Dynamics Based Control: An Introduction

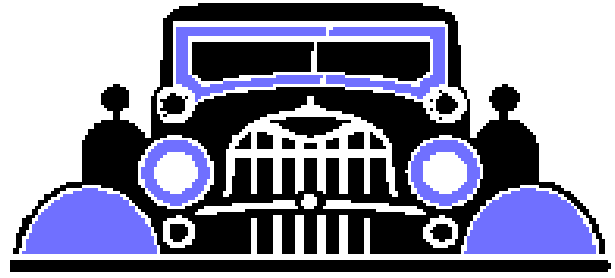
Zinovi Rabinovich
Jeffrey S. Rosenschein

School of Engineering and Computer Science
The Hebrew University of Jerusalem

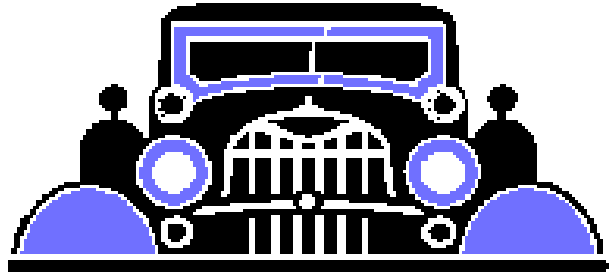
Agenda

- Motivational Example
- Dynamics Based Control (DBC)
- Planning Perspective
- Control Perspective
- Suboptimal DBC via Extended Markov Tracking
- Future Work

