Dynamics Based Control: Structure

Zinovi Rabinovich Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University, Jerusalem, Israel

Abstract

In this paper we introduce a novel approach to continual planning and control, called *Dynamics Based Control* (DBC). The approach is similar in spirit to the Actor-Critic [6] approach to learning and estimation-based differential regulators of classical control theory [13]. However, DBC is not a learning algorithm, nor can it be subsumed within models of standard control theory. We provide a general framework for applying DBC to discrete Markovian environments, and discuss the key differences between it and a popular alternative for this type of environment — Partially Observable Markov Decision Processes (POMDPs). We then show how a recently developed control scheme based on Extended Markov Tracking (EMT) [9, 10] can be seen as a suboptimal algorithm within the DBC framework, and discuss EMT's limitations relative to the general DBC approach.

1 Introduction

Consider NASA engineers who encounter a serious problem: two spacecraft, while attempting to dock in orbit, keep crashing or whirring out of control. At first, control schemes and sensors are blamed for being inexact, and great efforts are invested in perfecting the precision of spacecraft positioning. Bizarrely, as precision improves, the problem perseveres and becomes even more violent. A surprising solution is proposed that finally solves the problem — reduce positioning precision. As long as orbits do not change too much, docking can be done using a simple funnel-like mechanism. This kind of problem, control of change over time, or in other words *control of system dynamics*, is what we describe in this paper.

The central idea behind Dynamics Based Control (DBC) is that if laws governing changes in a system, the system dynamics, are beneficial in some sense, then neither state estimation precision nor the state itself matter as much. These system dynamics, like the funnel of a vortex, will channel the system into the desired configuration. The target of a DBC algorithm would be to select actions in such a way as to create or simulate the beneficial dynamics.

Recalling our spacecraft scenario, applying Dynamics Based Control (DBC) would mean that instead of attempting to set the orbit position directly, one should ensure that it has the tendency to recover from deviations from an ideal orbit. This tendency would be the beneficial system dynamics, and it requires much less effort than precise positioning of a spacecraft. A similar effect is observed while driving a car (the "small corrections principle"); there is no attempt to precisely position the car in the middle of a car lane, but rather it is directed to return there.

The "small corrections principle" has another interpretation, that of classical control theory [13]. Control theory has a concept of *neighboring control*. Should the system manifest itself in an ideal, noiseless way, the control signal (actions selected) would be clear. But if a (small) deviation in *system state* occurs, the control signal can be augmented by a small difference as well, usually proportional to the real or estimated state-difference, as observed in literature on linear systems [3, 13]. Though DBC uses terminology such as *system deviation*, the focus is on the rules that govern the system, rather than on the system state itself.

The main principle of action selection in DBC can be split into two phases: *dynamics estimation* and *dynamics correction*. Estimation can be done in numerous ways, starting from true system identification by a complete learning algorithm, and ending with lighter tracking algorithms, such as Extended Markov Tracking (EMT) [9, 10]. Given the way estimation is done, one can devise actions so that future system behavior will produce estimates similar to the ideal or beneficial system dynamics.

DBC-type action selection is actually quite widespread. Consider, for example, color creation on a computer screen. Human color perception (i.e., light frequency estimation) is based on three distinct receptors, each detecting a specific frequency or hue. This is used in color monitors, where most colors are not actually emitted by the screen. Instead three sources of constant hue are tapped into, and a mixture is created that simulates for the human eye the required color. This scheme parallels the phases of *estimation* and *correction* as performed by DBC.

A similar dual structure can be found in learning algorithms such as Actor-Critic algorithms [6]. A Critic estimates the value function, a system performance evaluator, while an Actor uses the estimation to refine its strategy. However, Actor-Critic algorithms are learning algorithms, and produce at the end a good value function estimate and the corresponding, usually static, strategy. The DBC framework by itself is not a learning algorithm, though it can be utilized in creating one (see below).

The rest of the paper is organized as follows. In Section 2 we provide a formal introduction to Dynamics Based Control, focusing especially on Markovian

stochastic environments. This is followed in Section 3 by comparison to a classical control alternative for these environments — Partially Observable Markov Decision Processes (POMDPs). Section 4 demonstrates how recently introduced EMT-based control fits the DBC architecture under the Markovian assumption, and also exposes the limitations of naive EMT-based control, as opposed to the general DBC framework. Section 5 provides some concluding remarks and directions for future developments of DBC.

