward of $+1$ every time the agent hits the puck with a paddle and a reward of -1 every time the puck leaves the playing surface. When the puck leaves the field it is reset to the center with an x velocity of 3.0 and a random y velocity in the range $(-0.5, +0.5)$.

During the next step of bootstrapping, the agent must account for the discrepancy between the background model $\bar{z}$ and present state of the agent's sensors. By finding connected components of active pixels, the agent constructs a model that accounts for the current active pixels. From each connected component, the agent extracts a set of properties, including the centroid position, area, and perimeter for each component [11]. We denote functions that extract properties from active pixel clusters *perceptual functions* and the resulting properties *percepts*.

The percepts for each component provide the agent with a model which accounts for some active pixel discrepancies. Each new timestep, however, results in a new set of discrepancies, and therefore a new set of connected components. To account for temporal discrepancy, the agent associates connected components by identifying clear successor components in subsequent timesteps. Temporal association of component feature sets gives rise to the notion of *trackers*.

Each tracker maintains a set of static and dynamic percepts. For a new tracker τ, all percepts for a connected component are in the the set of static percepts S_τ. The set of dynamic percepts D_τ is initially empty. The tracker maintains a set of percepts $\Delta_{\tau,t}$ that change at timestep t. At time T, the set of static and dynamic tracker percepts are given by

$$S_\tau = \bigcap_t^T (P - \Delta_{\tau,t}) \text{ and } D_\tau = \bigcup_t^T \Delta_{\tau,t}$$

where P denotes the set of all percepts.

At each time-point, the percepts of each newly-identified component are associated with an existing tracker if there is a clear best match within a threshold δ. Otherwise, a new tracker is created. We extend this by allowing a tracker to lose sensory support for a short time period, provided that the tracker reacquires sensory support within the same threshold. The current time-limit for tracker persistence is two timesteps. This method of temporal association is based on resource allocating

TABLE I

THREE CONNECTED COMPONENTS EMERGE FROM CLUSTERING ACTIVE PIXELS. WE USE THE STATIC PERCEPTS TO IDENTIFY THE ORDERING OF DYNAMIC STATE AT THE BEGINNING OF EACH EPISODE. THE DYNAMIC PERCEPTS ARE *surface variables*.

	Percepts	
Object	Static	Dynamic
Puck	$area_p, perimeter_p$	x_p, y_p
Left Paddle	$area_l, perimeter_l, x_l$	y_l
Right Paddle	$area_r, perimeter_r, x_r$	y_r

TABLE II

THE DERIVED VARIABLES RESULTING FROM APPLICATION OF DERIV AND DIFF TO SURFACE VARIABLES FROM TABLE I.

DERIV	$d(x_p), d(y_p), d(y_l), d(y_r)$
DIFF	$y_l - x_p, y_l - y_p, y_l - y_r$ $y_r - x_p, y_r - y_p, x_p - y_p$

vector quantization [12], though we expect that other forms of unsupervised clustering would work equally well [13], [14].

Between episodes in the reinforcement learning problem described below, the puck and paddles are reset to their respective starting positions. This discontinuous change results in new trackers. We use static percepts to establish a correspondence between new and old tracker state variables. Table I describes the distinguishing characteristics of dynamic objects in the environment.

By using trackers, the agent can create a model

$$z_t = \bar{z} + \sum_{\tau} \phi(\tau) + \epsilon_2 \tag{2}$$

that combines the previous static model $\bar{z}$ with the projection into pixel space ϕ of each currently active tracker τ. Pixels that are not explained by the tracker or static models are treated as noise, described by the ϵ_2 term.

B. Computational Scientific Discovery

As a result of tracking and categorization, the agent has access to a stable set of four continuous dynamic state variables corresponding to the puck coordinates (x_p, y_p) and the vertical locations y_l and y_r of the two paddles. In order to evaluate this state representation, we appeal to the natural reward structure of the simulated video game, and examine how well a reinforcement learning algorithm performs given this state representation.

