Scaling-up Knowledge for a Cognizant Robot

Thomas Degris[*], **Joseph Modayil**[†]
[*]Flowers, INRIA, 351 cours de la libration, 33405 Talence Cedex, France
[†]RLAI, University of Alberta, Edmonton, AB, Canada, T6G 2E8

Abstract

This paper takes a new approach to the old adage that knowledge is the key for artificial intelligence. A cognizant robot is a robot with a deep and immediately accessible understanding of its interaction with the environment—an understanding the robot can use to flexibly adapt to novel situations. Such a robot will need a vast amount of situated, revisable, and expressive knowledge to display flexible intelligent behaviors. Instead of relying on human-provided knowledge, we propose that an arbitrary robot can autonomously acquire pertinent knowledge directly from everyday interaction with the environment. We show how existing ideas in reinforcement learning can enable a robot to maintain and improve its knowledge. The robot performs a continual learning process that scales-up knowledge acquisition to cover a large number of facts, skills and predictions. This knowledge has semantics that are grounded in sensorimotor experience. We see the approach of developing more cognizant robots as a necessary key step towards broadly competent robots.

Knowledge: a Bottleneck for Cognizant Robots

Any robot with a closed-loop control has some awareness of its environment. In 1948, Walter's robot tortoises *Elmer* and *Elsie* could respond to their environment by avoiding obstacles or moving towards a light (Walter, 1950). More recently, robot vacuum cleaners detect dirty areas and spend more time cleaning them. Some are able to build maps of their local space. Autonomous cars are able to take appropriate action to react to changing traffic conditions (Urmson et al., 2008).

In comparison, most animals can be considered more cognizant than modern robots in many ways. First, even simple animals such as fruit flies, cockroaches and spiders are able to adapt their behavior to their environment through learning from everyday experience (Greenspan and van Swinderen, 2004). Second, animals can learn to anticipate what is going to happen next, as it has been demonstrated with dogs, pigeons, and insects (Pearce, 1997). Third, animals demonstrate a deep understanding of their sensorimotor experience by exhibiting complex behaviors. Cats can detect what is unusual in their environment and exhibit curiosity-driven behavior. Crows, bees, and rats can successfully remember locations and navigate between them. Fourth, some animals use a large number of diverse sensorimotor signals from the world. Mammals, for example, are able to take advantage of smell, sight, hearing, taste and touch, along with temperature, acceleration, hair movements and also their internal senses such as pain, hunger, fatigue, guts, and proprioception. In addition, they interact with their environment using a large number and variety of muscles and actuators.

To demonstrate comparable awareness of their environment, a cognizant robot would need to have lots of immediate knowledge. For instance, a reaction as basic as flinching in response to a moving object involves knowledge in the form of a ready to use flinching skill, and also, perhaps, a prediction of the trajectory of the object. Competent navigation involves a wide-range of knowledge about the topology and characteristics of the environment, as well as skills ranging from obstacle avoidance to planning a new route when a path is unexpectedly blocked. Detecting novelty and exhibiting curiosity-driven behaviour also involves knowledge: using existing knowledge to contextualize newly observed sensorimotor data, and a repertoire of behaviors to assess the nature of the novelty. The more knowledge to which a robot has immediate access, the more the robot can be cognizant.

The bottleneck in developing such robots is the lack of methods that are able to acquire and maintain a large quantity of immediately usable knowledge. A traditional approach is to manually provide the robot with knowledge about its environment in the form of models, skills, and features. But such methods are problematic for at least two reasons. One, scaling-up knowledge is difficult if it requires manual interventions for every modification. Two, such knowledge will be necessarily incomplete or inaccurate in an open environment. Moreover, knowledge is often encoded in a form that cannot be autonomously updated by the robot (e.g. hand-coded or learned-but-fixed features and skills).

