Dynamics Based Control with an Application to Area-Sweeping Problems

Zinovi Rabinovich and Jeffrey S. Rosenschein and Gal A. Kaminka

Critical MAS Research Group
The Hebrew University of Jerusalem

and

MAVERICK Research Group
Bar Ilan University
Agenda

- Area-Sweeping Scenario
- Dynamics Based Control (DBC)
  - DBC Components and Design
  - DBC for Markovian Environments
  - Extended Markov Tracking (EMT) Solution
- Experiment: EMT solving Area-Sweeping
- Conclusions
Area-Sweeping Scenario

- A graduate student needs to make a clip of a running cockroach.
- The student takes a torch and begins to scan his dark kitchen’s floor.
- But cockroaches do not like light and shy away from it.
- How the student should control his torchlight?
How the student should control his torchlight?

**Focus on Dynamics**

- Move the light until you spot one.
- Keep the light at the cockroach.
- Shift the light to keep the cockroach in the spotlight.
Importance of Dynamics

- Human vision relies on change to discover a scene’s composition.
- Not any particular “frame”, but the change between them.
- Dynamics, i.e. rules of the change, are recovered.
- The dynamics tag (or recognize) different parts of a scene.
Requirements Focus

How the student should control his torchlight?

Focus on Dynamics
- Move the light until you spot one.
- Keep the light at the cockroach.
- Shift the light to keep the cockroach in the spotlight.

Focus on Dynamics Impression
- Cockroach will run only if it thinks it’s in danger.
Requirements Focus

How the student should control his torchlight?

Focus on Dynamics
- **Move** the light until you spot one.
- **Keep** the light at the cockroach.
- **Shift** the light to keep the cockroach in the spotlight.

Focus on Dynamics **Impression**
- Cockroach will run only if it thinks it’s in danger.

The student should move the torchlight to impress upon the cockroach to run in a sleeve of light.
Dynamics Based Control (DBC)

- Formulated by three levels:
  - Environment Design level
  - User (strategic) level
  - Agent (tactical) level
Dynamics Based Control (DBC)

Formulated by three levels:

- Environment Design level
- User (strategic) level
- Agent (tactical) level

Formal specs and modeling of the environment
Dynamics Based Control (DBC)

Formulated by three levels:

- Environment Design level
- User (strategic) level
- Agent (tactical) level

Formal specs and modeling of the environment

A student with a torch enters a kitchen with a cockroach.
Dynamics Based Control (DBC)

- Formulated by three levels:
  - Environment Design level
  - User (strategic) level
  - Agent (tactical) level

- Dynamics estimator specs, the ideal (target) dynamics, and dynamics divergence measure
Dynamics Based Control (DBC)

- Formulated by three levels:
  - Environment Design level
  - User (strategic) level
  - Agent (tactical) level

- Dynamics estimator specs, the ideal (target) dynamics, and dynamics divergence measure

- The student knows
  - how the cockroach evaluates the situation,
  - preference on the torchlight motion,
  - measure the difference between them.
Dynamics Based Control (DBC)

Formulated by three levels:
* Environment Design level
* User (strategic) level
* Agent (tactical) level

Utilization of environment model and dynamics estimator to create target dynamics within environment.
Dynamics Based Control (DBC)

Formulated by three levels:
* Environment Design level
* User (strategic) level
* Agent (tactical) level

Utilization of environment model and dynamics estimator to create target dynamics within environment.

The student anticipates the cockroach perceptual reaction and movies the torch.
DBC: Design Cycle

Formulated by three levels:
- Environment Design level
- User (strategic) level
- Agent (tactical) level
DBC: Design Cycle

- Formulated by three levels:
  - Environment Design level
  - User (strategic) level
  - Agent (tactical) level

- The data flow between the levels can be depicted as:
DBC via Extended Markov Tracking

- DBC is very general
  - Multiple algorithmic solutions are possible
- When formulated for Markovian environments
  - Extended Markov Tracking (EMT) based control
  - A greedy approximate solution
DBC for Markovian Environment

For Markovian Environment it is possible to specify DBC in a more explicit form.

Environment Design: Markovian environment
\(< S, A, T, O, \Omega, s_0 >\)

User: \( L : O \times (A \times O)^* \rightarrow \mathcal{F}, \tau^* \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R} \).

