Dynamics Based Control and Continual Planning

Thesis submitted for the degree of
"Doctor of Philosophy"

by

Zinovi Rabinovich

Submitted to the Senate of the Hebrew University

September 2007

This work was carried out under the supervision of

Prof. Jeffrey S. Rosenschein

Abstract

Human society has long tried to control its environment, and this has also been one of the more prominent tasks of, and applications for, artificial intelligence (AI) systems. With the rise of the concept of an *agent* within AI, it became of even greater importance to endow these agents with the capability to control their environments. Given real-world limitations on agents, however, it became crucial to develop control mechanisms that go beyond standard control theory, incorporating bounded reasoning into these AI systems, recognising and utilising the agent's limitations regarding computational effort and complexity.

This thesis introduces Dynamics Based Control (DBC)—a novel framework for continual planning and control in stochastic environments. While it can be related to the principles of model-following and perceptual control (and in fact uses these principles as a part of its philosophical intuition), DBC directly targets system dynamics, rather than the system state. DBC views the sensory subsystem of an agent as a continual environment dynamics estimation and identification algorithm, and concentrates on the sensory subsystem as its control subject. As the sensory system limits the agent's ability to decipher the world, it makes little sense to attempt to invest more effort into controlling an agent's surroundings than can actually be detected. Thus the Dynamics Based Control (DBC) framework, following the perceptual control principle, has us design an agent's behaviour not to explicitly enforce preferred environment circumstances, but rather to create conditions within the environment that would be recognised by the sensory system as the complete preferred circumstances. This would result in completion of the

control task to the extent that can be detected, while economising on the effort to create refinements to the control task, which would not be detected even if they do take place.

Being a general and flexible framework, DBC can potentially have many algorithmic solutions and instantiations within different types of environments. The thesis concentrates on the DBC adaptation to Markovian stochastic environments, and formulates a specific system dynamics estimation algorithm for such environments — Extended Markov Tracking (EMT). EMT bases its estimate on two system state distribution vectors, which represent a single environment modification, and a previous system dynamics estimate. EMT thus performs a conservative update, producing a new dynamics estimate that explains away the change in the system state distribution, while remaining as close as possible to the previous estimate.

Based on the EMT estimator, and utilising its polynomial time performance, an approximate greedy algorithmic solution to the DBC control task is then developed and experimentally shown to be operational. EMT-based control utilises EMT to predict the effects of an action, and greedily selects an action that would bring the EMT estimate closest to the specified ideal system development. The resulting overall control scheme, in spite of being only an approximation, implements all basic elements and properties of the DBC framework.

During its operation, EMT-based control does not provide the EMT algorithm with the true system state transition data. Instead, the EMT algorithm is provided with the sequence of system state beliefs. This means that an EMT-based controller not only relies on EMT to identify the system dynamics, but also regards it as a filter, discarding noise from the dynamic system representation. This filtering

capability has been experimentally verified by a construction of an environment model calibration algorithm based on the EMT data.

The dynamics estimate of the environment, provided by EMT, has also enabled the construction of multi-agent and multi-target versions of the EMT-based controller. In the multiagent case, it is conjectured and empirically verified that in certain domains an implicit information transfer between the agents is formed, enabling efficient coordinated performance without explicit communication. In the multi-target version, EMT data forms a preference vector, enabling the merger of potentially conflicting behavioural requirements.

Besides being a greedy approximation, several additional limitations of the EMT-based control scheme, which stem from the EMT dynamics estimator itself, have been identified by this research. This has led to the formulation of additional directions of research, applying the DBC framework to other environment models and domains, including Predictive State Representations and repeated games with dynamically developing opponents.

Contents

List of Figures

Chapter 1

Introduction

"If you want to make God laugh, tell him about your plans."

Woody Allen

Ever since the dawn of humanity, people have tried to control their environment. The desire to see the world change for our benefit has pushed forward the technology of control and planning, from making a fire and plans to cooperatively hunt, to the flight-control computers of a modern aircraft, from narrative hunt stories to a variety of formal methods and mathematical models. It should then come as no surprise that among the very first questions posed for Artificial Intelligence systems was how to make computers plan automatically.

In the most classical AI view, the world is seen as one of a preset group of states, and an intelligent entity can deterministically move the world from one state to another. One instantiation of this model came to be known as State Oriented Domains (SODs) [83] and is governed by the Closed World Assumption (CWA) [87]. Under these conditions, planning and control were seen as a search for a sequence of actions that change the world to fit some constraints on the state.

Approaches like STRIPS [29] flourished.

However, researchers quickly realised that the CWA has severe limitations. Unexpected failures and uncertain environment responses spurred the improvement of planning and control methodologies. Contingency resistant planning appeared, giving rise to *conditional plans* [68, 23, 14], and planner systems like Weaver and PRODIGY [13, 100]. Yet, these plans suffer from scalability problems: tracing all possible exogenous events and their effects tends to increment plan size exponentially, sometimes making the entire exercise infeasible.

Refinement of plans and use of a plan hierarchy [30, 15, 62] became the next step in attempting to solve the scalability challenge. This ultimately resulted in a form of planning called *continual planning* [24], where a plan is generally updated and corrected over time, rather than just refined with details. Unfortunately, too many approaches to continual planning have used an *update upon failure* policy, which results once again in increased complexity and cost, since the plan has to be rebuilt. Though minimisation of plan changes is possible [27] and the re-planning approach has been applied in some complex domains [60], preventive treatment may be more beneficial over time despite the cost endured at each step. The *update upon failure* policy also lacks pro-activeness, an essential element of what we now consider an intelligent agent [109].

Another widespread framework for planning is that of decision theory [13, 14, 16]. Decision theory has introduced the notion of *utility*, or *preference*, into planning considerations [87]. Further interconnecting with stochastic game theory, it developed into one of the most used models for decision making under uncertainty: Markov Decision Processes (MDPs). MDPs [72] are capable of modelling uncertainty in action outcomes, as well as an elaborate range of utility gains over

time. An MDP model assigns utilities (reward/cost) to system state transitions, and reasons about average, accumulated, and accumulated discounted utility over finite or infinite time lines. The Bellman-Ford equation provides a state oriented description, and dynamic programming techniques provide the means [6, 9] to find an optimal action for every system state, thus forming a policy of action, or a plan.

MDP models have been also extended to include partial observability of the system state, or sensory aliasing, into so-called Partially Observable MDPs (POMDPs) [49, 20]. Armed with this new modelling ability, the MDP approach found many applications, including in robotics [42, 20, 43, 86, 92, 36]. Unfortunately, the wide range of solution techniques [58, 49], together with the focus on the value of a system state, inherited and multiplied the computational complexity of finding a solution, reaching even the condition of undecidability [16, 53, 52].

MDP-type modelling, however, gives us significant power in capturing a very wide variety of domains, and its analytical power maintains its appeal. Some complexity studies have suggested (e.g., [74, 73]) that it is the optimality criteria, with its momentary state focus and averaging, that are among the key ingredients to the computational complexity of the MDP approach.

It is interesting to see how the modern (PO)MDP[1] technology begins to turn towards continuous spaces of system states and actions, as well as the continuous time-line. It is hoped that analytical tools will assist (PO)MDPs to become more applicable to complex problem domains. This is, however, somewhat peculiar, for such a move makes (PO)MDPs resemble more than ever the classical control

[1] Partially Observable Markov Decision Problems (POMDPs) are a variant on MDPs, discussed further later in the dissertation.

theory.

Classical control theory [95, 21] has strong mathematical connections with physics, and usually describes and operates with continuous systems. A system's state is usually a vector that contains a sufficient set of parameters to describe a momentary snapshot of the system, while the system itself is seen as a function that describes the transformation of this vector over time, either by determining the state derivative for continuous time systems, or directly determining the state at the next time step for discrete time systems. The system state transformation function, termed *system dynamics*, can be analysed to discover a system's inherent stability and dynamic properties. Control theory's mathematical apparatus allows it to treat uncertainty in the system state by substituting a distribution over the system state into the system dynamics function, instead of a single system state vector.

As a result it becomes possible to describe a fairly complex (physical) environment by a rigorous mathematical formulation. But to describe a control problem within that environment, the system description is extended by the notion of utility and its expected accumulative over time, making the similarity to the (PO)MDP approach evident and obvious. Even the notion of *cost to go* that characterises dynamic programming solutions is relevant both to (PO)MDPs and to classical control theory.

Although the analytical properties of the system description allow control theory to create strong tools to deal with system state uncertainty (e.g., filters such as the Kalman filter and unscented transforms [111, 101, 39]), creating an optimal control signal for the expected accumulated utility is computationally hard, and a variety of assumptions and simplifications (e.g., linearity of the system dynamics,

a quadratic cost function) are made to make the process feasible.

It appears that among all the theories of planning and control a key ingredient disappeared along the way from a narrative hunt story to the formal plan and control methodologies. It is the perspective of this thesis that the missing ingredient is the focus on system or environment development dynamics. Even in classical control theory, where the notion of *system dynamics* is utilised, it is the system state (directly, or by the formulation of a utility function) that concerns the main body of approaches to control, starting from PIDs and ending with model following. Yet, capturing and operating directly in terms of system dynamics is crucial. This can be made evident by a simple example from the human vision system.

Consider a peculiar pattern on a dalmatian. Black, misshaped splashes on an otherwise white coat are the trademark of this breed of dog. Such a dog would usually stand out in our normal domestic environment. But assume for a moment that the dog sits in front of a screen bearing just the same kind of black splash pattern.[2] Motionless, the dog blends into its background and virtually disappears. However, once it starts to move we immediately spot it again. The variation, the change, the motion of black splashes we see is the very thing that identifies to us the presence of an entity and, most importantly, identifies the entity as a dalmatian. This nifty trick can be performed by the human vision system because it concentrates not on a momentary snapshot of the environment, but on the rules that guide and describe the *development* of the environment.

Our knowledge of how human vision analyses dynamic environments allows us to create a great variety of visual effects applied and utilised by Computer

[2]This is, in fact, a simulation of how the dog would look through the colour-blind vision of other animals (see Figure 1.1).

Figure 1.1: Running Dalmatian (photograph by R.C. James)

Graphics and the movie industry. Following the human ability to recognise moving splashes of colour as a dog, it is possible to create an impression of a dog just by moving the spots, without the actual dog being present. Similarly, the motion of certain forms of image components can create an illusion of a 3D object appearing on a 2D screen, and this fact is extensively used by 3D animators.

This thesis adopts the intuition of the human visual system's ability to concentrate on system development dynamics, resulting in a novel and effective framework for continual planning and control: the *Dynamics Based Control* framework.

Chapter 2

Background and Related Work

"The only thing more reliable than magik is one's friends!"

Macbeth

"There's something to be said for relatives ...it has to be said because it's unprintable!"

A. Einstein

(from R. Asprin epigraphs)

The *Dynamics Based Control (DBC)* framework and algorithms, which are the contribution of this thesis, have been created to address several shortcomings of available control and planning technologies in stochastic dynamic systems. We therefore overview the technological background of this field in order to understand the context and contributions of the DBC framework.

DBC is indeed a control framework, and as such uses a similar vocabulary to that of classical control theory. Terms such as *system identification* and *system dynamics* are used in the control theory sense. However, the DBC algorithms developed operate in a specific environment type—a discrete (time, state and action space) Markovian environment, which makes it more convenient to borrow

some terms from another control and planning framework—Partially Observable Markov Decision Problems (POMDPs). Terms such as *transition function* and *system state distribution*, though they retain their control theory meaning, were utilised by the Extended Markov Tracking (EMT) based implementation of DBC in their POMDP sense.

Both the general DBC framework and its EMT-based implementation contain an estimation algorithm component. This component in some sense summarises the environment's response to the control sequence applied, and makes such approaches as *model predictive control* and *fictitious play* relevant background as well.

In the sections that follow, the aforementioned approaches to a controlled interaction with an environment are briefly summarised. The complete volume of the background is so extensive that an exhaustive summary is virtually impossible. Therefore, we present only those elements which are deemed to assist the reader in understanding this thesis' contribution, the Dynamics Based Control framework. During this exposition, we will occasionally intersperse comparisons of the approaches being presented with the DBC approach.

2.1 Control Theory

Classical control theory [95, 21] has its mathematical apparatus rooted in physics. The first mathematically described dynamic systems did not have an external control input, and developed over time at their own accord. Such *autonomic* systems described the behaviour of a system according to the laws of physics: e.g., a bouncing ball, motion of a pendulum. Quickly it became apparent that, as long as

we can imagine that a mathematical law stands behind the development of a system, similar mathematical equations can describe the behaviour of non-physics related systems as well, e.g., the hunter-prey balance.

It became possible to analyse a system's behaviour, faults, and benefits. For some of the systems their behaviour did not only depend on their principal composition, but also on the specific parameters of the principal components. The simplest example would be perhaps the behaviour of a pendulum whose oscillation period depends (only) on its length. As many children quickly discover on a swing, for a swing is a pendulum, this parameter dependency makes it possible to vary the behaviour of the system—that is, to control it.

Control theory defines a system's state as a vector $\mathbf{x}(t) \in \mathcal{R}^n$ where each coordinate describes a single parameter of the system. A state vector is designed to contain *sufficient* data to describe a momentary snapshot of the system at time t. The system is then described by an equation of the form $\dot{\mathbf{x}} = \mathbf{f}[\mathbf{x}, \mathbf{u}, t]$, where f is the *system dynamics* function and $\mathbf{u} \in \mathcal{R}^m$ is the set of controlled parameters of the system. Since $\mathbf{u}$ is a function of time, it is also referred to as a *control signal* or *control input*. Once the control signal is fixed, so is the behaviour of the system: the system takes the autonomic form $\dot{\mathbf{x}} = \hat{\mathbf{f}}[\mathbf{x}, t]$, with its (autonomic) system dynamics $\hat{\mathbf{f}}$. Mathematical analysis of the system dynamics function can determine to what degree the system is controllable, that is, what kind of behaviours it is possible to create.

A formal system description, however, does not pose a control problem by itself. One needs to determine what behaviours, and to what degree those behaviours, are considered beneficial. To this end classical control theory makes a crucial decision—it is the properties of the control signal and system state over

time that determine the benefit of any given behaviour. In other words, a *utility* (or *cost*) *function* is defined of the form:

$$J = \phi[\mathbf{x}(t_f), t_f] + \int_{t_0}^{t_f} \mathcal{L}[\mathbf{x}(t), \mathbf{u}(t), t]dt$$

The control problem is then to find $\mathbf{u}(t)\big|_{t=t_0}^{t_f}$ so as to minimise the cost J. The solution to this problem is found by an application of the calculus of variations, and the principle is readily demonstrated by the Brachistochrone problem.

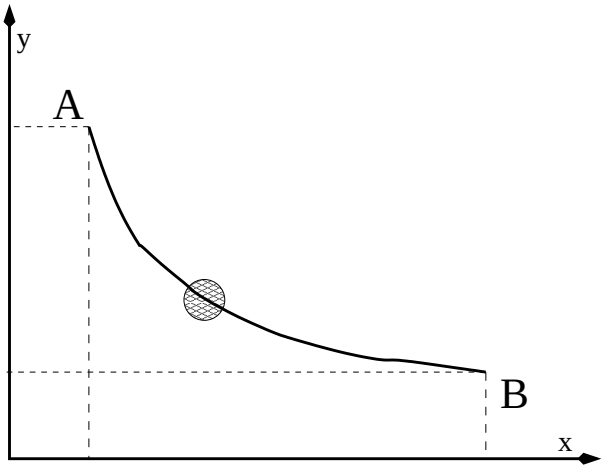


Figure 2.1: Brachistochrone problem: make the bead slide down in the least time possible.

The Brachistochrone problem can be formulated as follows: find the curve down which a bead sliding without friction, starting from zero speed, and effected by gravity alone, will slip in the least time possible (Figure 2.1). The problem can be viewed as a standard control problem if one takes the speed of the bead as constituting the system state, the curve function to be the control input, and the cost function computing the time needed for the bead to traverse the curve. Johann Bernoulli solved the problem, and then published it as a challenge to others

through Acta Eruditorum in June 1696. The challenge was accepted, and four other solutions were sent in by Isaac Newton, Jakob Bernoulli, Gottfried Leibniz, and Guillaume de l'Hôpital.