Since applying temporal difference methods to continuous state spaces directly is problematic, our agent must first generate a discrete representation of game state. Inspired by the results of computational scientific discovery [6], and the observation that good representations may require concepts that are not apparent from surface variables alone, the agent uses two heuristics to generate a set of derived variables. The first heuristic, DERIV, calculates the derivative $d(s_i)$ of each continuous surface variable. The second heuristic, DIFF, adds to this set the differences $s_i - s_j$ between every pair of continuous surface variables. The complete set of derived variables appears in Table II.

We note that for this scenario, the continuous state representation is Markov when we include derivative terms to the set of variables describing the game state.

C. Qualitative State

Due to the difficulty of learning from the continuous state space directly, we adopt a method of discretization drawn from qualitative reasoning [7]. After application of DERIV and DIFF, we generate qualitative values based on the resulting set of derived terms. These terms have natural landmark values at zero, indicating the steady state in the case of derivative terms and equality in the case of difference terms. Each landmark value divides the domain of each derived variable into three qualitative values, $[-], [0], [+]$, which serve as features for reinforcement learning.

Since the surface variables offer no natural landmarks, we do not include them in the set of qualitative features for reinforcement learning. Extensions to this method that allow for landmark distinctions based on experience are discussed later.

D. Reinforcement Learning

The process described above constitutes a method of state abstraction, taking 60,000 binary pixels (or $2^{60,000}$ possible states) to ten qualitative variables consisting of three values each (or 3^{10} possible states). In every state, the agent has three choices for actions $\mathcal{A} = \{up, down, steady\}$. To apply a method for solving the reinforcement learning problem described below, we consider each qualitative variable as a feature which can take on three values. We then apply linear gradient descent Sarsa(λ) over the qualitative feature set $\mathcal{QF}$ by associating each element of $\mathcal{QF} \times \mathcal{A}$ with a weight $\theta : \mathcal{QF} \times \mathcal{A} \longrightarrow \Re$ (Algorithm 3). The value of a given state and action is the sum of the currently active weights indexed by $\mathcal{QF} \times \mathcal{A}$.

E. Representation Refinement

Complete cognitive architectures that utilize temporal difference methods for reinforcement learning need to generate expressive representations over which value function estimation via temporal difference learning can take place. However, some of the generated features may not be necessary when learning an optimal policy. In fact, additional features increase the size of the search space and so can slow or disrupt learning. We utilize a method we term *representation refinement* (Figure 4) in order to prune the features generated in the application of DERIV, DIFF and qualitative abstraction.

By running with only subsets of the qualitative feature set enabled, an agent can evaluate the contribution of individual qualitative state variables to its overall performance. We refer to this as the *effective policy contribution* of a variable. We evaluate two heuristics for determining the effective policy contribution of a qualitative variable. The first, termed *leave one out*, disables the weights associated with the qualitative variable under consideration for removal. If the performance of the agent policy, e.g. the reward accumulated over the

$\theta \longleftarrow \bar{0}$
for all episodes **do**
  $\bar{e} \longleftarrow \bar{0}$
  $s, a \longleftarrow$ initial state and action of episode
  $\mathcal{QF}_s \longleftarrow$ qualitative values present in s
  repeat
    for $i \in \mathcal{QF}_s$ **do**
      $e(i,a) \longleftarrow e(i,a) + 1$
    end for
    Take action a, observe reward r and state s
    $\delta \longleftarrow r - \sum_{i \in \mathcal{QF}_s} \theta(i,a)$
    $\mathcal{QF}_s \longleftarrow$ qualitative values present in s
    for $b \in \mathcal{A}$ **do**
      $Q_s(b) \longleftarrow \sum_{i \in \mathcal{QF}_s} \theta(i,b)$
    end for
    $a \longleftarrow argmax_{b \in \mathcal{A}}(Q_s(b))$
    $\delta \longleftarrow \delta + \gamma Q_s(a)$
    $\bar{\theta} \longleftarrow \bar{\theta} + \alpha\delta\bar{e}$
    $\bar{e} \longleftarrow \gamma\lambda\bar{e}$
  until episode ends
end for