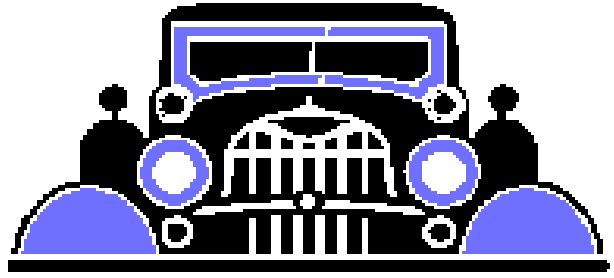
Car on a Road



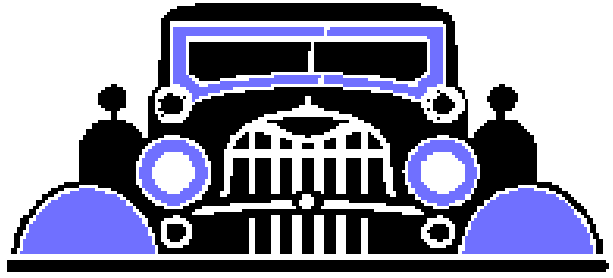
Car on a Road



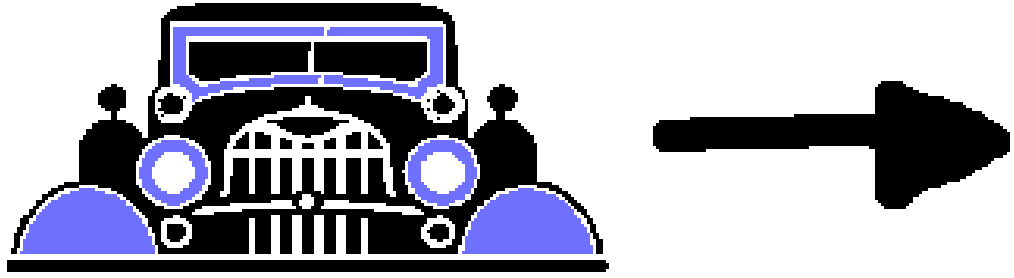
Car on a Road



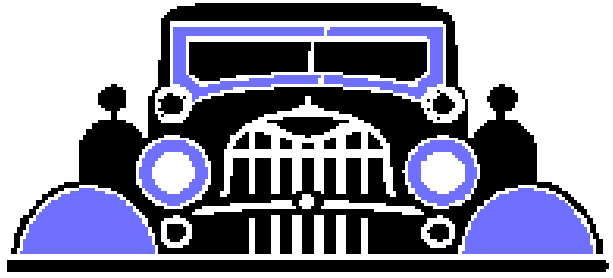
Car on a Road



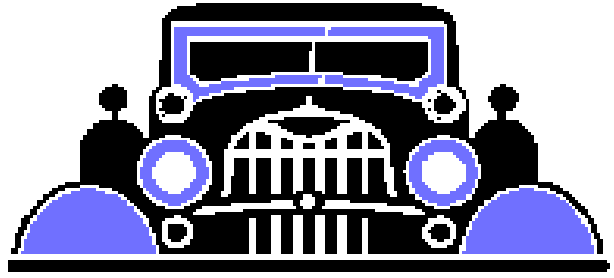
Car on a Road



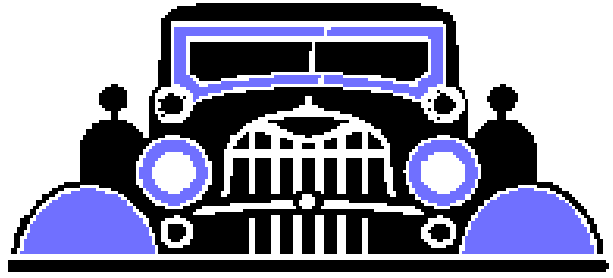
Car on a Road



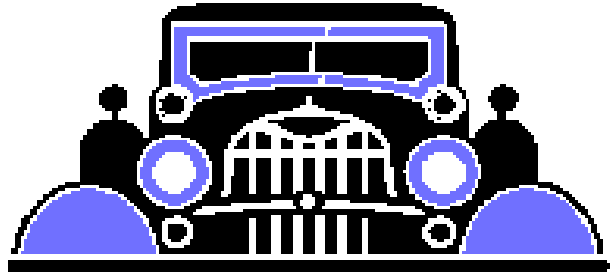
Car on a Road



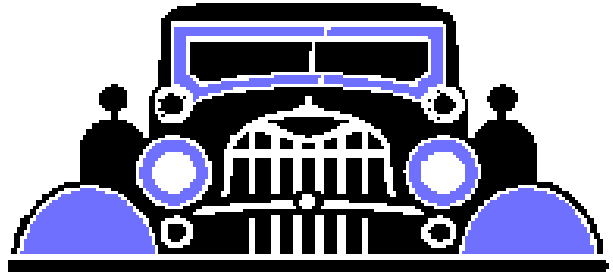
Car on a Road



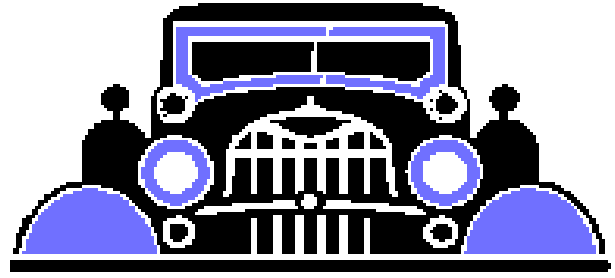
Car on a Road



Car on a Road



Car on a Road



Dynamics Based Control (DBC)

- Formulated by three levels:
  - Environment Design Level
  - User level
  - Agent level

Dynamics Based Control (DBC)

- Formulated by three levels:
  - Environment Design Level
    - Formal specs and modeling of the environment
  - User level
  - Agent level

Dynamics Based Control (DBC)

- Formulated by three levels:
  - Environment Design Level
  - User level
  - Agent level

Dynamics Based Control (DBC)

- Formulated by three levels:
  - Environment Design Level
  - User level
    - Ideal dynamics and dynamics estimator specs, and dynamics divergence measure
  - Agent level

Dynamics Based Control (DBC)

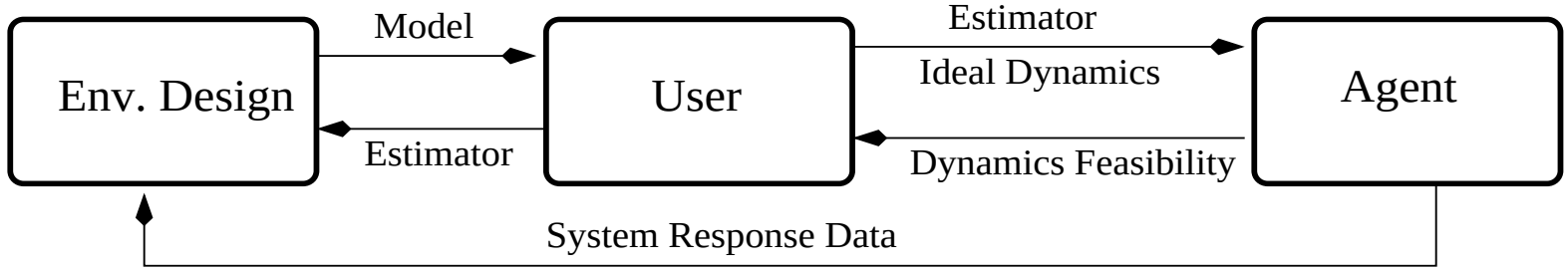
- Formulated by three levels:
  - Environment Design Level
  - User level
  - Agent level

Dynamics Based Control (DBC)

- Formulated by three levels:
  - Environment Design Level
  - User level
  - Agent level
    - Utilization of environment model and dynamics estimator to create ideal dynamics within environment.

