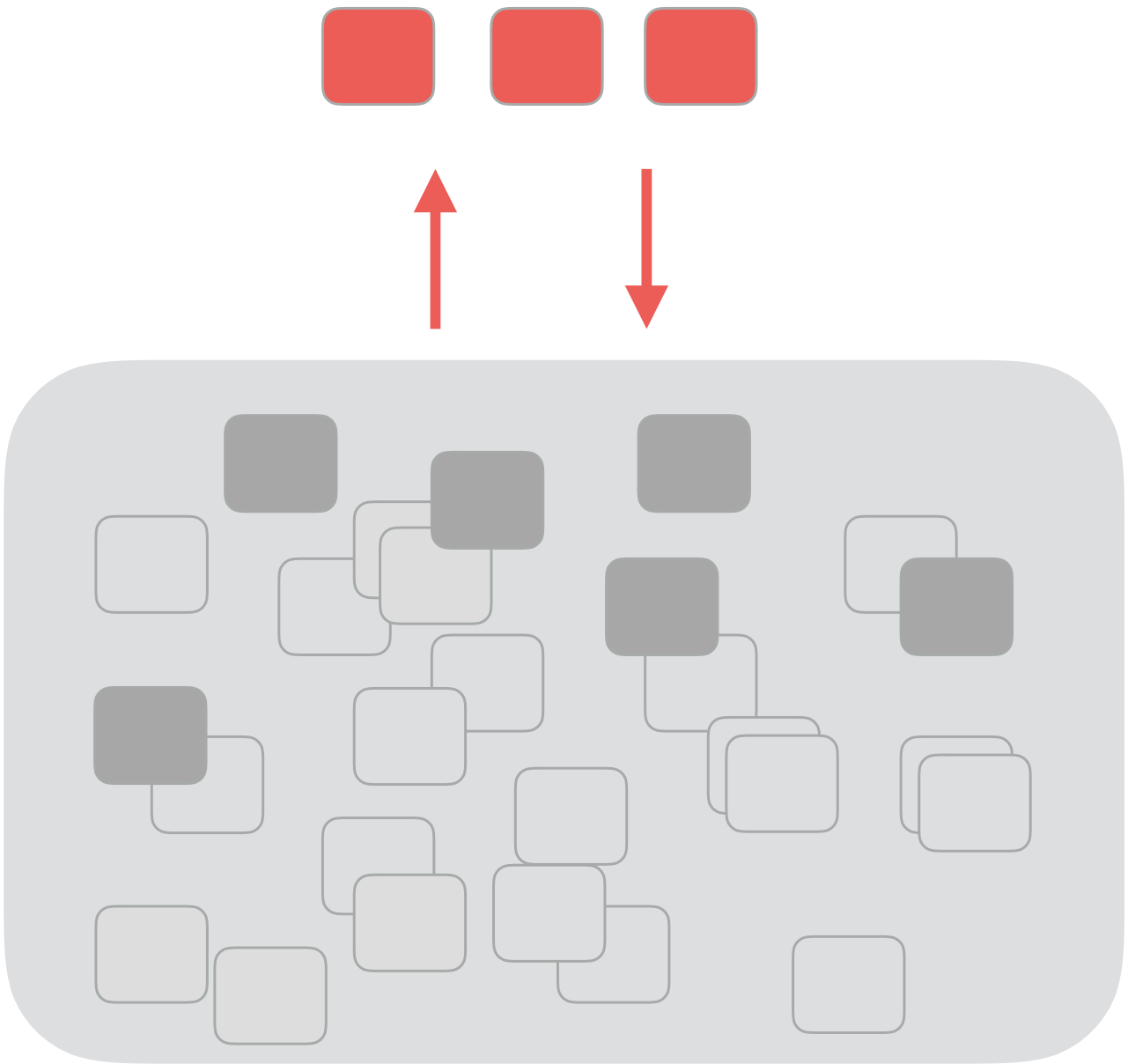
Modules Hypothesis

Routines can be learned with reinforcement and run in small groups



Forebrain contains a vast number of remembered modules

A module is an independent perception to action unit

A small number of active modules governed by working memory

Each module has its own rewards

Active modules may be swapped out on a 300ms time scale to meet changing contingencies

Reinforcement Learning Modules

Focus on human models: Embodied cognition

Results:

1. Basic modular RL
2. Visual sensing component
3. Credit assignment component
4. Fast Inverse RL with modules

RL Definitions

An MDP consists of a 4-tuple (S, A, T, R)

S being the set of possible states,

A the set of possible actions,

T the transition model describing the probabilities

$P(s_{t+1}|s_t, a_t)$ of reaching a state s_{t+1} from state s_t at time t and executing action a_t,

R is the expected value of the reward r_t, distributed according to $P(r_t|s_t, a_t)$ and is associated with the transition from state s_t to some state s_{t+1} when executing action a_t.

Modules have to agree on an action

$$\mathcal{M}_i = \{S_i, A, T_i, R_i\}$$

One way: Average

$$Q(s_t, a_t) = \sum_{i=1}^{M} Q(s_t^{(i)}, a_t^{(i)})$$

Better way: Softmax using

$$P(a_t^{(j)} | Q(s_t^{(1)}, a_t), ..., Q(s_t^{(M)}, a_t)) = \frac{e^{Q(s_t^{(j)}, a_t^{(j)})/\tau}}{\sum_{i=1}^{M} e^{Q(s_t^{(i)}, a_t^{(i)})/\tau}}$$

Q-Learning variant of Temporal Difference Learning

The goal of RL is to find a policy π that maps from the set of states S to actions A so as to maximize the expected total discounted future reward

$$V^{\pi}(s) = E^{\pi}\Big(\sum_{t=0}^{\infty} \gamma^t r_t\Big) \tag{1}$$

Alternatively, the values can be parametrized by state and action pairs, denoted by $Q^{\pi}(s, a)$.

$$Q^{*}(s, a) = \sum_{r} rP(r|s, a) + \gamma \sum_{s' \in S} P(s'|s, a) \max_{a'} Q^{*}(s', a') \tag{2}$$

Temporal difference learning

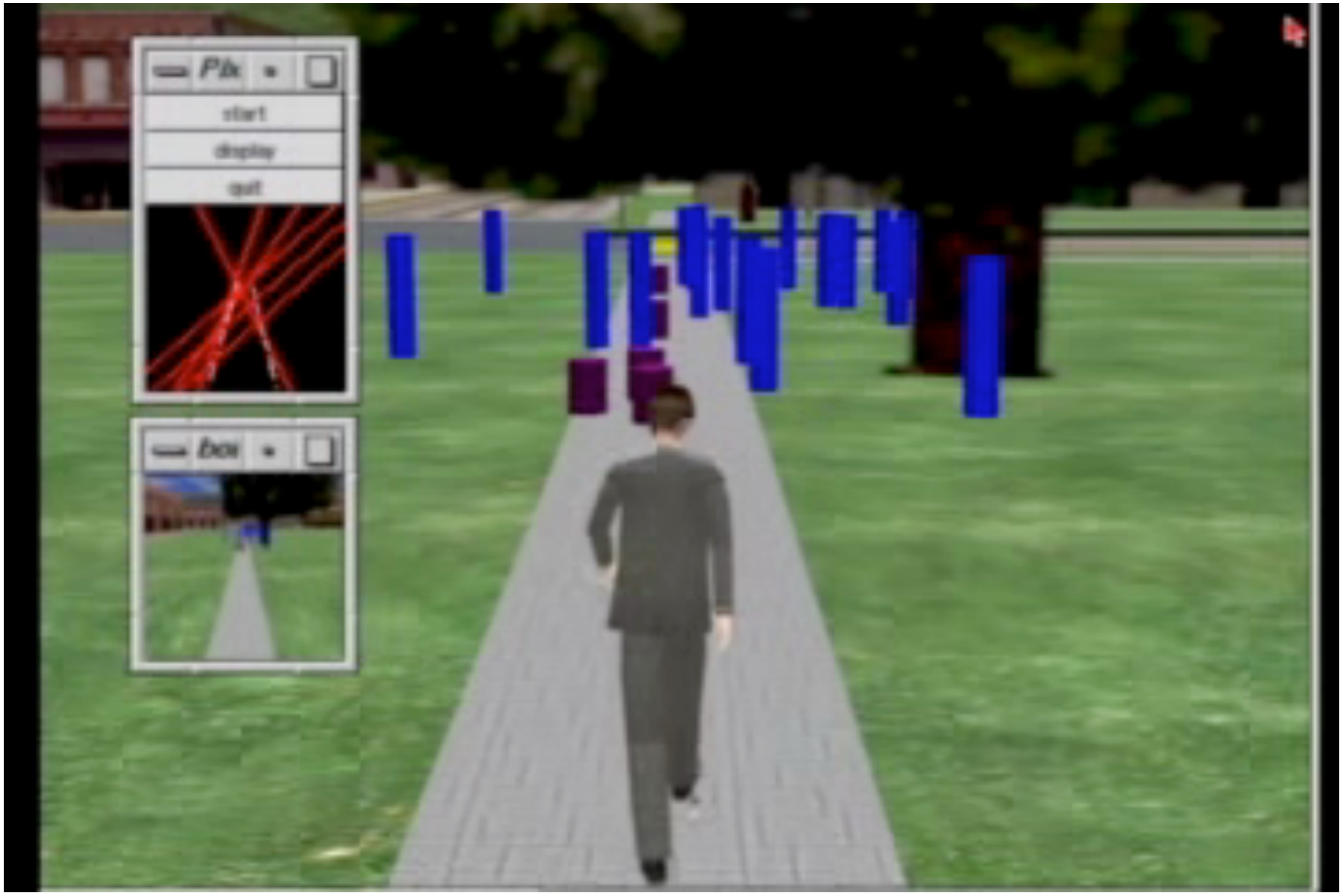
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \delta_Q \tag{3}$$

$$\delta_Q = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t). \tag{4}$$

1

Modules that compete for action choices have the promise of good scaling behavior

Agent “Walter” walks down a sidewalk, avoiding blue obstacles and picking up purple litter