Fig. 3. Linear gradient-descent Sarsa(λ) with qualitative features as applied to the simulated Pong domain. The set of weights θ and the eligibility traces $\bar{e}$ are indexed by the set of active qualitative values in the current state and action being considered. Note that the set of active qualitative values $\mathcal{QF}_s$ is updated after δ is initialized, since δ needs to be initialized using the weights associated with the state and action from the previous iteration. The $argmax$ of the action-value function Q_s determines the best action in the current state. $\gamma = 0.9$ is the discount factor used to determine the present value of future reward. $\alpha = 0.01$ is the learning rate, which determines the extent that temporal difference error affects the feature weights. $\lambda = 0.9$ determines the decay rate of the eligibility traces $\bar{e}$, modulating the amount of credit a past action should receive for a current reward. Adapted from [9].

episode, remains the same, we remove the qualitative variable from the representation. The second heuristic, termed *leave one in*, enables only the weights associated with the variable being evaluated. If the reward accumulated over the episode remains the same, indicating that the enabled weights encode a good policy, we keep the qualitative variable as part of the representation.

We note that the *leave one in* heuristic optimistically assumes that good policies do not depend on the interaction of two or more variables. The *leave one out* heuristic addresses this concern, but in practice seems to result in a more conservative refinement process, one that does not remove as many features. This method demonstrates correspondingly weaker performance in our experiments below.

Representation refinement uses the reward signal to evaluate the effective contribution of each state, though instead of updating a value function, we use reward received as a criteria for refining the state representation. This reduces the expressive power of our agent's representation, thus limiting the search space to only those states that demonstrate policy relevance. Our method of discovering irrelevant features falls into the class of π^*-irrelevant (policy irrelevant) abstraction methods [15].

$\mathcal{F} \longleftarrow$ set of state features
$refined \longleftarrow$ **false**
for all $episodes$ **do**
  $train\ policy\ over\ \mathcal{F} \times \mathcal{A}$
  if $episodic\ reward \geq threshold$ and $!refined$ **then**
    for all $f \in \mathcal{F}$ **do**
      if $contribution(f) \leq threshold$ **then**
        $remove\ f\ from\ \mathcal{F}$
      end if
    end for
    $refined \longleftarrow$ **true**
  end if
end for

Fig. 4. A pseudo-code description of the *representation refinement* process. This process requires a method for evaluating the policy contribution of a state feature and a method for measuring cumulative reward over an episode.

III. Evaluation

Using this qualitative state representation, and a reward signal that is $+1$ when the puck hits a paddle, -1 when the puck leaves the playing surface, and 0 otherwise, we train an agent using $Sarsa(\lambda)$ linear gradient descent with greedy action selection (Algorithm 3) using parameters $\lambda = 0.9, \gamma = 0.9, \alpha = 0.01$. At the beginning of an episode, both paddles and the puck are reset to their center positions. The x velocity of the puck is set to 3.0 and the y velocity is uniformly drawn from $(-0.5, +0.5)$. An episode ends when the puck leaves the playing surface. To limit computation time, episodes are capped at 1000 timesteps.

We compared four agents: a motor-babbling (random action) agent, an agent using $Sarsa(\lambda)$ and the full learned state representation (Table II), and two agents using $Sarsa(\lambda)$ with variants of representation refinement (Figure 4). We set the reward threshold to five. When an agent using representation refinement passes the threshold, it halts training and evaluates the effective policy contribution of each variable with either the *leave one in* or *leave one out* methods.

We evaluate the performance of the agent over 30 episodes, where performance is measured by the number of hits per episode. Out of 200 random trials, 168 of the agents using the entire state representation learn policies that reach the episode cap in at least one episode. In the aggregate, these agents clearly demonstrate policy improvement over the course of 30 episodes. Agents using representation refinement to improve their state representation perform better than agents using the entire state representation, since these agents limit their search for better policies to the state space most relevant to maximizing reward.