As a solution to this bottleneck, we consider Horde: a conceptually simple architecture that was introduced recently (Sutton et al., 2011). Horde proposes a theoretically sound formalism for representing knowledge. Horde allows a robot to learn autonomously from its everyday interaction with its environment. Horde is able to maintain and acquire a large number of situated, revisable, and expressive pieces of knowledge that are exploitable by a cognizant robot. To

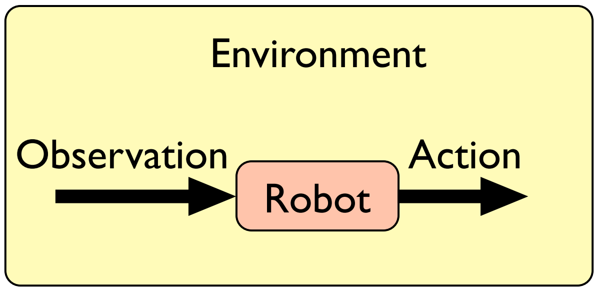


Figure 1: The robot is a situated agent that can only learn from its sensorimotor interaction stream of observations and actions.

reap these benefits, the robot should have an abundance of experience and the ability to autonomously explore its environment.

Defining Knowledge for a Cognizant Robot

To make our discussion about knowledge more concrete, we start by defining the interaction between the robot and its environment. We assume that this interaction is divided into discrete time steps. Despite the fact that this assumption does not characterize robots using analog electronics for their behavior (e.g. Walter's robot tortoises), it is a requirement for working with digital computers. Ideally, the length of these time steps should be as small as possible so that the robot has a short reaction time. Note that time steps do not need to be of constant length.

The environment for robots is the physical world, a large space with essentially unlimited forms of variation. At every time t, the robot receives a new observation $\mathbf{o}_t \in O$ from its environment (where O is a real vector space). The observation $\mathbf{o}_t$ represents the latest information available on all the sensors of the robot. Not all elements of $\mathbf{o}_t$ may have been updated because sensors can have different update times. In response, the robot sends back an action $\mathbf{a}_t \in A$ where A is also a real vector space. An action includes all the parameters for the different actuators available to the robot, including voltage on motors, parameters for pattern generators or colors on LEDs. Observations and actions are the only information exchanged between a robot and its environment, as illustrated in Figure 1; they do not constitute knowledge.

The first element of knowledge that we formalize is an *update function* $U : X \times A \times O \to X$ defined as

$$\mathbf{x}_t \leftarrow U(\mathbf{x}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t),$$

where X is the space of state of the agent. At a time step t, the function U takes the current state of the agent $\mathbf{x}_{t-1}$, the latest observation $\mathbf{o}_t$ and the last action $\mathbf{a}_{t-1}$ to produce an updated state $\mathbf{x}_t$. We do not restrict the state of the agent, which can be the current value of variables in a procedure or the weights of a feature vector for a learning algorithm.

A second element of knowledge is the *behavior* of the robot in the world, represented as a policy $b(\mathbf{a}_t|\mathbf{x}_t)$ defining the probability density of executing at time t the action $\mathbf{a}_t$ given its current state $\mathbf{x}_t$. We allow both U and b to depend on time but we do not include time indices for clarity.

Consequently, the goal of developing a cognizant robot requires the definition of only these two elements of knowledge: an internal state update function and a behavior policy. A robot will demonstrate a deep understanding of its environment by extracting information from observations to adequately update its internal state, so that the behavior policy will efficiently output adapted actions. Other elements of knowledge support the construction of these two functions.

Another important element of knowledge are *skills*. Independently of the robot's current behavior, we would like to represent the knowledge of being able to achieve some goals. We can formalize skills as policies $\pi(a|x)$. Similarly to the option framework, we can also define a termination function $C : X \to [0, 1]$ that, given the state of the agent, defines the probability of terminating a policy, maybe when a goal is achieved (Sutton et al., 1999). For a mobile robot, a useful skill such as "going back to the docking station" would be represented by a policy that the robot could follow to go back to its docking station and the skill would terminate either when the robot observes it has successfully connected or after a time out.

Given a policy and a termination function, another element of knowledge are the *predictions* about observable signals. An observable signal z is any function of the observations or the state of the agent. If the current time is t, we are interested in predicting the value z_T at the time T (where $T > t$) given the current state $\mathbf{x}_t$ of the agent. These predictions can be conditional on a skill, such as a prediction of the value z_T if a policy π is started from the current time t and terminated at time T. An example of such prediction for a mobile robot would be knowledge such as: if I go back to the docking station starting from now and until termination, will I observe being connected to it?