2 Dynamics Based Control

Dynamics Based Control (DBC) specification can be broken into three interacting levels: Environment Design Level, User Level, and Agent Level.

- **Environment Design Level** concerns itself with the formal specification and modeling of the environment. For example, this level would specify the laws of physics within the system, and set its parameters, such as the gravitation constant.

- **User Level** in turn relies on the environment model produced by Environment Design to specify the ideal or beneficial system dynamics it wishes to observe. The User Level also specifies the estimation or learning procedure for system dynamics and the measure of deviation. In our spacecraft docking scenario these would correspond to specifications of self-stabilization at an optimal orbit, and angular speed and radii difference evaluations.

- **Agent Level** in turn combines the environment model from the Environment Design level, and the dynamics estimation and ideal dynamics specification from the User Level, to produce a sequence of actions that create system dynamics as close as possible to the ideal one with respect to the deviation measure specified by the User Level.

As we are interested in the continual development of a stochastic system, such as happens in classical control theory [13] and continual planning [5], the question becomes how the Agent Level is to treat the deviation measurements over time. A classical approach would be to use expectation, to average over all possible system developments. However, we prefer to use a probability threshold alternative — that is, we would like the Agent Level to maximize the probability that the deviation measure will remain below a threshold.

Specific action selection would depend on system formalization. One possibility would be to create a mixture of available system trends, much like what

happens in Behavior-Based Robotic architectures [1]. The other alternative would be to rely on the estimation procedure provided by the User Level, and utilizing the Environment Design Level model of the environment to choose actions so as to manipulate the dynamics estimator to believe that a certain dynamics has been achieved. Notice that this manipulation is not direct, but via the environment. Thus, for strong enough estimator algorithms, successful manipulation would mean a successful (i.e., beyond discerning via the available sensory input) simulation of the ideal or beneficial system dynamics.

DBC levels can also have a back-flow of information (see Figure 1). For instance, the Agent Level could provide data about ideal dynamics feasibility, allowing the User Level to modify the requirement, perhaps focusing on attainable features of system behavior. Data would also be available about the system response to different actions performed; combined with a dynamics estimator defined by the User Level, this can provide an important tool for the environment model calibration at the Environment Design Level.

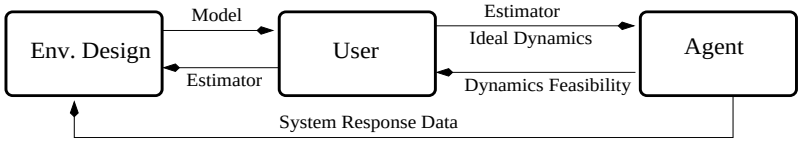

Figure 1: Data flow of the DBC framework

Extending the idea of Actor-Critic algorithms [6], DBC data flow can provide a good basis for design of a learning algorithm. For example, the User Level can operate as an exploratory device for a learning algorithm, inferring an ideal dynamics target from the environment model at hand that would expose and verify most critical features of system behavior. In this case, feasibility and system response data from the Agent Level would provide key information for an environment model update. In fact, the combination of feasibility and response data can provide a basis for the application of strong learning algorithms such as EM [2, 8].

2.1 DBC for Markovian Environments

For a Partially Observable Markovian Environment, DBC can be specified more rigorously. In this case, the phases or levels of DBC can be seen as follows:

- **The Environment Design** level specifies a tuple $< S, A, T, O, \Omega, s_0 >$:
  - S is the set of all possible environment states;

- s_0 is the initial state of the environment (which can also be viewed as a distribution over S);
- A is the set of all possible actions applicable in the environment;
- T is the environment's probabilistic transition function: $T : S \times A \to \Pi(S)$. That is, $T(s'|a, s)$ is the probability that the environment will move from state s to state s' under action a;
- O is the set of all possible observations. This is what the sensor input would look like for an outside observer;
- Ω is the observation probability function: $\Omega : S \times A \times S \to \Pi(O)$. That is, $\Omega(o|s', a, s)$ is the probability that one will observe o given that the environment has moved from state s to state s' under action a.

- **The User Level**, in the case of a Markovian environment, operates on the set of system dynamics described by a family of conditional probabilities $\mathcal{F} = \{\tau : S \times A \to \Pi(S)\}$. Thus ideal or beneficial dynamics can be described by $q \in \mathcal{F}$, and the learning or tracking algorithm can be represented as a function $L : O \times (A \times O)^* \to \mathcal{F}$; that is, it maps sequences of observations and actions performed so far into an estimate $\tau \in \mathcal{F}$ of system dynamics.