We found that representation refinement using the *leave one out* method resulted in refined representations that preserved, on average, more features than the *leave one in* method. We expect that the differing sizes of the resulting refined spaces explains the difference in performance between the two methods of refinement, and the agent that does no refinement. We also note that though refinement is a policy preserving process,

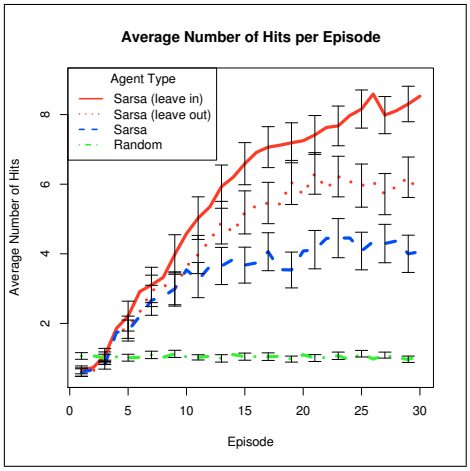

Fig. 5. We ran 200 trials of an agent using Sarsa($\lambda = 0.9$) for 30 episodes each. The mean performance per episode is shown, along with 95% confidence intervals. With each *representation refinement* strategy, the agent tested the policy contribution of each state variable after it achieves at least 5 hits in an episode, discarding those with minimal contribution. We omit the episodes used for representation refinement since no policy improvement takes place while testing the effective policy contribution of each state variable. We include a random agent for comparison. In this experiment, all learning agents perform significantly better than random, and agents that employ some form of representation refinement perform better than agents that do no representation refinement.

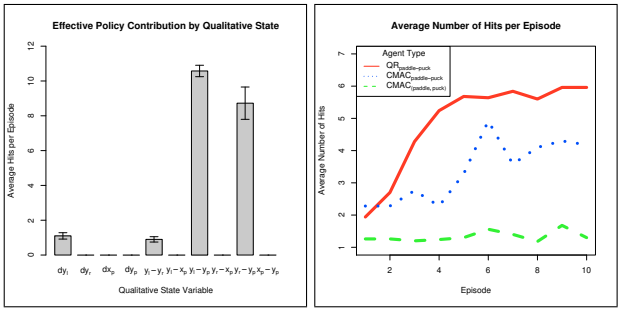

Fig. 6. The left chart shows the average number of hits over 40 episodes for each qualitative state variable using the *leave one in* method. The policy contributions of $y_r - y_p$ and $y_l - y_p$ generate significantly better performance than any other qualitative state variables. We view the "simplicity" of the best policy's state representation as indicating that our agent is employing the correct primitive operations to bridge the gap between surface variables and state features. On the right we compare a Sarsa(λ) agent using a qualitative representation of $y_{paddle} - y_{puck}$ space to an agent using tile coding (CMAC [16]) over the same space and over (y_{puck}, y_{paddle}). The number of hits for the first 10 episodes of training were averaged over 50 trials. We note that the discovery of the derived term $y_{paddle} - y_{puck}$ provides a significant increase in learning performance over the agent using tile coding in (y_{puck}, y_{paddle}) space. In addition, agents using a qualitative representation show increased performance over tile coding in the derived $y_{paddle} - y_{puck}$ space.

it is not a value function preserving process. High temporal difference error after refinement may serve to reinforce the refined policy, again resulting in a performance benefit.

We ran an additional experiment to compare the relative contributions of discovery and qualitative reasoning to learning performance over relevant state variables (Figure 6). We found that agents using tile coding and qualitative representations of $y_{paddle} - y_{puck}$ performed significantly better than the agents using tile coding over the original (y_{paddle}, y_{puck}) space. The use of a qualitative representation provided an additional learning performance increase over tile coding in the derived (discovered) state space.