Another form of predictions frequently used for knowledge, most notably in reinforcement learning, is the empirical return g_t, that is the prediction of the sum of an observable signal z all along the way while executing a skill:

$$g_t = \sum_{k=t}^{T} z_k. \tag{1}$$

Such knowledge can represent predictions such as: what is the number of time steps required to go back to the docking station? What is the total amount of energy required to reach the docking station?

A last element of knowledge commonly used is *features*. Part of the state of the agent can be a feature vector ϕ_t that is updated every time step. ϕ_t can include elements from the past experience of the robot, data extracted from sensors (e.g. edge detection from image, word recognized from audio) or other elements computed from observations, actions or more generally from the internal representation of the robot.

In summary, these elements comprise the robot's immediate knowledge: the robot's update function, behavior, skills, predictions, and features. These are not the only computational elements within the robot: slower timescale processes that may be present include planning and long-term memories. To be used by a cognizant robot, the results of these slower processes should eventually be made available as a form of immediate knowledge. Aspects of immediate knowledge can be identified in all mechanical robots, varying from Walter's tortoises, to map-building vacuums, to Shakey the robot. Analogues can also be identified in animals as described in the previous section.

The immediate knowledge forms the basis by which a cognizant robot can demonstrate the depth of its understanding of the environment. By requiring that the knowledge be demonstrable, we ensure that different approaches can be compared using clearly specified performance measures.

Cognizance: Situated, Revisable, Expressive Knowledge

Ideally, a cognizant robot's immediate knowledge would have several desirable properties, namely being situated, revisable, and expressive. These properties have largely been studied by different communities. To be situated is of primary importance for the robotics community who desire knowledge that pertains directly to the robot's interaction stream. To be revisable is of primary importance for the online machine learning community, who desire knowledge to have clear performance measures and mechanisms for incremental adaptation. To be expressive is of primary importance for the knowledge representation community, who desire knowledge that can be used flexibly to accomplish a wide range of tasks. As we describe below, requiring a system to possess all three properties suggests a substantially different approach from when these properties of knowledge are pursued in isolation.

Situated

Knowledge is situated if it pertains to the robot's stream of interaction with the environment. Many approaches to knowledge are not situated, including scraping information from the web, manually constructing knowledge bases, and mathematical theories. However, the need to be situated has been long understood in robotics (Brooks, 1986).

Imagine that a program accesses the Internet, scans a large (unstructured) knowledge database, and gets the factoid "A robot can walk on a sandy beach". Such knowledge cannot be used to infer what sensor values would arise from walking on a beach, nor to construct a gait for walking on the beach. There also remains a fundamental and difficult problem of associating the words "beach" and "sand" to the robot's immediate knowledge. Before making use of these non-immediate forms of knowledge, a robot needs a foundation of immediate knowledge that provides situated semantics for these terms.

Unlike forms of knowledge acquired from third parties, knowledge about sensorimotor interaction is tightly coupled to the current environment and physical embodiment of the robot. For instance, the response of a wheel motor to a voltage not only depends on the specification of the motor but potentially also on its age, its temperature, the shape and materials of the wheel, also most certainly on the kind of floor the wheel is on, the shape and weight of the robot. The amount of this information that is observable by the robot and taken into account will comprise how aware the robot is of its wheel response.

Revisable

Knowledge is revisable if it has an observable performance measure and the robot can modify the knowledge to improve its performance. Situated knowledge is ideally suited for revision, but the immediate knowledge in many robots relies on constructs whose semantics are not defined in the robot's interaction stream. Often, the robot is externally evaluated by the robot developer, but the robot cannot evaluate its own performance. In principle, even a robot vacuum cleaner could measure its success in cleaning the floor. Did the floor get brighter after cleaning? Did the dust-bin get heavier? Perhaps the robot vacuum cleaner could modify its behavior to spend more time in places that are more rewarding (have more dirt).

Knowledge is often considered to be static, as opposed to revisable. The static approach is appropriate for knowledge that has a clear logical notion of falsehood as is the case for knowledge about mathematics, events in the past, and the common knowledge within communication protocols. The fact that 2+2 is equal to 4 is always true, independent of the environment and the local condition of the robot. This is true also for historical events: "It rained in Edmonton on June 2, 2008". However, it is not appropriate for knowledge to be incapable of change when it pertains to the future.