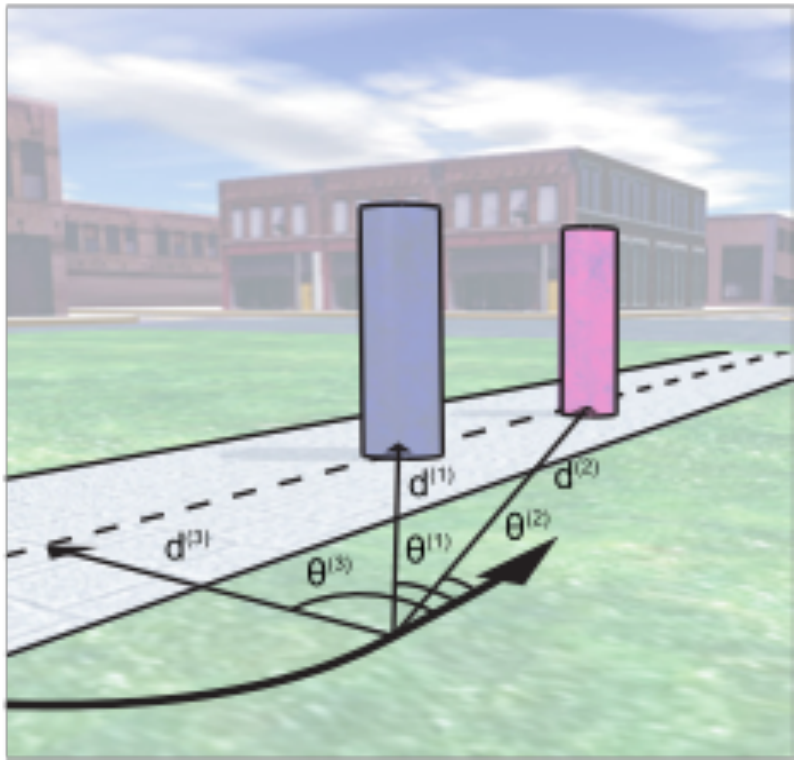


Sidewalk

Obstacle

Litter

State Spaces definitions

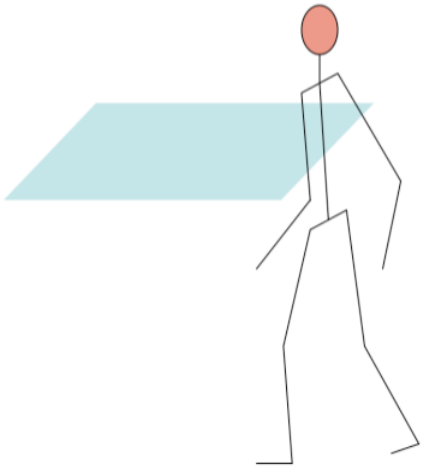


1. Visual Routine

→ θ,d

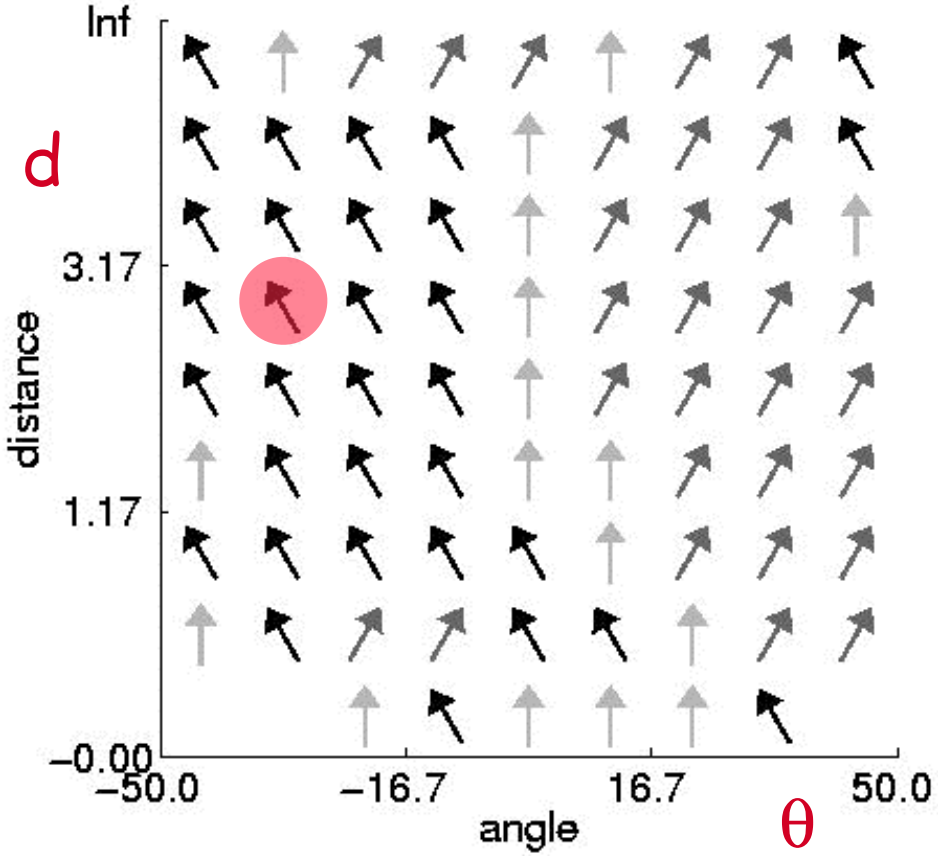
↓

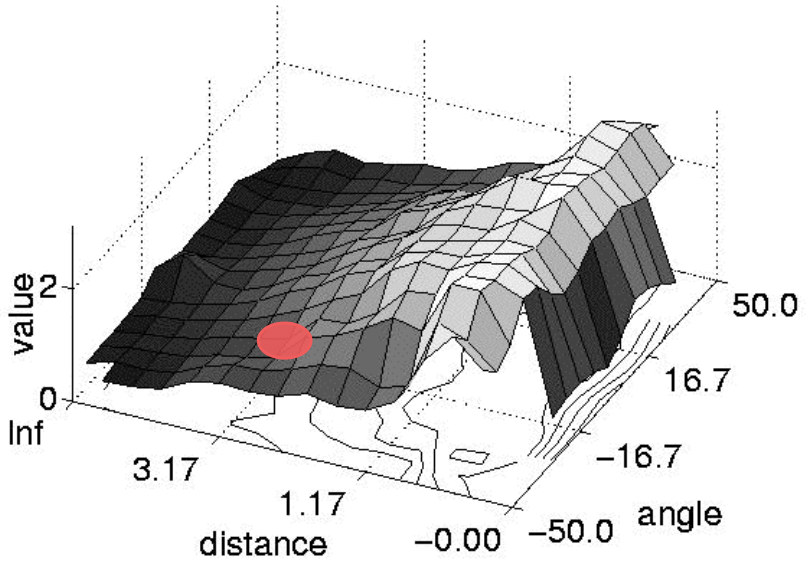
2a. Policy



Module for Litter Cleanup

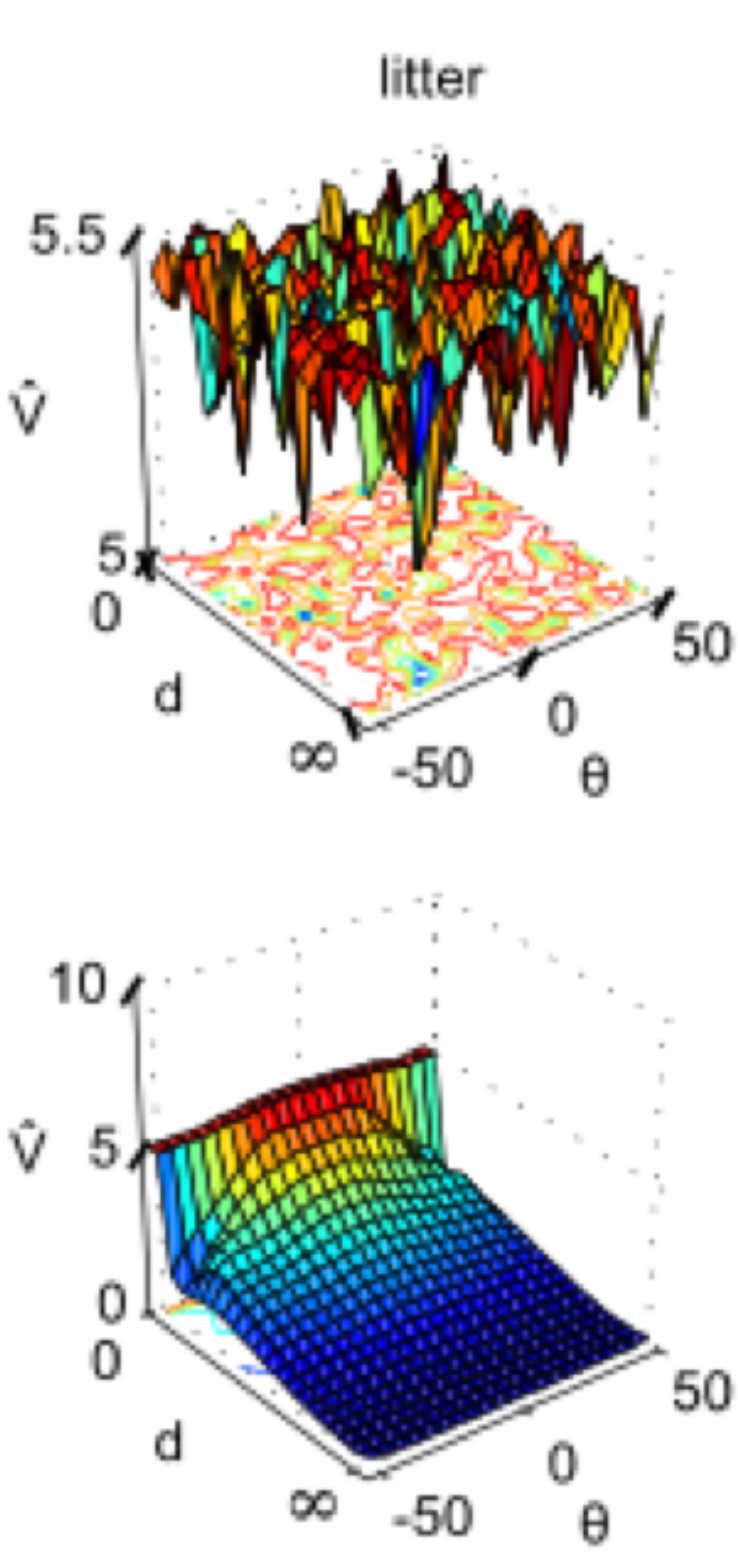
2b. V is value of Policy





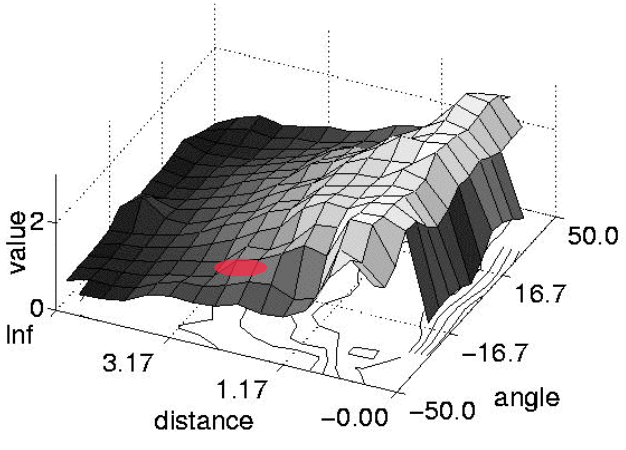
$V(s) = \max_a Q(s,a)$

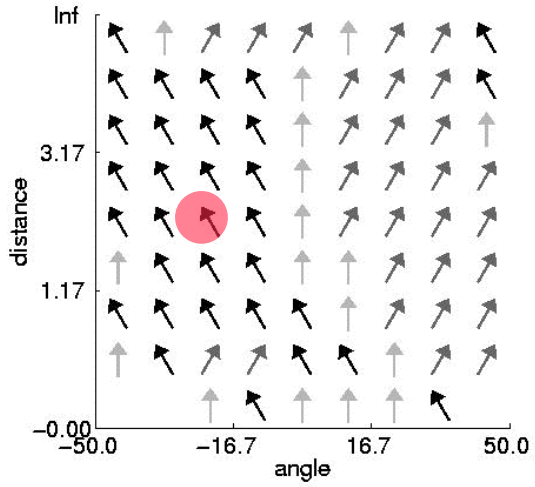
Heading from Walter's perspective


litter
5.5
$\hat{V}$
5
0
d
∞
-50
0
θ
50
10
$\hat{V}$
5
0
0
d
∞
-50
0
θ
50

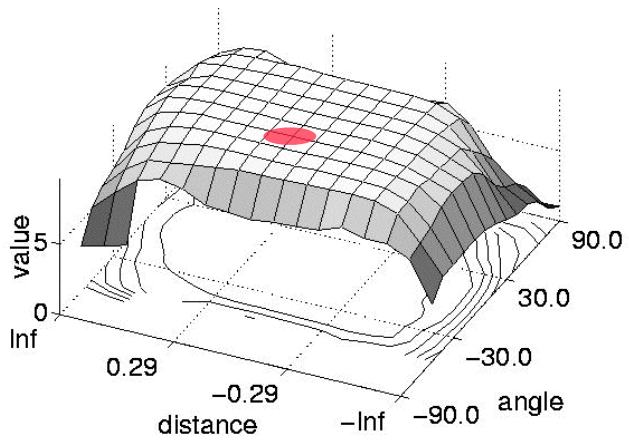
Learned Module Behaviors

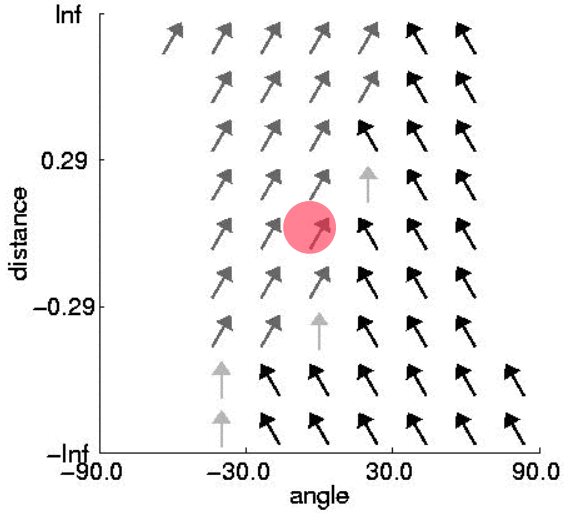
Litter



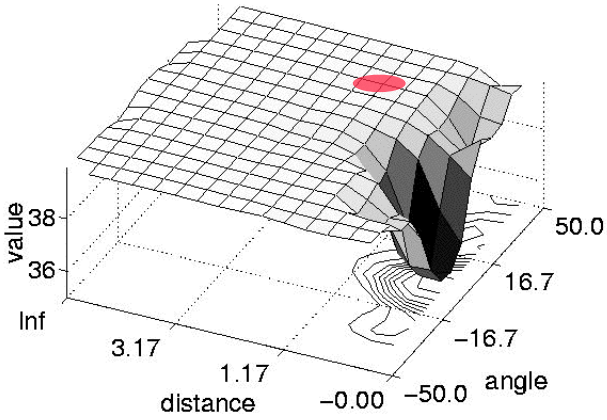


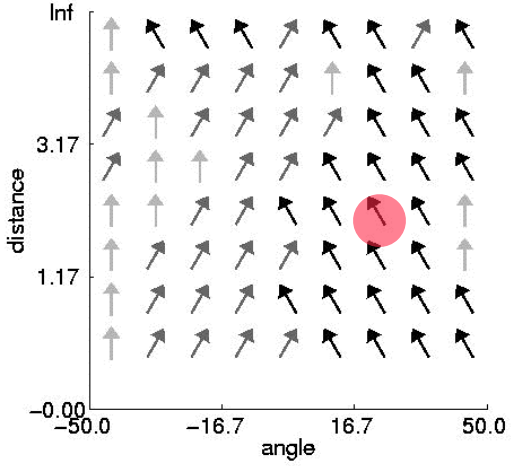
Sidewalk



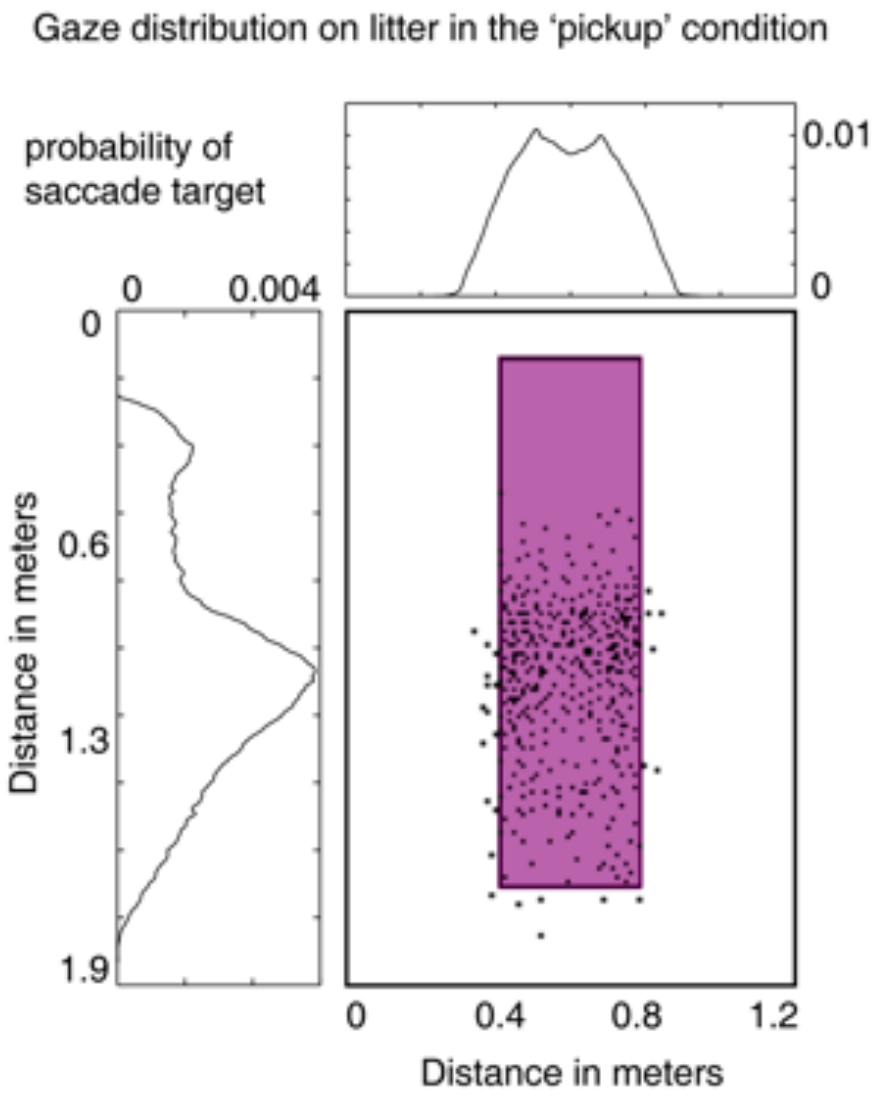


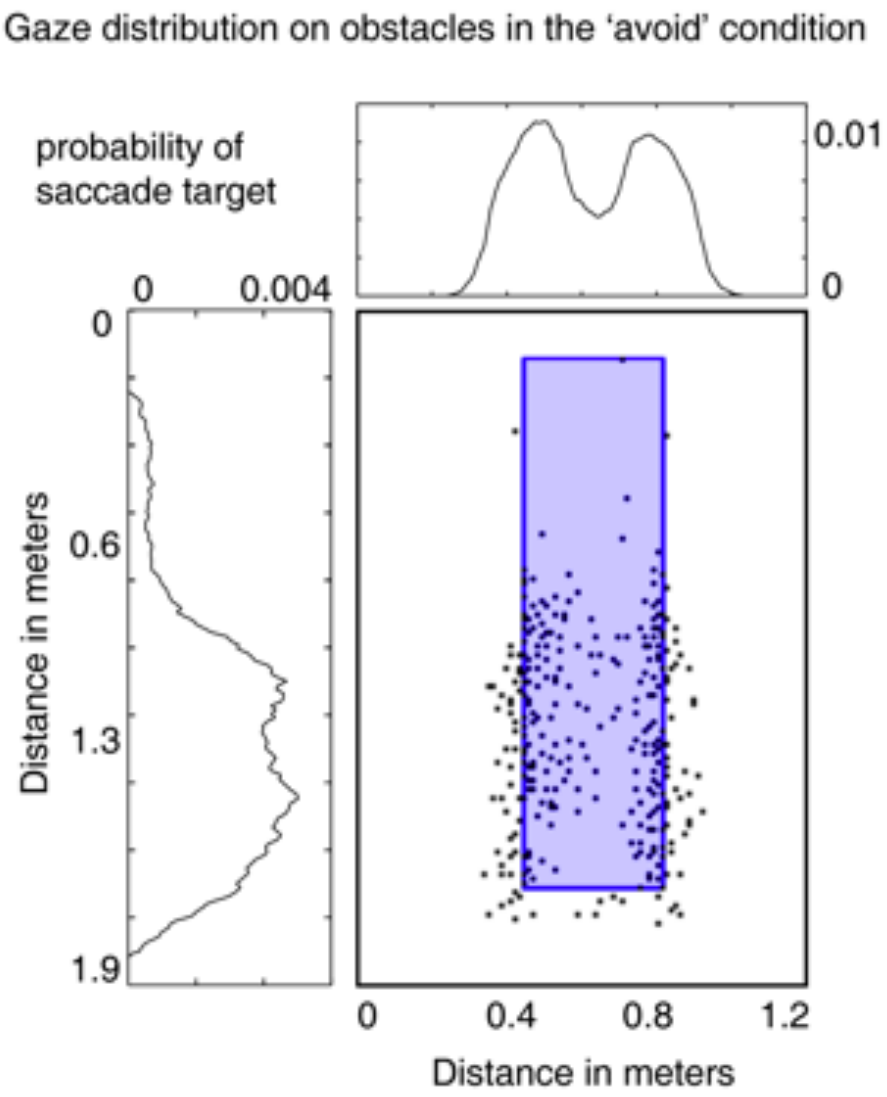
Obstacles





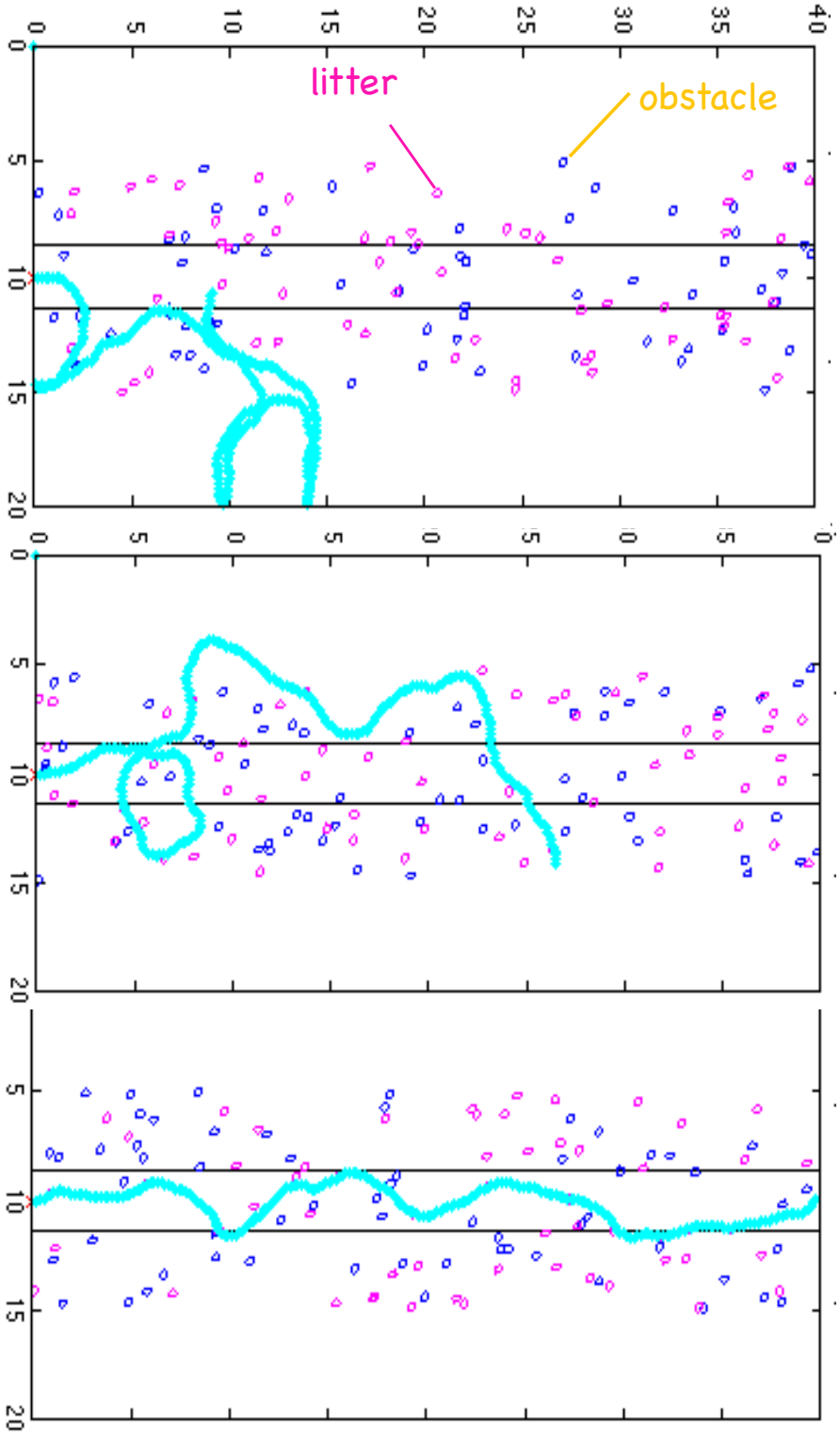
Human gaze data: Task determines fixation point





Rothkopf and Ballard *Journal of Vision* (2005) 5, 1-3

Overhead view of trajectory



Initial performance

After 100 episodes

After 150 episodes

Summary of Part 1:

In building human cognitive models, embodiment constraints allow the factorization of the overall problem into manageable pieces

2

Modules that compete for visual sensing

Visual sensing
Polling versus interrupts

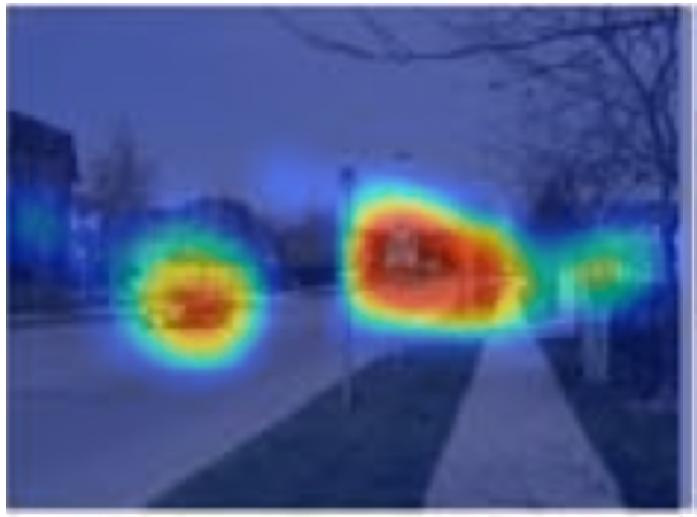
Interrupts: Eye gaze models invoke bottom up saliency; image features are the source of gaze locations

Polling: Modular RL is top-down agenda driven; active modules compete For the use of gaze to reduce uncertainty

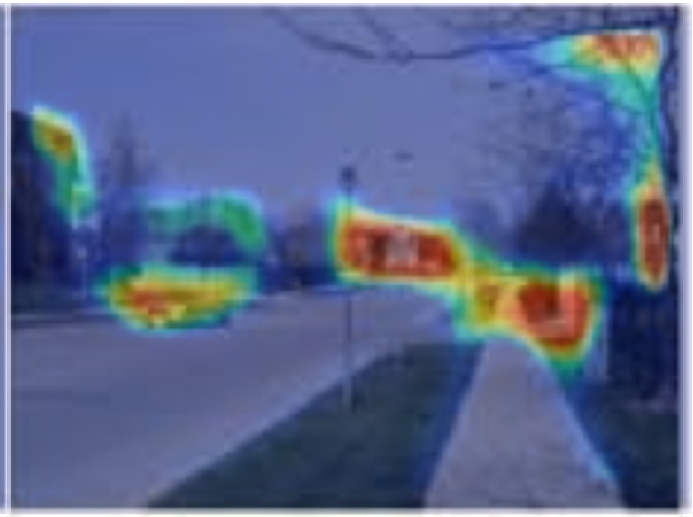
What directs gaze? One answer: Saliency

Salient locations are a good candidate for fixations but by and large,
The task dominates