Dynamics Based Control (DBC)

- Formulated by three levels:
  - Environment Design Level
  - User level
  - Agent level
- The data flow between the levels can be depicted as:



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: $L : O \times (A \times O)^* \rightarrow \mathcal{F}, q \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}$.
  - Agent: $a^* = \arg\min_a Pr(d(\tau_a, q) > \theta)$

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: $L : O \times (A \times O)^* \rightarrow \mathcal{F}, q \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}$.
  - Agent: $a^* = \arg\min_a Pr(d(\tau_a, q) > \theta)$

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: $L : O \times (A \times O)^* \rightarrow \mathcal{F}, q \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}$.
  - Agent: $a^* = \arg\min_a Pr(d(\tau_a, q) > \theta)$

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: $L : O \times (A \times O)^* \rightarrow \mathcal{F}, q \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}$.
    - $\mathcal{F} = \{\tau : S \times A \rightarrow \Pi(S)\}$ all possible dynamics
    - L is dynamics estimator
    - q is the ideal dynamics
    - d is the dynamics divergence measure
  - Agent: $a^* = \arg\min_a Pr(d(\tau_a, q) > \theta)$

DBC for Markovian Environment

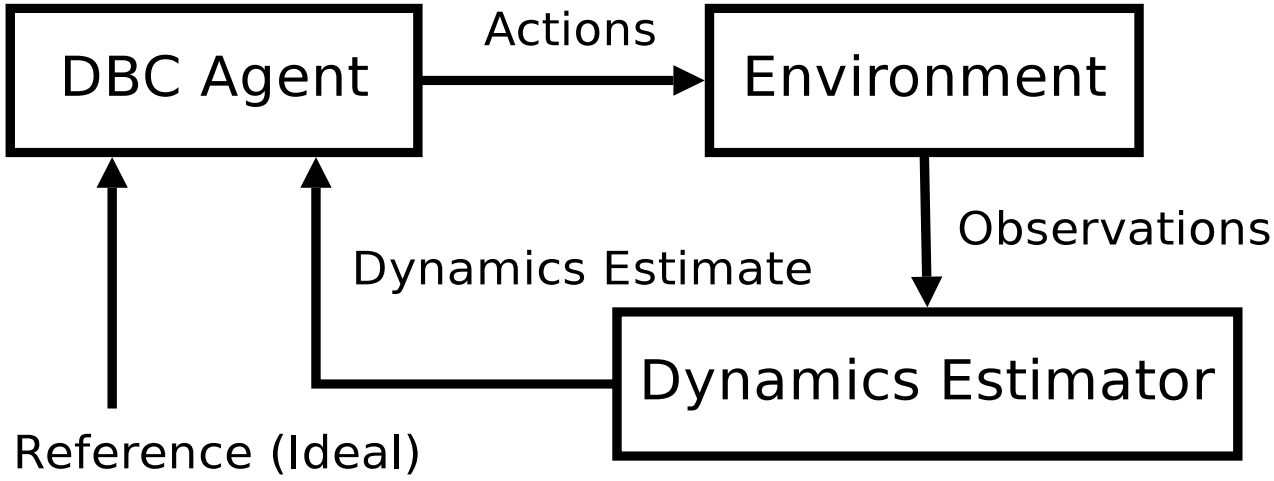
- For Markovian Environment it is possible to specify DBC in a more explicit form.
  - Environment Design: Markovian environment $< S, A, T, O, \Omega, s_0 >$
  - User: $L : O \times (A \times O)^* \rightarrow \mathcal{F}, q \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}$.
  - Agent: $a^* = \arg\min_a Pr(d(\tau_a, q) > \theta)$

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: $L : O \times (A \times O)^* \rightarrow \mathcal{F}, q \in \mathcal{F}, d : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}$.
  - Agent: $a^* = \arg\min_a Pr(d(\tau_a, q) > \theta)$
    - with θ coming from User level as well, or algorithm specific
    - alternatively $a^* = \arg\min_a d(d(\tau_a, q), \delta(0))$