A direct consequence of having an observable performance metric is that a robot should use its experience to revise its immediate knowledge. One might argue against this, by arguing that specifications for particular sensors are accessible to the robot designer. The designer can know a laboratory-calibrated function for converting from infrared reflection sensor readings to a distance between the sensor and an obstacle. But the specifications will never consider all the possible cases that the robot will encounter: the response of individual sensors might vary within a manufacturing batch, the response will also depend on the color and material on the wall, and the response might vary with more or less opaque walls. Such uncertainty is true for many sensors: how does an accelerometer respond to uneven ground? How does a laser range-finder respond to rain, fog, snow, mud and reflected mirages? A robot interacting in an open environment will eventually run into conditions that were not anticipated by the robot's designer. A robot needs the ability to incorporate new observed data with its existing knowledge, and this requires knowledge to be in a form that permits revision.

Expressive

The expressive power of a form of knowledge representation limits how flexibly the robot can achieve its objectives. Having expressive power is crucial for demonstrating competent behavior with limited resources. Expressivity constrains both how well a robot can generalize to novel situations, and the robot's ability to predict selected consequences of performing a temporally extended policy.

Expressive knowledge should enable a robot to generalize to novel situations. Since a robot never has access to the state of the physical world, the robot always has to generalize from past experience, but can never be certain in its generalizations. For any given task, some features will be relevant while many others will not, and the features that are relevant might vary over time. For example, navigating empty hallways at night will present a substantially different set of challenges from performing the same navigation task in the day when the hallways are filled with people.

Another aspect of expressive power is the ability to make predictions of temporally extended behavior. Many learning approaches focus on building a one-timestep dynamical system model of the robot's interaction. Using these models for temporally-extended planning requires substantial computation. In addition to their computational cost, they still may not make available the information the robot actually needs at the moment - for example, whether enough battery power remains to go back to the recharging station. Temporally extended predictions can also form a powerful method of abstraction. For example, if a robot can both predict the onset of collisions under various policies and how much time is required to come to a complete stop, then the robot can use this information to develop a skill for moving while avoiding collisions.

The Horde for Scaling-up Knowledge

Horde, proposed by Sutton et al. (2011), is an architecture for maintaining knowledge using reinforcement learning methods (Sutton and Barto, 1998). To learn policies and predictions, we need an idealized representation of the world of the robot to define objective functions. Horde assumes an underlying Markov Decision Process (MDP) which provides the dynamics of the world. The state of the world at time t is denoted $s_t \in S$.

By definition, the MDP representing the world satisfies the Markov assumption, namely that the next state does not depend on previous states, it depends only on the current state and the current action. This assumption may seem unrealistic for robots, as observations from the environment are an incomplete description of the world. Actually, this assumption is weak because it is only about the dynamics of the environment, not the robot. All the methods used in the Horde use function approximation. With this approach, Horde does not assume having access to the state of the environment. In practice, the main advantage of this formalism is the absence of strong assumptions about the data available to the robot, while still enabling mathematical proofs of convergence (with additional assumptions).

Horde introduces the concept of *general value functions* to formally define a prediction. A general value function is an expectation defined as:

$$V(s_t) = \mathrm{E}_{\pi,C}\left[R_t + Z_T | s_t\right].$$

Here R_t is a random variable that corresponds to the empirical return g_t as defined in Eq. 1 and Z_T is a random variable representing the value of the signal z_T at time T, where T is termination with respect to C.

Whereas a general value function V is defined given a MDP, Horde learns an *approximate* value function $\tilde{V}$ defined as the linear combination:

$$\tilde{V} = \mathbf{v}^\mathsf{T}\phi,$$

where $\mathbf{v}$ is a weight vector and ϕ is a feature vector. Horde uses gradient TD methods (Sutton et al., 2009) to learn the weights $\mathbf{v}$ given a general value function because these methods have a per-time-step complexity that is linear in the size of the feature vector ϕ.

