Where do actions come from? Autonomous robot learning of objects and actions

Joseph Modayil and Benjamin Kuipers
Department of Computer Sciences
The University of Texas at Austin

Abstract

Decades of AI research have yielded techniques for learning, inference, and planning that depend on human-provided ontologies of self, space, time, objects, actions, and properties. Since robots are constructed with low-level sensor and motor interfaces that do not provide these concepts, the human robotics researcher must create the bindings between the required high-level concepts and the available low-level interfaces. This raises the developmental learning problem for robots of how a learning agent can create high-level concepts from its own low-level experience.

Prior work has shown how *objects* can be individuated from low-level sensation, and certain properties can be learned for individual objects. This work shows how high-level *actions* can be learned autonomously by searching for control laws that reliably change these properties in predictable ways. We present a robust and efficient algorithm that creates reliable control laws for perceived objects. We demonstrate on a physical robot how these high-level actions can be learned from the robot's own experiences, and can then applied to a learned object to achieve a desired goal.

Motivation

This paper proposes a method for a robot to autonomously learn new high-level actions on objects. The motivation for this work is to understand how a robot can autonomously create an ontology of objects that is grounded in the robot's sensorimotor experience. This developmental learning approach is inspired by the observation that infants incrementally acquire capabilities to perceive and to act (Mandler 2004; Spelke 1990).

High-level symbolic AI has many techniques for learning, inference, and planning that operate on representations of self, space, time, objects, actions, and properties (Russell & Norvig 2002). However, robots have low-level sensor and motor interfaces that do not provide these conceptual abstractions. This raises the developmental learning question of how an agent can automatically generate these abstractions. Prior work has demonstrated methods for building ontologies of self and space (Pierce & Kuipers 1997; Philipona, O'Regan, & Nadal 2003). Building on research that demonstrates how objects representations can arise (Modayil & Kuipers 2004; 2006), this work demonstrates how actions for objects can be acquired.

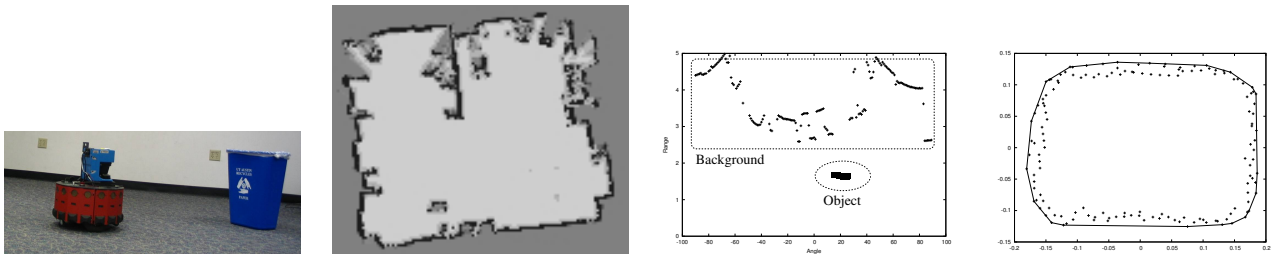
We propose to create actions by learning control laws that change individual perceptual features. First, the algorithm uses an unsupervised learning method to find an effective threshold of change for the feature. This threshold is used to find perceptual contexts and motor commands that yield a reliable change in the perceptual feature. After control laws are generated, the robot autonomously tests their performance.

The following sections describe the problem formulation, the algorithm, and the evaluation of the autonomous learning process.

Actions for Objects

People and robots use a finite set of sensor and actuator capabilities to interact with the effectively infinite state of the environment. To manage the inherent complexity, people generate high-level abstractions to facilitate reasoning about a problem and planning a solution. Abstraction must also occur within a developing agent in order to learn models of the reliable portions of its sensor and motor experience. As a concrete example, an infant gradually learns to perceive nearby objects and to manipulate them into desired states. Previous work has demonstrated how a robot can autonomously learn models of objects. These models introduce new state variables that the robot can compute from its observations, but it does not yet know how to control. This work shows how the robot can learn at least partial control over these new state variables.

