Kalman Based Temporal Difference Neural Network for Policy Generation under Uncertainty (KBTDNN)

Alp Sardag and H.Levent Akin
Bogazici University
Department of Computer Engineering
34342 Bebek, Istanbul, TURKEY

Abstract

A real world environment is often partially observable by the agents either because of noisy sensors or incomplete perception. Moreover, it has continuous state space in nature, and agents must decide on an action for each point in internal continuous belief space. Consequently, it is convenient to model this type of decision-making problems as Partially Observable Markov Decision Processes (POMDPs) with continuous observation and state space. Most of the POMDP methods whether approximate or exact assume that the underlying world dynamics or POMDP parameters such as transition and observation probabilities are known. However, for many real world environments it is very difficult if not impossible to obtain such information. We assume that only the internal dynamics of the agent, such as the actuator noise, interpretation of the sensor suite, are known. Using these internal dynamics, our algorithm, namely Kalman Based Temporal Difference Neural Network (KBTDNN), generates an approximate optimal policy in a continuous belief state space. The policy over continuous belief state space is represented by a temporal difference neural network. KBTDNN deals with continuous Gaussian-based POMDPs. It makes use of Kalman Filter for belief state estimation. Given only the MDP reward and the internal dynamics of the agent, KBTDNN can automatically construct the approximate optimal policy without the need for discretization of the state and observation space.

Introduction

Traditional learning approaches deal with models that define an agent and its interactions with the environment via its perceptions, actions, and associated rewards. The agent tries to maximize its long term reward when performing an action. This would be easier if the world model was fully observable, namely the underlying process was a Markov Decision Process (MDP). However, in many real world environments, it will not be possible for the agent to have complete perception. When designing agents that can act under uncertainty, it is convenient to model the environment as a Partially Observable Markov Decision Process (POMDP). This model incorporates uncertainty in the agent's perceptions that have continuous distributions, actions and feedback which is an immediate or delayed reinforcement. POMDP models contain two sources of uncertainty; stochasticity of the controlled process, and imperfect and noisy observations of the state (Kaelbling, Littman, & Moore 1996).

In POMDP's, a learner interacts with a stochastic environment whose state is only partially observable. Actions change the state of the environment and lead to numerical penalties/rewards, which may be observed with an unknown temporal delay. The learner's goal is to devise a policy for action selection under uncertainty that maximizes the reward. Although, the POMDP framework is useful in practically modeling a large range of problems, it has the disadvantage of being hard to solve, where previous studies have shown that computing the exact optimal policy is intractable for problems with more than a few tens of states, observations and actions. The complexity of POMDP algorithms grows exponentially with the number of state variables, making it infeasible for large problems (Boyen & Koller 1998; Sallans 2000).

Most of the methods dealing with POMDPs assume that the underlying POMDP parameters are known. However, collecting such information is impossible for most of the real world cases. Additionally, past work has predominately studied POMDPs in discrete worlds. Discrete worlds have the advantage that distributions over states (so called "belief states") can be represented exactly, using one parameter per state. The optimal value function (for finite planning horizons) has been shown to be convex and piecewise linear, which makes it possible to derive exact solutions for discrete POMDPs (Kaelbling, Littman, & Moore 1996).

In this study we aim to use POMDP as a model of the real environment. Since discrete case POMDPs are hard to solve, many studies focus on these type of problems. But the application of such algorithms to robotics which are continuous in nature is nearly impossible. There are a few studies on continuous POMDPs. Recently two studies have been presented. Vlassis, Porta and Spaan, in their work, assume discrete observation over continuous state space (Porta, Spaan, & Vlassis 2005). Hoey and Poupart assume discrete state space with continuous observation space (Hoey & Poupart 2005). In these two studies, they further assume that the world model is given. We have made two assumptions that do not exist in nearly none of the previous studies. First, we assumed that the POMDP parameters or world dynamics, such as observation and transition probabilities, are not known, and we expect our agent, simply using its internal

dynamics, to generate an optimal policy by making an approximation to exact value function over belief space. Second, since a large number of real world problems are continuous in nature, we are interested in POMDP's with continuous parameters which cannot be solved exactly.

In section 2 we formally describe conventional POMDP solution techniques. In section 3 we describe Kalman Based Temporal Difference Neural Network for Policy Generation under Uncertainty (KBTDNN) algorithm in detail. In section 4 we give the experimental results for some well-known POMDP problems. We conclude in section 5.