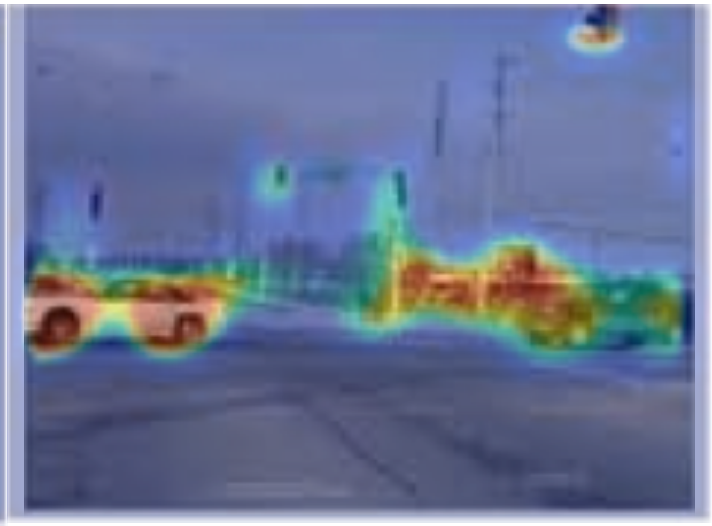
Image



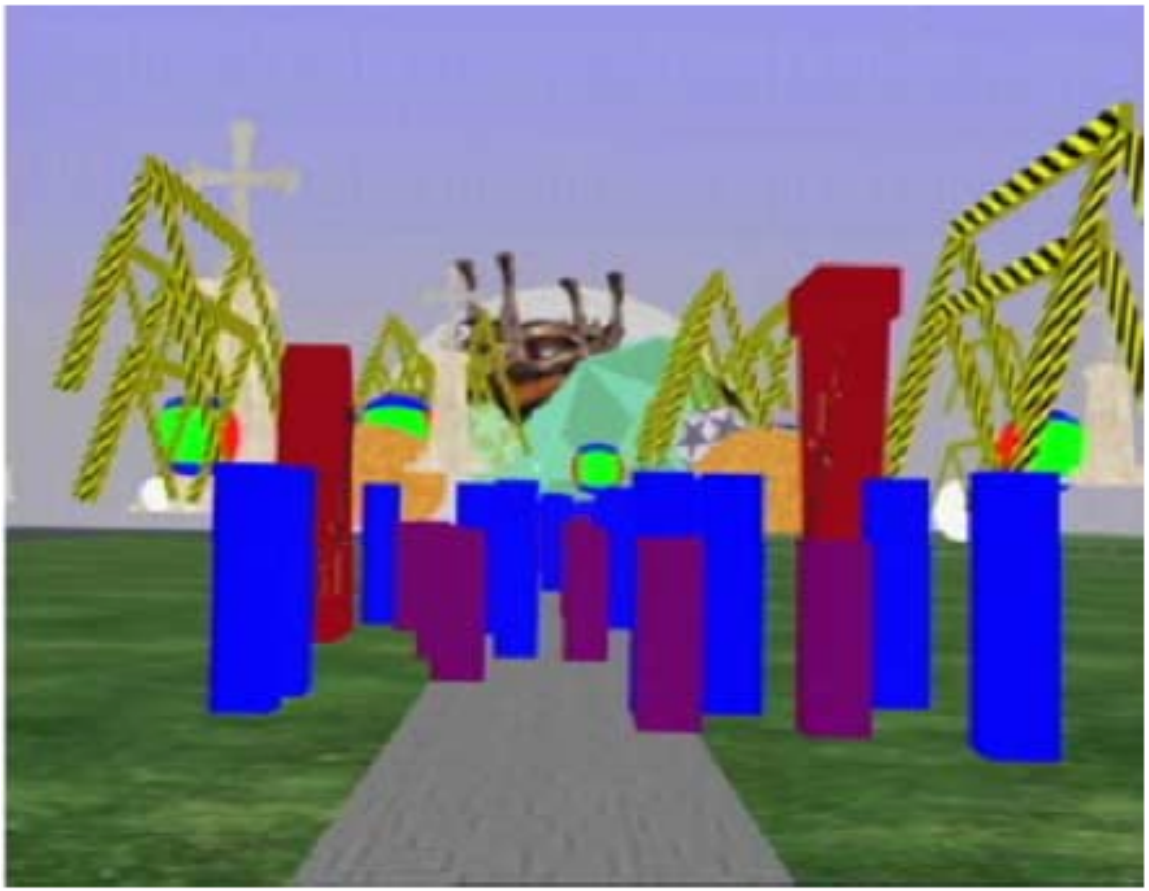
Human data



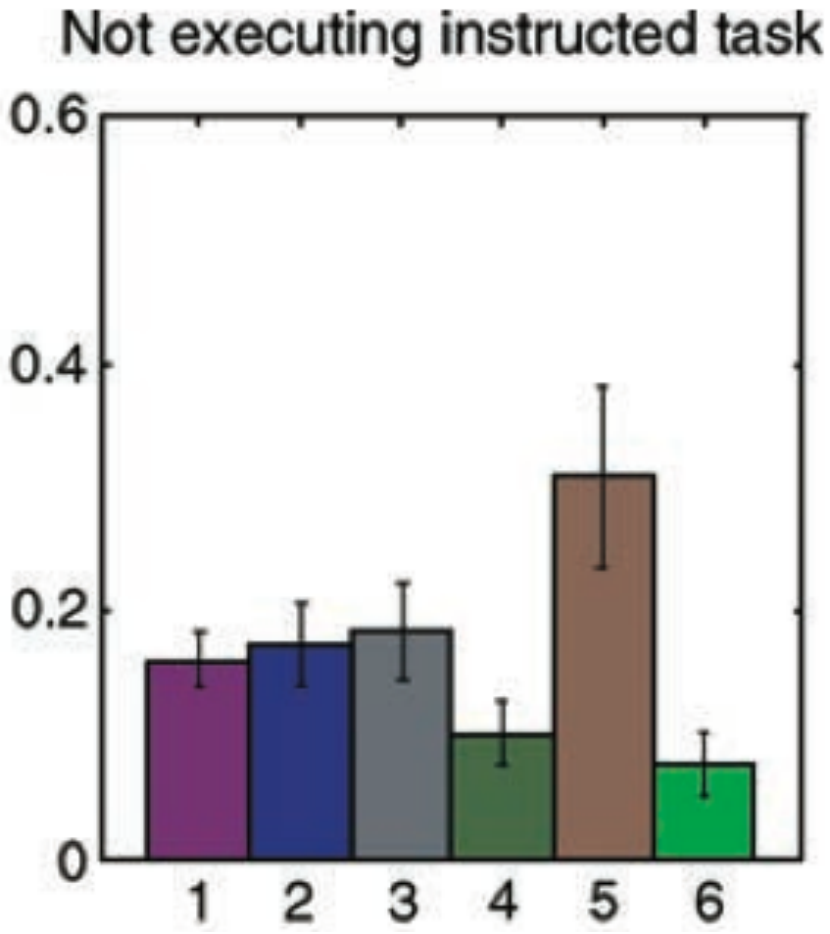
Saliency



Humans perform the sidewalk navigation task with visual distractions



Human gaze data



Litter

Obstacle

Footpath

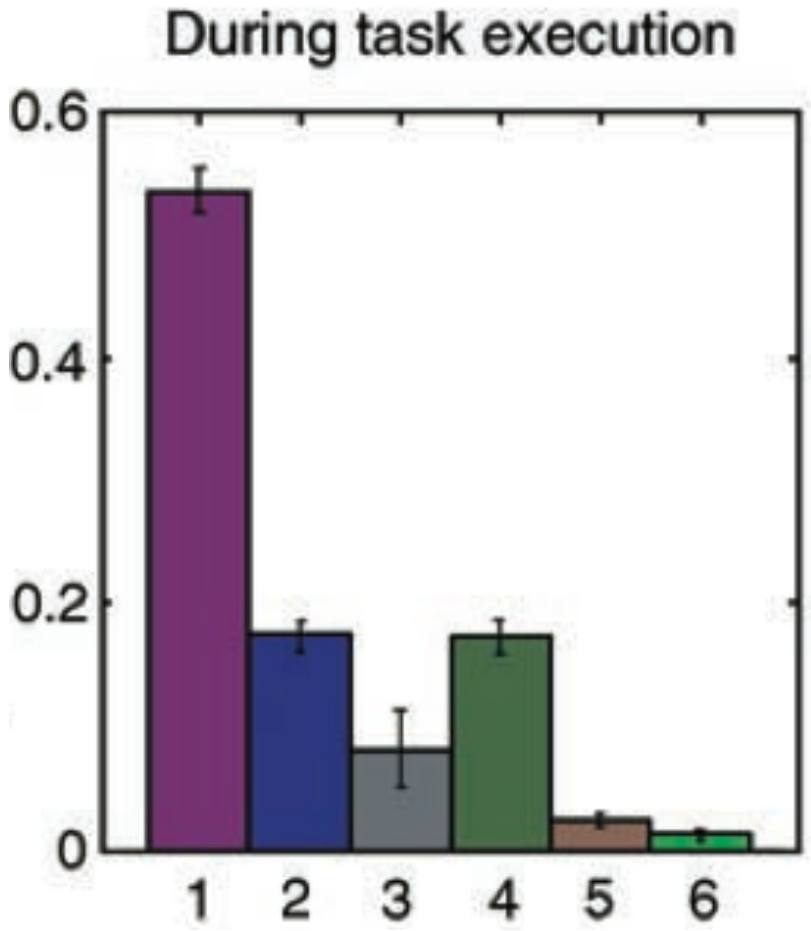
Grass

Other

grass

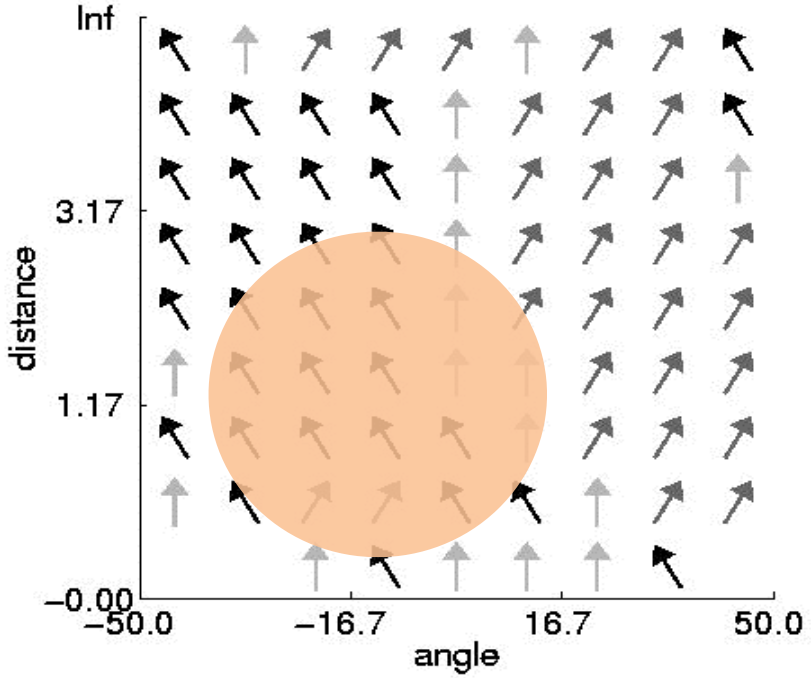
Human gaze data

Substitution



Which module should get the gaze vector?

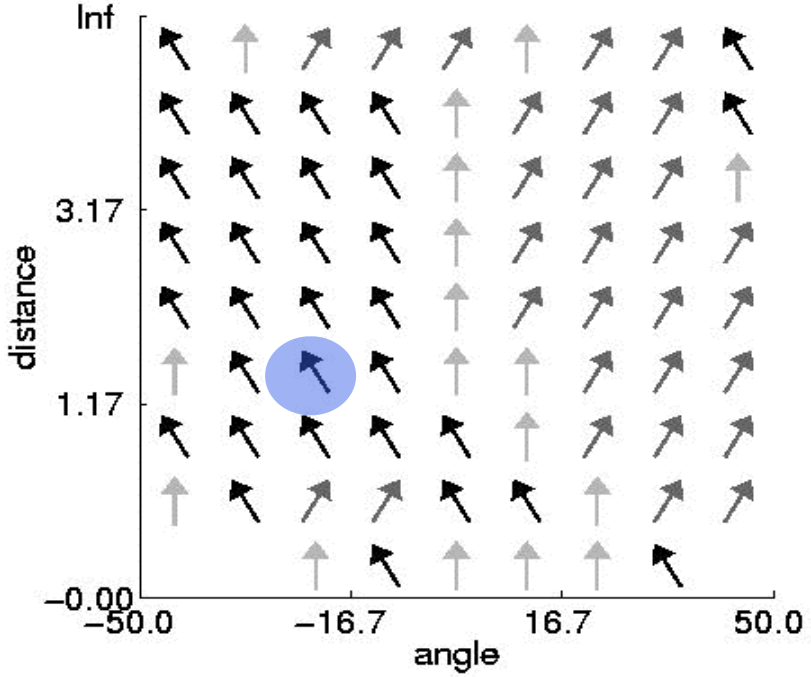
Before Observation



Without gaze, the location in state space becomes increasingly uncertain.

Which module should get the gaze vector?

After Observation



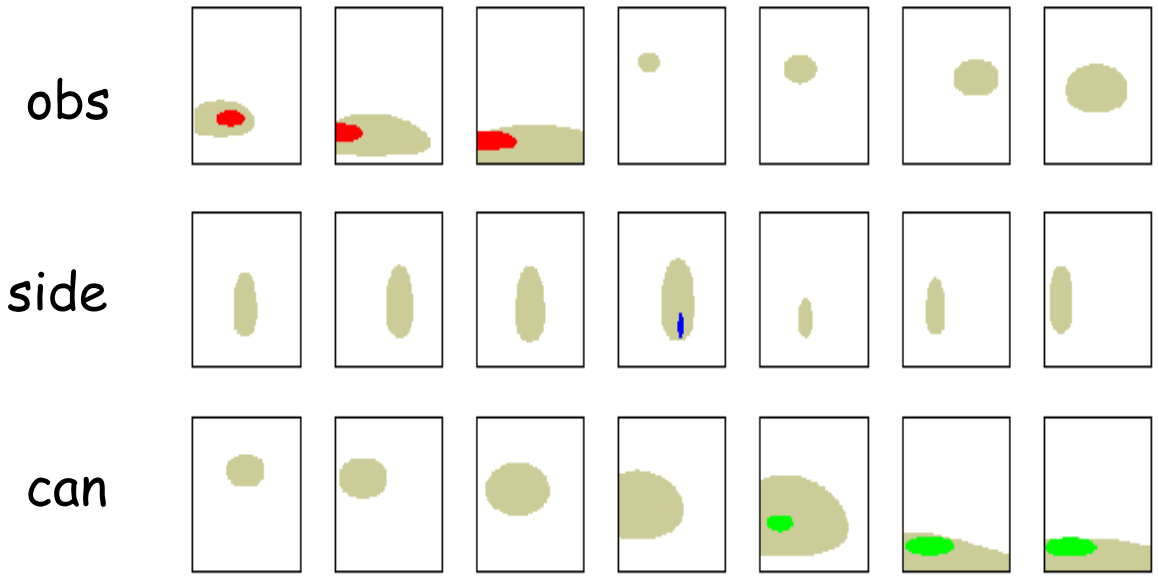
Making a measurement with a fixation reduces this uncertainty

Sprague's hypothesis

$$\text{gain}_b = E\left[\max_a \left(Q_b(s_b, a) + \sum_{i \in B, i \neq b} Q_i^E(s_i, a)\right)\right] - \sum_i Q_i^E(s_i, a_E)$$