This work thus helps to bridge the ontological gap between the representations of actions used in high-level logical formulations and those used for low-level motor control. At the high level, actions can be represented with add-lists and delete-lists for STRIPS-style world descriptions with discrete symbolic states. This representation is useful for creating deep plans with multiple types of actions, but it relies on the existence of hand-crafted control laws to realize the actions in the continuous world. At the low level, actions are often represented as control laws or forward motion models operating on continuous low-dimensional state spaces. These representations support motion planning, but do not explain how new state representations can be incorporated.



(a) Scene (b) Background (c) Sensor Image (d) Object Shape

Figure 1: A scene can be explained with a static background and moving objects. The sensor returns from a rigid object can be used to form a consistent shape model. The robot can infer the object's pose by using the shape model and the sensor image and the map.

In summary, people and robots both operate in a continuous world with a fixed set of actuator capabilities. Abstractions of this continuous experience can facilitate planning. The abstraction of the environment into objects is particularly useful for robots. For this paper, we demonstrate how a robot can autonomously acquire high-level object-directed actions that are grounded in the robot's sensor and motor system.

Object control laws for a developing robot

The complexity of the real world presents a daunting challenge for an autonomous agent attempting intelligent behavior. The agent must reason with finite resources about the consequences of actions while the results of procedural routines for perception and action depend implicitly on the effectively infinite state of the world. One approach to tackling this problem comes from looking at the origins of natural intelligence. Infants develop skills and concepts from unsupervised interaction with the environment. Some perceptual and motor skills are provided by nature (and thus learned on an evolutionary time-scale), while other skills and concepts are learned autonomously by the individual (Mandler 2004; Spelke 1990).

Work along these lines for robots have explored different aspects of development, including the formation of ontologies of space for the robot's body and environment (Pierce & Kuipers 1997; Philipona, O'Regan, & Nadal 2003). Other work has shown how the agent is capable of describing its observations of the dynamic environment in terms of objects (Biswas *et al.* 2002; Modayil & Kuipers 2004), but they do not provide actions that utilize these representations.

We describe the questions that must be addressed by an object representation system for robots. Since this work builds on the approach and representations developed in (Modayil & Kuipers 2004), we describe how these questions are answered by that work and how the approach is extended here.

What types of objects can the robot perceive? Non-static objects are perceived by identifying failures of a static world model. The robot maintains an occupancy grid (Figure 1b) that represents portions of space that have been observed to be vacant. When a sensor reading indicates this space is not vacant at a later time, then something must have moved into the previously vacant space. Thus, the presence of non-static objects is indicated by inconsistencies between the static world model and the robot's observations.

What objects are perceived in the scene? People are skilled at detecting objects (Figure 1a). The task is much harder for robots. Objects can be perceived (and defined) by clustering sensations that come from objects. In this approach, object hypotheses arise as explanations of sensorimotor experience, in particular, as explanations of the portion of experience that is not modeled by allocentric models of the static environment, but are sufficiently temporally stable to improve the agent's predictive capabilities.

Where are the objects located? The location of an object can be represented in multiple ways. In sensor coordinates, the image of the object has a location in the sensor array (Figure 1c). In world-centered coordinates, the object's center of mass has a location, and it has an extent. If the object has a rigid shape, then its orientation (hence pose) may also be represented.

What properties does an object have? A property for an individual object can be either variable or invariant. The position and orientation of an object is typically variable, while an object is classified as rigid if its shape is invariant. When an agent learns to act on objects, the conditions under which the action is successful can become an additional property of the object, its *affordance* for that action (Gibson 1979).

What is the identity of each object? An invariant property of an object may serve as a characteristic property, helping to determine when two observations are of the same individual object. A rigid object can be characterized by its shape (Figure 1d) (Modayil & Kuipers 2006), while constellations of features can characterize non-rigid objects such as faces and fingerprints.