Control Perspective

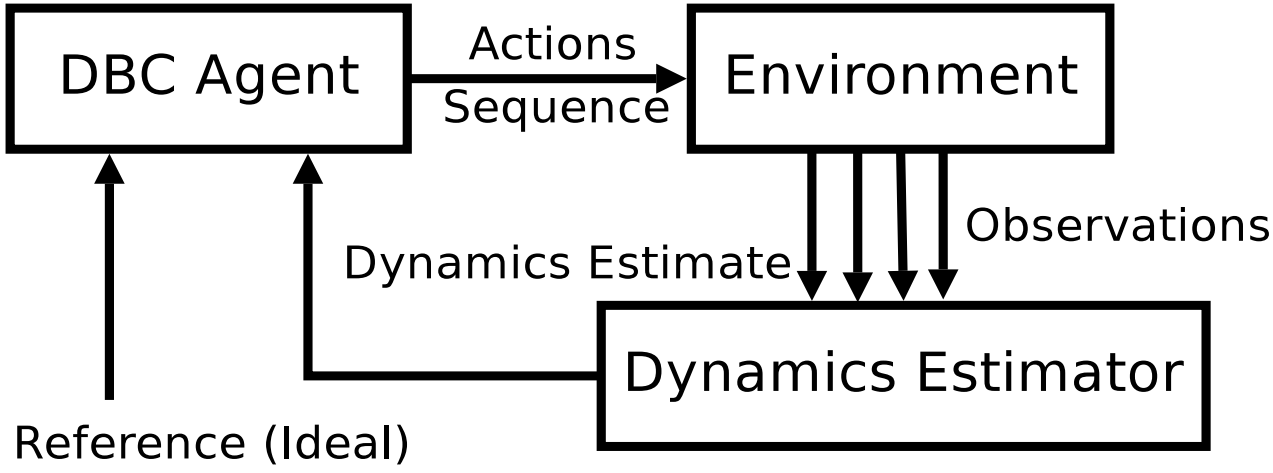
- Agent Level algorithm can be seen from control perspective:



- Control perspective requires dynamics estimation to change “smoothly”
- Not environment state, but rules of the change – system dynamics – are inferred.

Planning Perspective

- Appears if dynamics estimate is based on a non-trivial sequence of actions, rather then corrected.



- Dynamics Estimator is akin to plan recognition
- Agent level deals with sets of plans, and chooses one whose remainder is to achieve the ideal dynamics

DBC vs. and pro. POMDP

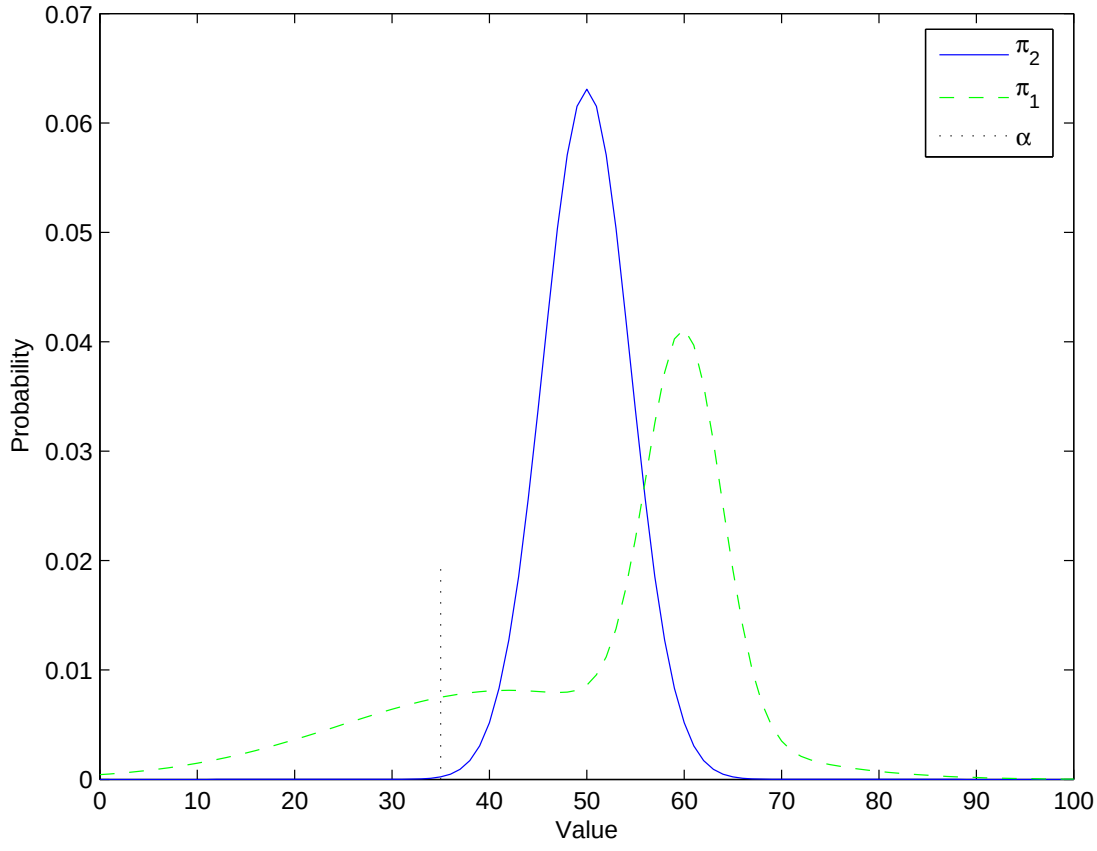
- In Markovian environment planning and control overlap
  - Except off-line vs. on-line property
- Alternative planning task exists: (PO)MDP
  – (Partially Observable) Markov Decision Process
  - Environment Design coincides with DBC
  - User: define reward structure $r : S \times A \rightarrow \mathcal{R}$
  - Agent: find plan (policy) such that
    $\pi^* = \arg\max_\pi E(\sum \gamma^i r_i)$
- DBC can be positioned both versus and pro POMDP
  - DBC as an on-line planning scheme vs. POMDP
  - DBC control is pro POMDP as plan implementation