Agent: \( a^* = \arg \min_a Pr(d(\tau_a, \tau^*) > \theta) \)
DBC for Markovian Environment

For Markovian Environment it is possible to specify DBC in a more explicit form.

Environment Design: Markovian environment $< S, A, T, O, \Omega, s_0 >$, where

- $S$ - set of possible environment states
- $s_0 \in \Pi(S)$ - the initial state (distribution)
- $A$ - set of possible actions applicable
- $T : S \times A \rightarrow \Pi(S)$ is the stochastic transition function
- $O$ - set of (partial) observations
- $\Omega : S \times A \times S \rightarrow \Pi(O)$ - stochastic observability function

User: $L : O \times (A \times O)^* \rightarrow \mathcal{F}, \tau^* \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}$.

Agent: $a^* = \arg \min_a Pr (d(\tau_a, \tau^*) > \theta)$
For Markovian Environment it is possible to specify DBC in a more explicit form.

Environment Design: Markovian environment
\[ < S, A, T, O, \Omega, s_0 > \]

User: \[ L : O \times (A \times O)^* \rightarrow \mathcal{F}, \tau^* \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}. \]

Agent: \[ a^* = \arg \min_{a} Pr(d(\tau_a, \tau^*) > \theta) \]
DBC for Markovian Environment

For Markovian Environment it is possible to specify DBC in a more explicit form.

Environment Design: Markovian environment

\[
< S, A, T, O, \Omega, s_0 >
\]

User: \( L : O \times (A \times O)^* \rightarrow \mathcal{F}, \tau^* \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R} \).

\( \mathcal{F} = \{ \tau : S \times A \rightarrow \Pi(S) \} \) all possible dynamics

\( L \) is dynamics estimator

\( \tau^* \) is the ideal dynamics (tactical target)

\( d \) is the dynamics divergence measure

Agent: \( a^* = \arg \min_a Pr(d(\tau_a, \tau^*) > \theta) \)
For Markovian Environment it is possible to specify DBC in a more explicit form.

Environment Design: Markovian environment
\[ < S, A, T, O, \Omega, s_0 > \]

User: \[ L : O \times (A \times O)^* \rightarrow F, \tau^* \in F, d : F \times F \rightarrow R. \]

Agent: \[ a^* = \arg \min_a Pr(d(\tau_a, \tau^*) > \theta) \]
DBC for Markovian Environment

For Markovian Environment it is possible to specify DBC in a more explicit form.

Environment Design: Markovian environment
\(< S, A, T, O, \Omega, s_0 >\)

User: \( L : O \times (A \times O)^* \rightarrow \mathcal{F}, \tau^* \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}. \)

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

with \( \theta \) coming from User level as well, or algorithm specific

alternatively \( a^* = \arg\min_a d(d(\tau_a, \tau^*), \delta(0)) \)
EMT-based control as DBC

- Environment Design
  - Markovian Environment \(< S, s_0, A, T, O, \Omega >\)

- User level
  - Dynamics divergence measure is Kullback-Leibler
  - Estimator is Extended Markov Tracking (EMT)

- Agent Level
  - **Greedy** action selection based on EMT predicted response
EMT: Intuition

- EMT assumes that the observed process is a non-controlled Markov chain.
- Space of all possible dynamics has the form $D : S \rightarrow \Pi(S)$.
- That is, the form of an explanation to a change in beliefs about the system state:
  $$p_{t+1}(s') = D(s' | s)p_t(s)$$
- Maintain the estimate by a conservative update w.r.t. Kullback-Leibler divergence
EMT: Tracking

- Given system state beliefs change from $p_t$ to $p_{t+1}$
  \[ p_{t+1}(s) \propto p(o|s, a) \sum_{s'} T(s'|a, s)p_t(s') \]

- Given previous system dynamics estimate $PD_t$, then the update of this belief is the solution to:
  \[
  PD_{t+1} = \arg\min Q \left[ E_{p_t(s)} \left[ D_{KL}(Q(\cdot|s) \parallel PD_t(\cdot|s)) \right] \right] \\
  \text{s.t.} \\
  p_{t+1} = Q \cdot p_t
  \]

- Denote $PD_{t+1} = H[p_{t+1}, p_t, PD_t]$. 
EMT-based Tactical Solution

Now that we have a means of understanding how the system develops, we can attempt to fix it to our liking, the liking of tactical target $\tau^* : S \rightarrow \Pi(S)$.
EMT-based Tactical Solution

Now that we have a means of understanding how the system develops, we can attempt to fix it to our liking, the liking of tactical target \( \tau^* : S \rightarrow \Pi(S) \).