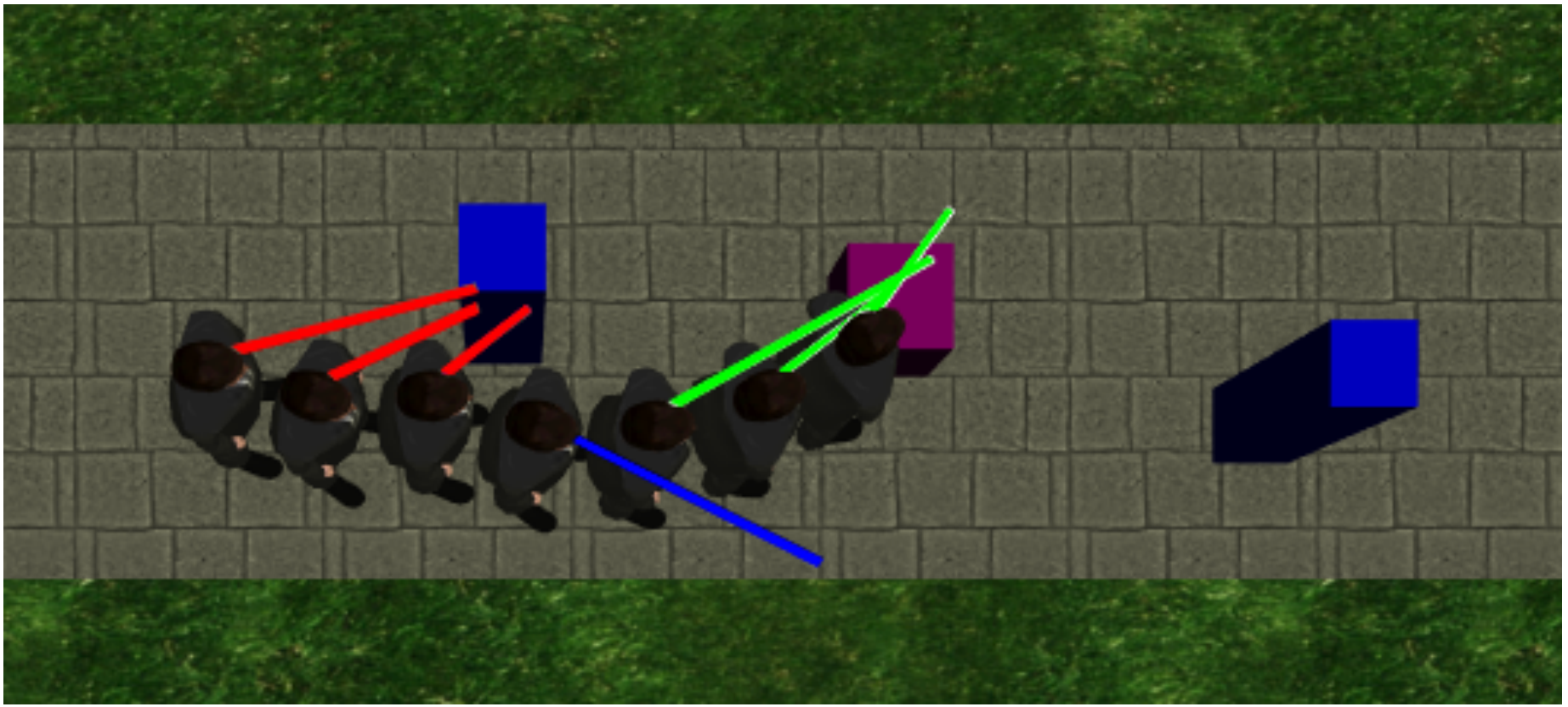
Reward where *b* updates using gaze

Average reward no updates

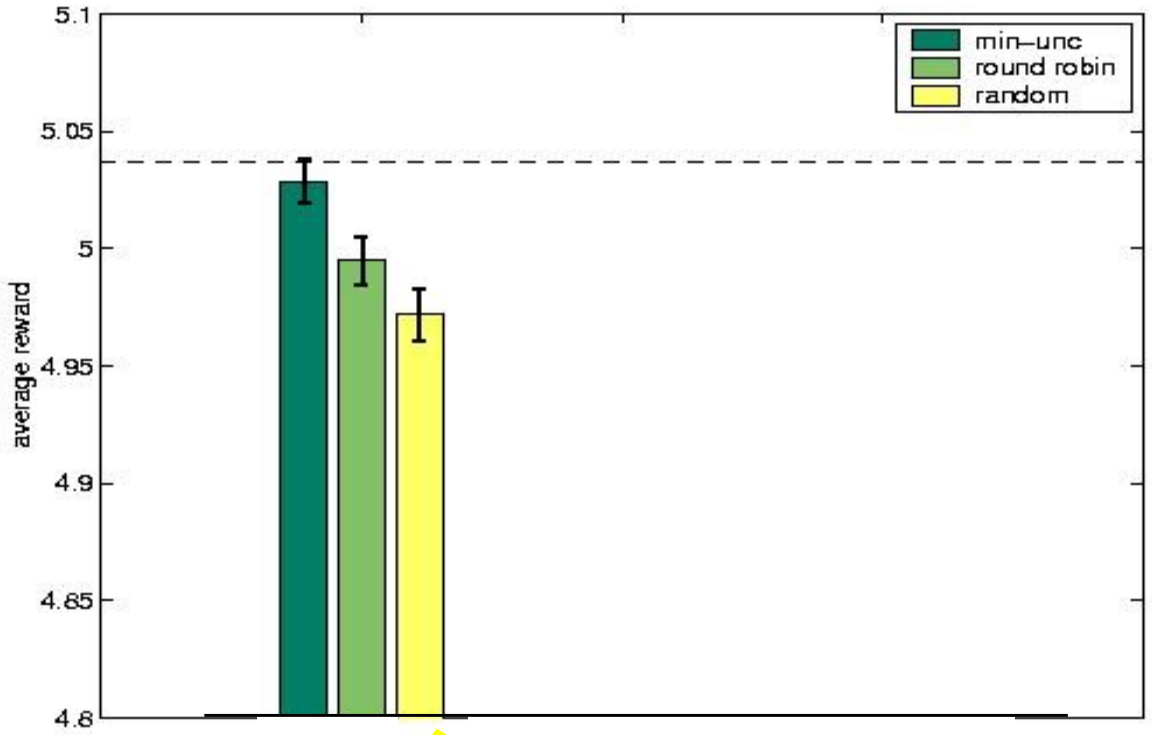


A priori uncertainty

A posteriori uncertainty



Performance Comparison



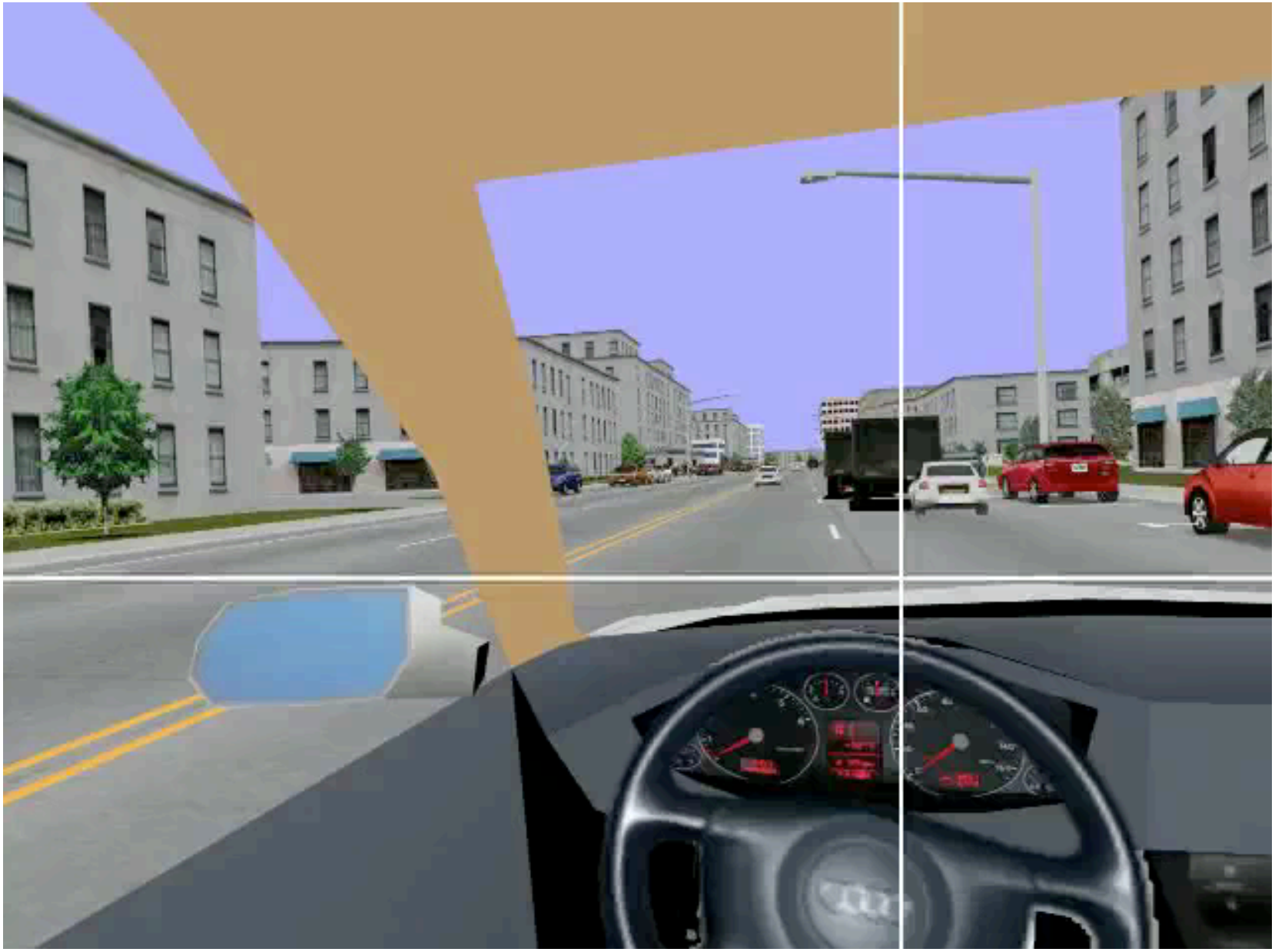
Gaze services random task

Gaze services tasks in order

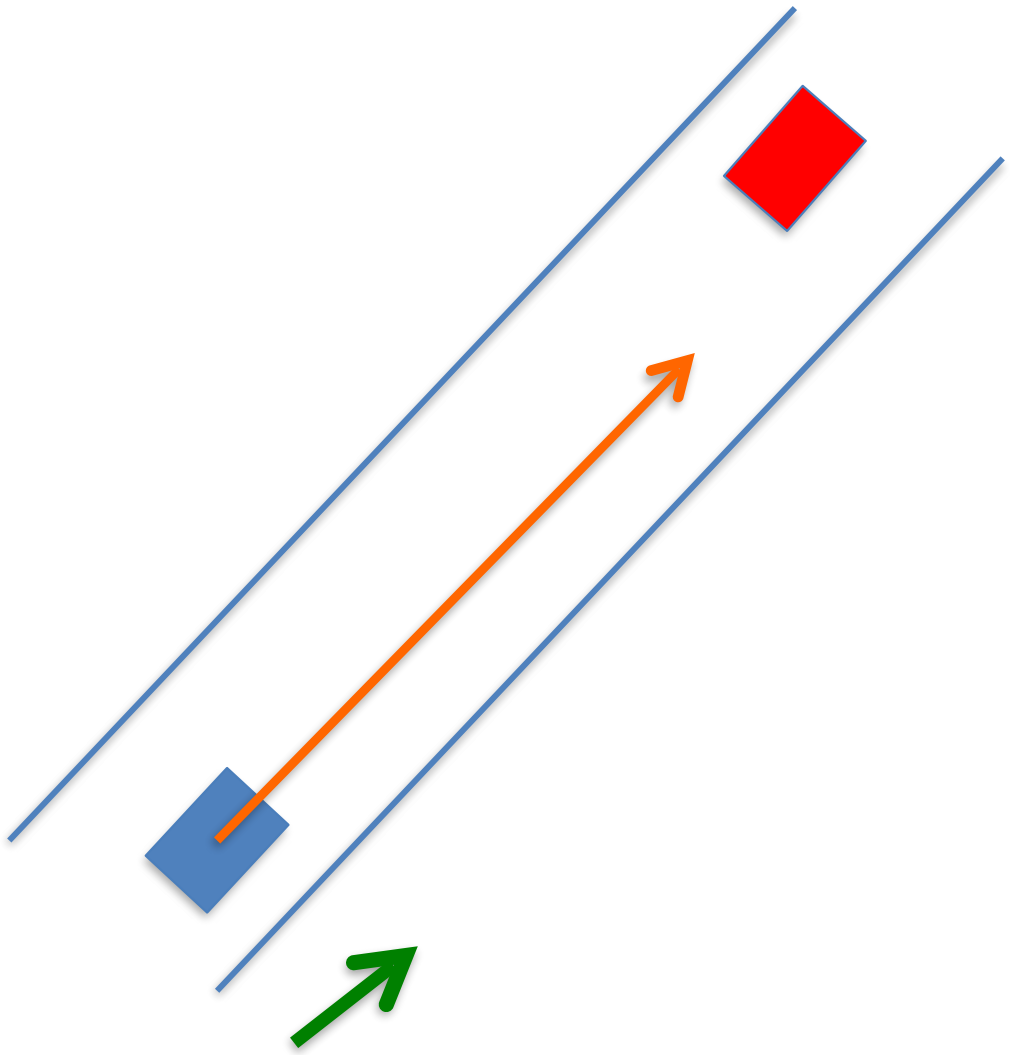
Gaze services task w highest reduction in reward uncertainty

Sprague et al ACM TAP 2007

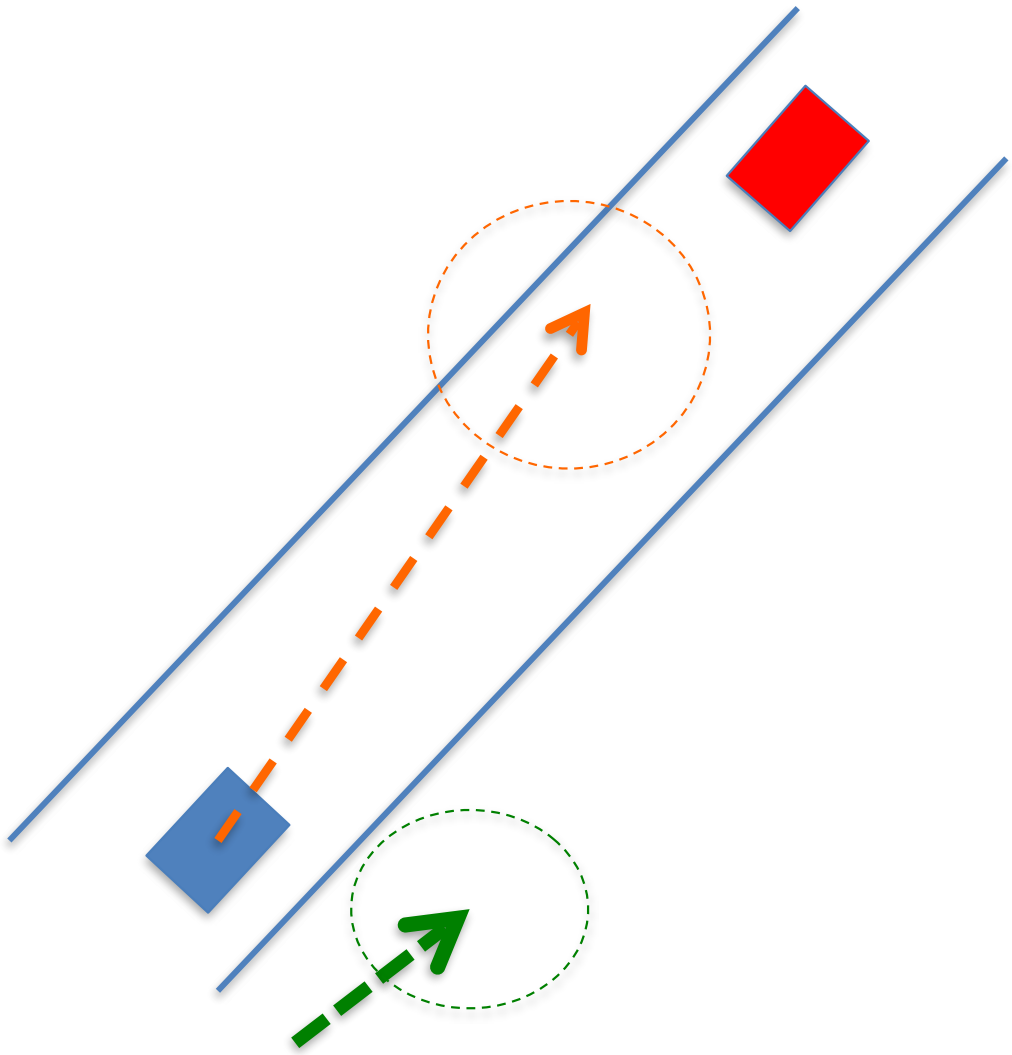
Predicting Human Visuomotor Behavior in a Driving Task



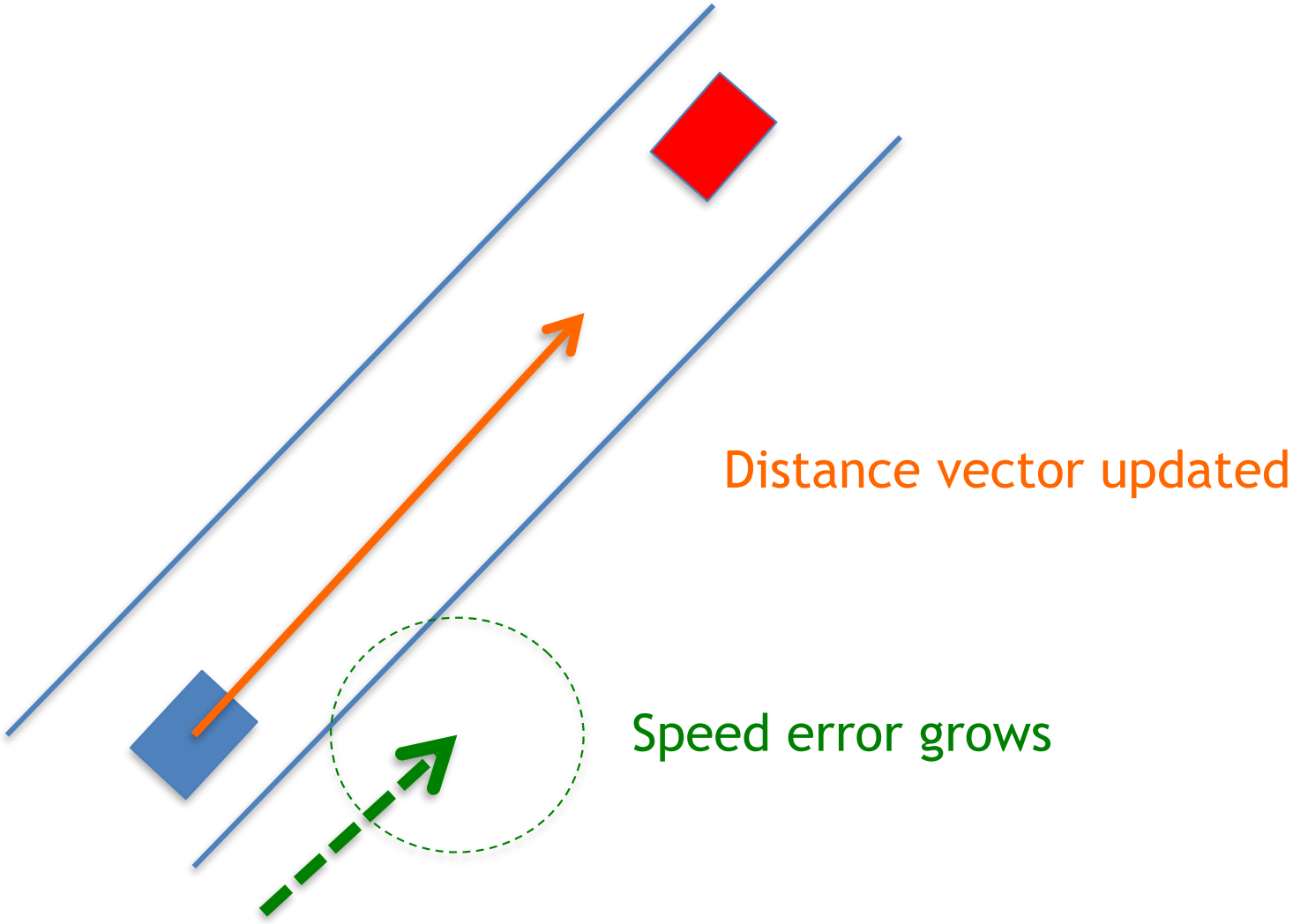
Tasks: Follow a lead car and maintain a speed



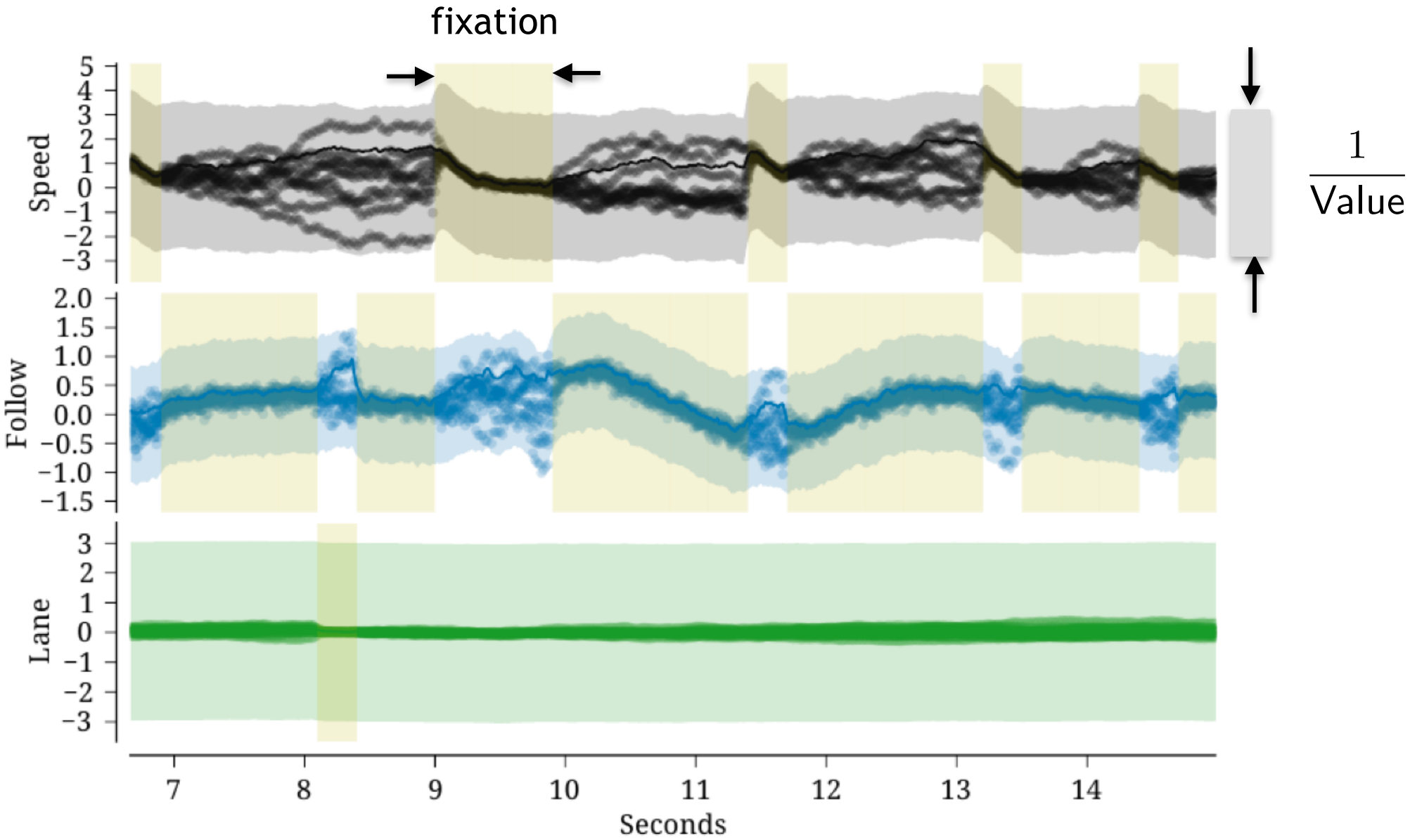
PD Control: In absence of gaze, set point estimates drift