The elegance of the continuous time solution nowadays gives way to the discrete time system formulation, mostly due to digital implementations. It somewhat changes the mathematics, but the structure remains the same: a system is described by an equation of the form $\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n, n)$, where n is the time step of the system.

Both the continuous and the discrete time system formulations have been extended to include random perturbations. This is done by introducing a stochastic process $\mathbf{w}(n)$, that encodes perturbations, into the system dynamics function and the expected cost function (here given in a discrete time form):

$$J = E\left(\phi[\mathbf{x}(n_f), \mathbf{w}(n_f), n_f] + \sum_{n=n_0}^{n_f} \mathcal{L}[\mathbf{x}_n, \mathbf{u}_n, \mathbf{w}_n, n]\right)$$

It is assumed that the random perturbation $\mathbf{w}$ has some convenient properties:

- $\mathbf{w}$ is relatively small and additive: $\dot{\mathbf{x}} = \mathbf{f}[\mathbf{x}, \mathbf{u}, t] + \mathbf{L}(t)\mathbf{w}(t)$
- $\mathbf{w}$ is a white-noise process:
  - $E[\mathbf{w}(t)] = \tilde{\mathbf{w}} = 0$
  - $E[\mathbf{w}(t)\mathbf{w}(t)^T] = \mathbf{W}(t)\delta(t-\tau)$, where δ is that of Kroeneger.

Introducing stochasticity requires the notion of the system state to be augmented. It can no longer be a deterministic (function) vector; instead, it too becomes a random variable, and the system dynamics function transforms the state's distribution through time, governing the dynamics of the overall stochastic process.

But control theory does not stop at this, and continues to complicate the theoretical setup further by introducing the notion of observability of the system state. This is modelled by introducing an observation vector $\mathbf{y} \in \mathcal{R}^k$, observation noise process $\mathbf{v}$, and the observation function, so that an observation at time n is given by an equation of the form $\mathbf{y}_n = \mathbf{h}[\mathbf{x}_n, \mathbf{u}_n, \mathbf{v}_n, n]$. Given that only partial knowledge about the system state is available, computing a control signal optimal with respect to the defined cost function becomes even more complex, giving rise to the concept of *filters*—mathematical and algorithmic constructs that allow efficient recovery of the system state distribution based on the observation data and the system dynamics models $\mathbf{f}$ and $\mathbf{h}$ (see e.g., [111, 101, 39]).

Scarcity of proper information about the system state and the system's stochastic nature make control feedback a necessity; that is, the control signal is not precomputed in advance, but rather continually corrected and reshaped in a *closed loop* with its application, based on the most recent system measurements and estimations. In the scope of this thesis, the most interesting version of this closed loop is the *model-following* control approach. Under the model following an ideal system dynamics, $\mathbf{f}^*$ is assumed to exist, and the system is controlled as a function of the error between the state space trajectory produced by the ideal and the factual system responses.

The model-following approach is considered among the more robust and effective control methods, but interestingly it also underlines the main shortcoming of classical control theory. At the origin of control theory, a utility function was used to differentiate beneficial system behaviour, but in the case of model-following the roles have been reversed—despite the fact that the optimal behaviour is defined, it is converted into a space state trajectory to fit a utility-based

solution concept.

Before we continue, it is important to underline this observation. Recalling the control of a pendulum or a swing—varying the length of the pendulum changes the *dynamics* of the system, rather than the system's state. The system continues to develop over time, but the laws of this development change—this is what makes swings fun for kids; they recreate a certain *dynamic* behaviour, and consider reaching that *dynamics* to be the optimality criteria. The same point of view is adopted by the Dynamics Based Control framework, introduced in this thesis—it focuses on the system *dynamics*, and chooses a control signal guided by proximity to an ideal *dynamics*.

2.2 Partially Observable Markov Decision Problems (POMDPs)

Markov Decision Processes and their partially observable variation have been introduced to model a completely discrete controlled system with an intrinsic stochastic property and limited memory. The model essentially sees the environment as a family of stationary stochastic processes with transition probabilities parameterised by some action space, where the processes develop over a discrete time-line. Formally, a Markovian Decision Process (or a Markovian environment) is defined by a tuple $< S, A, T, s_0 >$, where:

- S represents the set of all possible system (environment) states.
- $s_0 \in \Pi(S)$ determines the distribution from which the initial system state (at time zero) is sampled.

- A represents the set of all possible actions applicable within the environment. At times a division of the set A is defined: A_s for all $s \in S$, since not all actions are applicable in all possible system states.

- $T : S \times A \rightarrow \Pi(S)$ is the transition function that describes the probability distribution over the system states at the next time step given the current time step's system state, and the action applied.

In turn, Markov Decision Problems (MDPs) extend this environmental description by a utility (reward or cost) function, $R : S \times A \times S \rightarrow \mathcal{R}$, and an optimality criteria[1]. The utility function is designed to describe preferences over different controlled transitions of the environment from one state to another. The optimality criteria states how this utility has to be treated over time. A solution to a Markov Decision Problem (MDP) is then a policy of action selection based on the environment's observations that produces the best utility with respect to the defined optimality criteria.

By far the most popular optimality criteria are the following two:

- *Expected discounted reward accumulation over the infinite time horizon*:

  Let r_i^π be a random variable denoting the utility obtained at time i under action selection policy π, and let $0 < \gamma < 1$. Then the optimal policy under the criteria is

  $$\pi^* = \arg\max_\pi \mathbf{E}\left(\sum_{i=0}^{\infty} \gamma^i r_i^\pi\right)$$

- *Expected un-discounted reward accumulation over a finite time horizon*:

  Let r_i^π be a random variable denoting the utility obtained at time i under

[1] Note how this echoes classical control theory.

action selection policy π, and let $T > 0$ be some finite time step at which the system is stopped. Then the optimal policy under the criteria is

$$\pi^* = \arg\max_{\pi} \mathbf{E}\left(\sum_{i=0}^{T} r_i^{\pi}\right)$$

It has been shown that under the discounted reward criteria the optimal policy has the form of a mapping from system states to actions $\pi^* : S \rightarrow A$, that is in every system state there is a single optimal action. Furthermore the policy can be recovered from the Bellman-Ford equation for the optimal value function.

$$V^*(s) = \max_a \sum_{s' \in S} \left(R(s,a,s') + \gamma V^*(s')\right) T(s'|s,a).$$

The value function $V^{\pi} : S \rightarrow \mathcal{R}$ expresses the expected reward that can be obtained by applying a policy starting from any given system state. The expected reward obtained by the policy is then the expectation of the value function with respect to the initial state distribution s_0: $\sum_s s_0(s) V^{\pi}(s)$, and the value function in turn is maximised at all points by the optimal policy.

The duality between the value function and the applied policy give rise to MDP solution algorithms, such as *policy* and *value iteration*. The value iteration algorithm is a *dynamic programming* [9, 6] solution of the problem that uses the Bellman-Ford equation to backup the value:

- Initialise $V^0(s)$ to small random values.
- Iterate $V^{k+1}(s) = \max_a \sum_{s' \in S} \left(R(s,a,s') + \gamma V^k(s')\right) T(s'|s,a)$ until convergence.
- Compute the policy by

$$\pi^*(s) = \arg\max_a \sum_{s' \in S} \left(R(s,a,s') + \gamma V^*(s')\right) T(s'|s,a)$$

Policy iteration attempt to correct the policy directly during its iterations:

- Initialise policy to some random action selection π^0.
- Iterate until convergence:
  - Compute the value function that corresponds to the policy V^k by solving the set of equations:

$$V^k(s) = \sum_{s'} \left(R(s, \pi^k(s), s') + \gamma V^k(s') \right)$$

  - Compute new policy

$$\pi^{k+1} = \arg\max_a \sum_{s' \in S} \left(R(s, a, s') + \gamma V^k(s') \right) T(s'|s, a)$$

This however assumes that the policy is computed under two major assumptions: first, the world's model is known, and second, the policy is computed off-line, away from its application. To amend these two limitations, "hands-on" learning methods were introduced, such as SARSA, TD-, and Q-learning [97, 103]. Learning algorithms move through the space of possible policies guided by repeated interactions with an environment, assuming only that the environment is indeed Markovian with known state and action spaces, but without any knowledge about the relationship of these two spaces. By their way of thinking, most of these algorithms are close to the policy iteration algorithm, as the on-line experience can be seen as an empirical value function computation, followed by a correction of the policy.

Reinforcement learning algorithms, other than expanding the range of domains indirectly captured by MDPs, also expose a crucial weakness of the MDP approach: *indirect* encoding of the behavioural preference by a reward function.

To force a reinforcement learning algorithm produce a system behaviour which is considered proper, the reward function used by the algorithm has to carefully designed (see e.g., [55, 65, 54]). Although Dynamics Based Control, presented in this thesis, does assume the existence of an approximate environment model, it works with system dynamics directly, circumventing the inconvenience of secondary preference encoding through a reward function.

Addressing the question of partial knowledge of the environment, MDP technology turned to domains where the system state is not completely known at all times, since it is obscured and indirectly measured. This led to formulation of Partially Observable MDPs (or POMDPs for short).

A POMDP environment is defined by an extended tuple $< S, A, T, s_0, O, \Omega >$, where S, A, T and s_0 are as defined for an MDP, and the other two parameters are as follows:

- O is the set of all possible observations (or state measurement values);
- $\Omega : S \times A \times S \rightarrow \Pi(O)$ is the stochastic observability function, with $\Omega(o|s, a, s')$ determining the probability of an observation given that the system state underwent a certain controlled transformation.

POMDPs have received even greater attention for their capability to model more realistic domains that include sensory aliasing, including robotic applications (e.g., [86, 43]). Although it is possible to convert a POMDP into the MDP form, resulting in a so called *belief MDP*, the transformation explodes the state space to unmanageable proportions. Coupled with the fact that the optimal policy is no longer a simple mapping from states to actions (as it provably was in the completely observable case), it led to a series of approximation attempts and

theoretical research into the computational complexity of the Partially Observable problem.

Ranging from heuristic to more rigorous analytical approaches, the vast multitude [99, 58, 40, 26, 69, 28, 94, 1] of approximate solutions to POMDPs allows one to handle larger and larger domains with greater efficiency. But the stern answer of the equal multitude [50, 57, 12, 51, 34, 18, 67, 73, 52, 53] of computational complexity analysis remains unforgiving: many types of POMDPs are unsolvable or computationally hard and inapproximable.

Among the reasons for this austere complexity one can name the *expectation* (or other averaging) of the reward found in the formulation of the optimality criteria for (PO)MDPs. The Dynamics Based Control (DBC) framework removes this limitation by dealing directly with system behaviour distributions, rather than a single parameter of arduous computational effort. A similar trend, though preceded by DBC and rather less general than DBC, can be also found within a more classical view of MDPs, as the following section demonstrates.

2.3 Targeted Trajectory Distribution Markov Decision Processes (TTD-MDPs)

In [82] an alternative view of the target of the control procedure within MDP environments was introduced. Motivated by the need to establish a non-static response from an interactive game [5, 104, 64], authors have looked to MDPs to capture the necessary uncertainty both in user (gamer) actions and the desired system response. The resulting model was termed the Targeted Trajectory Distribution

Markov Decision Processes (TTD-MDP) and encompasses the need for a controller to create a distribution over the system trajectories, rather than maximise the expected utility, to create an effect of a surprise so valued in the interactive entertainment.

Formally a TTD-MDP is defined over a given MDProcess $< S, A, T, s_0 >$ by a tuple $< \mathcal{T}, A, P, P(\mathcal{T}) >$, where:

- $\mathcal{T}$ is a set of trajectories of the underlying MDProcess,
- A is the set of actions inherited from the underlying MDProcess,
- $P : \mathcal{T} \times A \rightarrow \Pi(\mathcal{T})$ is the stochastic transition function between different trajectories, given that a certain action has been taken. $P(\mathcal{T}'|\mathcal{T}, a)$ determines the probability that the trajectory $\mathcal{T}'$ will be a successor of the trajectory $\mathcal{T}$ after the action $a \in A$ has been taken.
- $P(\mathcal{T})$ denotes the desired distribution over the space of trajectories.

It is readily observed that the state space forms a tree, where each node is a partial system trajectory, and edges are marked by a pair of an action and a one-state extension to the trajectory. This structure allows the efficient computation and representation of the TTD-MDP transition function, since only for one pair of trajectories $t, t' \in \mathcal{T}$ the value of $P(t'|a, t)$ is non-zero. In fact, given that $t \in \mathcal{T}$ ends with state $s \in S$ of the underlying MDP, only for $t' = tas'$ with $s' \in S$ the transition probability $P(t'|a, t) = T(s'|a, s)$ and can be non-zero. This means that the policy of the form $\pi : \mathcal{T} \rightarrow \Pi(A)$, where $\pi(a|t)$ defines the probability of taking action $a \in A$ given that trajectory $t \in \mathcal{T}$ has been traversed, completely determines the policy dependent distribution, P^π, over the space of trajectories $\mathcal{T}$.

In [10, 82] several algorithms were provided to solve the problem of finding a policy that will produce a distribution over the trajectories as close as possible to the desired distribution $P(\mathcal{T})$. Each algorithm measured the distance between the policy induced distribution $P^{\pi}(\mathcal{T})$ and the ideal distribution $P(\mathcal{T})$ differently, ranging from the l_1 vector norm to Kullback-Leibler distances.

Although TTD-MDP provides an answer to some of the immediate needs of gaming to create a controlled form of surprise [19], its composition has encountered another difficulty inherent in the standard treatment of (PO)MDPs—because the approach attempts to create a distribution over (partial) trajectories of the system, the policy spans the *complete* tree of all possible system developments, which is exponential in the size of the state-action space of the underlying MDP. As a result it is computationally hard to procure a good TTD-MDP policy. This could be amended if the trajectory tree could be encoded succinctly, or the policy recreated on-line in a continual planning manner [24]. Both of these amendments are covered by the Dynamics Based Control (DBC) approach presented in this thesis.

However, DBC is not a mere extension of the TTD-MDP idea (which it precedes). The TTD-MDP approach differs conceptually on several deeper aspects from DBC and from the Extended Markov Tracking (EMT) instantiation of the DBC framework. First of all, DBC assumes no specific type of modelling of the environment, e.g., it would equally encompass a system modelled by a Markovian process and by a Predictive State Representation (PSR) [93, 108, 106, 105, 107]. Furthermore, EMT, although it assumes a Markovian environment model, is formulated over a domain with partial observability, rather than a complete one, as is the case with TTD-MDP. Finally, and perhaps most importantly, the DBC framework takes a crucial step towards encoding the *development* of the distribution

over the system state, and operates in terms of the system dynamics. Thus DBC explicitly represents the *source* of a distribution over different system trajectories. As a result, the task representation becomes a very versatile one, capable of capturing not only static system structures, but dynamic ones as well.

2.4 Fictitious Play

The Fictitious Play game theory concept was introduced by Brown [17] to denote an interleaved process of learning and estimation in an adversarial game. The original idea behind it was that a player would imagine a game unrolling against a set of potential adversaries, and construct his strategy in accordance with this *fictitious* play. However, the implementation this idea received reduced the imagination to one step only: a player selects a best-response action to a set of adversaries whose strategies were previously estimated, the action is then applied, and the estimates of the adversary policies are updated based on sensory information (e.g., the actions taken by other players).