As Figure 7 illustrates, the number of potential states in a naive tile coding representation of the state space of surface variables far exceeds the number required to capture useful policy components in this domain. Successful policies in our domain seek to maintain an invariant $y_{paddle} - y_{puck} = 0$ in the derived qualitative state of the game. We expect that synergies between the choice of state representation and effective policies may arise in many domains where rewards correlate with maintaining (possibly derived) invariants in the underlying dynamical system.

A bootstrapping agent autonomously generating its own state representation prior to reinforcement learning has no guarantee that all the qualitative states it generates are relevant to reward-maximizing policies. Through representation refinement, an agent is able to refine the state representation. In Figure 6, we see that the qualitative state $[y_{paddle} - y_{puck}]$ has the most impact on policy effectiveness in this domain.

IV. Discussion and Future Work

The qualitative state representation presented above provides an alternative to parametrized methods of sparse coding [17] and is similar to methods of defining discrete state in terms of distinguished values [18], [19]. A bootstrapping agent, however, cannot depend on a prior specification of "distinguished" values, or a highly-parametrized task-specific representation, since the continuous state space and task are unknown prior to the application of bootstrapping.

Our method relies on heuristics to generate derived terms and to define natural landmarks for the qualitative state representation. Scaling to larger physical domains will likely require the application of more heuristics for generating derived variables [6]. In addition, the division of qualitative variables into $[-], [0], [+]$ may not include the most useful underlying qualitative distinctions in other domains. Even if these simple qualitative distinctions are sufficient for representing good policies, they may not be sufficient for learning them [20]. We expect that methods of landmark identification [21], which have shown utility in planning over qualitative states [22], would allow this approach to extend to domains with non-zero qualitative landmarks.

The discovery of derived terms through data-driven exploration of the relationships between continuous state variables can potentially lead to a large number of state representations. We presented two methods above of evaluating the effective contribution of individual qualitative state variables (or subsets of variables). Pruning the state representation using these methods leads to better policies (Figure 5).

We expect that the heuristics we present for generating derived variables, and the qualitative distinctions we impose on those variables, would apply equally well to other scenarios involving high dimensional inputs, Newtonian physics and

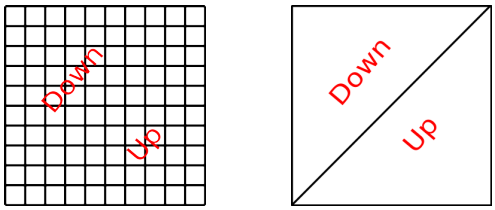

Fig. 7. Almost all successful policies employed to some extent the simple skill of tracking the y position of the puck with the paddles. Such policies attempt to minimize $y_{paddle} - y_{puck}$. We note in the above diagram that a naive tiling of (y_{paddle}, y_{puck}) space, unlike the derived qualitative state representation, would result in many more states than are required to represent this policy. We expect that methods which subdivide the ranges of existing variables in order to bridge the gap between surface variables and state features (e.g. tile coding) would require more experience to learn proper policies than the approach presented here.

complete observability. More complex domains may require improved heuristics for generating derived terms. The BACON system, which served as inspiration for our discovery method, was able to derive many complex physical laws (e.g. the ideal gas law [23]) from observations of surface variables.

In addition to a wider array of derived variables, we also expect that more complex domains would require more sophisticated methods of dividing the ranges of derived variables (e.g. parti-game [24], U-Trees [20], G Algorithm [25]). As we illustrate in Figure 7, such methods are not necessarily sufficient for generating efficient state representations, as they do not generate new state variables.

We also intend to explore extended coupling of representation refinement with the discovery of new derived terms, leading to the notion of *representation iteration*. The work above can be seen as one iteration of representation iteration, where we first generate a set of hypothesis state variables, then refine that set by attempting to train policies using TD learning over that set. An agent may repeat these two steps when, for example, new objects enter into the agent's perceptual field.

In much of the theoretical work on algorithms for reinforcement learning, the state representation is assumed to be fixed and given. In applications, however, the proper choice of state representation (and function approximation method) is a critical component of successful solutions to many reinforcement learning problems. Designers face a wide array of choices for both state and value function representations, choices that a bootstrapping agent must make autonomously. Though different domains may require the inclusion of different perceptual functions, clustering, and tracking algorithms, the work presented here provides a principled, automated method of going from pixel level observations to policies.