  There are many possible variations available at the User Level to define divergence between system dynamics; several of them are:

  - *trace (or L_1) distance* between two distributions p and q defined by

$$D(p(\cdot), q(\cdot)) = \frac{1}{2} \sum_x |p(x) - q(x)|$$

  - *Fidelity* measure of distance

$$F(p(\cdot), q(\cdot)) = \sum_x \sqrt{p(x)q(x)}$$

  - *Kullback-Leibler divergence*

$$D_{KL}(p(\cdot) \| q(\cdot)) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

  Note that the latter two are not actually metrics over the space of possible distributions, but still have meaningful and important interpretations. For

instance, Kullback-Leibler divergence is an important tool of information theory [4] that allows one to measure the "price" of encoding an information source governed by q, while assuming that it is governed by p. The User Level also defines the threshold of dynamics deviation probability θ.

- **The Agent Level** is then faced with a problem of selecting a control signal function a^* to satisfy a minimization problem as follows:

$$a^* = \arg\min_a Pr(d(\tau_a, q) > \theta)$$

where $d(\tau_a, q)$ is a random variable describing deviation of the dynamics estimate τ_a, created by L under control signal a, from the ideal dynamics q. Implicit in this minimization problem is that L is manipulated via the environment, based on the environment model produced by the Environment Design Level.

3 DBC vs. POMDPs

Comparison of the DBC approach to classical control can be done via comparison of DBC to Partially Observable Markov Decision Processes (POMDPs). Since POMDPs are a classical representative of the control theory of stochastic systems, and DBC has a well-formed formalization over Markovian environments, comparison between them illuminates key features of the DBC approach.

Structurally, POMDPs can also be fitted into the three levels of the Environment Design Level, User Level, and Agent Level. Furthermore, one can assume that the Markovian environment model produced by the Environment Design Levels of both approaches completely overlap (see Table 1 for a structural comparison). However, the User and Agent Levels differ significantly. At the User Level, instead of idealized system dynamics, the POMDP approach defines a reward function $r : S \times A \times S \to R$ to express preferences over different system transitions. POMDP at the User Level also defines an optimality criterion, determining how the reward should be treated over time, e.g., discounted accumulated optimality dictates that reward is additively collected with every next step being less profitable by a discount factor. The Agent Level then faces the problem of finding an action selection policy so as to maximize the expected optimum reward. For instance, in the case of discounted accumulated optimality this would be $\pi^* = \arg\max_\pi E\left[\sum \gamma^i r_i\right]$, where r_i denotes the reward obtained at time slice i under the policy π, and $0 < \gamma \leq 1$ is a discount factor.

Level	Approach	
	MDP	**Markovian DBC**
Environment **Design**	$< S, A, T, O, \Omega >$,where S — set of states A — set of actions $T : S \times A \to \Pi(S)$ — transition O — observation set $\Omega : S \times A \times S \to \Pi(O)$	
User	$r : S \times A \times S \to \mathcal{R}$ $F(\pi^*) \to r$ r — reward function F — reward remodeling	$q : S \times A \to \Pi(S)$ $L(o_1, ..., o_t) \to \tau$ q — ideal dynamics L — dynamics estimator θ — threshold
Agent	$\pi^* = \arg\max_\pi E[\sum \gamma^t r_t]$	$\pi^* = \arg\min_\pi Prob(d(\tau \| q) > \theta)$

Table 1: POMDP vs. Dynamics-Based Control in a Markovian Environment

In some cases the POMDP approach also defines a reward function remodeling procedure $F(\pi^*) \to r$, that transforms the reward function according to some features of the resulting action selection behavior. This way the POMDP approach achieves reward function composition that results in a system behavior with specific features. Contrast this with DBC, which specifies the idealized system behavior directly.

Available complexity results for POMDPs, such as [7], render many optimality criteria infeasible, and limit the choice to discounted accumulated optimality, which we assume for the rest of our discussion. The following two subsections discuss properties of POMDP and DBC-induced policies in the light of features such as user preference interpretation and similarity to controller mechanisms. The summary of this discussion is available in Table 2, with numbers in parentheses denoting the subsection in which specific issues are discussed.

