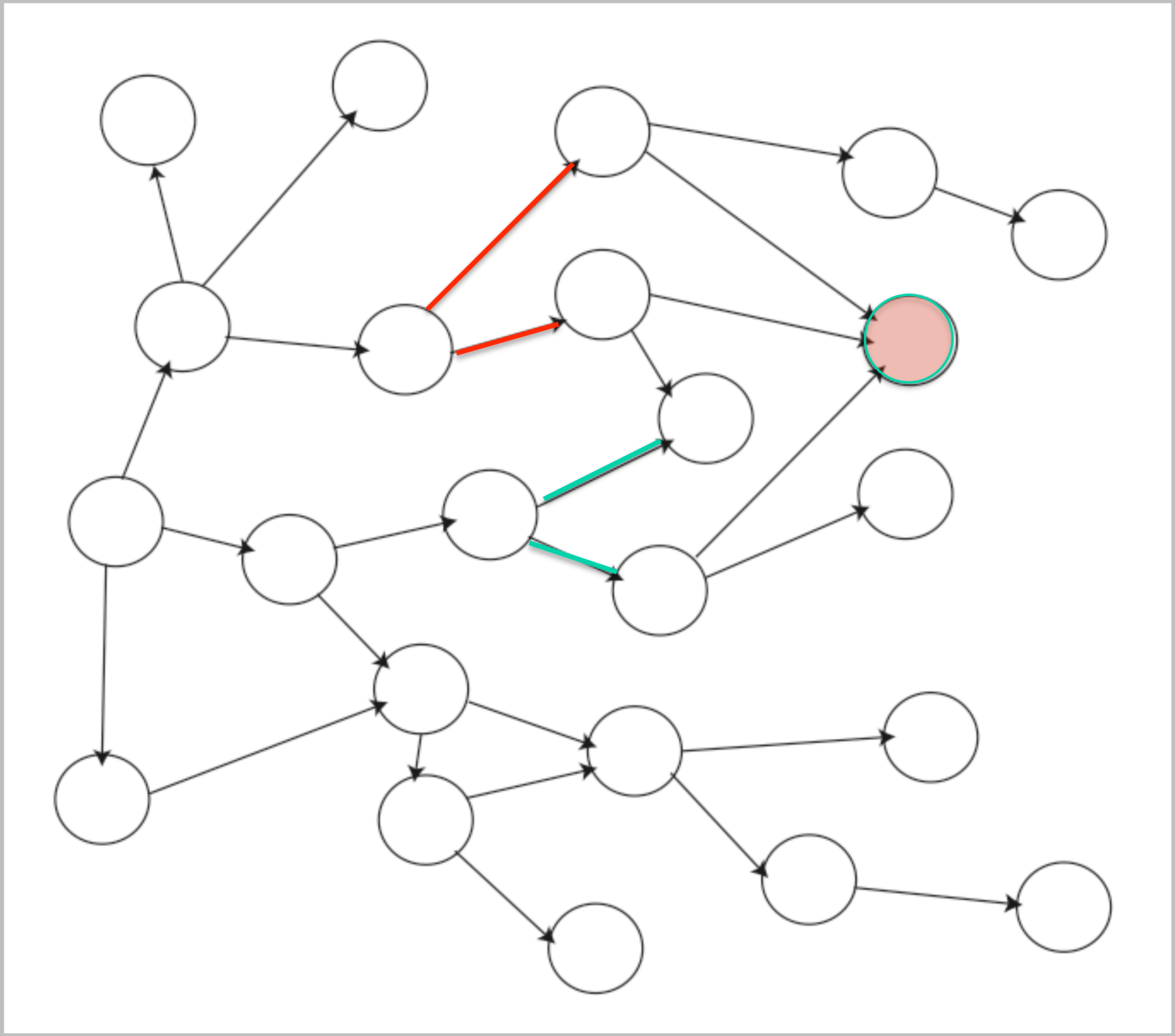
Basic RL Problems

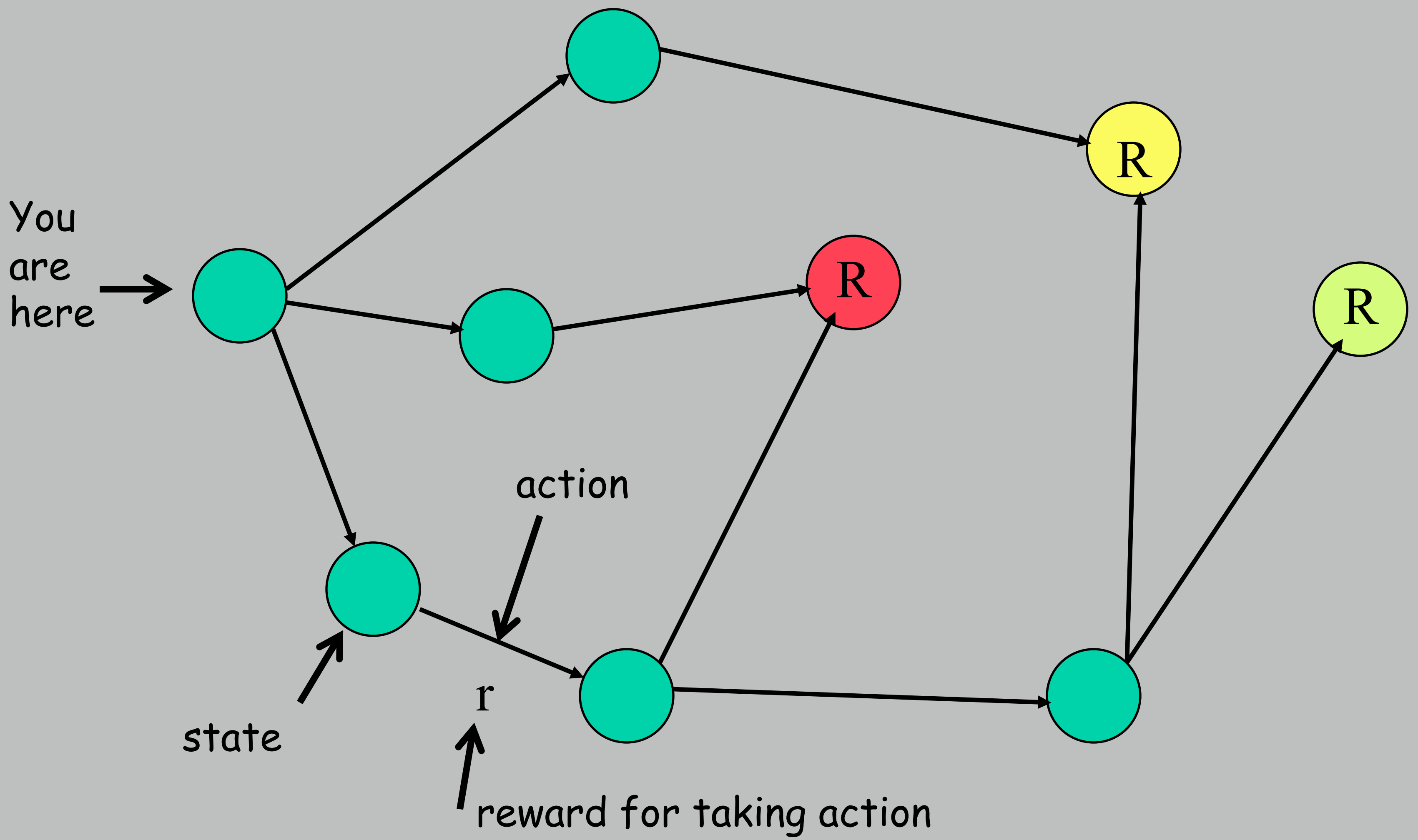