Although Fictitious Play has been interpreted within the framework of classical game theory, and projected into game theory ontology and structural elements, the concept itself is really much more general. Classically, this projection would mean that the interaction between the players would be described by a multi-dimensional utility function, $u : \prod_{i=1}^{n} A_i \rightarrow \mathcal{R}^n$, mapping actions independently selected by the players into a vector of rewards. That is, given that each player selected his action $a_i \in A_i$, then the utility would be

$$(u_1, ..., u_n) = \mathbf{u} = u(a_1, ..., a_n) \in \mathcal{R}^n,$$

and player i would receive utility u_i. This also assumes that there is no guarantee of this interaction to repeat. A player can of course toss a coin or two, that is, draw his actions from some distribution over his action space, $p_i \in \Pi(A_i)$; in that case, it is common to compute the *expected utility*[2]

$$(E(u_1), ..., E(u_n)) = E(u) = \sum_{\vec{a} \in \Pi A_i} u(\vec{a}) \Pi p_i(a_i).$$

In this setting, the best response means finding a distribution that would maximise the player's expected utility, and also reduces the Fictitious Play concept to the estimation of the distribution over the actions used by other players.

This game theoretic approach suffers from the same weakness as Markov Decision Problems—computing the average is hard, and instead of dealing with the action distributions of other agents directly, they are reduced to a single number, an agent's *expected utility*, to characterise the environmental setting it faces.

It is interesting to notice, however, that the Fictitious Play concept *itself* prescribes no specific way to estimate the strategy of the other players, but only assumes that a player has a mechanism to do so. Fictitious Play also states no specific representation of the other players strategy, or what that strategy might represent by itself. It even does not prescribe how to measure benefit of a response to obtain the best one.

This means that one could define 'best response' differently, and still remain within the framework of the Fictitious Play concept. For example, it is possible to define 'best response' with respect to the *distribution* of the player's utility, as opposed to the *expectation*, e.g., by preferring distributions which are proportional

[2]Notice, once again, how averaging plays the role of a fallback wherever distributions are considered—the same happened with Markov Decision Problems.

to e^{u_i}, where u_i is a utility value. If one relaxes the assumption of the non-repeated game, one can even estimate and use a stronger concept, namely the change of the utility distribution over time. That is, actions are chosen so that the distribution of utility will vary in time in a certain way—e.g., converging to be proportional to e^{u_i}.

A structurally similar process lies at the base of Dynamics Based Control: an estimator is determined to capture the system dynamics, and then a Fictitious Play type of computation is used to predict the response of the system to different actions, ultimately resulting in the action which is deemed to be best with respect to correcting the system dynamics estimate.

However, the Fictitious Play concept had left its Game Theory cradle long before Dynamics Based Control, and established itself in other fields. Since Fictitious Play is a concept of learning and adaptation, it naturally found its way into the field of multi-agent learning, discussed in the next section.

2.5 Multi-agent learning

Partial observability and decentralised, thus differing, perceptions of the environment make it hard for a team of agents to learn to behave beneficially in an unfamiliar environment. Even though all the team members wish to help each other, differences in their observations lead to dangerous mis-alignment, and a friend becomes a foe. In this context, fictitious play estimates the response of other team members and aligns the agent's actions with the rest of the team.

Although this can be made easier by information exchange (see e.g. [98]), in some cases the other team players are not even fully aware of the rest of the

team. For example, Sen et al. [89] show that mutually complementary strategies can be learned by two agents so as to perform a block pushing task over a given trajectory *without* sharing information, and, in fact, without mutual awareness. This is because the variation of the system development dynamics introduced by each agent could be overcome by the sensitivity (or actually the learning rate) of the other agent's learning procedure.

It is, however, the Game Theoretic framework, rather than the general Fictitious Play concept, which is of more frequent use in the multi-agent learning, largely due to Littman's contribution in [47]. In this paper, he takes the standard game theoretic point of view, but extends it by allowing the game to have an internal state which changes according to the action vector composed and applied by participating agent players. The resulting framework takes the form of an MDProblem with the utility and transition functions being parameterised by two action parameters, one for each agent within the modeled system. Furthermore the utility function is assumed to represent a *zero-sum game*—that is, one agent treats the function as representing reward, and wishes to maximise it, while the other sees it as cost, and seeks to minimise it. Under these assumptions, Littman formulates a combined version of the Q-learning algorithm [103, 102] and the MiniMax principle [66], resulting in a MiniMax-Q algorithm, in which both players dynamically change their response to one another, learning the best response, and in some cases converging to an equilibrium, at which point no agent has an incentive to change its policy any more (a so-called *Nash Equilibrium*). Later, Hu and Wellman [37, 38] extended Littman's approach to deal with general sum games, where the utility function is unconstrained in its interpretation by the players. The algorithm was also formally pro general sum games, and was named the

Nash Q-Learning algorithm.

It is important to notice that the concept of Nash equilibrium presumes so much mistrust and enmity between participating agents that each selects its actions independently from others. In the general case, where action selection is probabilistic, it means that distributions over agent's actions spaces are independent. This, however, need not be the case.

In his book "The Evolution of Cooperation" [3], Axelrod among other examples describes a peculiar behaviour in the trenches of the first World War. The opposing sides being roughly of equal power adopted a strangely synchronised behaviour. As if following a conductor's baton, solders on both sides barricaded themselves in bunkers, while the artillery shelled the other side relentlessly... and then again, following that invisible magical wand, everybody emerged from their cover to set up dinner plates and tea, completely secure that the other side does the same. Though surreal, this shows that even in most brutal of games, the action of players may be *correlated*. That is, action selection distributions of participating players are *dependent*, and the overall distribution over the joint action space my not be decomposable. Furthermore, it is possible that the joint distribution can be such that deviating from it will lead to utility reduction to the deviating player—if this situation occurs then the joint action selection distribution is a *correlated equilibrium* of the game.

Correlated equilibrium naturally extends and subsumes the notion of Nash equilibrium, and multi-agent learning algorithms have been developed that converge to such equilibria. Littman [48] has noticed that the assumption about the opponent in the game being friend or a foe is important both for convergence of the learning procedure and for the type of equilibrium to which the procedure

converges. Greedwald and Hall [35] then generalised the approach even further, introducing Correlated-Q-Learning (CE-Q) variations for general-sum games.

Correlated equilibrium is an important conceptual step, because action selection has a functional effect on the system response. Since the opponents' action selection is essentially parameterised by the player's action selection, selected action modulates the system response, which is a *dynamic* concept, though degenerate relative to the general *system dynamics*.

2.6 Multi-agent POMDPs

Fictitious Play can also be found in algorithms for distributed versions of POMD-Problems, e.g., [59, 61]. However, before we discuss what role Fictitious Play has there, we need to define what distributed POMDPs are. This section provides the necessary definitions and discussion.

Multi-agent POMDPs are conceptually a straightforward extension of the MDP idea to domains where the action space is factored by the multiple controllers that activate different portions of the action space. However, there are several parameters that can be extended in more than one way, which leads to a multitude of multi-agent POMDP models, rather than a single uniform extension. To quote the call for papers for the Multiagent Sequential Decision Making (MSDM) workshop [56]: "... i.e., MMDP, Dec-MDP, Dec-POMDP, Dec-MDP-Com, MTDP, COM-MTDP, R-MTDP, E-MTDP, I-POMDP, POSG, POIPSG, ND-POMDP, TI-Dec-MDP ...". Only two basic types will be presented here to demonstrate the major trends of the sub-field: ND-POMDPs and Dec-POMDPs.

The environment of general case Dec-POMDPs are defined by the tuple:

$< S, A = \times_{i=1}^{N} A_i, T, \{O_i\}_{i=1}^{N}, \{\Omega_i\}_{i=1}^{N} >$, where

- S is the set of all possible system states.
- $A = \times_{i=1}^{N} A_i$ is the joint space of actions. Each $(a_1, ..., a_N) = a \in A$ is composed of N elements, where $a_i \in A_i$ is set by agent i. Thus the system contains N agents.
- $T : S \times A \rightarrow \Pi(S)$ represents the stochastic transition function. Notice that the transition depends on a *joint* action $a \in A$.
- O_i represents the observation space of agent i,
- $\Omega_i : S \times A \times S \rightarrow \Pi(O_i)$ is the stochastic observability function that dictates the distribution of the i'th agent's observations based on the system transitioning from one state to another under the *joint* action.

It is important to notice and underline that agents depend on each other in two points in the environment. First, system transition depends on the joint action taken by the agents, which correlates their action selection processes. Second, any agent's observation depends on the *joint* action as well, which means that the agents' actions can produce mutual interference on their respective sensory activity.[3]

As is the case with MDProblems, the Dec-POMDProblem completes the environment description with a utility function and an optimality criteria with respect to that utility function:

[3]This is actually found in submersible vehicles, where sonar pitches, produced to help the vehicles orient themselves, interfere with one another.

- $R : S \times A \times S \rightarrow \mathcal{R}$ is the utility function (reward or cost) that depends on the system transition that occurred under the *joint* action taken by the agents.

- The utility is usually accumulated for a limited time T without discount, or for an infinite time with a discount factor $\gamma < 1$, and agents are set to maximise the expected value of this accumulation.

The policy, π_i, followed by agent i, is guided by the sequence of observations it received thus far, that is $\pi_i : O_i^* \rightarrow \Pi(A_i)$. The *joint policy* is then simply the multi-dimensional function $\pi = (\pi_1, ..., \pi_N) : (\times O_i)^* \rightarrow \Pi(\times A_i)$, with the control/planning task set to find the joint policy that satisfies the optimality criteria. Notice that this echoes, if not mirrors, the Game Theory view on mixed strategies under the Nash equilibrium principle: independently applied, uncorrelated (other than through the environment response), policies.

The result of this extension to MDPs is an extremely powerful modelling instrument. Unfortunately, the general case was proved to be NEXP-complete for an exact solution [8], and then proved to be inapproximable [74, 73]. The inapproximability result is based on an interactive proofs concept, where the system plays the role of the proof verifier and the agents try to cheat it. Since the agents can communicate only through the system, but not explicitly, and since the optimality criteria strongly depends on agent correlation with respect to the system state transitions, agent observations, and the utility structure, it is very easy for a verifier to catch agents cheating. Thus to resolve this complexity, either the optimality criteria or the correlation between agents needs to be modified. The Dec-POMDP literature focuses on modification of the inter-agent correlations, and spawns a va-

riety of limited models. These models have a significantly reduced computational complexity, but unfortunately lose as much modelling power (see e.g., [90, 33]).

One has to notice the following trends of Dec-POMDPs, and in fact of most of the POMDP-based multi-agent models:

- The policy of agent behaviour is precomputed off-line, and then is fixed during execution. As a result, if the environment model is imprecise with respect to the real world, the agents fail to cope with inconsistencies or to utilise opportunities.

- In spite of the optimality criteria containing an average, the policy is computed in the mind set of the worst-case scenario. That is, all possible deviations with respect to the average are counteracted, in great similarity to the way that MiniMax mixed strategies are computed. Nash, rather than the correlated, equilibrium intuition guides these policies.

- The optimality criteria is considered an invariable, and universally correct, measure with respect to the control task. As a result, the model components will be modified, and agent dependencies untied, in an attempt to overcome the computational complexity incurred, but not the optimality criteria or its design principles.

Although, several works attempt to defy the aforementioned trends, e.g., [61, 59, 84, 25, 32], they remain within the bounds of the original optimality criteria, and thus retain the major computational complexity trend.

MiniMax and the Nash equilibrium mindset is not the only heritage of Game Theory that crossed into use by the decentralised versions of POMDPs, and some

of them bear more positive charge. For instance, in ND-POMDPs [61], and in the underlying JESP algorithm [59], the principle of Fictitious Play combines with the distributed constraint satisfaction principles, and produces a distributed approach to the joint policy computation.

In JESP [59] agents start with random policies and then each agent in turn changes its policy to optimise utility, keeping other agent policies fixed. Notice that this is exactly in accordance with the alternating Fictitious Play principle.[4] For ND-POMDPs, it has been noticed that the correlation between agents induces a graph structure, thus making it sufficient to take into account only changes in the policy of the neighbouring agents with respect to that graph. This led to a distributed version of the JESP algorithm, reminiscent of simultaneous update Fictitious Play [61].

These works mark an important step towards an on-line, distributed solution to control in multiagent Markovian environments. However, all these modifications, even the most recent (e.g., [85])—the acknowledgement of the limited sphere of interest and effect, the need for distributed and on-line update of the policy—fail to cure the effect inflicted by the chosen (and preserved against all odds) optimality criteria inherited from single agent MDPs and Game Theory. Practical application of the distributed versions of POMDPs remains the matter of heuristic and approximate solutions, fine-tuned for a specific application or experiment. The theoretical inapproximability and complexity results [74, 73, 90, 33] loom over these algorithms, for none, with the notable exception of single agent work similar to [41, 40], provide a sound analysis of the approximation coefficient they

[4]The subtle difference between alternating and simultaneous Fictitious Play was noticed in, for example, [7]

give.

Dynamics Based Control (DBC), the subject of this thesis, proposes a new optimality criteria, justified both in terms of partial observability and knowledge, and in terms of limited reasoning resources of an agent. In its limited form, specialised for Markovian domains, this results in an on-line flexible and adaptive solution to the control problem both in single and multi-agent environments.

Chapter 3

The Dynamics Based Control (DBC) Framework

"Everything flows, nothing stands still."

"Nothing endures, but change."

Heraclitus, 535-475BC

"The universe is change; our life is what our thoughts make it."

Marcus Aurelius, 121-180AD

Before the details of Dynamics Based Control (DBC) are presented, we will look to a real-life mechanism for inspiration: human vision, and specifically motion-based separation. Motion-based separation allows us to discern a moving object otherwise perfectly camouflaged by its background. For instance, a quietly sitting dalmatian dog in front of a black spotted background would be hard to find, but once the dog moves, we immediately spot it. This happens because we **detect** that a certain subset of black spots have a different **dynamics** than the rest of them. These dynamics distinguish that part of the vision field which is occupied

by the dalmatian, and allow us to recognise it for what it is.

This dynamics recognition process can be, and in fact is, controlled. For example, in a Computer Graphics (CG) exercise, by moving a set of coloured dots on a two-dimensional screen, we can create a powerful impression of a three-dimensional object, e.g., a rotating sphere.

3.1 DBC Components

Dynamics Based Control follows the same patterns as the CG exercise above. Assume a controlled dynamic system $\mathbf{S}$ (the set of coloured dots), a tracking or a recognition algorithm $\mathbf{L}$ (human vision), and an ideal dynamics τ^* (the rotating sphere). The task of the controller would then be to feed into the system $\mathbf{S}$ a sequence of actions, so that based on the output from $\mathbf{S}$ the algorithm $\mathbf{L}$ will reproduce τ^* or a close alternative.

However, since we would like to mathematically formulate this control problem, we do not deal with these components directly, but rather model them, and base our decisions on these models:

- A model of the environment $\mathbf{S}$'
- The tracking/recognition algorithm $\mathbf{L}$'
- The target τ^*, and other possible outcomes of $\mathbf{L}$.

Since we allow the control solution to err, but would like to restrict that error, we also need to define a measure of proximity, $\mathbf{d}$, between two different outcomes of $\mathbf{L}$.

Given the mathematical formulation the control problem can be defined as following:

Definition 1 *Given $< S, L, \tau^*, d >$ as above, find a control method, based on the mathematical model of S, so that the algorithm L will recreate a dynamics model closest to τ^*.*

3.1.1 A Note on Versatility of System Dynamics

It is important to underline the strength and versatility of the dynamics-based task representation. As was noted in Section 2.1, the functional representation of system dynamics are capable of capturing a wide variety of physical systems, or, in fact, any system admitting an approximate mathematical description. Even without knowing whether any specific behaviour can be induced within a system, or what control signal would be required, a dynamics-based description allows an explicit description of the desired system behaviour.

Furthermore, some tasks are inherently dynamic. For instance patrolling a region (e.g., in a museum) requires complete coverage, but also requires stochasticity to reduce predictability by a potential intruder. Aerial vehicle behaviours, such as landing and acrobatic figures, can be comfortably described by an autonomic dynamic system, but would take a significant design effort to be described in terms of system state transition utilities.

The direct representation of the desired behaviour in the terms and mathematical vocabulary of the overall system description support faster feasibility and design cycles—a beneficial engineering outcome, utilised by the Dynamics Based Control architecture described in the next section.