3.1 Properties of POMDP-Induced Policy

Even within its framework, POMDP-induced policy has one important technical limitation — its optimization is *expectation* oriented, and is applied as is for all possible system developments. Computed off-line, POMDP-induced policy is, in a sense, an open-loop control mechanism.

Issue	**POMDPs** (see 3.1)	**DBC** (see 3.2)
Action Selector	Off-line computed policy	On-line action selection
Controller Similarity	Open-loop (3.1.2)	Closed-loop (3.2.1)
User preference interpretation	State value oriented (3.1.3)	Transition frequency oriented (3.2.2)
Optimality concept	Expectation oriented (3.1.1)	Context dependent, situated (3.2.3)
Complexity	PSpace	Unknown for DBC

Table 2: Features of POMDP and DBC-induced policies

3.1.1 Optimality Concept

POMDP policy selection dictates that a policy with a maximum *expected* utility be taken as optimal. Let us concentrate on this notion of *expectation*. Consider two policies π_1 and π_2 applied to the same POMDP, and distributions of the (accumulated) value under these two policies D_1 and D_2 respectively, as shown in Figure 2. POMDP optimization dictates that the policy π_1 should be selected and applied at all times. However, the variance of the value under π_1 is high, which means that if a critical value exists below which the gain is forbidden to drop (as denoted by α in Figure 2) — π_1 may not be suitable.

The expectation problem can be discussed from another point of view, that of system development. POMDP policies can be seen as a solution set for a Sequential Decision Making (SDM) problem. Each policy creates a distribution over system developments (state-action sequences), and one has to choose among different policies based on a comparison of distributions. An SDM problem defines a preference order over system development, in a sense, determining an ideal system development and a measure of divergence between development sequences. The POMDP concept of solution combines policy-induced distribution and SDM-induced preference over the state-action sequences by assigning real values to sequences and optimizing over the value expectation. Returning to our previous argument, we see that POMDP-optimal policy can have a very high probability of large deviations from an ideal system development sequence, a property unintended by, and not desired by, the SDM formulation.

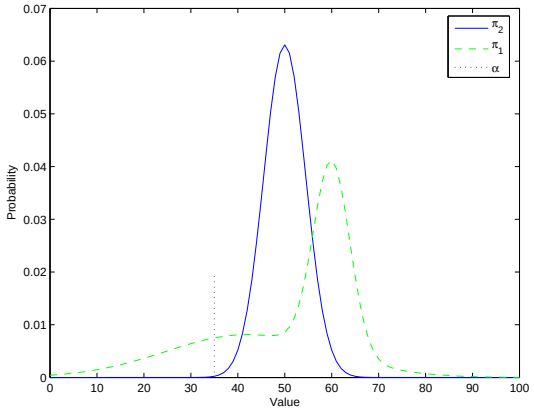

Figure 2: Value distribution under POMDP policies

3.1.2 Controller Similarity

Although POMDP policies can be history dependent, description length feasibility argues that a policy will have only finite (and rather small) variability in response to different system developments that actually take place. This, together with off-line computation and optimality based on average performance, underscores POMDP policies' inability to adapt and change for a specific system development that actually occurs, and which deviates from the *expected* development under application of the policy.

3.1.3 User Preference Interpretation

POMDP solutions also exhibit a form of user-preference distortion. By formal definition, reward/cost is obtained based on specific transitions exhibited by the system. However, a POMDP's induced policy is computed with respect to a Value Function, which is a state-oriented, rather than transition-oriented, concept. This may potentially cause distortion of User preferences when expressed by the reward structure.

3.2 Properties of DBC-Induced Policy

Dynamics Based Control (DBC) is a closed-loop control scheme, and action selection is bound to a system development that actually occurs at run-time. DBC is also directly tied to system-transition preference, expressed as the ideal system dynamics.

3.2.1 Controller Similarity

DBC is a situated, context-dependent scheme because it uses an estimation algorithm L that is used to identify the system development that actually takes place at run-time. In fact, DBC (and EMT-based control as a representative of a DBC scheme) takes this to an extreme, in that it views L as the "actual" system, and selects actions in a way that leads L to a required estimate.

It is important to note that L is not influenced directly; otherwise, its estimate and DBC action-selection make no sense. Rather, L is influenced only through the environment that it estimates. This mode of operation is most similar to a closed-loop differential controller, and is explicitly on-line. Note, also, that DBC formally is not a learning, tracking, or estimation algorithm by itself, although it has a tracking (estimator) component utilized with its structure.