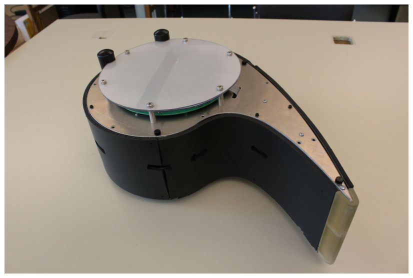
In addition to learning predictions, Horde can also learn policies. The task for the policy is defined by a reward function. A policy is learned using the Greedy-GQ algorithm (Maei and Sutton, 2010), which approximates a *general action-value function* with a linear function of the features. The complexity per-time-step of Greedy-GQ is also linear with the size of the feature vector.

The key idea of the Horde architecture to scale-up knowledge acquisition by learning in parallel many predictions and policies online, all from a single stream of experience. Indeed, Horde consists of independent *demons*, each learning a different prediction or policy using an independent weight vector. Demons are able to learn about policies not currently followed by the robot using off-policy learning.

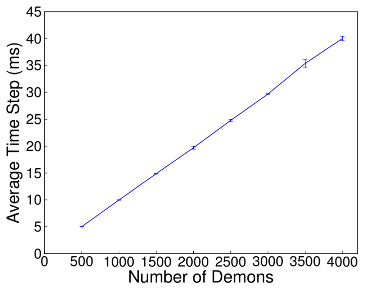
We implemented Horde on the Critterbot, a comma-shaped robot with a wide-range of low-level sensors (see Figure 2.a). These sensors include an accelerometer, infrared distance sensors, infra-red photodiodes, thermal sensors, a magnetometer, a gyroscope and ambient light sensors. Additional sensors provide information on the system bus voltage and motor velocity, current and temperature. The robot can move using three motors controlling omni-directional wheels. Sensor readings are generated every 10ms.

Figure 2.b shows the time (in ms) required for an update cycle with respect to the number of demons. The update cycle includes updating the current prediction given the latest feature vector and updating the weight vector. As expected, the update cycle grows linearly with the number of demons.

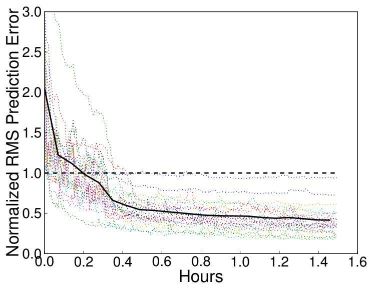
Figure 2.c, an early result, shows the normalized prediction error of 20 demons, that is the difference between the prediction and observed empirical return on a separate evaluation set, normalized for each demon by the standard deviation of the returns in the evaluation set. The behavior repeatedly selects a random action from a set of 7 predefined actions for a randomized duration while running in a wooden pen. Sensor readings were sampled at 10ms. Each demon is learning to predict the value of some sensor (one of the IR sensors, a motor speed, or a motor current), after following a constant action policy (for one of the 7 actions) with a spontaneous termination (with probabilities of 1, 0.5, 0.1, 0.05, or 0.01 spanning timescales from 10ms to 1s). These questions provide a sample of the sorts of knowledge that a robot might want to have about its experience. From the graph, we can observe that the prediction error decreases substantially in the first thirty minutes, with all the learned predictions eventually performing better than the best constant prediction. Moreover, the learning has no divergence problems, special feature selection, or parameter tweaking for individual predictions. With additional experience, learning continues to reduce the prediction error but slowly and with some variability.



(a)



(b)



(c)

Figure 2: A robot learning many pieces of knowledge. (a) The robot. (b) The time requirement scales linearly with the available computation, with timescales that are reasonable for robots. (c) Learning many predictions in parallel about different policies with substantial reductions in error in 30 minutes, and improving with additional experience. The prediction error is the measured difference between predictions and empirical returns on a separate evaluation run. For each question, the root-mean-square error is normalized by the sample standard deviation over the evaluation set. Thus, normalized errors below 1 indicate the portion of the variance in the empirical returns that the robot is able to predict.

Horde learns knowledge that is situated: predictions are a function of the feature vector of the robot and the predictions are about the sensors of the robot (or its internal state). Policies are defined as functions of the feature vector of the robot as well. Horde learns knowledge that is revisable: predictions and policies are defined using a weight vector updated in real-time from the sensorimotor stream. Finally, Horde learns knowledge that is expressive: predictions are conditional on policies that can be arbitrarily complex and temporally extended.

Future Directions: What are Demons For?