3.2 DBC Architecture

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

- **Environment Design Level** is concerned with the formal specification and modelling 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 target 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 a museum guard scenario, these would correspond to a stochastic sweep schedule, and a measure of relative surprise between the specified and actual sweeping.

- **Agent Level** in turn combines the environment model from the Environment Design level, the dynamics estimation procedure, the deviation measure and the target dynamics specification from the User Level, to produce a sequence of actions that create system dynamics as close as possible to the targeted specification.

As we are interested in the continual development of a stochastic system, such as happens in classical control theory [95] and continual planning [24], as well as in our example of museum sweeps, the question becomes how the Agent Level is to treat the deviation measurements over time. To this end, we use a probability

threshold—that is, we would like the Agent Level to maximise the probability that the deviation measure will remain below a certain threshold.

Specific action selection then depends on system formalisation. One possibility would be to create a mixture of available system trends, much like that which happens in Behaviour-Based Robotic architectures [2]. The other alternative would be to rely on the estimation procedure provided by the User Level—to utilise the Environment Design Level model of the environment to choose actions, so as to manipulate the dynamics estimator into believing 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 simulation of the specified target dynamics (i.e., beyond discerning via the available sensory input).

DBC levels can also have a back-flow of information (see Figure 3.1). For instance, the Agent Level could provide data about target dynamics feasibility, allowing the User Level to modify the requirement, perhaps focusing on attainable features of system behaviour. 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.

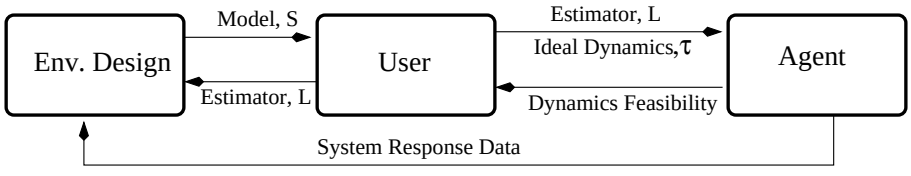


Figure 3.1: Data flow of the DBC framework

Extending upon the idea of Actor-Critic algorithms [44], DBC data flow can provide a good basis for the 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 behaviour. 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 [11, 63].

3.3 Control and Planning Perspectives

A control solution is devised and performed by the DBC Agent level, and it is that level upon which we will now concentrate. The task a DBC Agent faces can be viewed both as a closed loop control [95] problem, and as a continual planning [24] loop.

As Figure 3.2 shows, the closed loop control perspective is obvious: a DBC Agent will apply an action to which the environment will respond and provide an observation; the observation will be processed by the dynamics estimator, and result in an update of the system dynamics estimate; the dynamics estimate in turn will be compared to the reference of the ideal dynamics, and the control algorithm will produce another action.

The control perspective requires an additional assumption, absent from the general DBC framework. Dynamics estimates should not vary in a (strong) discontinuous fashion with respect to the action variation.

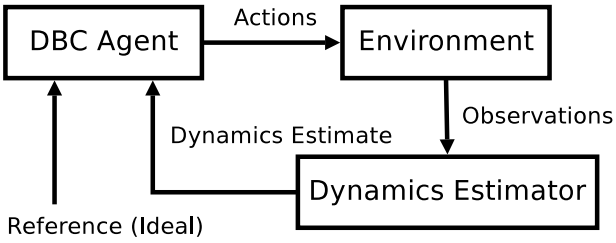


Figure 3.2: DBC Agent as a control loop

This assumption, however, naturally occurs in many physical motion models, and by itself would not constitute a major difference with DBC. The difference lies in the data that flows through the loop. In a classical control loop, the system state would be estimated and used as a basis for the control signal variation. In DBC, on the other hand, it is the estimate of **system dynamics** that is at the base of the action selection procedure.

Yet, with a wide array of control methods available, it is hard to avoid an attempt to resolve the DBC Agent problem using one of the classical methods. The most tempting among them is **model following**, since it also has a dynamic system as a reference.

3.3.1 The Model Following Perspective

Under the model following principle the system is controlled as a function of the error between the ideal and actual responses. Since the DBC User level provides the ideal dynamics τ^*, it may be possible to construct a model following controller with τ^* as the reference. It then seems feasible that the environment will follow τ^*, thus forcing the estimation algorithm $\mathbf{L}$ to reconstruct τ^*. That is, model following seems to be a trivial solution to DBC Agent control.

However, there are two problems with such an approach. First, it may be much easier to fool the estimation algorithm **L** than actually to control the environment. For example, trivial frequency analysis of the sequence $\{H, T, H, T, H, T, H, T\}$ would suggest that a fair coin was used to create it, in spite of the fact that the sequence is more likely to be of a deterministic origin.

Noise and stochasticity of the controlled environment provide the second reason for model following to fail as a DBC solution. The algorithm **L** can deviate away from its τ^* estimate, and then never return to it, even if the environment will strictly follow τ^* thereafter. Much stronger means may be necessary, as can be seen from the example of the noisy prisoner's dilemma [110]. A standard Tit-for-tat strategy is thrown off-sync by noise, and a modification by generosity or contrition is necessary.

The model following principle, however, does not have to concentrate on the environment as its control subject. Instead, it is possible to view the pair, formed by the environment and the dynamics estimator, as the control subject. In this case, one can argue that a DBC Agent would compute actions as a function of error between the response of the environment-estimator pair and the optimal dynamics τ^*. That is, one can see any DBC Agent as a form of model following. But in this case τ^* would need to be the response of an *ideal estimator*, which it is not—τ^* is a constant reference.

This analysis of the model following principle's failure with respect to the DBC Agent control problem exposes an interesting parallel to another control principle: Perceptual Control.

3.3.2 The Perceptual Control Perspective

Perceptual Control is a psychological theory of animal and human behaviour [71]. It debates the "mechanical" view that sees behaviour as a function of perceptions received by an organism. Instead, Perceptual Control states that an organism's behaviour is a means to control its perceptions.

Since the dynamics estimation algorithm **L** can be seen as part of an agent's perception system, it follows that control decisions made by a DBC Agent are directed at producing desired perceptions. In a sense it makes DBC a formal engineering counterpart of Perceptual Control theory.

Still, if we would like to deal with systems where variation is not continuous, or simply not metric, we may need to take the planning perspective.

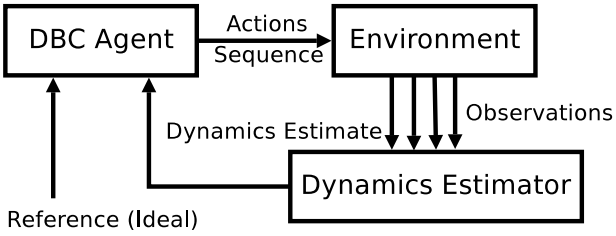


Figure 3.3: DBC Agent as a continual planning loop

3.3.3 The Planning Perspective

To see DBC as a form of planning, one simply has to modify the data flow diagram to that of Figure 3.3. Seen from the planning point of view, the dynamics estimator becomes akin to plan recognition, answering the question: "what plan is in effect to have caused the observed changes?". In this situation, the DBC agent may need to provide a sequence, rather than a single action, and the dynamics estimator may

receive more than one observation at any given time.

The DBC Agent receives the reaction of the dynamics estimator, which provides both plan failure and opportunity information. Furthermore, the DBC Agent has to utilise both failures and opportunities, otherwise it may fail to force the estimation algorithm $\mathbf{L}$ to recognise τ^*. As a result, the DBC Agent can be seen as a continual planner [24].

Chapter 4

DBC for Markovian Environments

"Should old acquaintance be forgot..."

Count of Monte Cristo

"...and then the fun began"

N. Bonaparte

(from R. Asprin epigraphs)

Without reducing from the general power of DBC, an application of the framework requires specification of a (type of a) mathematical model used to describe the environment. In this section, DBC will be applied to domains described as Partially Observable Markovian Environments. It is important to underline at this point the connection and the distinction between DBC and the Partially Observable Markov Decision Problems (POMDPs) discussed in Section 2.2. Both POMDPs and the Markovian projection of DBC rely on the same mathematical model of the environment: a Partially Observable Markovian Environment—and this is the connection between them. This is, however, the only common point. DBC has an entirely different control task specification and optimality criteria.

This, in fact, makes the difference so profound that, in spite of the environment description being the same, the two approaches cannot be formally compared.

Given the assumption that the environment is mathematically modelled by a Partially Observable Markovian Environment, DBC can be specified in a more rigorous manner. In this case, the phases or levels of DBC can be seen as follows:

- **Environment Design** level is to specify a tuple $< S, A, T, O, \Omega, s_0 >$, where:
  - 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:

    $$T : S \times A \rightarrow \Pi(S).$$

    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:

    $$\Omega : S \times A \times S \rightarrow \Pi(O).$$

    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.

- **User Level**, in the case of a Markovian environment, operates on the set of system dynamics described by a family of conditional probabilities $\mathcal{F} = \{\tau : S \times A \rightarrow \Pi(S)\}$. Thus ideal or beneficial dynamics can be described by $\tau^* \in \mathcal{F}$, and the recognition or tracking algorithm can be represented as a function $L : O \times (A \times O)^* \rightarrow \mathcal{F}$; 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 distance* or L_1 distance between two distributions p and q

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

  - *Fidelity* measure of distance

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

  - *Kullback-Leibler divergence*

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

  Notice that the latter two are not actually metrics over the space of possible distributions, but nevertheless have meaningful and important interpretations. For instance, Kullback-Leibler divergence is an important Information Theory tool [22] 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 the dynamics deviation probability θ.

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

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

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

4.1 The Extended Markov Tracking (EMT) Solution

Extended Markov Tracking (EMT) is a specific form of system dynamics estimation, and EMT-based control instantiates DBC in Markovian environments.

- **Environment Design** produces a Markovian partially observable tuple

$$< S, A, T, O, \Omega, s_0 >$$

- **User Level** of EMT-based control defines a limited-case target system dynamics independent of action:

$$\mathcal{F} = \{\tau : S \rightarrow \Pi(S)\}.$$

It then utilises the Kullback-Leibler divergence measure to compose a momentary system dynamics estimator—the Extended Markov Tracking (EMT) algorithm. The EMT 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 minimisation problem:

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

- **Agent Level** in EMT-based control is suboptimal 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} is the auxiliary Bayesian system state estimator, then the EMT-based control choice is described by

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

4.1.1 Intuition and Mathematics of EMT

Extended Markov Tracking perceives the world as a homogeneous Markov chain, and tries to recover the transition matrix that governs it. The algorithm bases its incremental update on examples of two consecutive distributions over the system state, p_{t-1}, p_t. EMT assumes that the second distribution is obtained from the first one by an application of the true transition matrix of the Markov chain it attempts to recover. However, since there is a continuous subspace of matrices that would provide the same transition effect for any two distribution vectors, a reference matrix is needed, and EMT uses its previous estimate, τ_{EMT}^{t-1} as such a reference. As a result, the mathematical problem EMT faces is the recovery of a joint distribution with given marginals, as is described in [45].

$$
\begin{aligned}
\tau_{EMT}^{t} &= H[p_t, p_{t-1}, \tau_{EMT}^{t-1}] \\
&= \arg\min_{\tau} D_{KL}(\tau \times p_{t-1} \| \tau_{EMT}^{t-1} \times p_{t-1}) \\
&s.t. \\
&\quad p_t(s') = \sum_{s} (\tau \times p_{t-1})(s', s) \\
&\quad p_{t-1}(s) = \sum_{s'} (\tau \times p_{t-1})(s', s)
\end{aligned}
$$

The mathematical program thus obtained is a convex optimisation problem over a convex domain, and is solvable in polynomial time. Furthermore, in [45], Kullback provides an iterative procedure, later termed *the iterative proportional fitting*, that provably [46, 45] converges to the solution.

The following is then the finite precision adaptation of the algorithm to the discrete Markov chain problem of EMT:

0. Initialisation:
   - Set precision ϵ (e.g., $= 5 * 10^{-5}$) and $t = 0$
   - Set base matrix from corrected target $Q_t(i,j) = \tau_{EMT}^{t-1} p_{t-1}(j)$
1. Compute $tmp_1(i) = \sum_j Q_t(i,j)$
2. Set $Q_{t+\frac{1}{2}}(i,j) = p_t(i)(tmp_1(i))^{-1} Q_t(i,j)$
3. Compute $tmp_2(j) = \sum_i Q_{t+\frac{1}{2}}(i,j)$
4. Set $Q_{t+1}(i,j) = p_{t-1}(j)(tmp_2(j))^{-1} Q_{t+\frac{1}{2}}(i,j)$
5. Set t=t+1
6. if $\|tmp_1 - p_{t-1}\| + \|tmp_2 - p_t\| >= \epsilon$
   - Goto 1
7. Set $\tau_{EMT}^t(i|j) = \frac{Q_t(i,j)}{p(j)}$

4.1.2 The EMT-based Agent Level Control Algorithm

Given that the mathematical optimisation problem of EMT can be solved by the iterative algorithm above or other convex optimisation methods, the overall DBC Agent Level algorithm is formalised as follows:

0. Initialise estimators:
   - the system state estimator $p_0(s) = s_0 \in \Pi(S)$,

– system dynamics estimator

$$\tau^0_{EMT}(\bar{s}|s) = prior(\bar{s}|s)$$

Set time to $t = 0$.

1. Select action a^* to apply using the following computation:
   – For each action $a \in A$ predict the future state distribution
   $$\bar{p}^a_{t+1} = T_a * p_t$$
   – For each action, compute
   $$D_a = H(\bar{p}^a_{t+1}, p_t, \tau^t_{EMT})$$
   – Select $a^* = \arg\min_a \langle D_{KL}(D_a \| r) \rangle_{p_t}$
2. Apply the selected action a^* and receive an observation $o \in O$.
3. Compute p_{t+1} due to the Bayesian update.
4. Compute $\tau^{t+1}_{EMT} = H(p_{t+1}, p_t, \tau^t_{EMT})$.
5. Set $t := t + 1$, goto 1.

To demonstrate how EMT-based control works, an aircraft carrier landing scenario has been formalised as a Markovian environment and solved using an EMT-based controller.

4.1.3 Validation Experiment: Aircraft Landing

Consider an airplane approaching the landing deck of an aircraft carrier (Figure 4.1). To land safely the airplane has to keep its angle of approach in a narrow

corridor of about half a degree. However, the landing deck pitches up and down, air streams are unstable, and the aircraft may be damaged; as a result, the approach angle constantly changes, and the airplane pilot has to adjust the approach constantly. To assist the pilot, a set of coloured lights is projected into the sky, colour coding different angular sectors of approach.

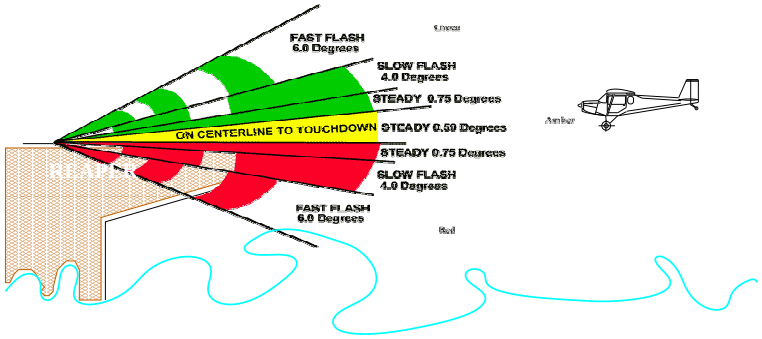


Figure 4.1: Landing Scenario for EMT control

This landing scenario can be easily modelled by a random walk over a linear graph (Figure 4.2), and defines the following Markovian model $\{S, A, T, O, \Omega, s_0\}$:

- S is a (discrete) set of valid approach angles.
- A is a (discrete) set of adjustment actions taken by the pilot. In this experiment, the set $A = [1 : 2 * F_{max} + 1]$ was created to reflect the effort exerted to increase or decrease the approach angle, where the F_{max} parameter allowed finer granularity of the action space, as will be evident in the transition function parametrisation.
- $T : S \times A \rightarrow \Pi(S)$ is the stochastic transition function, that models random landing deck pitch, and aircraft response. For any action, $T(\cdot|a, \cdot)$ represented a k-step simple random walk along the linear graph (see Figure 4.2).