Acknowledgments

This work has taken place in the Intelligent Robotics Lab at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Intelligent Robotics lab is supported in part by grants from the Texas Advanced Research Program (3658-0170-2007), from the National Science Foundation (IIS-0413257, IIS-0713150, and IIS-0750011), and from the National Institutes of Health (EY016089).

References

[1] D. Pierce and B. Kuipers, "Map learning with uninterpreted sensors and effectors," *Artificial Intelligence*, vol. 92, no. 1-2, pp. 169–227, 1997.

[2] L. Olsson, C. Nehaniv, and D. Polani, "From unknown sensors and actuators to actions grounded in sensorimotor perceptions," *Connection Science*, vol. 18, no. 2, pp. 121–144, 2006.

[3] Z. Pylyshyn, "The Role of Location Indexes in Spatial Perception:A Sketch of the FINST Spatial Index Model," *Cognition*, vol. 32, pp. 65–97, 1989.

[4] S. Coradeschi and A. Saffiotti, "An introduction to the anchoring problem," *Robotics and Autonomous Systems*, vol. 43, no. 2-3, pp. 85–96, 2003.

[5] J. Modayil and B. Kuipers, "Autonomous development of a grounded object ontology by a learning robot," in *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07)*, 2007.

[6] P. Langley, H. Simon, G. Bradshaw, and J. Zytkow, *Scientific Discovery: Computational Explorations of the Creative Processes*. MIT Press, 1987.

[7] B. Kuipers, *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, 1994.

[8] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys (CSUR)*, vol. 38, no. 4, 2006.

[9] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[10] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial IntelligenceResearch*, vol. 4, pp. 237–285, 1996.

[11] W. Snyder and H. Qi, *Machine Vision*. Cambridge University Press, 2004.

[12] F. Linaker and L. Niklasson, "Sensory Flow Segmentation using a Resource Allocating Vector Quantizer," *Advances in Pattern Recognition: Joint IAPR International Workshops SSPR2000 and SPR2000*, pp. 853–862, 2000.

[13] B. Fritzke, "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems*, vol. 7, pp. 625–632, 1995.

[14] J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, 1991.

[15] L. Li, T. Walsh, and M. Littman, "Towards a unified theory of state abstraction for MDPs," *Proceedings of the ninth international symposium on AI and mathematics*, 2006.

[16] J. Albus *et al.*, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *Journal of Dynamic Systems, Measurement and Control*, vol. 97, no. 3, pp. 220–227, 1975.

[17] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," *Advances in Neural Information Processing Systems*, vol. 8, pp. 1038–1044, 1996.

[18] D. Michie and R. A. Chambers, "BOXES: An Experiment in Adaptive Control," in *Machine Intelligence 2*. Edinburgh University Press, 1968, pp. 137–152.

[19] A. Barto, R. Sutton, and C. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Computer Society Neural Networks Technology Series*, pp. 81–93, 1990.

[20] A. McCallum, "Reinforcement Learning with Selective Perception and Hidden State," Ph.D. dissertation, University of Rochester, 1996.

[21] U. Fayyad and K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, vol. 2, 1993.

[22] J. Mugan and B. Kuipers, "Learning distinctions and rules in a continuous world through active exploration," in *7th International Conference on Epigenetic Robotics (Epirob-07)*, 2007.

[23] P. Langley, G. Bradshaw, and H. Simon, "Rediscovering chemistry with the BACON system," *Machine Learning: A Multistrategy Approach*, 1994.

[24] A. Moore and C. Atkeson, "The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces," *Machine Learning*, vol. 21, no. 3, pp. 199–233, 1995.

[25] D. Chapman and L. Kaelbling, "Input generalization in delayed reinforcement learning: An algorithm and performance comparisons," *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp. 726–731, 1991.