Barrier models control which estimate is updated

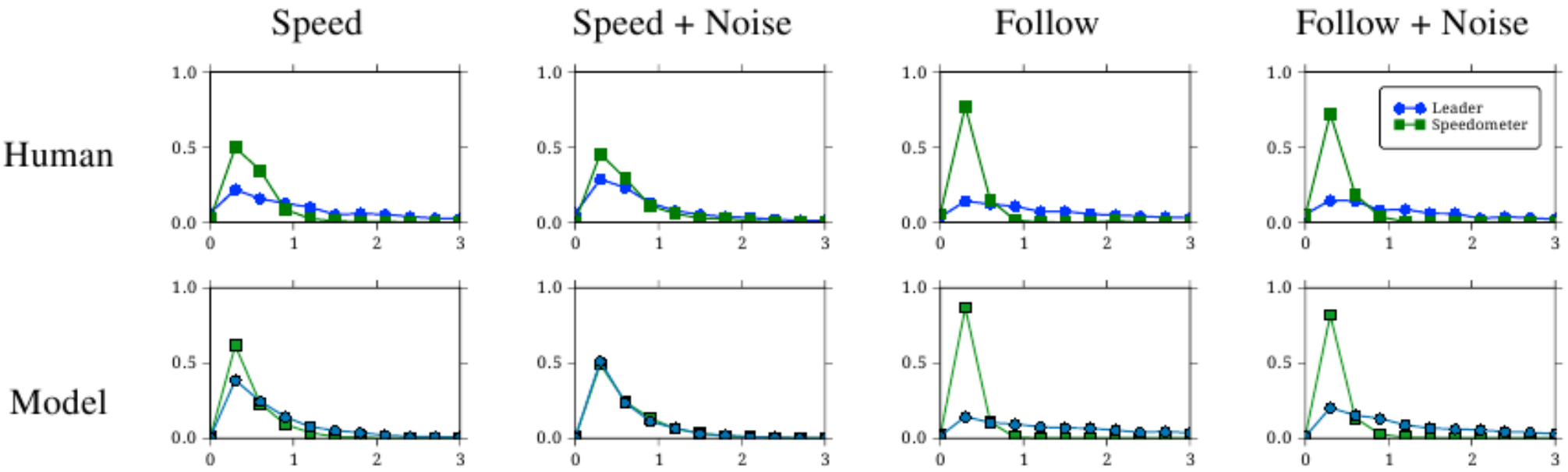


Multi-particle boundary decision model



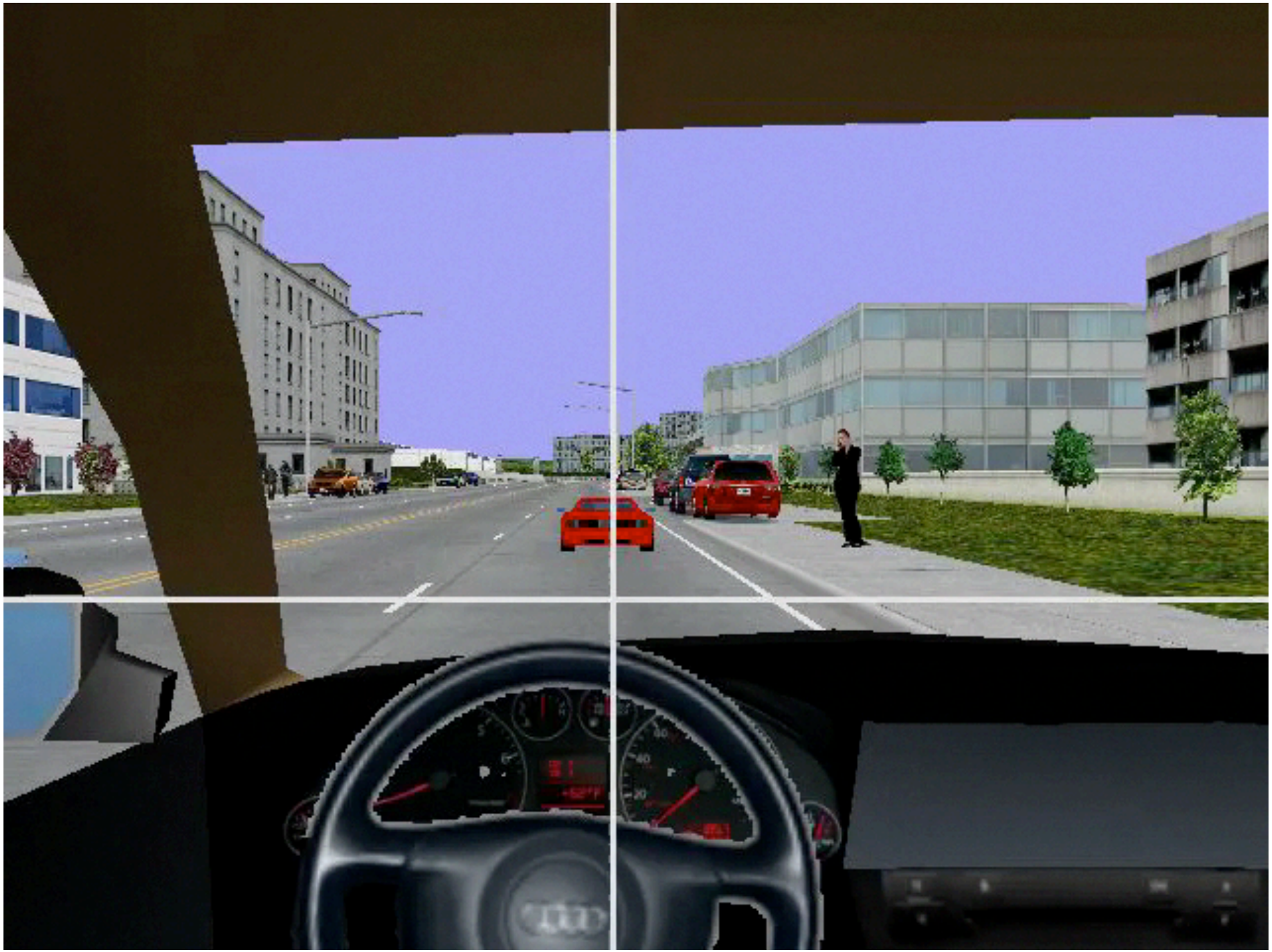
Johnson et al *Proc. Roy Soc B* 2014

KL divergence shows a very close match of fixation interval distributions between human data and boundary drift model



(a) KL =0.12 (b) KL = 0.09 (c) KL = 0.05 (d) KL = 0.10

The model gets to drive the car



Summary of Part 2:

RL modules provide an account of gaze allocation

Without gaze, internal state drifts

Module with maximum expected reward gets the gaze vector

3
Modules that solve the credit assignment problem

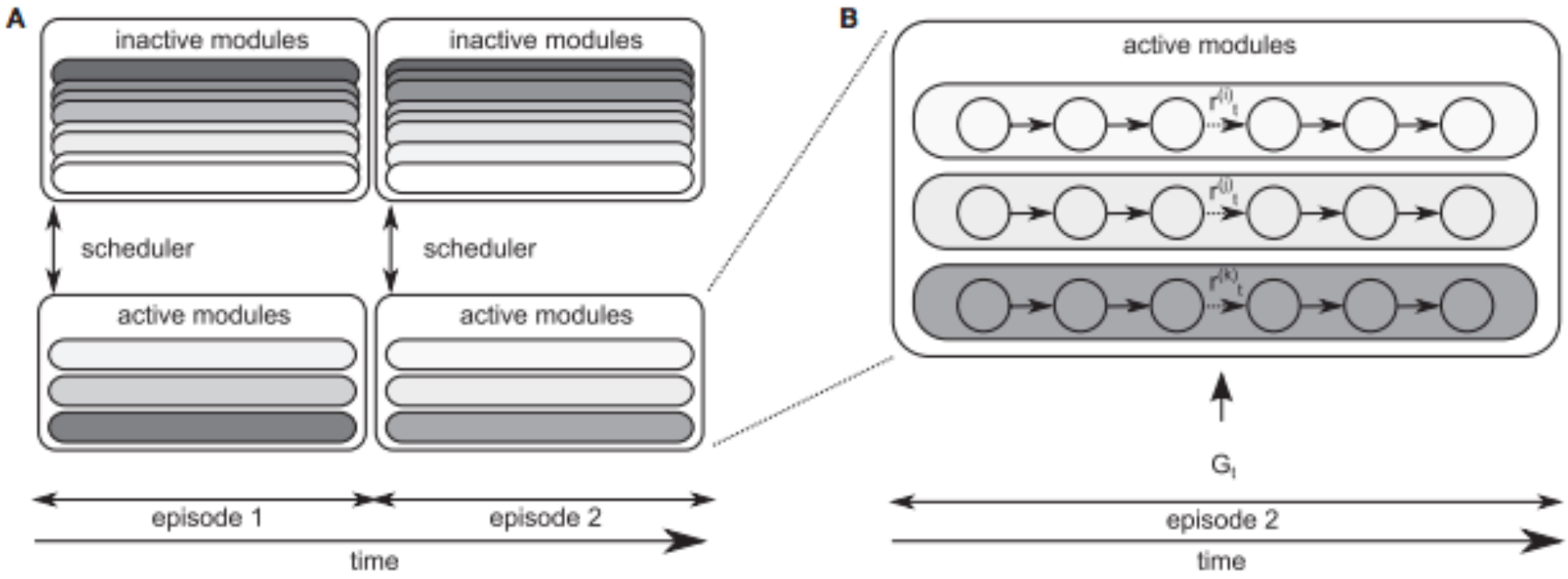
Credit Assignment: how do modules learn their share of reward?

Situations where you need to calibrate reward:

Solving a problem with a new set of modules

Using a set of modules where only the global reward is known and the problem is to assign credit appropriately

Modules architecture: Episodes



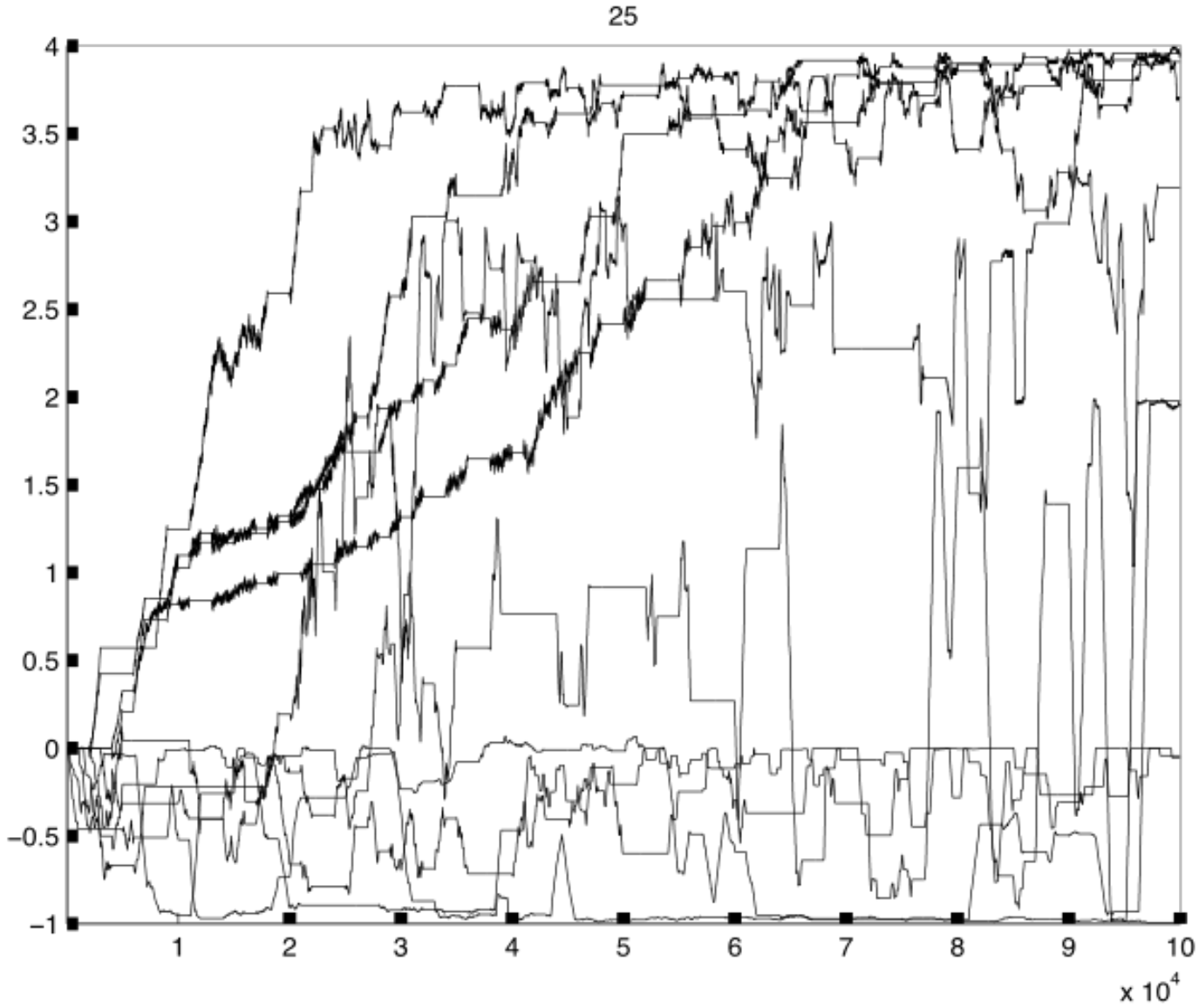
Is this really a problem?

In a multi-agent RL setting, a Kalman fitering model allows agents to learn the correct policy even when they can't solve the credit assignment problem.

It is a problem when *different sets of modules* are used, since small errors in reward estimates can lead to large offsets in state values and consequently inconsistent policies

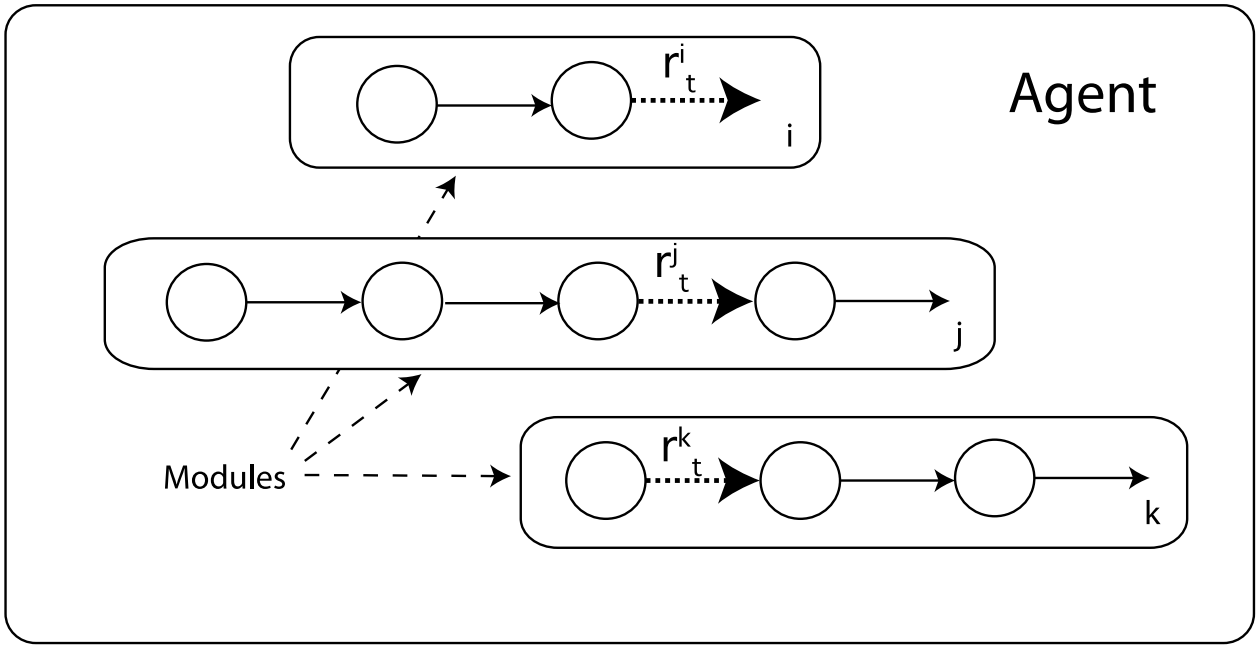
$$V'^{\pi}(s) = \frac{c}{1-\gamma} + V^{\pi}(s)$$

Ten modules working in subsets fail to compute learn correctly on a simple grid world task



Module rewards can be found by bootstrapping

Key assumption: a module knows the estimated rewards of its co-activated module set



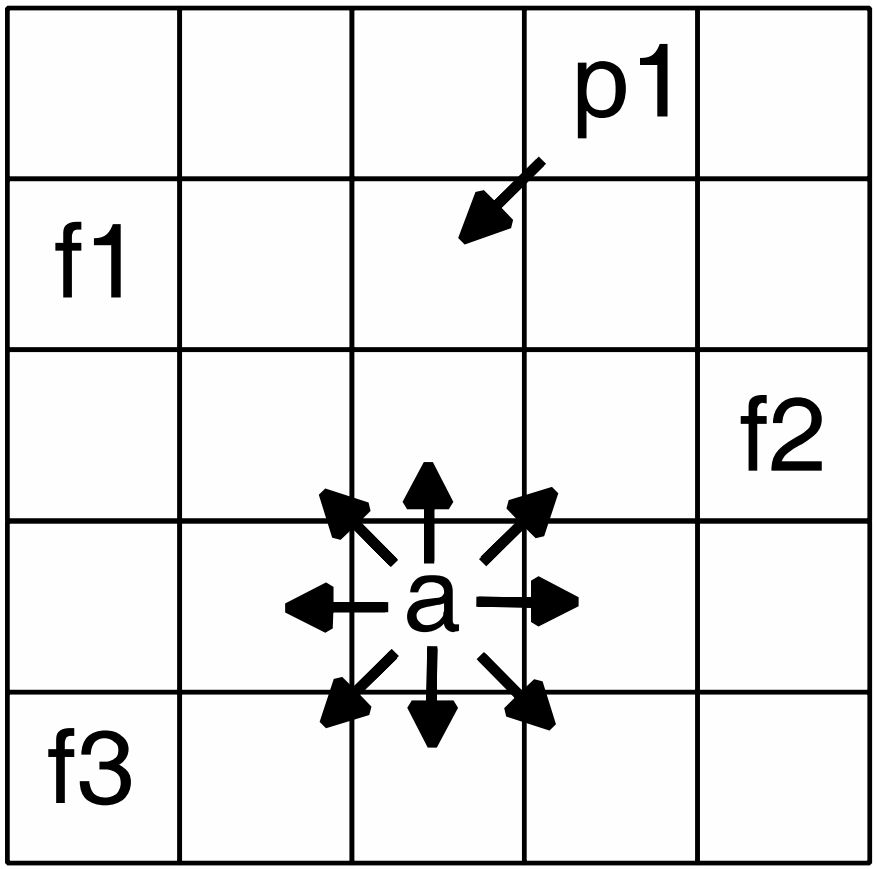
G_t

Module's estimate

Estimate of the other modules

$$\hat{r}^{(i)} \leftarrow (1-\beta)\hat{r}^{(i)} + \beta\Big(G - \sum_{j\in\mathcal{M}, j\neq i} \hat{r}^{(j)}\Big)$$