Each random step's left, right, and stay probabilities were computed based on the action applied.

Denote by p^+ the probability to move right (increase the approach angle), by p^- the probability to move left (decrease the approach angle), and by p the probability of the approach angle remaining the same. Then given action $a \in [1 : 2 * F_{max} + 1]$, these probabilities were computed as follows:

- Let p^0 be the parameter that denotes the "natural" tendency of the aircraft to maintain the angle of approach.
- Let $F = \frac{a - F_{max} - 1}{F_{max}}$, and $\alpha = \frac{F+1}{2}$. Thus α denotes the normalised effect of $a \in A = [1 : 2 * F_{max} + 1]$.
- Then $p = p^0 * (1 - \frac{\|F\|}{2})$, $p^- = \alpha * (1 - p)$, $p^+ = (1 - \alpha) * (1 - p)$.

- O is the set of all observations; in this case, it can be the colour of the light beam through which the airplane is currently flying. In the experiment, the set of observations was taken to be equivalent to that of the state $O = S$.

- $\Omega : S \times A \times S \rightarrow \Pi(O)$ is the observability function, which takes into account, for example, light beam failures or colour variation due to weather conditions. Several observability versions were experimented with, with similar results:
  - The immediate neighbourhood of the true state were equiprobable.
  - The observations are distributed by a discrete Gaussian with the mean being the true state.
  - The discrete Gaussian was clamped to a limited neighbourhood of the state, and the distribution renormalised.

Ideally, given this model, the pilot would manage to counteract any deviation from the ideal approach angle in one step, though, since the pilot is only human, some tolerance has to be admitted. Thus the ideal system dynamics can be expressed by the following: $\tau^*(s'|s) = \begin{cases} \frac{1}{Z} & s' = ideal \\ \frac{\epsilon}{Z} & otherwise \end{cases}$, where Z is a normalisation factor and ϵ is the error tolerance.

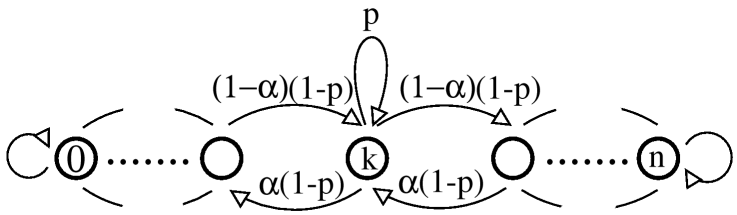


Figure 4.2: Random walk model for the landing scenario. α denotes transition probability change due to an action application.

If no forces were applied, the distribution over the system states (approach angles) would be almost uniform. However, under the application of the EMT-based controller a Gaussian-like distribution is obtained (Figure 4.3).

4.2 The Multi-Agent EMT Algorithm

Having observed that EMT-based control works in a single agent domain, and noticing that the algorithm takes only polynomial time with respect to the size of the Markovian environment model, it becomes increasingly intriguing to see whether it would be as effective in a multi-agent domain.

To test this, the environment model has to be modified to account for multiple agents acting simultaneously. This is done by replacing the single action space

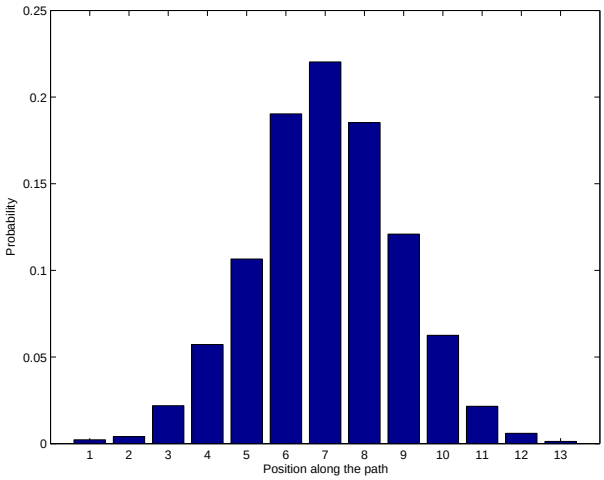


Figure 4.3: Distribution of the approach angle under EMT controller application.

with a Cartesian product, where each term corresponds to the action set of some agent. Observability functions are also augmented accordingly. Thus a tuple $< S, s_0, A, T, \{O_i\}_{i=1}^n, \{\Omega_i\}_{i=1}^n >$ describes a multiagent Markovian environment where:

- S — the set of system states, $s_0 \in S$ is the initial system state;
- $A = \times_{i=1}^n A_i$ — where A_i is the set of actions applicable by the agent i;
- $T : S \times A_1 \times \cdots A_n \rightarrow \Pi(S)$ — the system transition function;
- O_i — the set of possible observations for agent i;
- $\Omega_i : S \times A \times S \rightarrow \Pi(O_i)$ — the observation probability distribution for agent i.

Noticing that each agent can still represent its beliefs about the system state at time t by a probability distribution $\vec{p_t} \in \Pi(S)$, and system dynamics by a conditional distribution $\tau : S \times A \rightarrow \Pi(S)$, EMT can be applied straightforwardly,

but with one small correction. Every agent can compute the best joint action tuple $(a_1, ..., a_n)$, but this will be the best choice only from the agent's local point of view, and he will only be able to apply his action element of the joint action tuple. Thus the overall multiagent EMT algorithm performed by each agent $0 \leq i \leq n$ is as follows:

0. Initialise estimators:

   - the system state estimator $p_{0,i}(s) = s_0 \in \Pi(S)$,
   - system dynamics estimator

   $$\tau_i^0(\bar{s}|s) = prior(\bar{s}|s)$$

   Set time to $t = 0$.

1. Select action $a^* \in A$ to apply using the following computation:

   - For each action $a \in A$ predict the future state distribution

   $$\bar{p}_{t+1,i}^a = T_a * p_{t,i},$$

   where T_a is the transition function limited to action a;

   - For each action, compute

   $$D_a = H(\bar{p}_{t+1,i}^a, p_{t,i}, \tau_i^t)$$

   - Select $a^* = \arg\min_a \langle D_{KL}\left(D_a \| \tau^*\right)\rangle_{p_{t,i}}$

2. From the selected actions $a^* = (a_1, \ldots, a_N)$ apply action $a_i \in A_i$, and receive an observation $o_i \in O_i$.

3. Compute $p_{t+1,i}$ due to the Bayesian update.

4. Compute $\tau_i^{t+1} = H(p_{t+1,i}, p_{t,i}, \tau_i^t)$.

5. Set $t := t + 1$, goto 1.

4.2.1 Experiment: Springed Bar Balance

Consider a long bar resting with its ends on two equal springs, and two agents of equal mass standing on the bar. Their task is to shift themselves around so that the bar will be level, as shown in Figure 4.4. At each time step of the system, each agent has the choice of three actions: moving left one step, moving right one step, or staying put. Every movement of an agent has a non-zero probability of failing, and the probability is biased by the inclination of the bar. That is, an uphill motion will have less probability of succeeding than if the bar were levelled, and downhill motion will have more probability of succeeding than if the bar were levelled. Notice that bar inclination depends on the current agent positions on the bar, thus creating a correlation between the effects of the agent actions, and provides implicit information transfer between the agents.

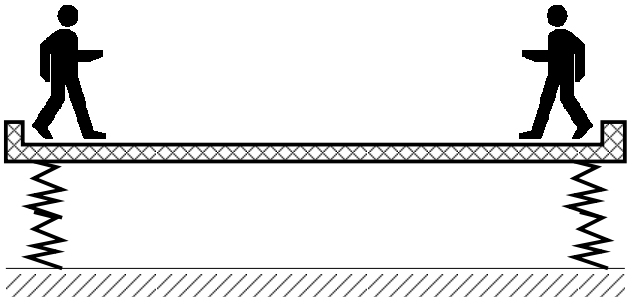


Figure 4.4: Springed bar setting

Formally the system state is described by the positions of the two agents on

the bar $S = [1 : d_{max}]^2$, where d_{max} is the length of the bar in "steps", and the initial state is an unbalanced one $s_0 = (1, \frac{d_{max}}{2} + 1)$. The action sets are $A_i = \{left, stay, right\}$, and the transition probability is built according to the physics of motion as follows:

- It is assumed that the agents are of the same mass, and the joint mass is m.
- The springs are assumed to have coefficients k_1 and k_2, which are the parameters of the model.
- Given that the bar has $d = d_{max} - 1$ units of length, the inclination of the bar is computed
$$\sin\theta = \frac{mg}{d^2}\left(\frac{l_2}{k_1} - \frac{l_1}{k_2}\right)$$
where l_1 and l_2 are the relative shifts of the centre of mass with respect to the first (leftmost) and second (rightmost) springs, and g is the standard gravity coefficient.
- Let $p \in [0, 1]$ be a parameter determining the general mobility of an agent. Then the probabilities of successful right, p^+, and left, p^-, steps are computed by:
$$p^+ = 0.5 * (1 - 2 * p) * sin_t^2 - 0.5 * sin_t + p$$
$$p^- = 0.5 * (1 - 2 * p) * sin_t^2 + 0.5 * sin_t + p$$

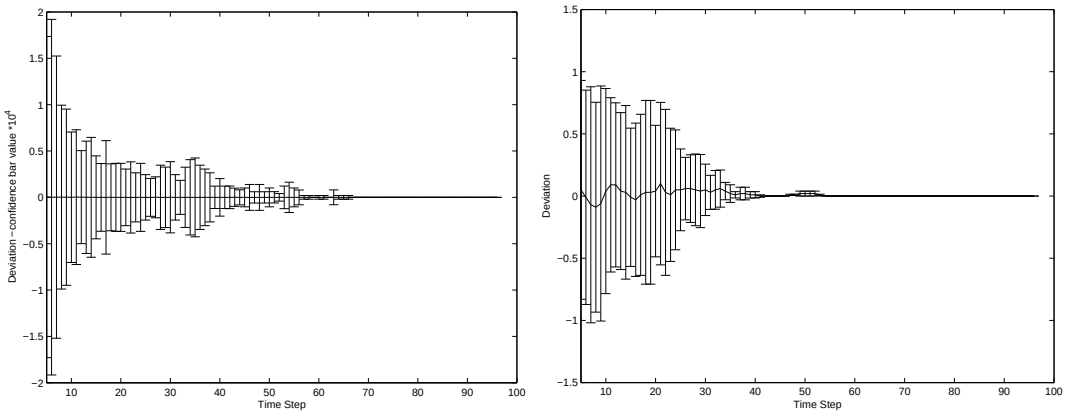
Two observation schemes are considered:

1. $O_i = S = \{$all positions of the two agents$\}$, $\Omega_1 = \Omega_2$ and creates uniform noise over the immediate neighbourhood of the real joint position of the agents.

2. $O_i = [1 : d_{max}]$ and represents the position of the observing agent.

   Ω_i creates a uniform noise over the immediate neighbourhood of the observing agent's real position.

In the first observations scenario, agents converge to a symmetric position around the ideal centre of mass (given that the springs and masses are equal, this is the centre of the bar). An example run can be seen at Figure 4.6. Average deviation with confidence bars is shown at Figure 4.5 (left).



(a) Observational Scenario I (b) Observational Scenario II

Figure 4.5: Multiagent Scenario: Deviation from the ideal centre of mass.

In the second observation scenario, where agents have only noisy observations of their own position, an interesting form of behaviour is engendered. Agents cannot step off the bar; any action that attempts to do so fails. Together with the symmetric nature of the problem, this creates a Schelling focal point [88], where each agent occupies the far end of the bar, thus balancing it. Agents' positions in the second observational scenario quickly converge to this focal point.

What's even more interesting is the way they do so. The initial state of the system places one of the agents at the far end of the bar, while the other stands

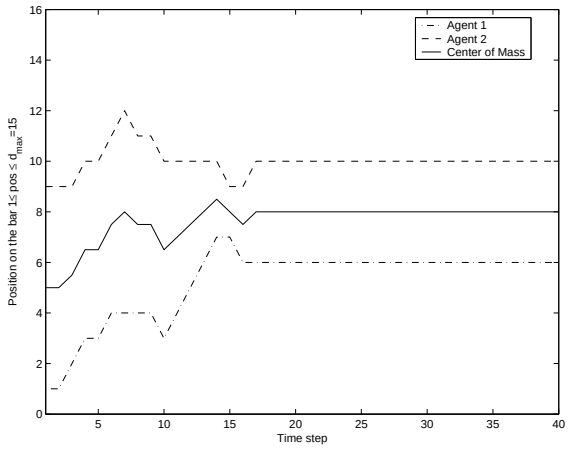


Figure 4.6: Multiagent observational scenario I.

quite close to the middle. The intuitive way to move towards the focal point position (two far ends of the bar) would be for the second agent to move away from the centre, while the first agents stays put, especially since agents do not see each other. Recall though that the bar would then be tilted, slowing the second agent down. EMT Control compensates for that, and in many experiments moves the first agent towards the middle of the bar, thus helping the second agent to reach its destination; it then "recalls" the first agent to the original far end position. This behaviour can be seen in the example run in Figure 4.7 (left).

However, because of observational noise, the first agent sometimes overshoots, moving too far. EMT Control of the second agent detects that and moves the second agent towards the centre of the bar, allowing the first agent to correct its mistake. At its extreme, this behaviour can cause "switching", where agents switch their relative position, passing one another at the centre as shown in Figure 4.7 (right). However, EMT Control agents **always** manage to balance the bar, as shown by the statistical data in Figure 4.5 (right).

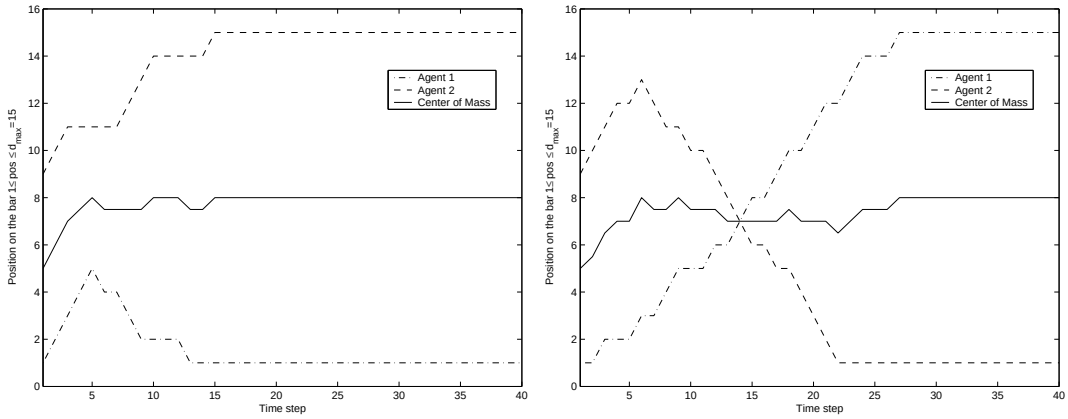


Figure 4.7: 'Helping' (left) and 'Switching' (right) behaviours

4.3 Multi-Target EMT Algorithm

At times, there may be several behavioural preferences. For example, in the case of multi-robot movement in formation, two preferences on motion direction exist—one dictated by formation keeping, the other by obstacle collision avoidance. Successful formation navigation requires a robot to adhere to, and balance, both of these behaviours. For EMT-based control, this would mean facing several tactical targets $\{\tau_k^*\}_{k=1}^K$, and the question becomes how to merge and balance them. A balancing mechanism can be applied to resolve this issue.