Which properties can be controlled and how? What has not been shown in prior work is how these newly defined properties can be controlled. A robot may be able to reliably change a variable property of an object. We define

	Property	Dim
Robot pose	position(robot,map)	2
	heading(robot,map)	2
Object image	mean(index(object image)))	1
	min(distance(object image))	1
Object pose	position(object shape,map)	2
	heading(object shape,map)	2

Figure 2: Perceptual properties learned previously by the agent. In learning its *self* model, the agent has already learned to control the robot pose properties. For the example in this paper, the agent learns to control the properties of the object image on its sensor array (the mean index of the object image measures the egocentric heading to the object). The robot also learns to control the object position in the environment (Figure 4).

such a property to be *controllable*. Finding controllable properties increases the portion of the perceptual space over which the robot can form plans. A prime example of a controllable property is the position of a free standing object which the robot can move. The perceptual properties that the robot attempts learn to control are listed in Figure 2.

Representing Features and Controls

At the low level, a robot and its environment can be modeled as a dynamical system:

$$\begin{array}{rcl} x_{t+1} & = & F(x_t, u_t) \\ z_t & = & G(x_t) \\ u_t & = & H_i(z_t) \end{array} \quad (1)$$

where x_t represents the robot's state vector at time t, z_t is the raw sense vector, and u_t is the motor vector. The functions F and G represent relationships among the environment, the robot's physical state, and the information returned by its sensors, but they are not known to the robot itself (Kuipers 2000).

The robot acts by selecting a control law H_i such that the dynamical system (1) moves the robot's state x closer to its goal, in the context of the current local environment. When this control law terminates, the robot selects a new control law H_j and continues onward. An *action* is a symbolic description of the preconditions, the transfer function H_i, and the effects of a reliable control law, suitable for planning.

The raw sensorimotor trace is a sequence of raw sense vectors and motor controls.

$$\langle z_0, u_0 \rangle, \langle z_1, u_1 \rangle, \cdots \langle z_t, u_t \rangle, \cdots \quad (2)$$

Perceptual Features

In a more detailed model of the robot dynamical system, the control laws H_i depend on perceptual features p_j, rather than the raw sense vector z. Let $P = \{p_j | 1 \leq j \leq n\}$ be a set of perceptual features,

$$p_j(z_t) = y_t^j \in \Re^{n_j} \cup \{\perp\} \quad (3)$$

defined over the sense-vector z and returning either a real-valued vector of dimension n_j or failure ($\perp$). These perceptual features are created by the learning process that individuates objects from their background and characterizes their shapes (Modayil & Kuipers 2004; 2006). The features used in our examples are listed in Figure 2.

The robot model (1) is updated:

$$\begin{array}{rcl} x_{t+1} & = & F(x_t, u_t) \\ z_t & = & G(x_t) \\ y_t^j & = & p_j(z_t) \text{ for } 1 \leq j \leq n \\ u_t & = & H_i(y_t^1, \cdots y_t^n) \end{array} \quad (4)$$

(A control law H_i will typically depend on only a few of the available perceptual features $\{y_t^1, \cdots y_t^n\}$.)

The learning algorithm describes changes in these perceptual properties and learns how to control them. We define the term $\Delta p_j(t)$ to refer to the change in p_j.

$$\Delta p_j(t) \equiv \begin{cases} p_j(z_t) - p_j(z_{t-1}) & \text{when defined,} \\ \perp & \text{otherwise.} \end{cases} \quad (5)$$

For each perceptual feature p_j, we can use the sequence of values of that feature,

$$p_j(z_0), p_j(z_1), \cdots p_j(z_t), \cdots \quad (6)$$

and the sequence of changes between adjacent values

$$\Delta p_j(1), \Delta p_j(2), \cdots \Delta p_j(t), \cdots \quad (7)$$

We also define *constraints* on perceptual features. For a scalar perceptual feature $y_t^j \in \Re^1$, we define an *atom* to be an atomic proposition of the form $y_t^j \geq \theta_j$ or $y_t^j \leq \theta_j$, where θ_j is some appropriate threshold value.