Vs. POMDP

- Optimality Concept
- Controller Similarity
- Preference Interpretation

Vs. POMDP

- Optimality Concept
  - POMDP selects **maximum expected** payoff



Vs. POMDP

- Optimality Concept
  - DBC optimality is based on probability threshold
  - Direct comparison between induced and ideal dynamics

Vs. POMDP

- Optimality Concept
- Controller Similarity
  - POMDP has off-line policy computation, blindly applied based on value expectation — Open loop control

Vs. POMDP

- Optimality Concept
- Controller Similarity
  - DBC is explicitly on-line, with continual policy update with complete integration of sensory information — Closed loop control

Vs. POMDP

- Optimality Concept
- Controller Similarity
- Preference Interpretation
  - POMDP has user preference expressed as $r : S \times A \times S \rightarrow \mathcal{R}$
    - (if normalized) constitutes dynamics preference
  - Solution is based on state oriented Value Function
    - possible preference distortion

Vs. POMDP

- Optimality Concept
- Controller Similarity
- Preference Interpretation
  - DBC directly compares induced and estimated system dynamics
    - No preference distortion
  - Can translate into direct comparison of value distributions of policies

DBC via Extended Markov Tracking

- Complete DBC algorithm is yet available
- Approximation with all DBC features exists
  - Extended Markov Tracking (EMT) based control
- EMT is a dynamics estimator
  - Conservative with respect to Kullback-Leibler divergence measure
  - Continuous in the environment state beliefs space

$$\tau_{EMT}^{t} = H[p_t, p_{t-1}, \tau_{EMT}^{t-1}] = \arg\min_{\tau} D_{KL}(\tau \times p_{t-1} \| \tau_{EMT}^{t-1} \times p_{t-1})$$

$$s.t.$$

$$p_t(s') = \sum_{s} (\tau \times p_{t-1})(s', s)$$

$$p_{t-1}(s) = \sum_{s'} (\tau \times p_{t-1})(s', s)$$

EMT as DBC

- Environment Design
  - Markovian Environment
- User level
  - Estimator is EMT
  - Dynamics divergence measure is Kullback-Leibler
- Agent Level
  - Greedy action selection based on EMT predicted response

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

EMT: Success and Limitations

- EMT has been successfully applied to balancing problems, both single and multi-agent scenarios.
- EMT is limited relatively to DBC in general
  - EMT can not directly deal with negative preference
  - EMT can not directly deal with pure sensory actions
    - Pure sensory actions virtually do not exist in robotics

Future Work

- Extensive theoretical study of DBC framework
  - General and special case complexity
    - EMT is polynomial, but this is an approximation
  - Convergence rates, and conditions
- Algorithmic implementations
  - Modification and extension of EMT-based version
  - Alternative estimators integration
  - Problem specific implementations

THANK YOU