3.2.2 User Preference Interpretation

The DBC discussion and POMDP discussion above can be unified by taking the point of view of distributions over different system development sequences. Unlike POMDP, the DBC framework attempts a direct comparison between distributions over system development sequences. This is done by considering "inductor" stochastic functions, that represent different stochastic behaviors and thus different distributions over system development sequences. DBC assumes that the ideal system development sequence and the measure of deviation from it can be combined into such an "inductor" function, q — *the ideal system dynamics*. The learning/tracking algorithm L is then used to seek the same form of representation for the actual system development, τ.

3.2.3 Optimality Concept

A DBC policy considers τ and q, that is, the "estimated actual" and the "ideal" distributions over system development sequences, and selects an action based on their direct comparison. The DBC concept of optimality is based on a threshold probability, and an optimal policy is the one that is capable of keeping deviation from the ideal system dynamics below the threshold with highest probability.

4 EMT-based Control as a DBC

Recently, a control algorithm was introduced called *EMT-based Control* [9, 10], which fits perfectly into the DBC framework. Although it presents an approximate

greedy solution in the DBC sense, initial experiments using EMT-based control have been encouraging [11]. EMT-based control is based on the Markovian environment definition, as in the case of POMDPs, but its User and Agent Levels are of the DBC type.

- **The User Level** of EMT-based control defines a limited-case, idealized system dynamics independent of action: $q_{EMT} : S \rightarrow \Pi(S)$. It then utilizes the Kullback-Leibler divergence measure to compose a momentary system dynamics estimator — the Extended Markov Tracking (EMT) algorithm. The algorithm keeps a system dynamics estimate τ^t_{EMT} that is capable of explaining recent change in an auxiliary Bayesian system state estimator from p_{t-1} to p_t, and updates it conservatively using Kullback-Leibler divergence. Since τ^t_{EMT} and $p_{t-1,t}$ are respectively the conditional and marginal probabilities over the system's state space, "explanation" simply means that $p_t(s') = \sum_s \tau^t_{EMT}(s'|s)p_{t-1}(s)$, and the dynamics estimate update is performed by solving a minimization problem:

$$\tau^t_{EMT} = H[p_t, p_{t-1}, \tau^{t-1}_{EMT}] = \arg\min_{\tau} D_{KL}(\tau \times p_{t-1} \| \tau^{t-1}_{EMT} \times p_{t-1})$$
$$s.t.$$
$$p_t(s') = \sum_s (\tau \times p_{t-1})(s', s)$$
$$p_{t-1}(s) = \sum_{s'} (\tau \times p_{t-1})(s', s)$$

- **The Agent Level** in EMT-based control is sub-optimal with respect to DBC (though it remains within the DBC framework), performing greedy action selection based on prediction of EMT's reaction. The prediction is based on the environment model provided by the Environment Design level, so that if we denote by T_a the environment's transition function limited to action a, and p_{t-1} the auxiliary Bayesian system state estimator, then EMT-based control choice is described by

$$a^* = \arg\min_{a \in A} D_{KL}(H[T_a \times p_t, p_t, \tau^t_{EMT}] \| q_{EMT} \times p_{t-1})$$

Note that this follows the DBC framework precisely; a dynamics estimator (EMT in this case) is manipulated via action effects on the environment to produce an estimate close to the idealized system dynamics. Yet naive EMT-based control is suboptimal in the DBC sense, and has several additional limitations that do not exist in the general DBC framework.

4.1 EMT-based Control Limitations

EMT-based control is a sub-optimal (in the DBC sense) representative of the DBC structure. It limits the User by forcing EMT to be its dynamic tracking algorithm, and replaces Agent optimization by greedy action selection. This kind of combination, however, is common for on-line algorithms. Although further development of EMT-based controllers is planned, evidence so far suggests that even the simplest form of the algorithm possesses a great deal of power, and trends that are optimal (in the DBC sense of the word).

There are two further, EMT-specific, limitations to EMT-based control that are evident at this point. Both already have partial solutions and are subjects for future research.

The first limitation is the problem of negative preference. In the POMDP framework, for example, this is captured simply, through the appearance of values with different signs within the reward structure. For EMT-based control, however, negative preference means that one would like to *avoid* a certain distribution over system development sequences; EMT-based control, however, concentrates on getting as *close* as possible to a distribution. Avoidance is thus unnatural in native EMT-based control.