POMDP and Conventional Solution Techniques

POMDPs contain two sources of uncertainty: the stochasticity of the underlying controlled process, and imperfect observability of its states via a set of noisy observations (Sondik 1978). Noisy observations and the ability to model and reason with information gathering actions are the main features that distinguish the POMDPs from the widely known Markov decision process (MDP).

The formal definition of a POMDP is a 6 tuple (*S, A,* Θ*, T, O, R*) where;

- *S* is the set of states,
- *A* is the set of actions,
- Θ is the set of observations,
- *R* is the reward function $S \times A \mapsto R$,
- *O* is the set of observation probabilities that describe the relationships among observation states and actions $S \times A \times \Theta \mapsto [0,1]$,
- *T* is the state transition function $S \times A \mapsto \Pi(S)$, where a member of $\Pi(S)$ is a belief state that is a probability distribution over the set *S*.

The immediate rewards, received when the agent takes an action at any given belief state, are fed in order to give hint to the agent about what is right and what is wrong. The goal of a POMDP agent is to optimize the expected discounted reward that is the discounted infinite sum of instant rewards, given in Eq. 1.

$$E\left(\sum_{t=0}^{\infty} \gamma^t r(t)\right) \tag{1}$$

The methods for learning within the POMDP (Kaelbling, Littman, & Moore 1996) framework may be classified as exact but intractable methods and approximate methods. There are two versions of the POMDP training problem: learning when a model of the POMDP is known, and the much harder problem, learning when a model is not available. Obtaining the world dynamics for complex environments is very difficult. Since the final policy is built on world dynamics, a wrong or idealized world model will lead to poor policies which will cause serious problems in real world environments. Therefore, most of the suggested solution techniques for discrete state space POMDPs where the world model is given are not appropriate for robotics in which world dynamics can not be calculated exactly and is continuous in nature.

Recently two studies, one assuming discrete observation over continuous state space (Porta, Spaan, & Vlassis 2005) and one assuming discrete state space with continuous observation space (Hoey & Poupart 2005) have been presented. Although, they assume that the world model is given, both of them use point based value iteration technique (Pineau, Gordon, & Thrun 2003) for Gaussian-based POMDPs. Point based value iteration (PBVI) is an approximate solution technique that requires belief point backups chosen based on a heuristic. The solution is dependent on the initial belief and, need a time consuming belief point set expansion and pruning phase for α-vectors.

Kalman Filters

In the POMDP approaches, the agent's state is not fully observable; the agent tries to convey uncertain or incomplete information about the state of the world. The agent remembers past observations up to some finite number of observations, called a *history* denoted by the set, h_{t-1} , h_{t-2} , . . . , h_{t-n}

Astrom (Astrom 1965) conceived an alternative to storing histories: a *belief state*, which is the probability distribution over the state space, *S*, summarizing the agent's knowledge about the current state. The belief state is sufficient in the sense that the agent can learn the same policies as if it had access to history (Kaelbling, Littman, & Moore 1996).

$$\sum\nolimits_{s \in S} b_t(s) = 1 \tag{2}$$

Here $b_t(s)$ is the probability that the world is in state *s* at time *t*. Knowing $b_t(s)$ in Eq. 2, an action, *a* and an observation, *o*, the successor belief state can be computed by Eq. 3.

$$b_{t+1}(s) = \frac{\sum_s b_t(s) P(s' \,|s, a) P(o \,|s, a \,, s')}{\sum_{s,s'} b_t(s) P(s' \,|s, a) P(o \,|s, a \,, s')} \tag{3}$$

Eq. 3 is equivalent to Bayes filter, and its continuous case forms the basis of Kalman filters (Howard 1960). The Kalman filter (Grewal & Andrews 1993; Maybeck 1979) addresses the general problem of trying to estimate the state of the agent. Nearly all algorithms for spatial reasoning make use of this approach.

Figure 1: Graphical representation of POMDPs.

The Kalman filter is a recursive estimator. This means that only the estimated state from the previous time step, action and the current measurement are needed to compute the estimate for the current state. In contrast to batch estimation techniques, no history of observations and/or estimates is required. A belief state is represented by Kalman filter with a mean, the most probable state estimation and a covariance, the state estimation error.