Prey & Predators problem

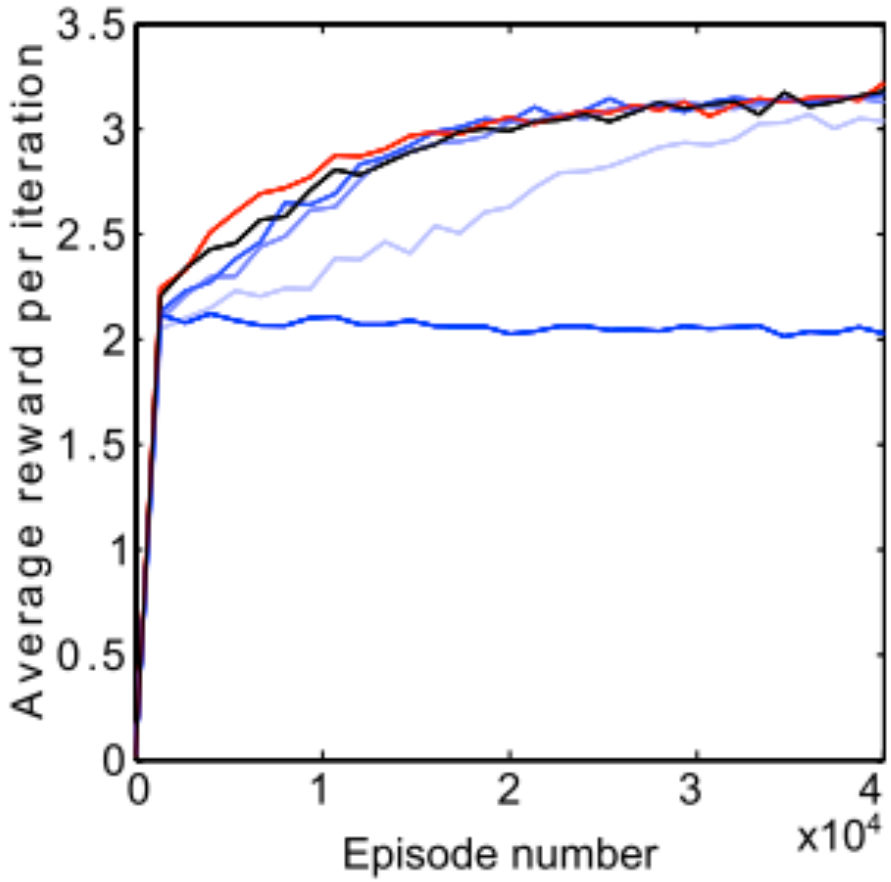


15 food sources

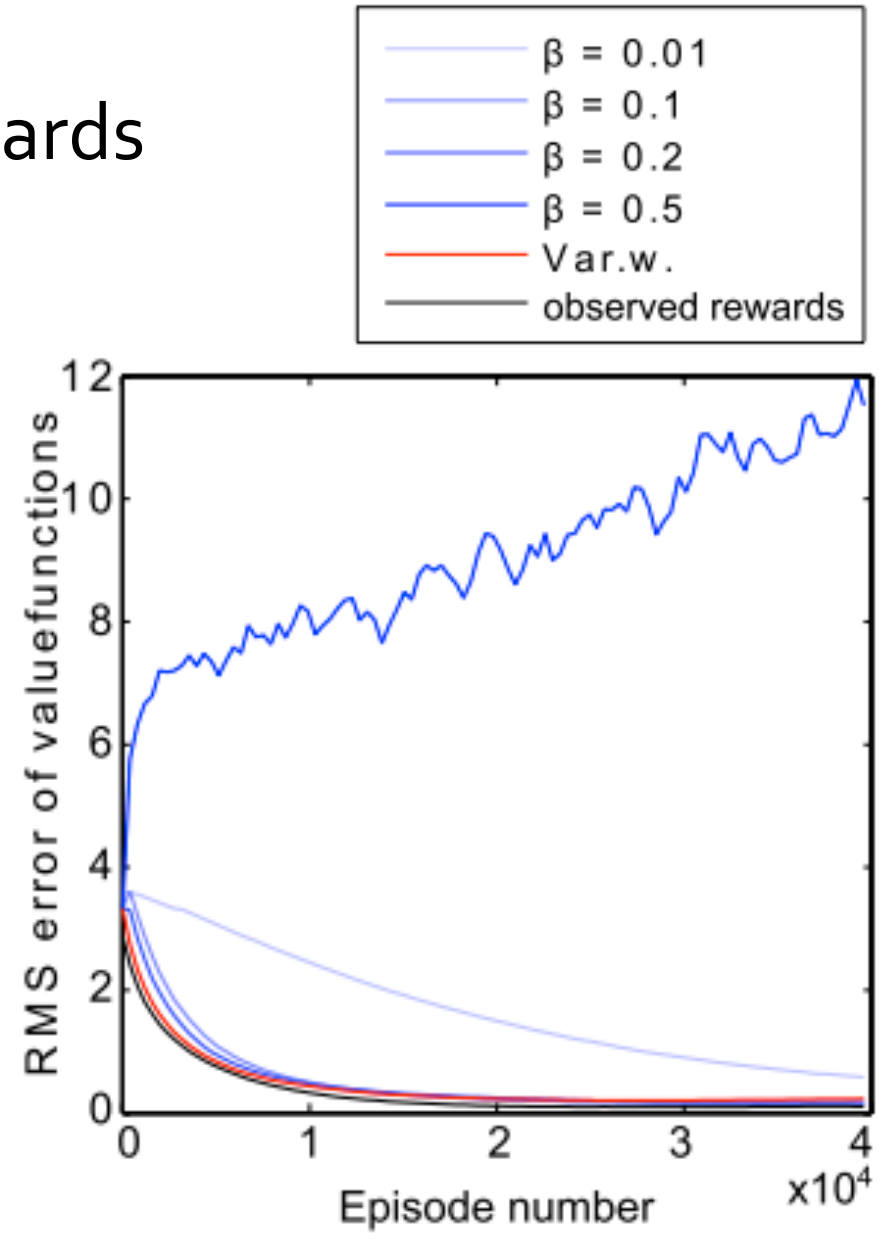
5 Predators

Adapted from Singh and Cohn

Results of learning module rewards



Total reward



Error between observed rewards and estimates

Variance weighted estimates are superior

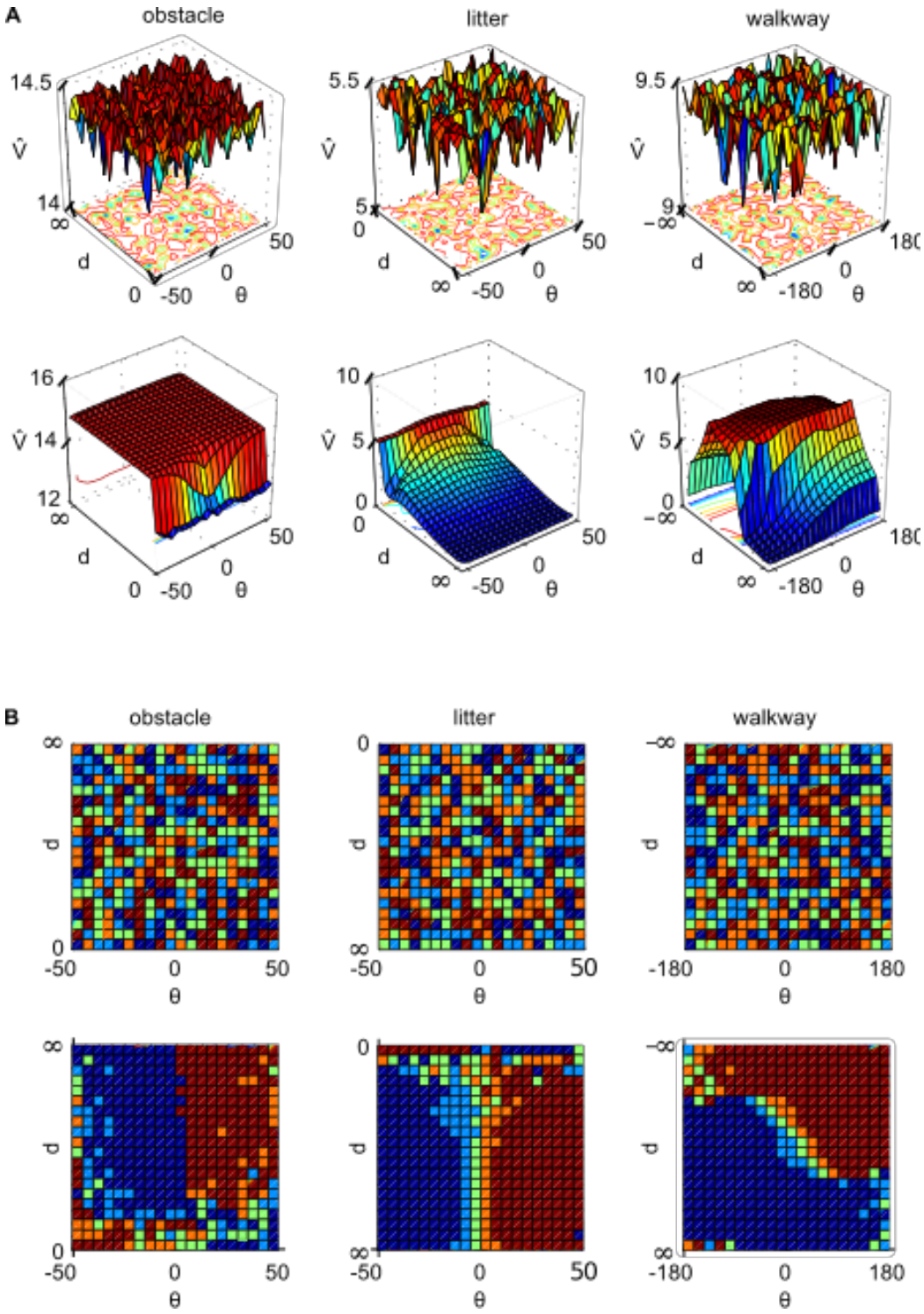
Problem: A module that has a high variance estimate can corrupt others' estimates

Solution: weight estimates by the variances in reward

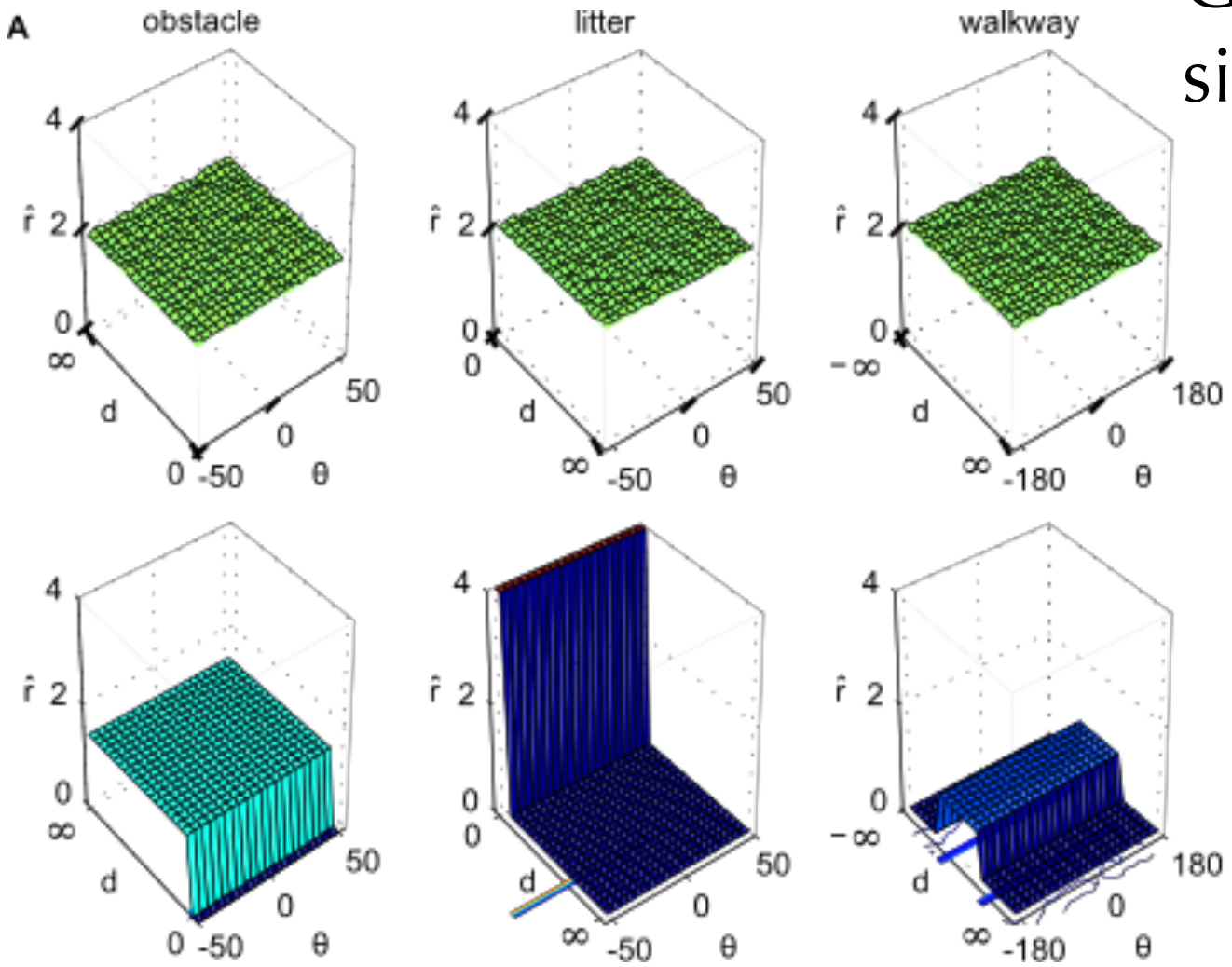
$$\beta_i = \frac{(\sigma^{(i)})^2}{\sum_{j=1}^{N} (\sigma^{(j)})^2}$$

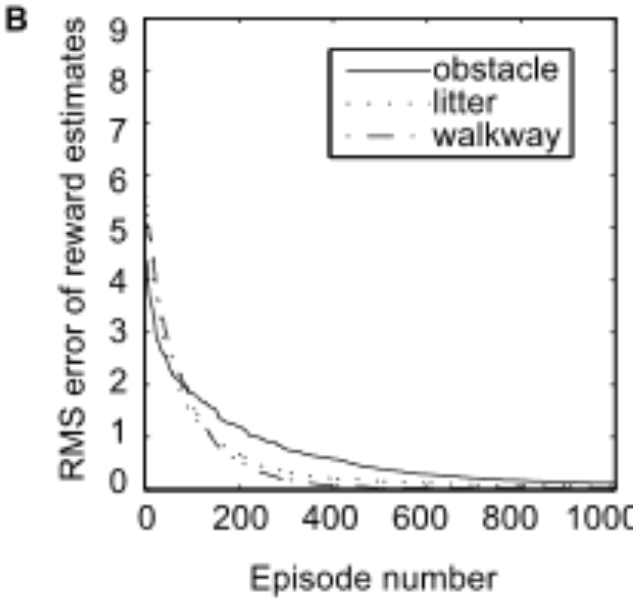
$$= \frac{(\sigma^{(i)})^2}{\sum_{j \neq i}^{N} (\sigma^{(j)})^2 + (\sigma^{(i)})^2}$$

Learning Q tables
And policies given only
Global reward in sidewalk venue



Learning rewards given only
Global reward for sidewalk venue





Summary of **3**: Global Reward Calibration

Modules work in different subsets each episode.

Modules know each other's reward estimates

Modules weight estimates by variance estimates

4

Modules and Inverse Reinforcement Learning

Bayesian IRL

Observations from a subject:

$$O_{\mathcal{X}} = \{(s_1, a_1), (s_2, a_2) \ldots (s_k, a_k)\}$$

Probability of seeing the individual states/actions, given the reward, is a product of individual probs.

$$\begin{aligned} Pr_{\mathcal{X}}(O_{\mathcal{X}}|\boldsymbol{R}) &= Pr_{\mathcal{X}}((s_1, a_1)|\boldsymbol{R}) Pr_{\mathcal{X}}((s_2, a_2)|\boldsymbol{R}) \\ &\quad \ldots Pr_{\mathcal{X}}((s_k, a_k)|\boldsymbol{R}) \end{aligned}$$

Bayesian IRL

Use Bayes thm

$$Pr_{\mathcal{X}}(\boldsymbol{R}|O_{\mathcal{X}}) = \frac{Pr_{\mathcal{X}}(O_{\mathcal{X}}|\boldsymbol{R})P_R(\boldsymbol{R})}{Pr(O_{\mathcal{X}})}$$

Bayesian IRL

Lets make the assumption of the following PDF:

$$Pr_{\mathcal{X}}(O_{\mathcal{X}}|\boldsymbol{R}) = \frac{1}{Z} e^{\alpha_{\mathcal{X}} E(O_{\mathcal{X}}, \boldsymbol{R})}$$

And furthermore:

$$E(O_{\mathcal{X}}, \boldsymbol{R}) = \sum_i Q^*(s_i, a_i, \boldsymbol{R})$$

So that:

$$Pr_{\mathcal{X}}((s_i, a_i)|\boldsymbol{R}) = \frac{1}{Z_i} e^{\alpha_{\mathcal{X}} Q^*(s_i, a_i, \boldsymbol{R})}$$

Fast Bayesian IRL

Modular rewards are scaled

Search the space of scale factors to make observed data most probable

$$P(s_j, a_j | Q^*) = \frac{1}{Z} e^{\sum_i c_i Q(s_j^{*(i)}, a_j)}$$

Bayesian IRL

Priors that emphasize sparse rewards:

$$P_{Laplace}(\boldsymbol{R}(s) = r) = \frac{1}{2\sigma} e^{-\frac{|r|}{2\sigma}}, \forall s \in S$$

$$P_{Beta}(\boldsymbol{R}(s) = r) = \frac{1}{(\frac{r}{R_{max}})^{\frac{1}{2}}(1 - \frac{r}{R_{max}})^{\frac{1}{2}}}, \forall s \in S$$

Bayesian IRL Algorithm

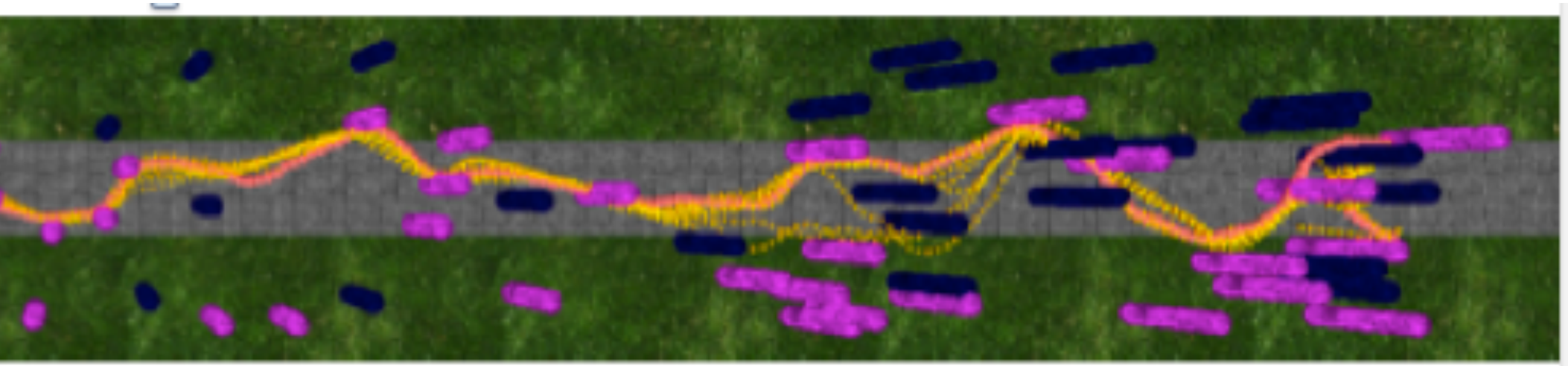
Ramachandran &Amir

Algorithm `PolicyWalk`(Distribution P, MDP M, Step Size δ)

1. Pick a random reward vector $\boldsymbol{R} \in \mathbb{R}^{|S|}/\delta$.
2. $\pi := \texttt{PolicyIteration}(M, \boldsymbol{R})$
3. Repeat
   (a) Pick a reward vector $\tilde{\boldsymbol{R}}$ uniformly at random from the neighbours of $\boldsymbol{R}$ in $\mathbb{R}^{|S|}/\delta$.
   (b) Compute $Q^{\pi}(s, a, \tilde{\boldsymbol{R}})$ for all $(s, a) \in S, A$.
   (c) If $\exists (s, a) \in (S, A), Q^{\pi}(s, \pi(s), \tilde{\boldsymbol{R}}) < Q^{\pi}(s, a, \tilde{\boldsymbol{R}})$
      i. $\tilde{\pi} := \texttt{PolicyIteration}(M, \tilde{\boldsymbol{R}}, \pi)$
      ii. Set $\boldsymbol{R} := \tilde{\boldsymbol{R}}$ and $\pi := \tilde{\pi}$ with probability $\min\{1, \frac{P(\tilde{\boldsymbol{R}}, \tilde{\pi})}{P(\boldsymbol{R}, \pi)}\}$
      Else
      i. Set $\boldsymbol{R} := \tilde{\boldsymbol{R}}$ with probability $\min\{1, \frac{P(\tilde{\boldsymbol{R}}, \pi)}{P(\boldsymbol{R}, \pi)}\}$
4. Return $\boldsymbol{R}$