Control Laws

Useful control laws are learned by collecting descriptions of the effects of motor commands on perceptual features, in the form of tuples,

$$\langle p_j, Q, C, R, H_i \rangle \quad (8)$$

- p_j is the specific perceptual feature whose controlled behavior is the focus of this control law.
- The *qualitative description* Q is a high-level description of how the control law affects a particular perceptual feature p_j. For a scalar feature, $Q \in \{up, down\}$. For a vector feature, $Q \in \{direction[p_k] \mid p_k \in P\}$, which is used to signify that the feature p_j changes in the direction of feature p_k.
- The *context* C describes the region in which the control law is applicable, expressed as a conjunction of atoms, $C = \wedge_j (y_t^j \rho_j \theta_j)$, where the relation symbol $\rho_j \in \{\geq, \leq\}$.
- The *result* R describes the region of motor space from which the control law draws its motor signals, also expressed as a conjunction of atoms, $R = \wedge_i (u_t^i \rho_i \theta_i)$, where the relation symbol $\rho_i \in \{\geq, \leq\}$.
- The *transfer function* H_i is a process that takes the values of one or more perceptual features y_t^j defining C and generates a motor signal u_t in R.

Learning Actions

The learning algorithm has the following steps:

1. Collect a trace of sensory and motor data (eqns 2, 6, 7) from random or exploratory actions in the environment
2. Identify transitions in the trace where particular perceptual features p_j exhibit a particular qualitative change Q.
3. Using these as positive training examples (and the rest as negative training examples), apply a classification learner to the space of sensory and motor signals to learn the region $C \times R$ in which the qualitative direction of change Q is reliable.
4. Define a transfer function $H : C \rightarrow R$, to specify the motor output for a given perceptual input.
5. Evaluate each of the learned control laws by collecting new observations while running that law.

Identify Qualitative Changes

For each scalar feature p_j, select a threshold $\epsilon_j > 0$. Label some of the $\Delta p_j(t)$ in (7) with $Q \in \{up, down\}$.

$$\begin{array}{rcl} \Delta p_j > \epsilon_j & \rightarrow & up \\ -\Delta p_j > \epsilon_j & \rightarrow & down \end{array} \tag{9}$$

For a vector feature p_j, $Q \in \{direction[p_k]\}$. For some thresholds $\epsilon_j, \epsilon'_j > 0$,

$$||\Delta p_j|| > \epsilon_j \;\wedge\; \frac{\langle \Delta p_j, p_k \rangle}{||\Delta p_j|| \cdot ||p_k||} > 1 - \epsilon'_j \;\rightarrow\; direction[p_k] \tag{10}$$

In order to provide qualitative labels for a sequence of changes (7), we must determine the relevant values for ϵ_j. For a given feature p_j, a histogram of values $\{\Delta p_j(t)\}$ is collected and smoothed with a Gaussian kernel. If possible, ϵ_j is chosen corresponding to a significant local minimum in this distribution, defining a natural division between values near zero and those above. If this is not possible, then ϵ_j is set to the value about two standard deviations above the mean of $\{||\Delta p_j(t)||\}$. The value of ϵ'_j is set similarly, but a search for a control only proceeds if a natural minimum exists for some property p_k.

Classification Learning

We train a classifier to predict the occurrence of qualitatively significant change in $\Delta p_j(t)$. Using the sensor trace, significant changes are labelled as positive examples, and all other examples are labelled as negative. Then, standard supervised classifier learning methods (Mitchell 1997) can learn to distinguish between portions of the sensorimotor space where the desired qualitative change is likely, and those portions where it is not.