Note that EMT-based control, while selecting an action, creates a preference vector over the set of actions based on their predicted performance with respect to a given target. If these preference vectors are normalised, they can be combined into a single unified preference. We thus replace the stage 1 of EMT-based control (the action selection stage) by the following:

1. Given a set of tactical targets $\{\tau_k^*\}_{k=1}^K$, and their corresponding weights $w(k)$, select action a^* based on the following computations:

- For each action $a \in A$ predict the future state distribution

$$\bar{p}^a_{t+1} = T_a * p_t;$$

- For each action, compute

$$D_a = H(\bar{p}^a_{t+1}, p_t, \tau^t)$$

- For each $a \in A$ and τ^*_k tactical target, denote

$$V(a,k) = \langle D_{KL}\left(D_a \| \tau^*_k\right) \rangle_{p_t}.$$

  Let $V_k(a) = \frac{1}{Z_k} V(a,k)$, where $Z_k = \sum_{a \in A} V(a,k)$ is a normalisation factor.

- Select $a^* = \arg\min_a \sum_{k=1}^{k} w(k) V_k(a)$.

The weight vector $\vec{w} = (w_1, ..., w_K)$ allows the additional "tuning of importance" among tactical targets without the need to redesign the targets themselves. This balancing method is also seamlessly integrated into the EMT-based control flow of operation, and is compatible with its multi-agent extension. This compatibility makes possible the following experiment.

4.3.1 Experiment: Multi-Target Bar Problem

To test the multi-target version of the EMT-based control algorithm, recall again the Spring Bar multi-agent environment described in Section 4.2.1 limited to only one observation scenario: both agents receive independent noisy observations about their joint position. That is, $O_i = S = \{$all positions of the two agents$\}$, $\Omega_1 = \Omega_2$ and creates uniform noise over the immediate neighbourhood of the real joint position of agents.

Two conflicting targets are set. One is to balance the springed bar, while the other is to maintain a preset distance between themselves. Note that these targets are not requirements of the system state, but of the laws governing its behaviour. For instance, distance maintenance is expressed by a dynamics matrix, that shifts any given state into one that possesses the correct distance property, and EMT-based control sets out to achieve this kind of constraint within the system.

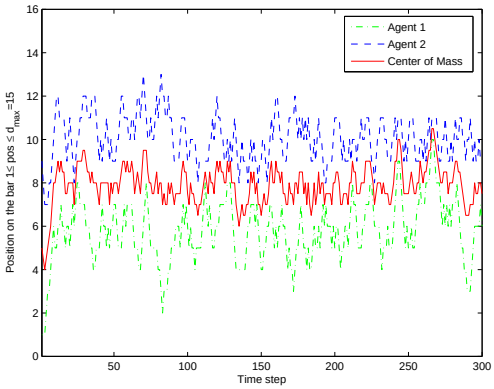


Figure 4.8: Example Run of Dual-Target Springed-Bar Problem

Although the two behavioural targets are compatible, that is, there exists a position of agents on the bar that satisfies both, the targets are indeed conflicting and interfering. For example, assume that we want agents to be at a distance of 4 from each other. Denote by 0 the coordinate line at the centre of the bar, and assume that the system's noisy response forced the agents into positions -2 and $+3$. In this case, the balancing target can encourage the motion of the left agent from -2 to -3 thus balancing the bar, but violating the distance constraint. On the other hand, the distancing target could be satisfied by the same agent moving right to -1, violating bar balancing even further.

Despite the constant conflict between the two targets, multiagent EMT-based

control equipped with multi-target action selection managed to maintain both targets quite closely. Although relentless system noise caused fluctuations, as seen from an example run at Figure 4.8, these fluctuations occurred around the only common position that satisfies the demands of both targets. In fact, the mean values of distance between agents and the position of the centre of mass almost perfectly matches the ideal, as can be seen from the value distributions in Figure 4.9 (left graph), which in this experiment would be 4 and 8 respectively.

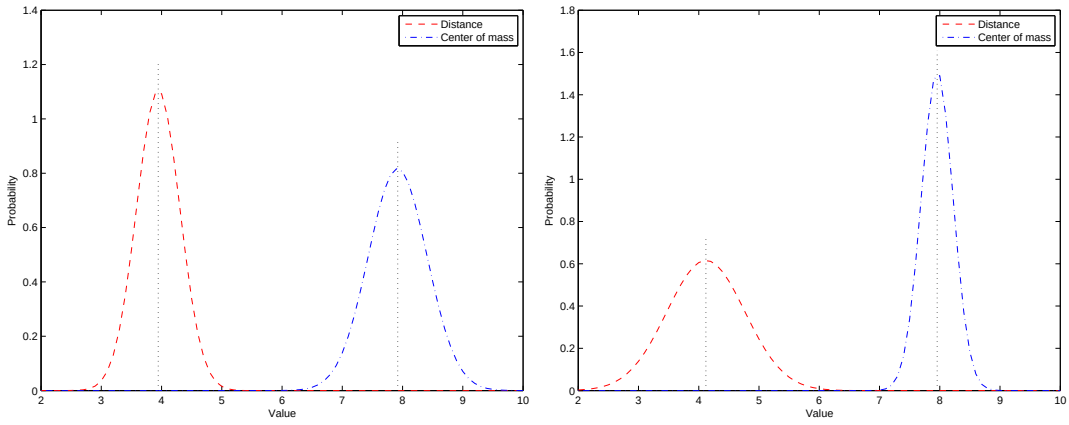


Figure 4.9: (Fitted normal) Distribution of distance and centre of mass in dual-target springed-bar problem with (0.2, 0.8) (left) and (0.4, 0.6) (right) balancing

Using the bar setting, multiple balancing vectors have been tested. For example, the distribution in Figure 4.9 was obtained from the weight vector $\vec{w} = (0.2, 0.8)$, that is, setting the balancing target at 0.2 and the distancing target at 0.8. Changing the weight vector to be $\vec{w} = (0.4, 0.6)$ exposes an interesting property of multi-target EMT-based control. Since the targets were weakly compatible, the algorithm maintained both targets with the new balancing vector, as it did with the old one. The difference occurred when the control algorithm had to correct system behaviour in response to noise—the algorithm was more ready to

briefly deviate from the distancing target than the balancing one, as shown by the distributions in Figure 4.9 (right graph).

However, as the weight of the the balancing target increased, not only the variance of the distance between the agents changed, but also the mean. EMT-based control began to lean strongly towards the balancing target, almost abandoning the distancing target. This can be seen clearly from Figure 4.10, which depicts changes of the distance and centre of mass distributions mean, as a function of weight of the target.

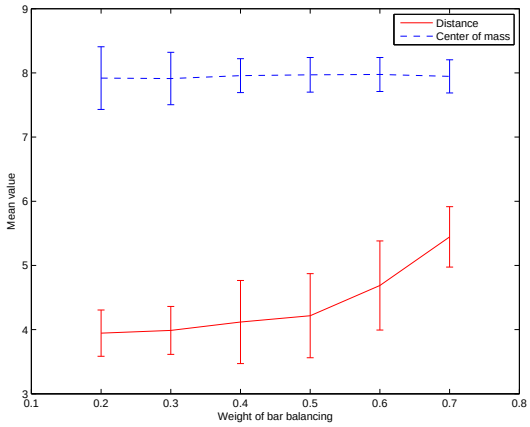


Figure 4.10: Means of distributions with respect to the weight of the balancing target. Error bars depict variance.

Note that the balancing target had a very strong presence at the weight 0.4, as expressed by the mean and variance of the centre of mass distribution. On the other hand, the distancing target at the weight of 0.4 had much less attention from EMT-based control. This inequality in attention to targets with respect to symmetric weight vectors has been contributed to the varying *strength* of preference expressed by the targets. Due to their construction, the balancing target included preferential differences of dozens of orders of magnitude over different

transitions, while the distancing target featured preference differences of only 2–4 orders of magnitude.

As a result, any change in system dynamics had much higher impact with respect to the balancing behavioural preference, which thus had a stronger effect on proper system behaviour. The distance tactical target was very mild in comparison, and thus had less influence on action choices.

4.3.2 Experiment: EMT Playing Tag

Multiple targets can also play the role of "basic behaviours", such as those found in the Behaviour Based Robotics paradigm [2]. In this case, the single-agent EMT-based control algorithm plays the role of a behaviour selector and mixer, as is demonstrated by the following experiment using the Game of Tag.

The Game of Tag was first introduced in [70]. It is a single agent problem of capturing a quarry, and belongs to the class of area sweeping problems. An example domain is shown in Figure 4.11.

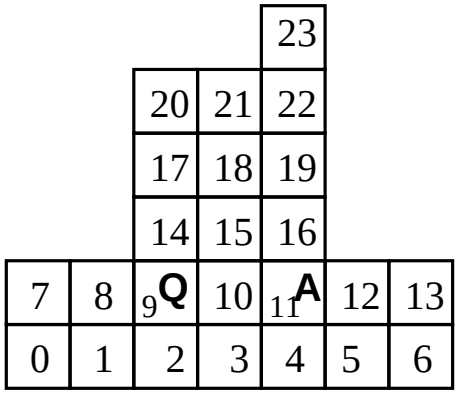


Figure 4.11: Tag domain; an agent (A) attempts to seek and capture a quarry (Q)

The Game of Tag severely limits the agent's perception, so that the agent is

able to detect the quarry only if they are co-located in the same cell of the grid world, in which case the game ends. Both the agent and its quarry have the same motion capability, which allows them to move in four directions, North, South, East, and West. These form a formal space of actions within a Markovian environment.

The state space of the formal Markovian environment is described by the cross-product of the agent and quarry's positions. For Figure 4.11, it would be $S = \{s_0, ..., s_{23}\} \times \{s_0, ..., s_{23}\}$.

The effects of an action taken by the agent are deterministic, but the environment in general has a stochastic response due to the motion of the quarry. With probability q_0[1] it stays put, and with probability $1 - q_0$ it moves to an adjacent cell further away from the agent. So for the instance shown in Figure 4.11 and $q_0 = 0.1$:

$$P(Q = s_9 | Q = s_9, A = s_{11}) = 0.1$$

$$P(Q = s_2 | Q = s_9, A = s_{11}) = 0.3$$

$$P(Q = s_8 | Q = s_9, A = s_{11}) = 0.3$$

$$P(Q = s_{14} | Q = s_9, A = s_{11}) = 0.3$$

Although the evasive behaviour of the quarry is known to the agent, the quarry's position is not. The only sensory information available to the agent is its own location.

For the Game of Tag, one can easily formulate three major trends: catching the quarry, staying mobile, and stalking the quarry. This results in the following

[1]The experimental data was obtained with $q_0 = 0.2$.

three target dynamics:

$$T_{catch}(A_{t+1} = s_i | Q_t = s_j, A_t = s_a) \propto \begin{cases} 1 & s_i = s_j \\ 0 & otherwise \end{cases}$$

$$T_{mobile}(A_{t+1} = s_i | Q_t = s_o, A_t = s_j) \propto \begin{cases} 0 & s_i = s_j \\ 1 & otherwise \end{cases}$$

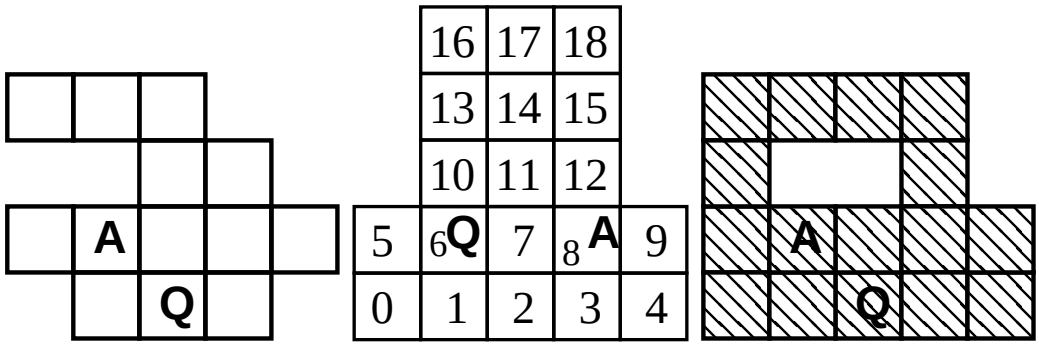
$$T_{stalk}(A_{t+1} = s_i | Q_t = s_o, A_t = s_j) \propto \frac{1}{dist(s_i, s_o)}$$

Note that none of the above targets are directly achievable; for instance, if $Q_t = s_9$ and $A_t = s_{11}$, there is no action that can move the agent to $A_{t+1} = s_9$ as required by the T_{catch} target dynamics.

Three configurations of the domain shown in Figure 4.12 were used to test EMT performance in the Tag Game, each posing a different challenge to the agent due to partial observability. In each setting, a set of 1000 runs was performed with a time limit of 100 steps. In every run, the initial position of both the agent and its quarry was selected at random; this means that as far as the agent was concerned, the quarry's initial position was uniformly distributed over the entire domain cell space.

Two variations of the environment observability function were used. In the first version, an observability function mapped all joint positions of hunter and quarry into the position of the hunter as an observation. In the second, only those joint positions in which hunter and quarry occupied different locations were mapped into the hunter's location. The second version in fact utilised and expressed the fact that once hunter and quarry occupy the same cell, the game ends.

The results of these experiments are shown in Table 4.1. Balancing the catch,



(a) Multiple Dead-end (b) Irregular Open Arena (c) Circular Corridor

Figure 4.12: These configurations of the Tag Game space were used

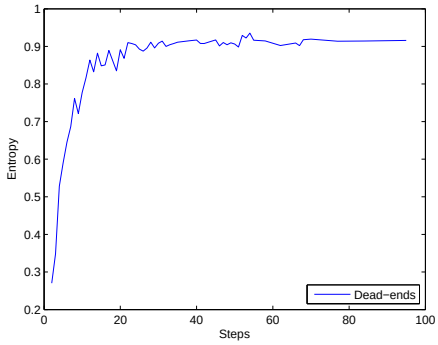
move, and stalk target dynamics described earlier by the weight vector $[0.8, 0.1, 0.1]$, EMT produced stable performance in all three domains.

Table 4.1: Performance of the EMT-based solution in three Tag Game domains and two observability models.

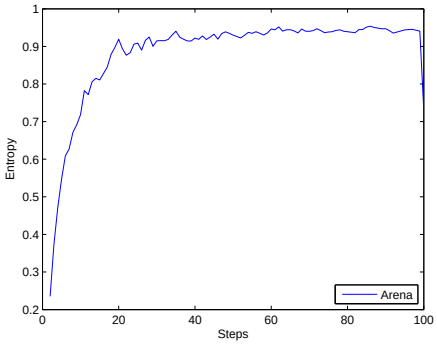
Model	Domain	Capture%	$E(\text{Steps})$
I	Dead-ends	100	14.8
omniposition	Arena	80.2	42.4
quarry	Circle	91.4	34.6
II	Dead-ends	100	13.2
quarry is not	Arena	96.8	28.67
at hunter's position	Circle	94.4	31.63

The behaviour cell frequency entropy, empirically measured from trial data, was also recorded. As Figure 4.13 and Figure 4.14 show, empirical entropy grows with the length of interaction. For runs where the quarry was not captured immedi-

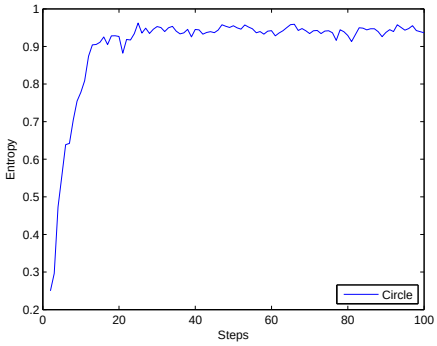
ately, the entropy reaches between 0.85 and 0.95[2] for different runs and scenarios. As the agent actively seeks the quarry, the entropy never reaches its maximum.