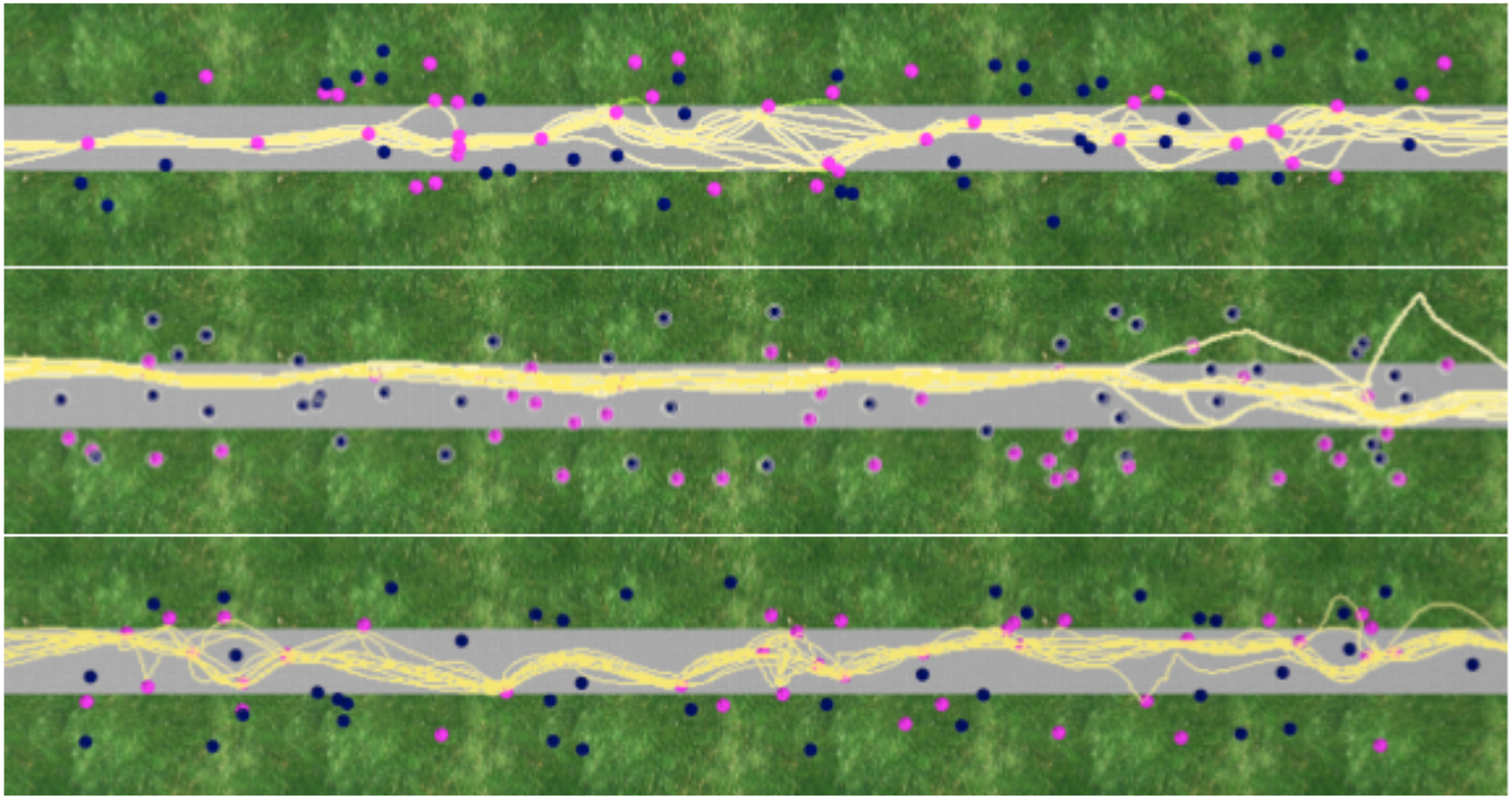
Fast Inverse Reinforcement learning

Choose the module weighting that makes the observed data the most probable



- Original trajectory (data)
- Generated trajectories using recovered weight estimates

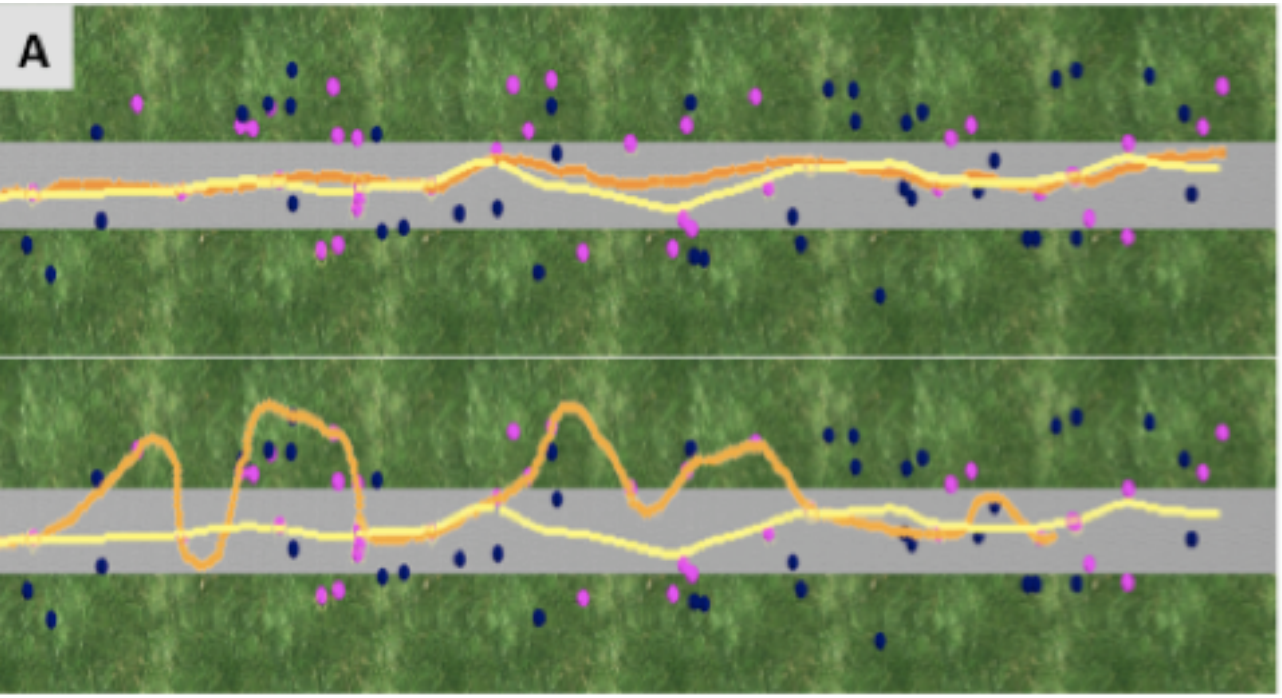
Human performance data shows unexpected regularities