The graphical representation of POMDPs is given in Fig. 1. The Kalman filter relates the current state, s_t, to previous state, s_{t-1}, and action a_{t-1}, disturbed by actuator noise using a state transition equation. Likewise, the observation o_t is a function of the current state and disturbed by sensor noise. Both the actuator and the sensor suite of real world agents are subject to some noise. Kalman assumes that these are Gaussian as given in equations 4 and 5. In practice, the process noise (v) covariance, Q, and measurement noise (n) covariance, R, matrices might change with each time step or measurement, however in this research we assume that they are constant.

$$p(v) \sim N(0, Q) \tag{4}$$

$$p(n) \sim N(0, R) \tag{5}$$

In this research, we have linearized the motion model. The general linearized motion model equation is given in Eq. 6.

$$s_t = As_{t-1} + Ba_{t-1} + v_{t-1} \tag{6}$$

Likewise, we have also linearized the sensor model. The general linearized sensor model equation is given in Eq. 7.

$$o_t = Hs_t + n_t \tag{7}$$

The matrix A in Eq. 6 relates the state at the previous time step to the state at the current step, in the absence of either a driving function or process noise. Note that in practice A might change with each time step, but here we assume it is constant. The matrix B relates the optional control input to the state s. The matrix H in Eq. 7 relates the state to the measurement o. In practice H might change with each time step or measurement, but in this research we assume that it is also constant.

To compute the minimum mean square error estimate of the state and covariance, the following equations are used. The estimate of the state variables,

$$s^-_{t|t-1} = As_{t-1} + Ba_{t-1} \tag{8}$$

Where the subscript $(t|t-1)$ denotes that the value at time step t is predicted using the value at time t-1. The estimate of the sensor reading,

$$o_{t|t-1} = Hs_{t|t-1} \tag{9}$$

The covariance matrix for the state P,

$$P_{t|t-1} = AP_{t-1}\ A^T + Q \tag{10}$$

Kalman filter calculates parameters of a Gaussian distribution, the mean representing the most probable state and the covariance representing the uncertainty. In our study, recursively calculated Gaussian distribution represents a belief state. At each time step, the state and covariance is calculated recursively. This feature matches the belief state defined by Astrom (Astrom 1965), as a sufficient statistic representing the whole history. The Kalman filter divides these calculations into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback, i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. The final estimation algorithm resembles that of a predictor-corrector algorithm.

The specific equations of time update functions are given in Eqs. 8 - 10, and the specific equations of measurement update functions are as follows:

$$K_t = P_t^-\ H^T\ (HP_t^-H^T + R)^{-1} \tag{11}$$

$$s_t = s_t^- + K_t(o_t - Hs_t^-) \tag{12}$$

$$P_t = (I - K_tH)P_t^- \tag{13}$$

The superscript - at the Eqs. 11 - 13 denotes that the value calculated at the prediction phase will be updated in the correction phase. K_t, the Kalman gain, determines the degree of estimate correction. It serves to correctly propagate or magnify only the relevant component of the measurement information. Detail of the derivation of Kalman gain can be found in (Maybeck 1979) .

After each time and measurement update pair, the process is repeated with the previous a posteriori estimates used to project or predict the new a priori estimates. This recursive nature is one of the very appealing features of the Kalman filter; making it very suitable for POMDP problems which has to keep history. Moreover, it keeps the history implicitly by summarizing all previous states in the current state estimate and its covariance. This is obviously superior to some approaches that keep time window over past history.

Value Function

A policy is simply a mapping from beliefs to actions. Let us demonstrate this with an example. Imagine a clinic where a patient applies with symptoms of sneezing and fatigue. Dr. John has to diagnose and suspects that the patient suffers from either flu or allergy. For this POMDP problem we have two states: having flu or having allergy. In order to make the right decision, Dr. John may require tests. After each test, he may require additional tests or may write a prescription. The decision tree of Dr. John is given in Fig. 2 where each ellipse represents an action that is valid for a belief range over having flu given in parenthesis. The top ellipse represents the case when Dr. John's belief that the patient's sickness is flu is between 0.4 and 0.6. For this belief state, Dr. John's decision is to require additional tests.

Figure 2: Diagnostic decision tree.

If a policy, π, represented by a decision tree, collects maximum reward in the long term, it is called the *optimum policy*, π^*. Each node in the decision tree has a value representing the benefit of choosing the action for that belief

range. The benefit calculated by the value function for discrete POMDPs given in Eq. 14 and Eq. 15.

$$V_t^{\pi^*}(b) = \max_a Q_t^{\pi}(b,a) \tag{14}$$

Where $Q_t^{\pi}(b,a)$ is the value of performing action a at a belief state b at time step t. The calculation of this value is given in Eq. 15.

$$Q_t^{\pi}(b,a) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|b_t = b, a_t = a\right\} \tag{15}$$

where n is the planning horizon, γ is the discount factor, $0 \leq \gamma \leq 1$.