Because of our focus on foundational learning, the representations for C and R are both simple conjunctions of constraints, and the constraints are found by a simple greedy algorithm. The process is similar to splitting a node in decision-tree learning. For each perceptual feature p_k (including the feature p_j that we are trying to learn to control), we consider all possible thresholds θ and the two constraints $p_k \geq \theta$ and $p_k \leq \theta$. In the same way, we consider each component u_t^i of the motor vector, and compare it with possible thresholds θ, considering the constraints $u_t^i \geq \theta$ and $u_t^i \leq \theta$. When a constraint significantly improves the separation of positive from negative examples, we add it to the definition of C or R. The greedy algorithm terminates when improvement stops.

More sophisticated classifier learning algorithms, such as SVM or boosting, could be used to define C and R, but their value in a foundational learning setting has yet to be evaluated.

CL1:	Property:	object-position
	Description :	direction [robot-heading]
	Context:	min-distance $\leq$ 0.25
		min-distance $\geq$ 0.21
	Result:	motor-drive $\geq$ 0.15
CL2:	Property:	min-distance
	Description :	down
	Context:	min-distance $\geq$ 0.46
		mean-index $\leq$ 0.93
		mean-index $\geq$ -0.93
	Result:	motor-drive $\geq$ 0.15
CL3:	Property:	mean-index
	Description :	up
	Context:	mean-index $\leq$ 0.73
	Result:	motor-turn $\geq$ -0.30
CL4:	Property:	mean-index
	Description :	down
	Context:	mean-index $\geq$ -0.76
		min-distance $\geq$ 0.27
	Result:	motor-turn $\geq$ 0.30

Figure 3: Control laws (CL1-CL4) that are learned from the initial exploration. Each control law describes the expected change to a property when a motor command from the result is executed in a given perceptual context. Spurious constraints from over-fitting (min-distance in the min-index down control) can be removed with additional experience.

Defining the Transfer Function

Once C and R have been specified, a procedural version of the transfer function $u_t = H(z_t)$ can be defined trivially:

$$\text{if } z_t \in C \text{ then return some } u_t \in R \tag{11}$$

However, given that $H : C \rightarrow R$, it is possible to use standard regression methods to find a specific function H that optimizes some selected utility measure.

Evaluation

The robot evaluates each learned control law using the F-measure, which is is the geometric mean of precision and recall.

$$\begin{array}{ll} \textit{precision} & Pr(Q(\Delta p_j(t)) \mid C(z_t), u_t = H_i(z_t)) \\ \textit{recall} & Pr(C(z_t), u_t = H_i(z_t) \mid Q(\Delta p_j(t))) \end{array}$$

The *applicability* of the control law measures how commonly the constraints in its context are satisfied. The applicability $Pr(C(z_t))$ is determined not only by the context,

Property	dim	D	ϵ (ϵ')	applicability	F-measure
mean-index	1	down	0.17	0.36	0.90
mean-index	1	up	0.17	0.44	0.93
min-distance	1	down	0.08	0.29	0.85
min-distance	1	up	0.08	0.00	-
object-position	2	dir[robot-heading]	0.06 (0.14)	0.24	0.85
object-heading	2	-	-	-	-

Figure 4: Perceptual features and their control laws as learned by the robot. The learned control laws were executed, and their performance was evaluated using the F-measure function. The robot learns to control the location of the object on its sensor array (mean-index up/down), and to approach the object (min-distance down). The robot does not learn to back away from the object since the robot was prevented from driving backwards. The robot learns that the object moves in the same direction as the robot (object-position dir[robot-heading]), but does not find a control law that can reliably change the object's orientation.

but also by the robot's ability to fulfill the conditions in the context. When an overly specific control law is learned and the conditions in its context can not be readily met, it can be discarded due to its low observed applicability.

Evaluation: Autonomous Learning

The previous section described how control laws can be mined from data traces of robot experience. This section describes an experiment with a physical robot, demonstrating how these data traces are generated, subsequently mined, and how the control laws can be individually tested and applied.

The robot used is a Magellan Pro with a non-holonomic base. Using the techniques described earlier, the robot perceives and recognizes nearby objects. The laser rangefinder returns z_t, an array of distances to obstacles measured at one degree intervals. The sensory image in z_t of the object of interest is used to compute perceptual features (Figure 2), including the position and orientation of the object, the minimum distance to the object and the mean angle to the object.