(a) Multiple Dead-end



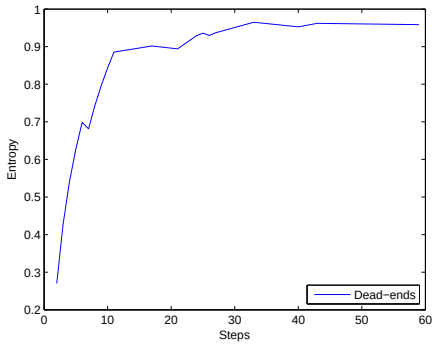
(b) Irregular Open Arena



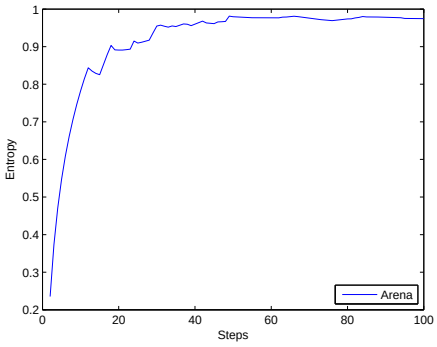
(c) Circular Corridor

Figure 4.13: Observation Model I: Omniposition quarry. Entropy development with length of Tag Game.

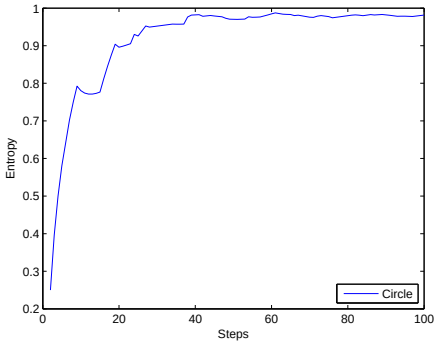
[2]Entropy was calculated using $\log$ base equal to the number of possible locations within the domain; this properly scales entropy expression into the range $[0, 1]$ for all domains.



(a) Multiple Dead-end



(b) Irregular Open Arena



(c) Circular Corridor

Figure 4.14: Observation Model II: quarry not observed at hunter's position. Entropy development with length of Tag Game.

Chapter 5

Empirical Stability of EMT-based Control

"Only constant and conscientious practise in the Martial Arts will ensure a long and happy life"

B. Lee

(from R. Asprin epigraphs)

5.1 EMT Resistance to Model Incoherence

It must be noted that in all previously described experiments, the controlled system was simulated exactly as the mathematical model of the environment prescribed. However, to apply a control method in the real world one has to account for the possibility that the environment model will not be precisely correct. In this chapter, rather than once more modifying the algorithm, the performance of the basic EMT-based controller, and the data it provides, are scrutinised. To test how EMT

would cope with an incoherent environment model, a simple robot-following scenario was simulated within the Player/Stage simulation environment [31].

Robot Green is given the task of following Robot Red at a preset distance. To achieve this task, two independent EMT controllers, $EMTC_1$ and $EMTC_2$, were applied to linear and rotation speed modulation of a single (simulated) Pioneer-2X robot (Robot Green). The sensory information was received through a *blob finder*—an on-robot camera with basic image analysis that makes possible the detection of colour blobs within the picture. Camera information was approximately mapped onto the observation sets: colour blob relative area and centring within the picture. Thus, the observation distributions provided state meanings of linear distance for $EMTC_1$, and angular distance for $EMTC_2$.

Both controllers used the environment model originally composed for the aircraft landing scenario, since both keeping visual angle and linear distance adhere to the same balancing logic. The aircraft-landing model of cause was not entirely coherent with the real-world behaviour of the visual angle and distance to Robot Red.

Incoherence with the real-world transitions, and also interdependence between the visual angle and linear distance, influenced the EMT Controller's performance; however, it was still able to successfully perform the tracking task. A sample run of the EMT-controlled robot can be seen in Figure 5.1, depicting three positions of the robots at different times. In this run, Robot Red performed a constant loop, and the EMT-controlled Robot Green that followed it managed to capture this motion. Robot Greed traced a smaller loop, concentric with the one traced by Robot Red. However, due to action model incoherence, Robot Green did not perform optimally and reacted to the change in Robot Red's position with insufficient

correcting actions.

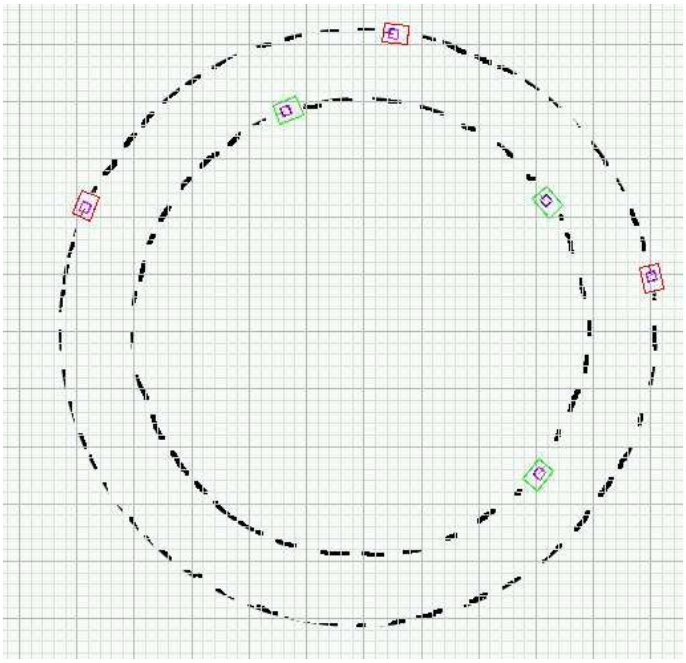


Figure 5.1: Target Following with a Weakly Coherent Model

5.2 EMT-Based Action Model Calibration

Although EMT robot control suffers from an inaccurate action model, EMT can also provide a remedy to the model—calibration. The simplest way to calibrate a model would be to accumulate data on state transitions and build an action model from statistics. Though exact system state knowledge is unavailable, EMT can still estimate the transitions based on the partial information. Then, EMT-based action model calibration becomes a matter of data accumulation and statistics:

0. Assume a Markovian environment model $< S, A, T, O, \Omega, s_0 >$ to be calibrated. For each action $a \in A$ let:

- $\bar{t}_a$ be the accumulator of the EMT dynamics estimators, initialised $\bar{t}_a = T_a$.
- N_a the counter, initialised $N_a = 1$.

Set time $t = 0$

1. Select and perform an action $a \in A$.
2. Assume that system state beliefs changed from $p \in \Pi(S)$ to $\bar{p} \in \Pi(S)$.
3. Using the EMT procedure, obtain an explanation $D = H(\bar{p}, p, Prior)$.
4. Let $\bar{t}_a := \bar{t}_a + D$, $N_a := N_a + 1$ and $t := t + 1$
5. If $t \geq t_{calibration}$
   - For all $a \in A$ let $T_a = \frac{1}{N_a}\bar{t}_a$

   else goto 1

5.2.1 Calibrated Target Following

The $EMTC_1$ controller, responsible for linear speed modulation in the robot-follows-robot experiment, was calibrated for a *stationary* Robot Red, and the resulting environment model was then used in tracking a moving target. During the calibration, the robot alternately walked to and from the target, switching direction if the target became point-like (went too far away), or if the target effectively blocked the camera view (went too close). This procedure is essentially as in [96], with variation of the stopping criteria.

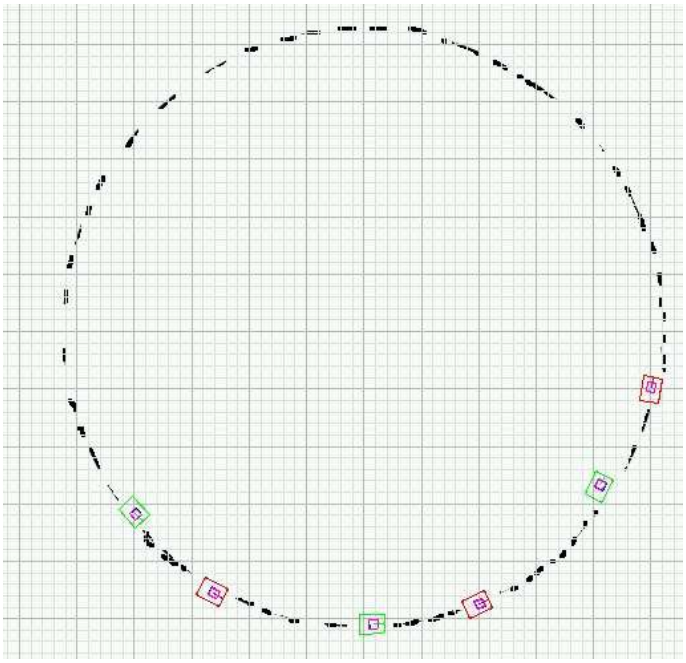


Figure 5.2: Target Following with a Calibrated Model

Even without the calibration of the angular controller, the ability of the system to follow a moving object was greatly improved. This can be seen in the example (Figure 5.2) of Robot Red moving circularly. Robot Green has almost completely matched the speed and trajectory of Robot Red. In fact, Robot Green cannot simply choose to move at the same speed as Robot Red; that speed is not available in the action set. Instead, Robot Green exhibits more sophisticated behaviour, it alternates appropriately between two speeds that bracket Robot Red's speed, matching the latter on average. The distance between the robots was still greater than required, but it is hypothesised that this is explained by residual incoherence of the internal model (since calibration occurred for a stationary object and only one controller model).

Chapter 6

A Short Remark on the Technical Limitations of EMT-based Control

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 optimisation by greedy action selection. This kind of combination, however, is common for on-line algorithms. Although further development of EMT-based controllers is necessary, evidence so far suggests that even the simplest form of the algorithm possesses a great deal of power, and displays 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.

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 modelling 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.

Chapter 7

Summary and Future Work

"All's well that ends well"

E. A. Poe

(from R. Asprin epigraphs)

7.1 Discussion and Summary

The Dynamics Based Control (DBC) framework, introduced by this thesis in Section 3, is directed at bringing together the concepts of perceptual control and dynamic systems. Perceptual control brings into the framework the idea that the task of the controller is concentrated on the sensory system, rather than on the true environment situation. DBC thus states that the control task is to enforce a certain output of the sensory system, which can be affected only indirectly through an uncertain and stochastic environment. The second crucial contribution of DBC is restating the sensory system of an agent as a system (dynamics) identification tool. That is, a sensory system does not concentrate on recovering an environ-

ment's momentary state, but rather on identification of the environment by the dynamics that govern the change within the state.

Concentration on the sensory system as the subject for control is essentially a recognition by the agent of its sensory limitations, which it does not try to overcome, but rather embraces those limitations to simplify the task at hand. As the sensory system limits the agent's ability to decipher the world, it makes little sense to attempt to invest more effort into controlling the agent's surroundings than can actually be detected. Thus the Dynamics Based Control (DBC) framework, following the perceptual control principle, dictates us to design an agent's behaviour not to explicitly enforce preferred environment circumstances, but rather to create conditions within the environment that would be recognised by the sensory system as the complete preferred circumstances. This would result in completion of the control task to the extent that can be detected, while economising on the effort to create refinements to the control task, which would not be detected even if they do take place.

DBC models the sensory system as a system dynamics estimation algorithm, and imposes several mild assumptions on the algorithm. First, the algorithm's decision has to be mutable, in the sense that it will readily modify its estimate of the system dynamics given new data. For example an infinite memory frequency statistics would not be usable, as it requires larger and larger amounts of data to modify its estimate. The second assumption is that the algorithm converges or at least exhibits convergent properties for a stable system. That is, given complete observations of a non-perturbed dynamic system, the algorithm produces a good estimate of that system dynamic identity. This means that a single observation estimate without memory of the past is also unlikely to be of use within the DBC

framework.

Dynamics Based Control, as a framework, does not assume any limitation on the environment in which an agent is situated. However, once an implementation for a specific class of environments is required, DBC adopts necessary assumptions from that class of domains. In this thesis, DBC has been adapted for Markovian discrete time and state space environments with partial observability. Taking into account the properties of Markovian environments, such as dependency of the environment development on a finite history of its state and a finite history of the agent's actions, Markovian class-specific definitions of the DBC components have been constructed (Section 4).

Given the DBC adaptation to the Markovian class environments, a system dynamics estimation algorithm was constructed—Extended Markov Tracking (EMT) (Section 4.1). EMT bases its estimate on two system state distributions vectors, representing a single environment modification, and a previous dynamics estimate. EMT thus performs a conservative update, producing a new dynamics estimate that explains away the change in the system state distribution, while remaining as close as possible to the previous estimate.

Extended Markov Tracking relies on an optimisation procedure first stated in [45]. Its analysis, together with the fact that Kullback-Leibler divergence is dual to likelihood, means that EMT tends towards a more likely explanation of the observed change, with respect to the old dynamics estimate as a reference point. Furthermore, properties of the EMT minimisation procedure suggest that for a constant underlying dynamics, a sequence of EMT updates weakly converges to that dynamics or its dynamic limit.

EMT has been applied as the estimator algorithm base for a greedy approxi-

mation to the DBC framework in Section 4.1.2. EMT-based control utilises EMT to predict the effects of an action, and greedily selects an action that would bring the EMT estimate closest to the specified ideal system development. The resulting overall control scheme, in spite of being only an approximation, implements all basic elements and properties of the DBC framework. Since EMT-based control prefers actions that produce dynamics which are closer (more likely with respect to) the specified ideal dynamics, τ^*, the sequence of EMT updates over the controlled system is forced towards τ^* as well, and potentially converges to it.

An important positive aspect of EMT-based control is that the EMT optimisation procedure at its base is time polynomial in the size of the discrete environment model it uses. The same is true for the multi-target and multi-agent cases, introduced in Sections 4.2 and 4.3, where the procedure remains polynomial in all parameters except the number of agents in the system.

One has to take notice that EMT-based control does not provide the EMT algorithm with the true system state transition data. Instead, the EMT algorithm is provided with the sequence of system state beliefs. This means that an EMT-based controller does not only rely on EMT to identify the system dynamics, but also serves as a filter, discarding noise from the dynamic system representation. This noise, in fact, need not come from sensory noise; instead, it can be a result of interference by another agent within the system.

The Multiagent EMT-based control version, introduced in Section 4.2, prescribes that each agent scan the joint action space, and perform its respective element of the optimal action tuple. If the environment correlates the agents, and the effect of the joint multi-agent activity is (partially) observable, then it is conjectured that estimating system dynamics creates an implicit information transfer

between the agents, and facilitates coordination. This conjecture is supported by a successful application of the multiagent EMT-based controller in a multiagent balancing scenario (Section 4.2.1).

EMT-based control also has a multi-target version, presented in Section 4.3, where the control task cannot be described by a single ideal system dynamics. Instead, echoing the principles of Behaviour Based Robotics [2], the desired system dynamics are formulated as a set of heuristic behaviour types that need to be interleaved, combined, and fused together to achieve the desired performance. The target fusion is achieved by creating preference vectors over the action space with respect to each target, and then linearly combining them with respect to some specified weight factors. Sections 4.3.1 and 4.3.2 present experimental data demonstrating that EMT-based action preference data can be successfully used to fuse distinct, and weakly conflicting, targets.

Looking back at the data flow of the DBC framework presented in Figure 3.1, and repeated below, Section 4.3 experiments have, in fact, also confirmed that it would be possible to use EMT-produced data at the User Level to augment and redesign the target system dynamics. Further following back-flow of data within the DBC framework, Section 5 examined the effects of faulty world modelling at the Environment Design level. EMT-based control is applied in a simulated robotic domain with an incoherent world model, and exhibits resistance to this incoherence. By calibration scheme construction, Section 5.2 demonstrated that EMT data procured at the Agent Level can be also effective for environment model calibration.