Proposed Approach

In this study, we aim to use POMDP as a model of a real environment in which an agent exists. We further assume that POMDP parameters are not known and that these parameters are continuous. These two assumptions make the problem even harder and such POMDPs cannot be solved exactly.

We use a Kalman filter to calculate Gaussian-based POMDP belief states, under the assumption of *A, H, Q* and *R* are time-invariant. This filter returns the best state estimate and its covariance, a measure of uncertainty around the best estimate. The uncertainty gradually converges depending on the measurement, process noise and initial uncertainty value. This feature eliminates the unvisited uncertainty values from value calculation. The Gaussian-based belief state calculated by the Kalman filter and the action which is chosen according the Boltzmann Softmax strategy (Sutton & Barto 1998), is fed to the temporal-difference neural network (TDNN) (Waibel, Sawai, & K. 1989) to calculate the $Q_t(b,a)$ for each logically grouped action for the current belief state. By the end of the tuning phase, at each decision step, TDNN's output which is the $Q_t(b,a)$ is used to choose the best action which is the one with the maximum Q-value.

Belief State Representation

Assume a one dimensional maze as shown in Fig. 3. For a discrete POMDP, initially the agent is in either one of the four cells, and the agent's belief can be represented as a vector given in Eq. 16.

$$b = [\ 0.25 \quad 0.25 \quad 0.25 \quad 0.25\ ] \tag{16}$$

Figure 3: One dimensional maze.

In the continuous state case it is impossible to represent each belief as a vector, since there are infinitely many states. So we have assumed the belief as a Gaussian probability distribution where the mean is the most probable current state and the covariance is the state estimation error or uncertainty. Belief formulation at step t is given in Eq. 17, where s_t is the most probable current state and P_t is the covariance.

Figure 4: One dimensional maze belief state space representations: (a) discrete case (b) continuous case with parametric distribution.

$$b_t(s) = N(s_t, P_t) \tag{17}$$

In the discrete case, the belief state space dimension is one less than the number of states as shown in Fig. 4(a). It is normally expected that for the continuous case since the number of states is infinite, the dimension of the belief state space is also infinite. Using a parametric distribution, i.e. Gaussian, for the belief state representation of the one dimensional maze reduces the infinite dimensions to two dimensions as shown in Fig. 4(b). In our experiments, the initial belief point covariance or uncertainty is $P_0 = P_{max}$. At the next decision steps Kalman filter decreases the uncertainty level until $P_{converge}$. Although $P_{converge}$ is always greater than 0 when an observation noise exists, the case where P equal 0 is given in the figure since we calculate the reward function from the definition for the MDP case as explained in subsequent sections.

The Reward Value Calculation

The reward for POMDP problems is always defined for the MDP case. For a MDP, whether discrete or continuous, the reward of taking action a for state s is constant. In the one dimensional maze POMDP problem illustrated in Fig. 3, there exists four discrete states and the goal state is marked with a circle. The reward definition for this problem is +1 for being at the goal state and 0 elsewhere for each possible action. The reward function can be represented by a four dimensional vector [0 0 0 1]. To calculate the reward at belief state [0.25 0.25 0.25 0.25], the vector product of the belief state vector and the reward vector for MDP case is taken which is $0.25 \times 0 + 0.25 \times 0 + 0.25 \times 0 + 0.25 \times 1 = 0.25$.

The formal definition of reward *r(b)* for a belief state *b* is given in Eq. 18, where n is the number of states.

$$r(b) = \sum_{k=0}^{n-1} b_k(s) r_k(s,a) \tag{18}$$

To calculate the reward for the belief state *N(s, P)*, we compute an infinite sum of the product of the MDP reward by its probability as shown in Fig 5. Since the reward in MDP case is constant and the state probabilities are represented as Gaussian, this infinite sum is equivalent to the integral in Eq. 19, which is equivalent to cumulative distribution function (cdf) of Gaussian across the range where reward is non-zero.

$$\lim_{\Delta s \to 0} \sum_{s=-\infty}^{s=\infty} b(s) r_a(s) \Delta s = \int_s b(s) r_a(s) ds \tag{19}$$

In our study, the reward is calculated at the completion of the action chosen according to Boltzmann Softmax strategy

Figure 5: The continuous case reward value calculation at a belief state.

using Eq. 19 which is the cdf of Gaussian, representing current belief state, across the range where reward is non-zero. The reward is used in the tuning phase of the TDNN at the decision layer.