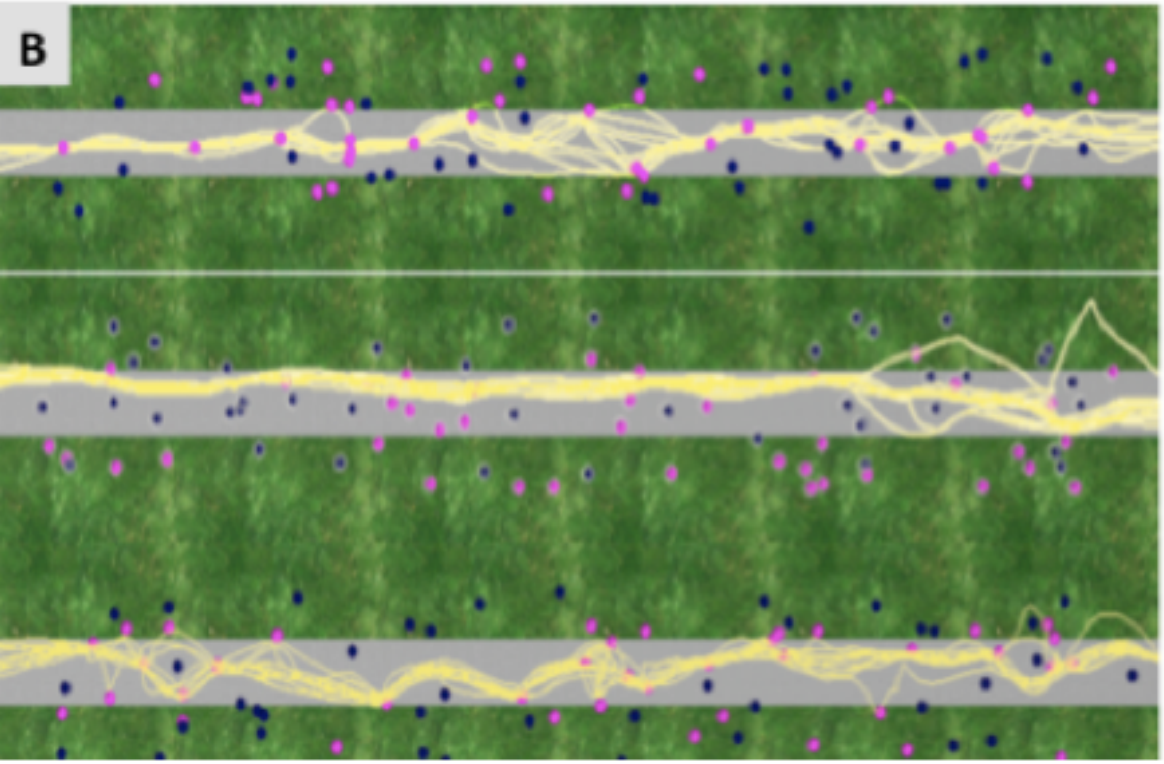


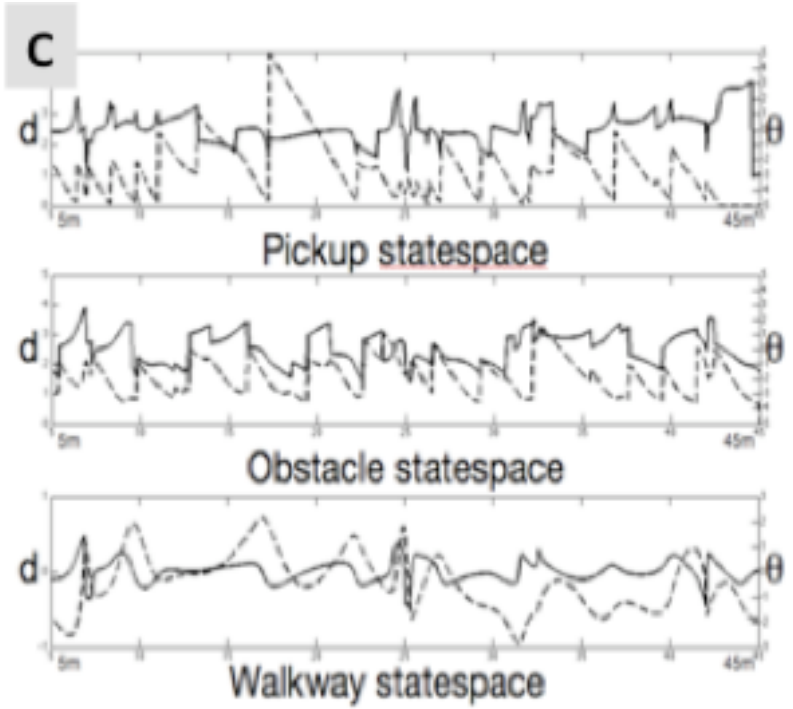
pickup

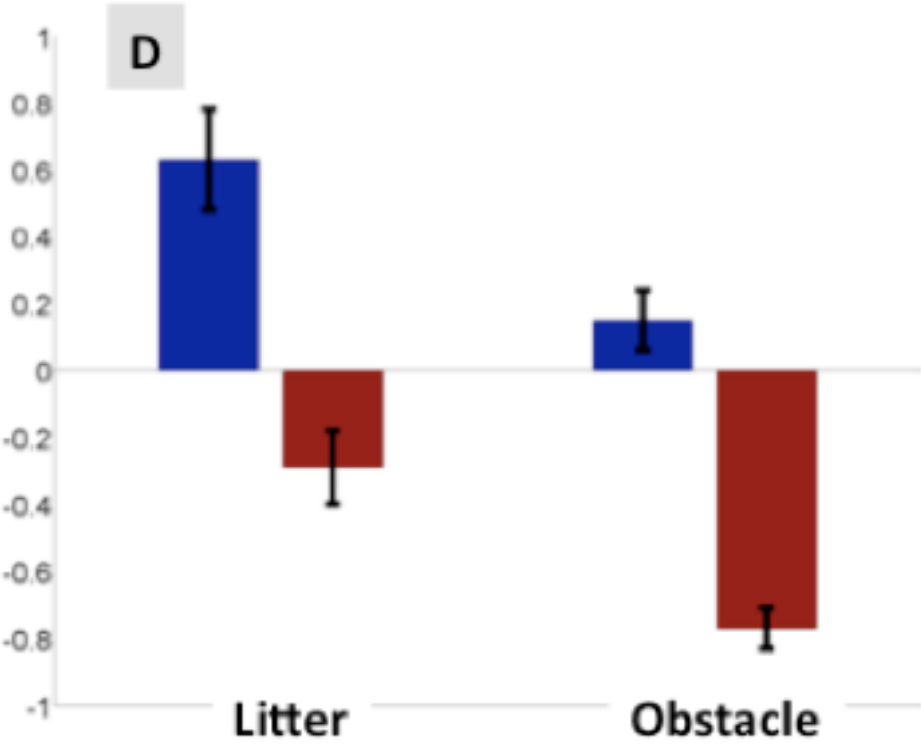
avoid

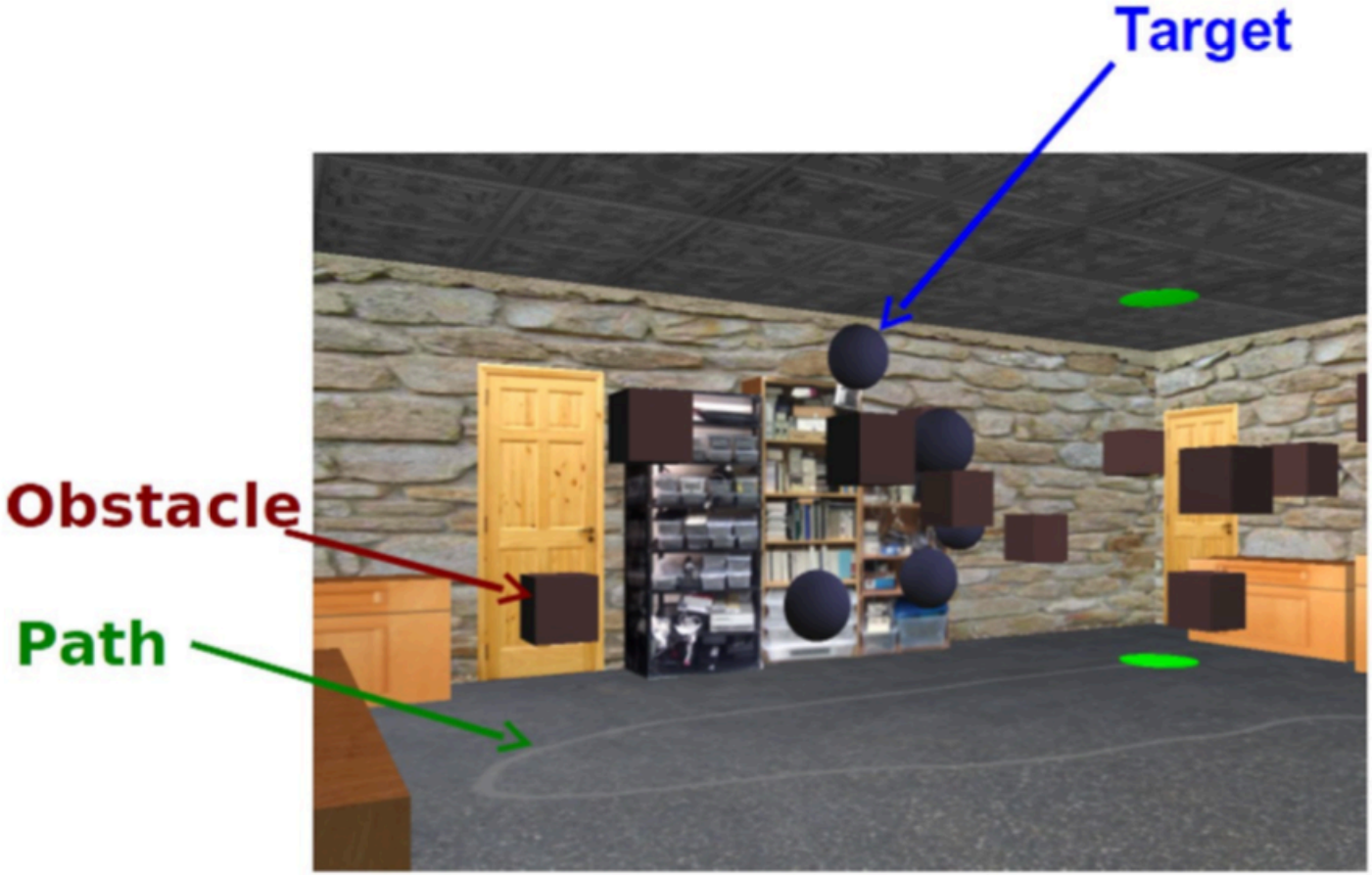
pickup
+
avoid

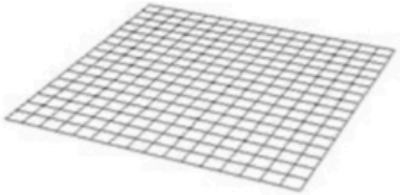

A


B

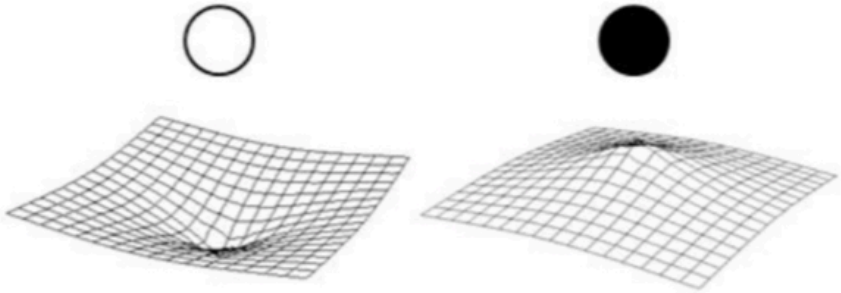

C
d
θ
5m
45m
Pickup statespace
Obstacle statespace
Walkway statespace


D
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Litter
Obstacle

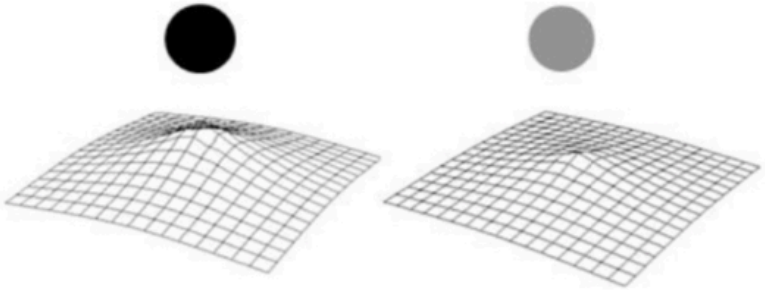

Target
Obstacle
Path



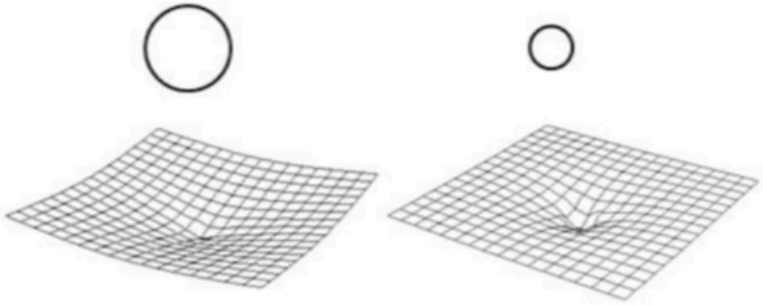
(A) Initial value surface



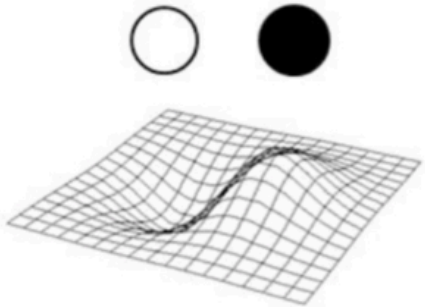
(B) Postive reward vs. negative reward



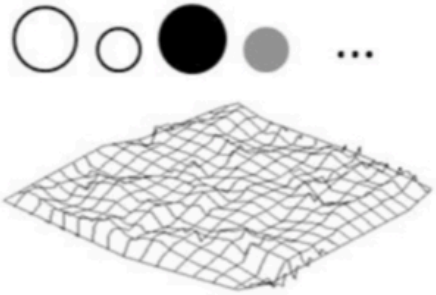
(C) Large reward vs. small reward



(D) Large discount factor γ vs. small γ

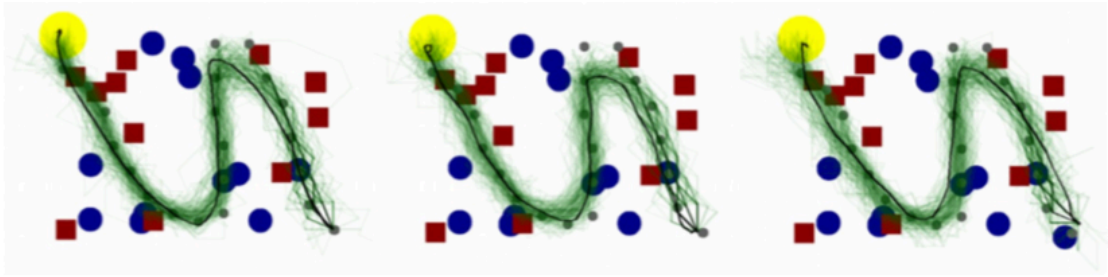


(E) Composed surface with two objects

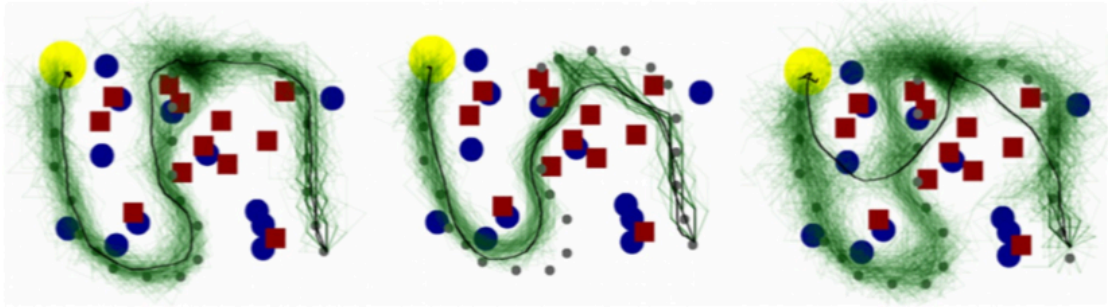


(F) Composed surface with many objects

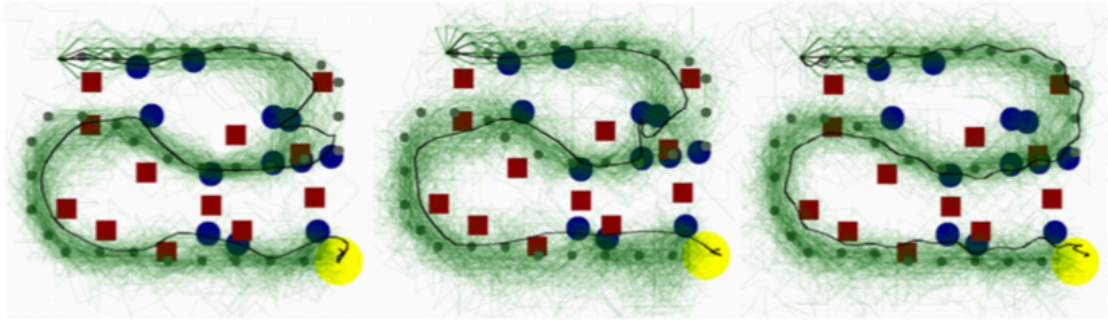
Fig 3. Maximum likelihood modular inverse reinforcement learning. (A) From an observed trajectory (a sequence of state-action pairs), the goal of modular IRL is to recover the underlying value surface. (B) Maximum likelihood IRL assumes that the probability of observing a particular action (red) in a state is proportional to its Q-value among all possible actions as in Eq (5).



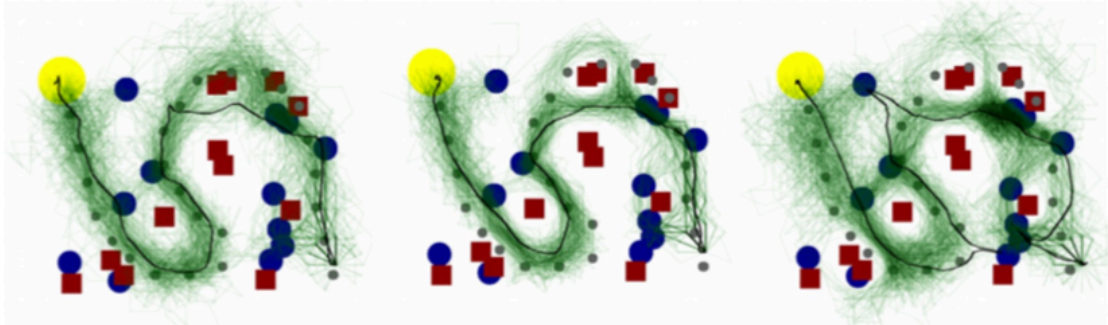
(A) $r : (0.11, 0.38, \mathbf{0.52})$ $\gamma : (0.85, 0.99, 0.88)$ **(B)** $r : (0.11, 0.25, \mathbf{0.63})$ $\gamma : (0.88, 0.99, 0.94)$ **(C)** $r : (0.07, 0.39, \mathbf{0.54})$ $\gamma : (0.82, 0.99, 0.94)$

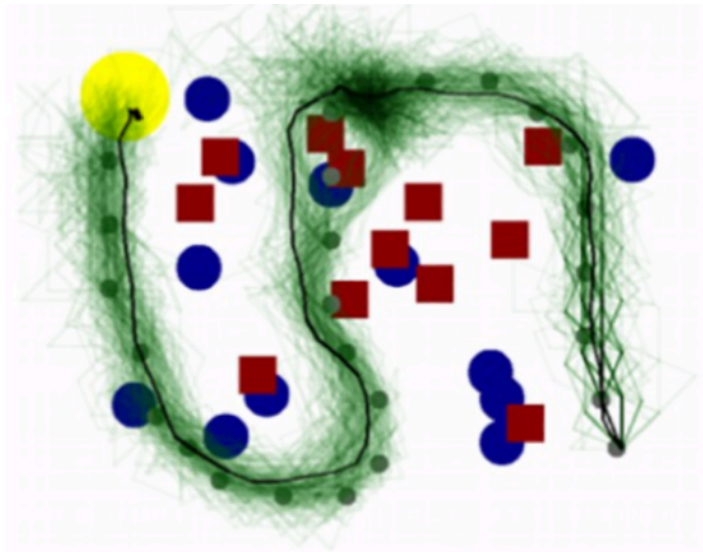


(D) $r : (0.00, \mathbf{0.57}, \mathbf{0.43})$ $\gamma : (0.00, 0.69, 0.91)$ **(E)** $r : (0.02, \mathbf{0.57}, \mathbf{0.41})$ $\gamma : (0.99, 0.59, 0.97)$ **(F)** $r : (0.06, \mathbf{0.75}, \mathbf{0.19})$ $\gamma : (0.95, 0.60, 0.88)$

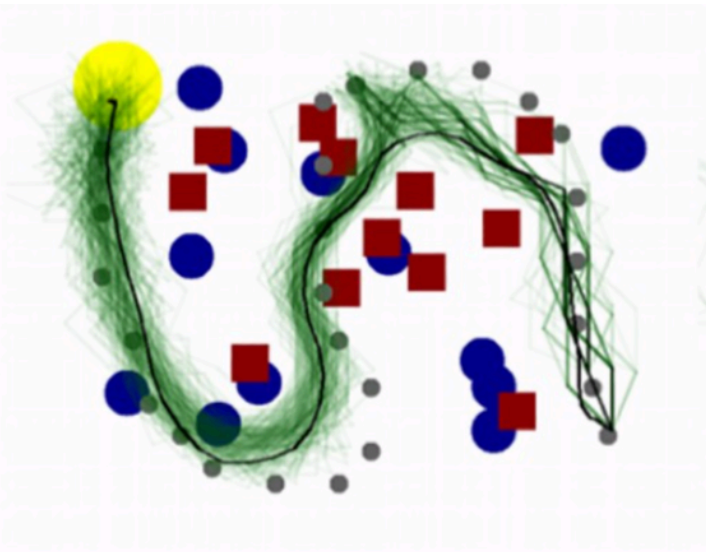


(G) $r : (\mathbf{0.30}, 0.22, \mathbf{0.48})$ $\gamma : (0.77, 0.72, 0.89)$ **(H)** $r : (\mathbf{0.27}, 0.29, \mathbf{0.45})$ $\gamma : (0.69, 0.73, 0.96)$ **(I)** $r : (\mathbf{0.30}, 0.17, \mathbf{0.52})$ $\gamma : (0.76, 0.99, 0.89)$

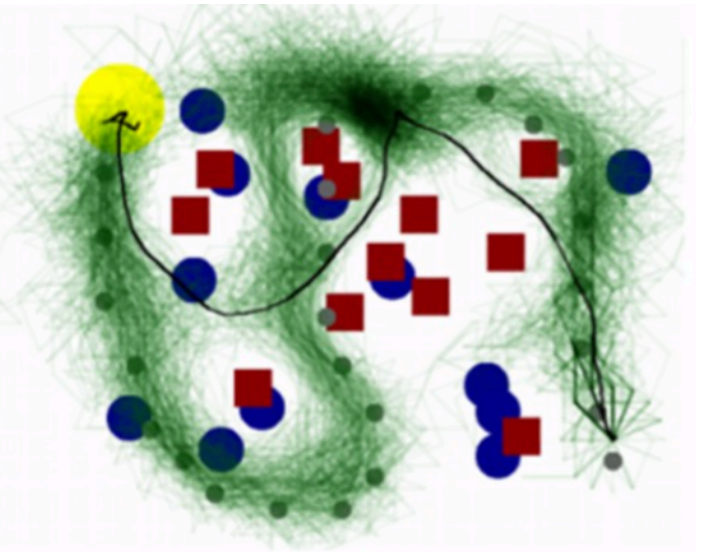




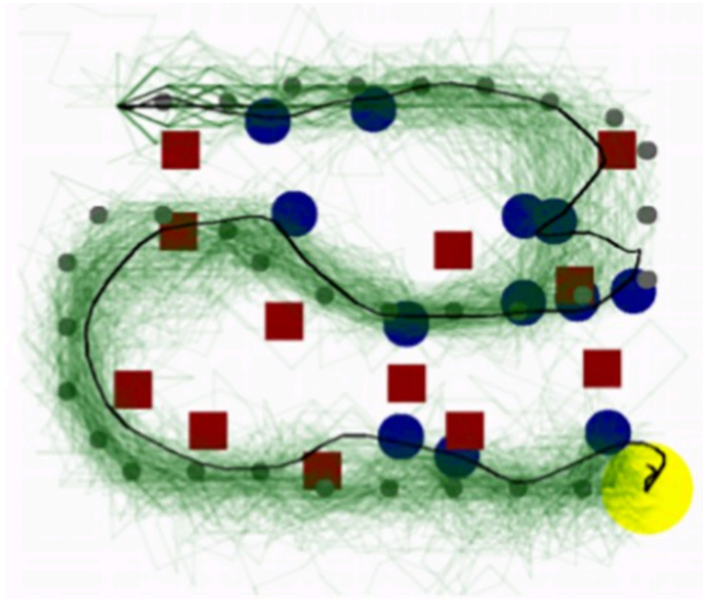
(D) $r : (0.00, \mathbf{0.57}, \mathbf{0.43})$
$\gamma : (0.00, 0.69, 0.91)$



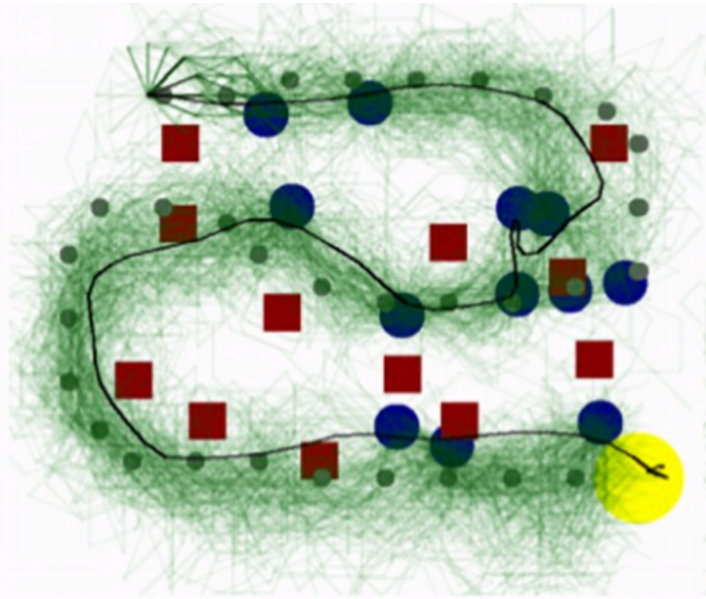
(E) $r : (0.02, \mathbf{0.57}, \mathbf{0.41})$
$\gamma : (0.99, 0.59, 0.97)$



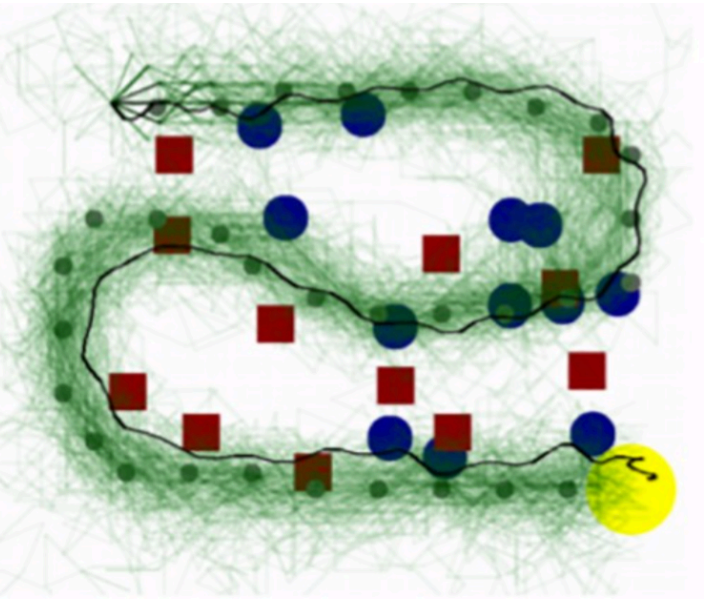
(F) $r : (0.06, \mathbf{0.75}, \mathbf{0.19})$
$\gamma : (0.95, 0.60, 0.88)$



(G) $r : (\mathbf{0.30}, 0.22, \mathbf{0.48})$
$\gamma : (0.77, 0.72, 0.89)$



(H) $r : (\mathbf{0.27}, 0.29, \mathbf{0.45})$
$\gamma : (0.69, 0.73, 0.96)$



(I) $r : (\mathbf{0.30}, 0.17, \mathbf{0.52})$
$\gamma : (0.76, 0.99, 0.89)$

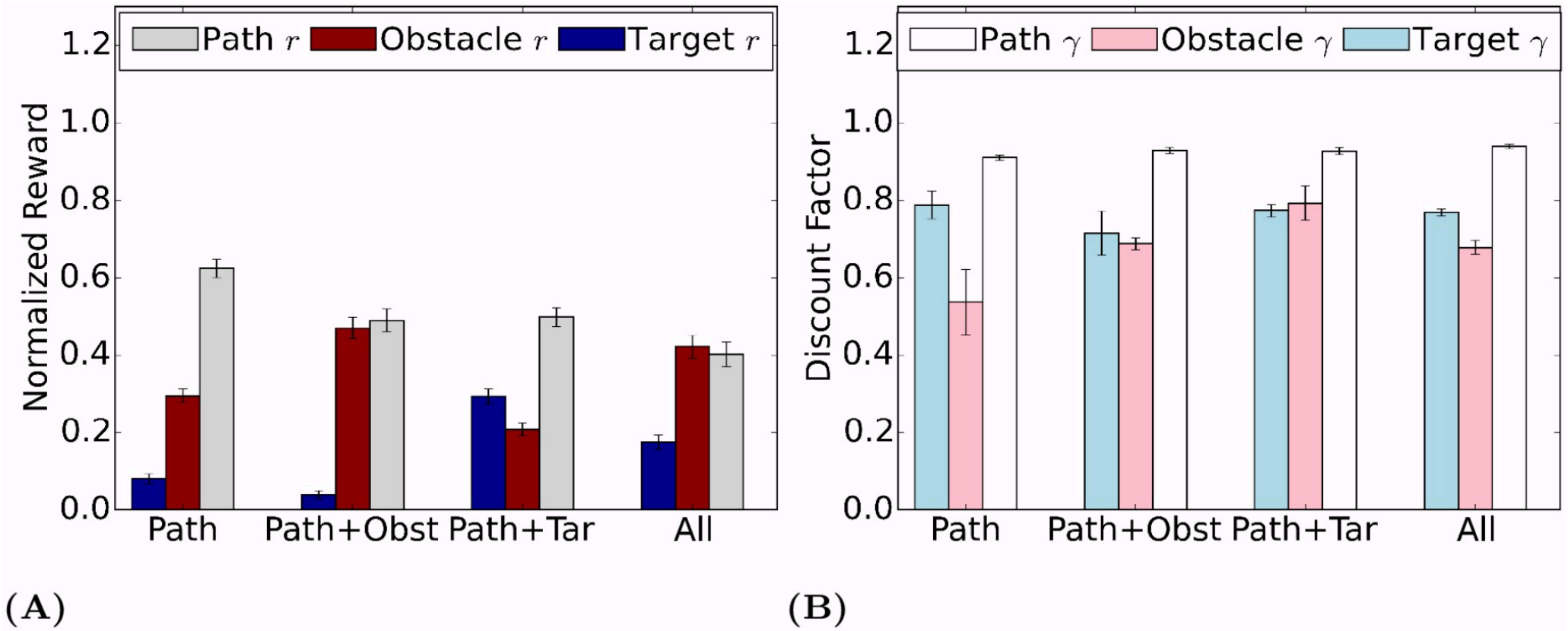


Fig 5. (A) Normalized average rewards across different task instructions. The error bar represents the standard error of the mean between subjects ($N = 25$). The obstacle module has negative reward, but to compare with the other two modules its absolute value is taken. The estimated reward agree with task instructions. (B) Average discount factors across different task instructions. The error bar represents the standard error of the mean between subjects ($N = 25$).