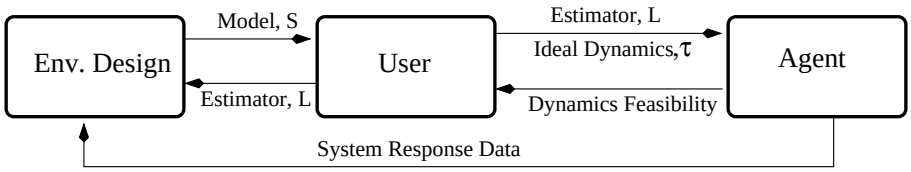


Figure 7.1: Data flow of the DBC framework

7.2 Conclusions and Future Work

The Dynamics Based Control (DBC) Framework concentrates on achieving system dynamics, rather than a specific state or a sequence of states deemed optimal with respect to some optimality criteria. It operates with respect to a given dynamics estimator, and if the estimator is efficient enough, the controlled environment itself is forced to undergo an appropriate controlled variation as well. The philosophical foundation behind DBC is dictated by the perceptual control principle, and allows the control scheme to be efficient with respect to the efforts it invests to achieve a *detectable* degree of task completion.

Though it is still future work to develop a general control solution for DBC, the control solution based on the Extended Markov Tracking (EMT) estimator provides a good approximation with promising theoretical, computational, and practical trends. EMT-based variations to multiple correlated tasks and multiple agents have been shown to work well even with incoherent environment models.

Formulation of a preference vector over actions, which allowed for the multi-target controller version, can also be used to combine multiple environment models with a common action space. This would further link EMT-based control, and DBC itself, with robotic applications. In such applications, multiple robots have to correlate the effects their actions have on their coordination and the actual task

performance. Task completion and coordination would be described by different models, but are likely to have similar action spaces, making the multi-model version of a DBC controller highly applicable.

Several EMT shortcomings are evident at this time, such as the inability to directly handle pure sensory actions. This problem, however, can be easily remedied by changing the type of the environment model. The integration of Predictive State Representations (PSRs) [93] with the DBC framework opens up a promising perspective, both in generalising existing DBC algorithms, and in extending DBC applicability beyond Markovian environments. A thorough theoretical analysis of EMT is also needed to establish convergence rates and stability, and may also reveal additional methods to amend EMT shortcomings.

It will also be important to explore the effects DBC principles may have within domains that are not directly formulated as a control problem. For instance, in repeated games with dynamic opponents the concept of *teaching* may be interpreted as a form of control [91]. Given that the game is repeated and the participants have dynamically changing attitudes, the problem can be roughly reformulated as a control problem over a dynamic system, making it a domain suitable for the application of the DBC framework.

Bibliography

[1] Giogos Apostolikas and Spyros Tzafestas. Improved Q-MDP policy for partially observable Markov decision processes in large domains: Embedding exploration dynamics. *Intelligent Automation and Soft Computing*, 10(3):209–220, 2004.

[2] Ronald C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.

[3] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[4] Michael Bach. 71 optical illusions and visual phenomena. WWW Page http://www.michaelbach.de/ot/, 2007.

[5] J. Bates. Virtual reality, art, and entertainment. *Presence: The Journal of Teleoprators and Virtual Environments*, 1(1):133–138, 1992.

[6] Richard Ernest Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[7] Ulrich Berger. Brown's original fictitious play. In *Evolutionary Game Dynamics Workshop*, Banff, Canada, June 10-15 2006.

[8] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[9] Dimitri P. Bertsekas. *Dynamic programming : deterministic and stochastic models*. Prentice-Hall, 1987.

[10] Sooraj Bhat, David L. Roberts, Mark J. Nelson, Charles L. Isbell, and Michael Mateas. A globally optimal algorithm for ttd-mdps. In *Proceedings of the 6th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1196–1203, 2007.

[11] Jeff 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, Department of Electrical Engeineering and Computer Science, University of California at Berkeley, 1998.

[12] Vincent D. Blondel and John N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, September 2000.

[13] Jim Blythe. *Planning under uncertainty in dynamic domains*. PhD thesis, Carnegie Mellon University, Computer Science Department, 1998.

[14] Jim Blythe. Decision-theoretic planning. *AI Magazine*, 20(2):37–54, 1999.

[15] C. Boutilier and R. Dearden. Using abstractions for decision-theoretic planning with time constraints. In *Proceedings of the 12th AAAI*, pages 1016–1022, 1994.

[16] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[17] G. W. Brown. Iterative solutions of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, 1951.

[18] D. Burago, M. de Rougemont, and A. Slissenko. On the complexity of partially observable Markov decision processes. *Theoretical Computer Science*, 157(2):161–183, 1996.

[19] Andrew S. Cantino, David L. Roberts, and Charles L. Isbell. Autonomous nondeterministic tour guides: Improving quality of experience with TTD-MDPs. In *Proceedings of the 6th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 91–93, 2007.

[20] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI)*, pages 1023–1028, 1994.

[21] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, 1999.

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

[23] Thomas L. Dean and Michael P. Wellman. *Planning and Control*. Morgan Kaufmann, 1991.

[24] Marie E. desJardins, Edmund H. Durfee, Charles L. Ortiz, and Michael J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 4:13–22, 1999.

[25] Prashant Doshi, Yifeng Zeng, and Qiongyu Chen. Graphical models for online solutions to interactive pomdps. In *Proceedings of the International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 809–816, 2007.

[26] Eyal Even-Dar, Sham M. Kakade, and Yishay Mansour. Planning in POMDPs using multiplicity automata. In *Proceeings of 20th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 185–192, 2005.

[27] Ariel Felner, Alex Pomeransky, and Jeffrey S. Rosenschein. Searching for an alternative plan. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 33–40, Melbourne, Australia, July 2003.

[28] Zhengzhu Feng and Shlomo Zilberstein. Region-based incremental pruning for POMDPs. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 146–153, 2004.

[29] R. E. Fikes and N. J. Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[30] P. E. Friedland and Y. Iwasaki. The concept and implementation of skeletal plans. *Journal of Automated Reasoning*, 1(2):161–208, 1985.

[31] Brian Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR-03)*, pages 317–323, 2003.

[32] Piotr Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in mutliagent settings. *Journal of Artificial Intelligence Research (JAIR)*, 24:49–79, 2005.

[33] Claudia V. Goldman and Shlomo Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research (JAIR)*, 22:143–174, 2004.

[34] Judy Goldsmith and Martin Mundhenk. Complexity issues in Markov decision processes. In *Proceedings of IEEE Conference on Computational Complexity*, 1998.

[35] Amy Greenwald and Keith Hall. Correlated-Q learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 242–249, 2003.

[36] Kaijen Hsiao, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Grasping POMDPs. In *Proceedings of the IEEE Conference on Robotics and Automation*, 2007.

[37] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In J. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning (ICML)*, pages 242–250. Morgan Kaufmann, 1998.

[38] Junling Hu and Michael P. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[39] Simon Julier and Jeffrey K. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, Robotics Research Group, Department of Engineering Science, University of Oxford, 1996.

[40] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. Approximate planning in large POMDPs via reusable trajectories. In *Advances in Neural Information Processing Systems (NIPS-12)*, pages 1001–1007, 2000.

[41] Michael Kearns and Satinder Singh. Finite-sampling convergence rates for Q-learning and indirect algorithms. In *Neural Information Processing Systems (NIPS) 12*. MIT Press, 1999.

[42] Sven Koenig. Optimal probabilistic and decision-theoretic planning using Markovian decision theory. Master's thesis, Computer Science Department, University of California at Berkeley, 1991.

[43] Sven Koenig and Reid G. Simmons. Xavier: A robot navigation architecture based on partially observable Markov decision process models. In

D. Kortenkamp, R. Bonasso, and R. Murphy, editors, *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pages 91–122. MIT Press, 1998.

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

[45] S. Kullback. Probability densities with given marginals. *The Annals of Mathematical Statistics*, 39(4):1236–1243, 1968.

[46] S. Kullback and M. A. Khairat. A note on minimum discrimination information. *The Annals of Mathematical Statistics*, 37:279–280, 1966.

[47] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ML11: 11th international conference on machine learning*, pages 157–163, 1994.

[48] Michael L. Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 322–328, 2001.

[49] Michael L. Littman. Value-function reinforcement learning in Markov games. *Journal of Cognitive Research*, 2:55–66, 2001.

[50] Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 394–402, 1995.

[51] M.L. Littman, J. Goldsmith, and M. Mundhenk. The complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

[52] Christopher Lusena, Judy Goldsmith, and Martin Mundhenk. Nonapproximability results for partially observable Markov decision processes. *Journal on Artificial Intelligence Research,* 14, 2001.

[53] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence Journal*, 147(1-2):5–34, July 2003. submitted for publishing April 2001.

[54] Maja J. Mataric. Reward functions for accelerated learning. In *Proceedings 11th International Conference on Machine Learning*, pages 181–189. Morgan Kaufmann, 1994.

[55] Francisco S. Melo and Isabel Ribeiro. Transition entropy in partially observable Markov decision processes. In *Proceedings of the 44th IEEE Conference on Decision and Control (CDC) and European Control Conference (ECC-2005)*, 2005.

[56] *Multiagent Sequential Decision Making (MSDM) workshop at AAMAS*, 2007.

[57] Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of finite-horison Markov decision process problems. *Journal of the ACM*, 47(4):681–720, 2000.

[58] Kevin P. Murphy. A survey of POMDP solution techniques. Technical report, University of British Columbia, 2000.

[59] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 705–711, 2003.

[60] Ranjit Nair, Milind Tambe, and Stacy Marsella. Role allocation and reallocation in multiagent teams: Towards a practical analysis. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 552–559. ACM Press, 2003.

[61] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, pages 133–139, 2005.

[62] Dana S. Nau, Stephen J. J. Smith, and Kutluhan Erol. Control strategies in HTN planning: Theory versus practice. In *AAAI/IAAI*, pages 1127–1133, 1998.

[63] Radford M. Neal and Geoffrey 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.

[64] Mark J. Nelson, David L. Roberts, Jr Charles L. Isbell, and Michael Mateas. Reinforcement learning for declarative optimization-based drama manage-

ment. In *Proceedings of the 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 775–782, 2006.

[65] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, pages 278–287, 1999.

[66] Guillermo Owen. *Game theory*. Academic Press, 3rd edition, 1995.

[67] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, August 1987.

[68] M. A. Peot and D. E. Smith. Conditional nonlinear planning. In *Proceedings of the 1st International Conference on Artificial Intelligence Planning Sysytems*, pages 189–197, 1992.

[69] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value interation: An anytime algorithm for POMDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1030, 2003.

[70] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032, August 2003.

[71] William T. Powers. *Behavior: The control of perception*. Aldine de Gruyter, Chicago, 1973.

[72] Martin L. Puterman. *Markov Decision Processes*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics Section. Wiley-Interscience Publication, New York, 1994.

[73] Zinovi Rabinovich, Claudia V. Goldman, and Jeffrey S. Rosenschein. Non-approximability of centralized control. Technical Report 2002-29, Leibniz Center for Computer Science, Hebrew University, Jerusalem, Israel, 2002.

[74] Zinovi Rabinovich, Claudia V. Goldman, and Jeffrey S. Rosenschein. The complexity of multiagent systems: The price of silence. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1102–1103, Melbourne, Australia, July 2003.

[75] Zinovi Rabinovich and Jeffrey S. Rosenschein. Extended Markov Tracking with an application to control. In *The Workshop on Agent Tracking: Modeling Other Agents from Observations, at the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 95–100, New York, July 2004.

[76] Zinovi Rabinovich and Jeffrey S. Rosenschein. Dynamics based control: An introduction. In *The Third European Workshop on Multi-Agent Systems, EUMAS'05*, pages 323–331, Brussels, Belgium, December 2005.

[77] Zinovi Rabinovich and Jeffrey S. Rosenschein. Multiagent coordination by Extended Markov Tracking. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 431–438, Utrecht, The Netherlands, July 2005.

[78] Zinovi Rabinovich and Jeffrey 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.

[79] Zinovi Rabinovich and Jeffrey S. Rosenschein. Dynamics based control: Structure. In *Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM 2006)*, pages 148–161, Hakodate, Japan, May 2006.

[80] Zinovi Rabinovich and Jeffrey S. Rosenschein. On the response of EMT-based control to interacting targets and models. In *The Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 465–470, Hakodate, Japan, May 2006.

[81] Zinovi Rabinovich, Jeffrey S. Rosenschein, and Gal A. Kaminka. Dynamics based control with an application to area-sweeping problems. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, Honolulu, Hawaii, May 2007. To appear.

[82] David L. Roberts, Mark J. Nelson, Jr Charles L. Isbell, Michael Mateas, and Michael L. Littman. Targeting specific distributions of trajectories in MDPs. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1213–1218, 2006.

[83] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, Cambridge, Massachusetts, 1994.

[84] Maayan Roth, Reid Simmons, and Manuela Veloso. Decnetralized communication strategies for coordinated multi-agent policies. In Lynne E. Parker, Frank E. Schneider, and Alan C. Shultz, editors, *Multi-Robot Systems: from Swarms to Inelligent Automata*, volume III, pages 93–106. Springer, 2005.

[85] Maayan Roth, Reid Simmons, and Manuela Veloso. Exploiting factored representations for decentralized execution in multi-agent teams. In *Proceedings of the International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 457–463, 2007.

[86] Nicholas Roy and Sebastian Thrun. Coastal navigation with a mobile robot. In *Proceedings of Conference on Advances in Neural Information Processing Systems (NIPS)*, 1999.

[87] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A modern approach*. Prentice Hall, 1995.

[88] Thomas Schelling. *The Strategy of Conflict*. Harvard University Press, Cambridge, MA, 1960.

[89] Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In *Proceedings of 12th national conference on AI (AAAI-94)*, pages 426–431, 1994.

[90] Jiaying Shen, Raphen Becker, and Victor Lesser. Agent interaction in distributed mdps and its implications on complexity. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 529–536, 2006.

[91] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, 2007. Special issue on Foundations of Multi-Agent Learning.

[92] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1080–1087, 1995.

[93] Satinder Singh, Michael R. James, and Matthew R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 512–519, 2004.

[94] Trey Smith and Reid Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 542–555, 2005.

[95] Robert F. Stengel. *Optimal Control and Estimation*. Dover Publications, 1994.

[96] Daniel Stronger and Peter Stone. Simultaneous calibration of action and sensor models in a mobile robot. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 4563–4568, 2005.

[97] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An introduction*. The MIT Press, 1998.

[98] Ming Tan. Multi-agent reinforcement learning: independent vs. cooperative agents. In *Proceedings of International Conference on Machine Learning (ML10)*, pages 330–337, 1993.

[99] Georgios Theocharous and Leslie Pack Kaelbling. Approximate planning in POMDPs with macro-actions. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS-16)*, Cambridge, MA, 2004. MIT Press.

[100] Manuela Veloso, Jaime Carbonell, Alicia Prez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1), 1995.

[101] Eric A. Wan and Rudolph van der Merwe. The unscented kalman filter for nonlinear estimation. In *IEEE Symposium on Adaptive Systems for Signal Processing, Communications, and Control*, pages 153–158, 2000.

[102] C. J. Watkins. *Models of Delayed Reinforcement Learning*. PhD thesis, Psychology Department, Cambridge University, 1989.

[103] C. J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[104] P. Weyhrauch. *Guiding Interactive Drama*. PhD thesis, School of Computer Science, Carnegie-Mellon University, 1997.

[105] David Wingate and Satinder Singh. Kernel predictive linear Gaussian models for nonlinear stochastic dynamical systems. In *Proceedings of the In-*

ternational Conference on Machine Learning (ICML), pages 1017 – 1024, 2006.

[106] David Wingate and Satinder Singh. Mixtures of predictive linear Gaussian models for nonlinear stochastic dynamical systems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 524–529, 2006.

[107] David Wingate and Satinder Singh. On discovery and learning of models with predictive representations of state for agents with continuous actions and observations. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1128–1135, 2007.

[108] Britton Wolfe and Satinder Singh. Predictive state representations with options. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 1025–1032, 2006.

[109] Michael Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, February 2002.

[110] Jianzhong Wu and Robert Axelrod. How to cope with noise in the iterated prisoner's dilemma. *Journal of Conflict Resolution*, 39(1):183–189, 1995.

[111] James R. Van Zandt. A more robust unscented transform. Technical Report MS-M210, MITRE Corporation, 202 Burlington Road, Bedford, USA, July 2001.