Main Architecture

Our architecture has two layers as shown in Fig. 6.

Figure 6: Two layered KBTDNN architecture.

The TDNN at the first layer, shown in Fig. 7 is organized as a multilayer perceptron (MLP). The MLP is a generic nonlinear function approximator for q-values.

Figure 7: TDNN layer.

The TDNN architecture has one hidden layer with three inputs and one output nodes. The number of nodes in the hidden layer varies depending on the problem. Each node in the hidden layer uses hyperbolic tangent function given in Eq. 20 as the activation function and identity activation function for the output node.

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{20}$$

During the learning phase, the TDNN itself is used to decide on actions. At each time step, the neural network calculates q-values for each action given the current belief state calculated by the Kalman filter. The q-values are used in the action selection mechanism using the Boltzmann Softmax strategy, where probability of choosing an action *a* for a belief *b* is given in Eq. 21, where τ is a positive parameter called *temperature*. When τ is high, all actions have almost the same probability of being chosen. When $\tau \rightarrow 0$, the Boltzmann Softmax exploration more likely chooses the action *a* that has a high Q-value.

$$p(b, a) = \frac{e^{\frac{Q(b,a)}{\tau}}}{\sum\limits_{\forall a'} e^{\frac{Q(b,a')}{\tau}}} \tag{21}$$

The training procedure of the TDNN is as follows : The current belief state and the action from the previous step is fed as input vector v_t to the TDNN. For each input vector v_t a corresponding q-value which is the TDNN output is calculated. At each time step, the temporal-difference algorithm is applied to update the network's weights. The weight update rule is given in Eqs. 22, 23 and 24, where α is the learning rate, *W* is the weight of a link, a_j is the activation of a hidden unit, I_k is an input.

$$\Delta_i = r + Q_{t+1}(b', a') - Q_t(b, a) \tag{22}$$

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i \tag{23}$$

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times I_k \times \tanh'(in_j) \sum_i W_{j,i} \Delta_i \tag{24}$$

The reward, *r*, in Eq. 22 is calculated using the technique in section . The second layer is a Kalman filter (Grewal & Andrews 1993; Maybeck 1979), which addresses the general problem of trying to estimate the state of the agent. It is governed by stochastic equations Eq. 6 and Eq. 7. At each decision step, after the agent decides and executes the action according to Boltzmann Softmax strategy in the tunning phase, Kalman filter computes the minimum mean square error estimate of the current state. The estimate is the current belief state which is represented by most probable state *s* and the covariance or state estimation error *P*. The output of the Kalman filter which is the current belief state is again fed to the TDNN layer.

Experimental Results

The algorithm is run on some well known POMDP problems, the tiger problem and the one dimensional maze problem. The learning rate for all experiments is 0.1.

The Tiger Problem

The tiger problem has been converted to continuous observation over discrete states case by Hoey and Poupart (Hoey & Poupart 2005). Although our algorithm works for continuous observation over continuous states, we modeled the tiger problem as defined in (Hoey & Poupart 2005). We assumed that *B* is equal to the zero matrix, *A* and *H* are equal to the identity matrix and only the observation noise exists, *N*$(0, 0.625)$, i.e., there is no actuator noise. The penalty of opening the door with the tiger is -100, the reward of opening the door without the tiger is +10 and the cost of listening is -1. s_0 is 0 and P_0 is 0.625. TDNN has five nodes in the hidden layer for this problem. The resulting policy decides on a door after listening for three steps where the state estimation error reduces to 0.1563. The decision for selecting the door without the tiger depends on the observation. Fig. 8 shows a sample test run, where the tiger is located at the right door. The agent listens for three steps until it estimates the tiger's most probable location greater than 1.0, then it opens the door located at -1.0 which is the left door.

Figure 8: Representative belief states for the tiger problem.