The training environment consists of a virtual playpen that is a circle with a 60cm radius. The small size of the playpen permits the robot to sense and contact the object frequently even while engaged in random motor babbling. During the initial exploration phase, the robot uses its location to select from two control laws. When the robot is outside of the playpen, the robot executes a control law that turns the robot towards the playpen and drives forward. When the robot is in the playpen, it executes a random motor command, u_t, that consists of a drive and turn velocity selected from:

$$M = \{(0.15, 0.0), (0.0, -0.3), (0.0, 0.3)\}(m/s, rad/s)$$

A new motor command is selected every two seconds. A human experimenter places one of two objects into the playpen (a recycling bin and a trash can), and returns them to the playpen when the robot pushes them outside. During the initial exploration phase, the robot gathers four episodes of experience, each of which is five minutes long.

The evaluation process uses the same experimental setup as is used in training. The robot executes each control law repeatedly during one five minute episode, in the condition where the robot would normal execute a random motor command. If all constraints in the context of the control law are satisfied by the current sensory experience, then the robot sends the result of the transfer function to the motors. If a constraint is not satisfied, then the robot attempts to select a control law that will reduce the difference between the feature-value and the threshold. The robot executes the selected control if it exists; otherwise the robot sends a random motor command to the robot.

The data gathered from the initial exploration is used to learn control laws. Control laws are generated by applying the learning algorithm from the previous section to the exploration data. Both up and down controls are generated for each scalar property. The performance of each control law is evaluated by executing it. A generated control law is discarded if its F-measure performance is less than 0.8, or the applicability is less than 0.05. A summary of the learned control laws is listed in Figure 4.

The result of the learning process is a set of control laws (Figure 3). Each control law can be easily interpreted. For example CL3 is a control law that causes the mean-index to increase, meaning that the image of the object on the sensor array moves to the left. The control law achieves this effect by turning to the right (motor-turn $\geq$ -0.30). Since the robot has a limited field of view, this control law is only reliable when the object is not already on the far left on the sensor array. Hence, there is an additional constraint in the context (mean-index $\leq$ 0.73).

Note that the control laws support back-chaining: if a constraint of one control law of the form $p_j \leq \theta$ is not satisfied, then the control law with property p_j and description down may be used to reduce the difference. These chaining conditions can be expressed in terms of a dependency graph (Figure 5).

Here is an example of how the robot performs the chaining. Suppose the robot has the goal of moving an object. When it tries to execute the object-position direction[robot-heading] control law, the object violates the constraint min-distance $\leq$ 0.25. Hence, the object probably will not move at this time step, and the robot applies the min-distance down control law. However, the object lies to the side of the robot, and violates the mean-index constraint. Hence, the robot will probably not get closer to the object at this time step, but can execute the mean-index up control law to bring the object image closer to the center of the field of view. On subsequent time steps, the robot goes

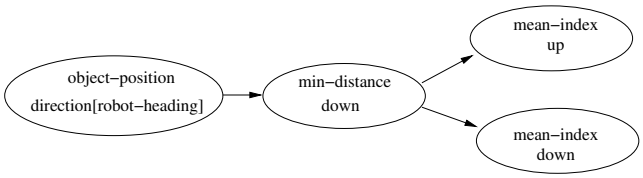


Figure 5: Dependencies between learned control laws arise as each inequality constraint in a control law's context may be fulfilled by an up or down control law.

through the same series of calls until it succeeds in pushing the object.