Though the idea of using TD methods to acquire knowledge might seem unusual at first glance, we have demonstrated that this approach yields empirical knowledge that can be learned accurately and flexibly with practical amounts of training experience. Although we demonstrated this approach on a research robot, the same method can be applied directly to robots with more complex sensory and motor systems. Moreover, as we discuss below, the demons have several possible uses in modifying the behaviour and state-update function: for novelty-detection, as partial models for planning, for feature selection, or to develop new features.

First, demons can be used for detecting novelty. Each piece of knowledge maintained by a demon represents an expectation of the robot of its environment. When the sensorimotor stream does not match the predictions, the discrepancy can be detected and the robot can modify its behavior to express surprise, to assess the newly discovered novelty—an idea with connections to encouraging a curiosity-drive, or even to change its update function to imprint on the novelty. A robot that is able to respond to novelty on any on its sensors can be considered more cognizant than a robot that would not.

Second, demons can be used for planning. Each prediction demon is a partial model of the consequences of following an option, and multiple demons could be considered jointly to enable temporally extended planning. As a slow process, planning could use multiple partial models to update the behavior policy of the robot. The ability to flexibly reason with partial models would provide a compelling discriminative alternative for tasks where generative Bayesian models are commonly used.

Third, demons can be used for feature selection. In Horde, all demons compute a general value function with a shared feature vector. As the demons have different predictive tasks, this provides a selection pressure on the features to be relevant for many tasks. Examining the reliance across all the demons on each feature can help to determine what features to change or remove as part of a representation discovery process. In short, a feature used by many demons predicting different signals may be more informational and valuable compared to a feature only used by only a few demons.

Fourth, demons can be used as features themselves. Predictive state representations and TD networks have been shown to be more general for knowledge representation than other concurrent approach such as POMDP or n^{th}-order Markov models (Singh et al., 2004). In such setting, the prediction of a demon can be used directly as a feature for other demons (including itself). Demons connected together in this manner could define a TD network that is able to represent abstract concepts built from the sensorimotor stream.

The argument we propose in this paper, that knowledge is important and that it should be closely related to sensorimotor experience, has been acknowledged for a long time in the robotics community. The novel contribution of the Horde is to provide a combination of features that enable a robot to efficiently learn a broad set of competencies. 1. Horde is the first architecture able to represent abstract temporally-extended knowledge within a well-defined mathematical framework that supports learning under function approximation (an incomplete representation of the environment). 2. Horde is scalable: the computational and memory requirements grow linearly with the number of demons, all demons always learn from one single common sensorimotor stream but are able to efficiently learn about different ways of behaving. 3. Demons are easy to create and delete to build more abstract knowledge. Finally, this paper is about developing low-level knowledge and abstractions from low-level knowledge on a robot, and it is not obvious precisely how this supports high-level deliberative tasks like reading and playing chess.

Conclusion

This paper describes a roadmap for building a cognizant robot. We have defined a cognizant robot as one that possesses vast amounts of situated, revisable and expressive knowledge. We have argued that such knowledge ought to be learned for scalability and for maintaining accuracy in an open environment. We have described a framework for acquiring knowledge that is conceptually simple and scalable in practice. Our results so far show that this approach scales well, and updates in real-time. Thus our approach to build knowledge for a cognizant robot is concrete, implemented, and works. In some ways, it is surprising that we can use reinforcement learning methods for acquiring a broad array of knowledge. Although there are still many open questions for future research, we believe working towards more cognizant robots will provide a firm foundation for more abstract cognition.

References

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Trans. on Robotics and Automation*, RA-2(1):14–23.

Greenspan, R. J. and van Swinderen, B. (2004). Cognitive consonance: complex brain functions in the fruit fly and its relatives. *TRENDS in Neurosciences*, 27(12):707–711.

Maei, H. R. and Sutton, R. S. (2010). GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proceedings of the Third Conference on Artificial General Intelligence*.

Pearce, J. (1997). *Animal Learning and Cognition: an Introduction*. Psychology Press.

Singh, S., James, M., and Rudary, M. (2004). Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 512–519. AUAI Press.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th International Conference on Machine Learning*.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Urmson, C., Anhalt, J., Bagnell, D., et al. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466.

Walter, W. G. (1950). An electro-mechanical animal. *Dialectica*, 4(3):206–213.