The second limitation comes from the fact that standard environment modeling can create *pure sensory actions* — actions that do not change the state of the world, and differ only in the way observations are received and the quality of observations received. Since the world state does not change, EMT-based control would not be able to differentiate between different sensory actions.

Notice that both of these limitations of EMT-based control are absent from the general DBC framework, since it may have a tracking algorithm capable of considering pure sensory actions and, unlike Kullback-Leibler divergence, a distribution deviation measure that is capable of dealing with negative preference. Furthermore, as was mentioned, EMT-based control itself has a number of possible extensions that promise to be less prone, or not at all prone, to the aforementioned limitations.

5 Conclusions and Future Work

In this paper we have formally introduced a novel control framework — *Dynamics Based Control* (DBC). Unlike existing approaches to control and planning, DBC is directly involved with the rules that govern change within the controlled system

— system dynamics. Instead of limiting its attention to quantitative changes of system state and their estimation, DBC attempts to estimate and vary the qualitative features of system modulations. A Dynamics Based Controller can operate either as a direct mixer of available sources of system dynamics, or as a simulator with respect to a given dynamics estimator.

The DBC framework, even limited to well-studied Markovian environments, significantly differs in its approach and the resulting control signal (action policy) from the classical control approach, as represented (for example) by Partially Observable Markov Decision Processes.

It is important to note, however, that DBC and POMDPs are not rival, but rather parallel, approaches. Furthermore, the DBC approach, since it is related to on-line noise-resistant control methods, can be used together with POMDP. For instance, one can configure DBC to maintain the system dynamics induced by a POMDP solution policy, forming a kind of "neighboring control" [13], which exhibits both maximum payoff expectation and performance guarantees.

Although at this point a complete and exact solution of the DBC framework is not available, a promising beginning in this direction exists — control based on Extended Markov Tracking (EMT) [9, 10]. EMT-based control is computationally efficient, and in fact is polynomial in environment description parameters. Initial experiments using EMT-based control have been encouraging [11], and the trend continues with further development [12].

EMT-based control does have limitations, the main two being negative dynamics preference and pure sensory actions. It seems, however, that extended state representations may resolve these problems, and this direction is currently under investigation.

DBC also requires extensive feasibility and complexity studies, paralleling those done for POMDPs, as well as utilizing its potential for the composition of learning algorithms. The latter would be significant for the field of robotics, providing flexible automated techniques for environment model calibration simultaneous with task performance. In this direction it would be interesting to take a closer look at the application of EMT combined with EM, as the latter can use Kullback-Leibler divergence for its operation, thus directly making use of information from an EMT-based controller.

6 Acknowledgment

This work was partially supported by Israel Science Foundation grant #039-7582.

References

[1] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.

[2] J. A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and Hidden Markov Models. Technical Report TR-97-021, Dept. of EECS, Univ. of Calif., Berkeley, 1998.

[3] C.-T. Chen. *Linear System Theory and Design*. Oxford Univ. Press, 1999.

[4] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley, 1991.

[5] M. E. desJardins, E. H. Durfee, C. L. Ortiz, and M. J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 4:13–22, 1999.

[6] V. R. Konda and J. N. Tsitsiklis. Actor-Critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[7] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence Journal*, 147(1–2):5–34, July 2003.

[8] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

[9] Z. Rabinovich and J. S. Rosenschein. Extended Markov Tracking with an application to control. In *Workshop on Agent Tracking: Modeling Other Agents from Observations, at AAMAS'04*, pages 95–100, New York, 2004.

[10] Z. Rabinovich and J. S. Rosenschein. Multiagent coordination by Extended Markov Tracking. In *AAMAS'05*, pages 431–438, Utrecht, July 2005.

[11] Z. Rabinovich and J. S. Rosenschein. Robot-control based on Extended Markov Tracking: Initial experiments. In *The Eighth Biennial Israeli Symposium on the Foundations of Artificial Intelligence*, Haifa, Israel, June 2005.

[12] Z. Rabinovich and J. S. Rosenschein. On the response of EMT-based control to interacting targets and models. In *AAMAS'06*, Hakodate, Japan, May 2006. To appear.

[13] R. F. Stengel. *Optimal Control and Estimation*. Dover Publications, 1994.