Related Work

The above results show that a robot can learn to control object percepts. Several other researchers have explored similar ideas. Work in locally weighted learning has explored learning continuous control laws, but with supervised learning and careful selection of relevant state variables (Atkeson, Moore, & Schaal 1997). Work in developmental learning through reinforcement learning techniques have also explored defining control laws for novel state. (Barto, Singh, & Chentanez 2004) shows how reusable policies can be learned, though with significantly more data and in simulation. In (Hart, Grupen, & Jensen 2005), symbolic actions are provided to the robot and the probability that the actions are executed successfully is learned. Work by (Oudeyer *et al.* 2005) shows how curiosity can drive autonomous learning, again with thousands of training examples but on physical robots. Previous work also shows how a robot can learn to push objects around (Stoytchev 2005), but from only a fixed viewpoint. Work in (Langley & Choi 2006) shows how hierarchical task networks can be autonomously learned in a manner that is similar in approach to the learning we perform here. That work differs in that the task is driven by an externally specified goal.

Conclusions

We have presented the requirements for an object ontology, described an algorithm for learning object control laws, and demonstrated how a robot can autonomously learn these control laws. The results show how control laws can be learned autonomously.

This paper makes several contributions towards learning about objects and actions. One is that a perceptual feature is more useful if control laws can be found that cause reliable changes in the feature. A second contribution lies in demonstrating how backchaining of learned control laws can occur when each precondition in the context of one control law can be satisfied by another control law. A third contribution lies in the algorithm for learning the control law which demonstrates how an unsupervised learning stage can be used to generate a training signal for supervised learning.

An important extension on the current work will be to test this process on robot platforms with a wider range of perceptual properties and actuation.

Acknowledgements

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 National Science Foundation (IIS-0413257 and IIS-0538927), from the National Institutes of Health (EY016089), and by an IBM Faculty Research Award.

References

Atkeson, C. G.; Moore, A. W.; and Schaal, S. 1997. Locally weighted learning for control. *Artificial Intelligence Review* 11(1/5):75–113.

Barto, A.; Singh, S.; and Chentanez, N. 2004. Intrinsically motivated learning of hierarchical collections of skills. In *International Conference on Developmental Learning*.

Biswas, R.; Limketkai, B.; Sanner, S.; and Thrun, S. 2002. Towards object mapping in non-stationary environments with mobile robots. In *IEEE/RSJ Int. Conf on Intelligent Robots and Systems*, volume 1, 1014–1019.

Gibson, J. J. 1979. *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin.

Hart, S.; Grupen, R.; and Jensen, D. 2005. A relational representation for procedural task knowledge. In *Proc. 20th National Conf. on Artificial Intelligence (AAAI-2005)*.

Kuipers, B. J. 2000. The Spatial Semantic Hierarchy. *Artificial Intelligence* 119:191–233.

Langley, P., and Choi, D. 2006. Learning recursive control programs from problem solving. *Journal of Machine Learning Research*.

Mandler, J. 2004. *The Foundations of Mind: Origins of Conceptual Thought*. Oxford University Press.

Mitchell, T. M. 1997. *Machine Learning*. Boston: McGraw-Hill.

Modayil, J., and Kuipers, B. 2004. Bootstrap learning for object discovery. In *IEEE/RSJ Int. Conf on Intelligent Robots and Systems*, 742–747.

Modayil, J., and Kuipers, B. 2006. Autonomous shape model learning for object localization and recognition. In *IEEE International Conference on Robotics and Automation*, 2991–2996.

Oudeyer, P.-Y.; Kaplan, F.; Hafner, V.; and Whyte, A. 2005. The playground experiment: Task-independent development of a curious robot. In *AAAI Spring Symposium Workshop on Developmental Robotics*.

Philipona, D.; O'Regan, J. K.; and Nadal, J.-P. 2003. Is there something out there? Inferring space from sensorimotor dependencies. *Neural Computation* 15:2029–2049.

Pierce, D. M., and Kuipers, B. J. 1997. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence* 92:169–227.

Russell, S., and Norvig, P. 2002. *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Spelke, E. S. 1990. Principles of object perception. *Cognitive Science* 14:29–56.

Stoytchev, A. 2005. Behavior-grounded representation of tool affordances. In *IEEE International Conference on Robotics and Automation (ICRA)*.