Tactical solution performs continual loop of
- EMT estimation of system development
- Action choice

\[
a^* = \arg \min_a D_{KL}( H[T_a \ast p_t, p_t, PD_t] \parallel \tau^*)
\]

Application of \( a^* \).
Example: Game of Tag

Agent $A$ tries to tag the quarry $Q$.

$Q$ randomly chooses direction away from $A$ and takes a probabilistic step.

$A$ can take a (deterministic) step in either of four directions.
Game of Tag: Environment Design

- State space is a Cartesian product
  \[ S = \{c_0, ..., c_n\} \times \{c_0, ..., c_n\} \]
- Actions \( A = \{\text{North, South, West, East}\} \)
- \( T : S \times A \times S \) with accordance to random motion of the quarry \( Q \) and the step taken by agent \( A \).
- Observations are \( \{c_0, ..., c_n\} \) corresponding to the quarry location
  - Scenario I: Blind sweep: all observations are equiprobable
  - Scenario II: Shortsighted: all observations are equiprobable except the location of the agent which has zero probability.
Game of Tag: Targets

- Three targets:
  - Catch the Quarry:
    \[ T_{catch}(A_{t+1} = s_i|Q_t = s_j, A_t = s_a) \propto \begin{cases} 
    1 & \text{if } s_i = s_j \\
    0 & \text{otherwise}
  \end{cases} \]
  - Stay mobile:
    \[ T_{mobile}(A_{t+1} = s_i|Q_t = s_o, A_t = s_j) \propto \begin{cases} 
    0 & \text{if } s_i = s_j \\
    1 & \text{otherwise}
  \end{cases} \]
  - Stalk the Quarry:
    \[ T_{stalk}(A_{t+1} = s_i|Q_t = s_o, A_t = s_j) \propto \frac{1}{\text{dist}(s_i, s_o)} \]
  - Balancing [0.8, 0.1, 0.1]
Game of Tag: Domains

Dead-Ends

Arena

Circle

Dynamics Based Control with an Application to Area-Sweeping Problems – p.18/22
## Results

### Success Rate:

<table>
<thead>
<tr>
<th>Model</th>
<th>Domain</th>
<th>Capture%</th>
<th>$E$(Steps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Dead-ends</td>
<td>100</td>
<td>14.8</td>
</tr>
<tr>
<td></td>
<td>Arena</td>
<td>80.2</td>
<td>42.4</td>
</tr>
<tr>
<td></td>
<td>Circle</td>
<td>91.4</td>
<td>34.6</td>
</tr>
<tr>
<td>II</td>
<td>Dead-ends</td>
<td>100</td>
<td>13.2</td>
</tr>
<tr>
<td></td>
<td>Arena</td>
<td>96.8</td>
<td>28.67</td>
</tr>
<tr>
<td></td>
<td>Circle</td>
<td>94.4</td>
<td>31.63</td>
</tr>
</tbody>
</table>
Results

Empirical agent location entropy:
Observation Scenario I: Omniposition Quarry: Dead-End

![Graph showing entropy over steps for Dead-ends]
Results

Empirical agent location entropy:
Observation Scenario I: Omniposition Quarry: Arena

![Graph showing entropy over steps for Arena dynamic control application to area-sweeping problems.](image-url)
Results

Empirical agent location entropy:
Observation Scenario I: Omniposition Quarry: Circle
Results

Empirical agent location entropy:
Observation Scenario II: Not at Agent Location: Dead-End

![Graph showing the entropy over steps for dead-ends scenario.](image-url)
Results

Empirical agent location entropy:
Observation Scenario II: Not at Agent Location: Arena
Results

Empirical agent location entropy:
Observation Scenario II: Not at Agent Location: Circle
Conclusions

- Dynamics Based Control (DBC)
  - Focus on System Dynamics Impression
- DBC Implementation
  - Based on the Extended Markov Tracking (EMT)
- Natural applicability to dynamic domains
Current and Future Work

- Generic solution to DBC
  - Theoretical convergence and stability
- Non-Markovian environments
  - Predictive State Representations (PSRs)
  - Dynamic (non-linear) systems
THANK YOU