The One Dimensional Maze Problem

In this problem, the goal is placed at the right end. The agent, can move *EAST* and *WEST*. The agent has a single range finder. We assumed *A*, *B* and *H* are equal to the identity matrix and the sensor and the actuator noises are *N*$(0, 0.625)$ and *N*$(0, 0.625)$ respectively. Initially E(s_0) is 0.5, and P_0 is 1.0. TDNN assumes three nodes in the hidden layer for this problem. Fig. 9 shows a sample test run after

the policy has converged. The agent always moves right till the goal location. Note that as the agent moves right it is more certain of its state.

Figure 9: Representative belief points for the one dimensional maze problem.

Collected Rewards

For each experiment the approximate policy of KBTDNN and the MDP case optimal policy is compared in terms of expected reward. The expect reward for approximate policy of KBTDNN is the average of 150 sample runs. The results are given in Table 1. For the tiger problem, the MDP case optimal policy where no noise exists, the agent listens only once and then opens the right door. Likewise, for the one dimensional maze problem, for the optimal policy MDP case the agent the agent always collects the maximum reward, 1.

.	Tiger	1D Maze
Optimal Policy (MDP)	9	1
Approximate Policy	5.06 ± 0.34	0.733 ± 0.06

Table 1: Comparison of collected rewards with MDP case.

Conclusion

The KBTDNN algorithm introduced in this study deals with continuous Gaussian-based POMDPs, and as a new approach suggests making use of Kalman filter for state estimation and temporal-difference neural network at the decision layer. The Kalman filter approach assumes that the motion and sensor models are given and their noise can be modeled as Gaussian. It enables easy pruning of unreachable belief points.

Nearly all previous studies assumed discrete POMDPs. As a comparative work, Porta, Spaan and Vlassis (Porta, Spaan, & Vlassis 2005) suggested a method for Gaussian-based POMDPs with continuous state space and discrete observation space. Alternatively, Hoey and Poupart (Hoey & Poupart 2005) suggested a method for Gaussian-based POMDPs with discrete state space and continuous observation space.

Both assume that the world model is given, and use PBVI technique (Pineau, Gordon, & Thrun 2003) for Gaussian-based POMDPs. They calculated value iteration on a pre-collected set of beliefs, while the Bellman backup operator is analytically computed given the particular value function representation. Depending on the POMDP problem, they have to keep a large number of belief points and corresponding α-vectors, in order to calculate the next horizon value function, rather KBTDNN keeps a small number of weights which composes the TDNN.

Moreover, KBTDNN handles continuous state and observation space. For generating the optimal policy, KBTDNN does not keep α-vectors and has no belief point set expansion and pruning phase for α-vectors, rather KBTDNN keeps only the weights of TDNN. In summary, KBTDNN is time and space efficient, as opposed to other point based methods.

In real world applications, motion and sensor noises are rarely constant. For example, when an agent navigates on a carpet, it is normally expected to have a different noise characteristic compared to navigation on a marble floor. Our study assumed that motion and sensor noises are assumed to be constant. As a future work, we plan to adapt the method for dynamically changing noise characteristics.

References

Astrom, K. 1965. Optimal control of markov decision process with incomplete state estimation. *Mathematical Analysis and Applications* 10:174–205.

Boyen, X., and Koller, D. 1998. Tractable inference for complex stochastic processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 33–42.

Grewal, A., and Andrews, C. 1993. *Kalman Filtering*. Prentice Hall.

Hoey, J., and Poupart, P. 2005. Solving pomdps with continuous or large discrete observation spaces. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 1332–1338. Edinburgh, Scotland: IJCAI.

Howard, R. 1960. *Dynamic Programming and Markov Processes*. The MIT Press, Cambridge MA.

Kaelbling, L.; Littman, M.; and Moore, A. P. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.

Maybeck, P. 1979. *Stochastic models, estimation and control*. Academic Press.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for pomdps. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Acapulco, Mexico: IJCAI.

Porta, J.; Spaan, M.; and Vlassis, N. 2005. Robot planning in partially observable continuous domains. In S. Thrun, G. S. S., and Schaal, S., eds., *Robotics: Science and Systems I*, 217–224. MIT Press.

Sallans, B. 2000. Learning factored representations on partially observable markov decision process. In *Neural Information Processing Systems*, 1050–1056. MIT Press.

Sondik, E. 1978. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs,. *Operations Research* 26:282–304.

Sutton, R., and Barto, A. G. 1998. *Reinforcement Learning I: An Introduction*. The MIT Press, Cambridge MA.

Waibel, A.; Sawai, H.; and K., S. 1989. Modularity and scaling in large phonemic neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing* 12:1888–1898.