Location of reward uncertain

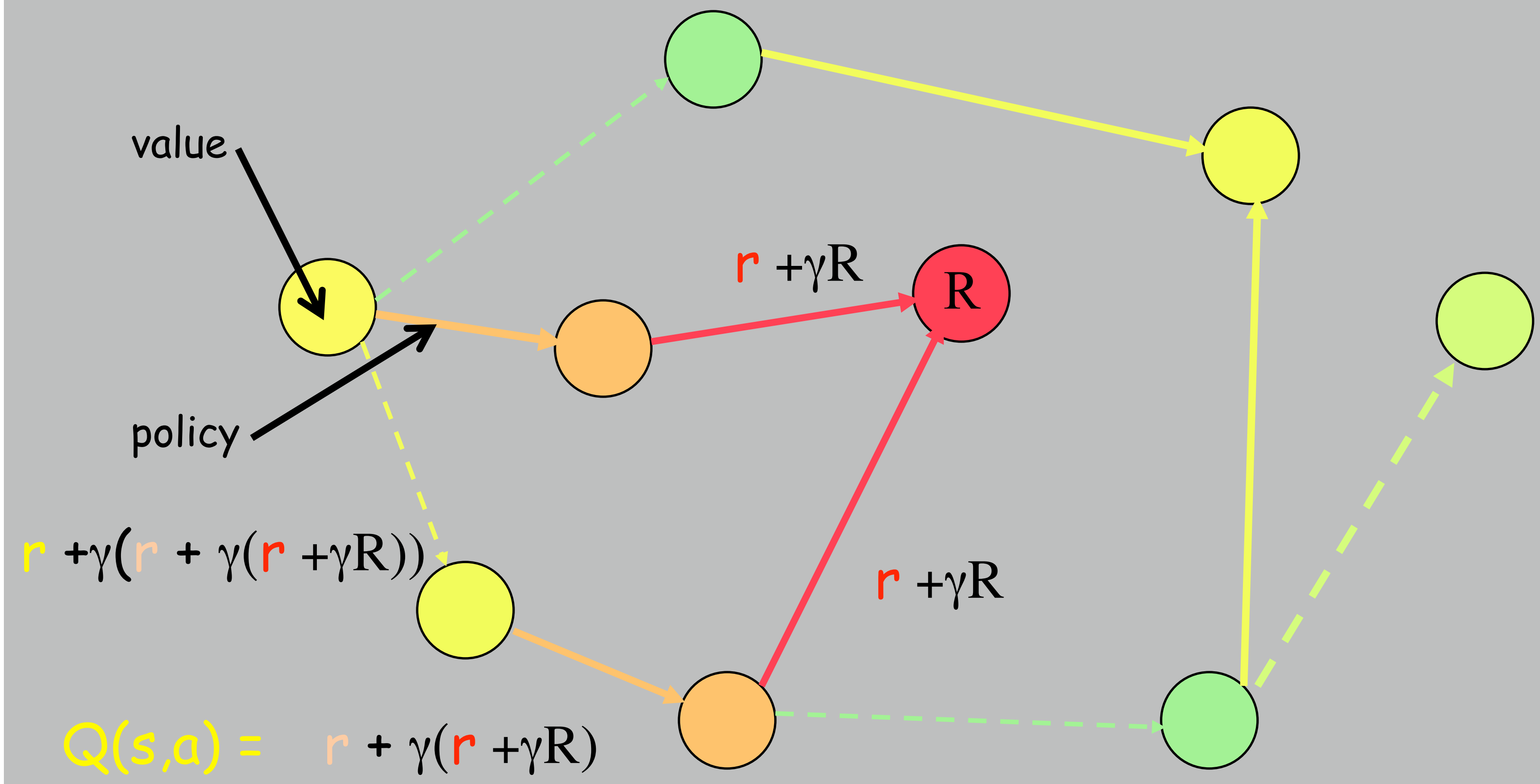
Transitions between states uncertain

Policy constantly changing

Reinforcement Learning Primer : Before Learning

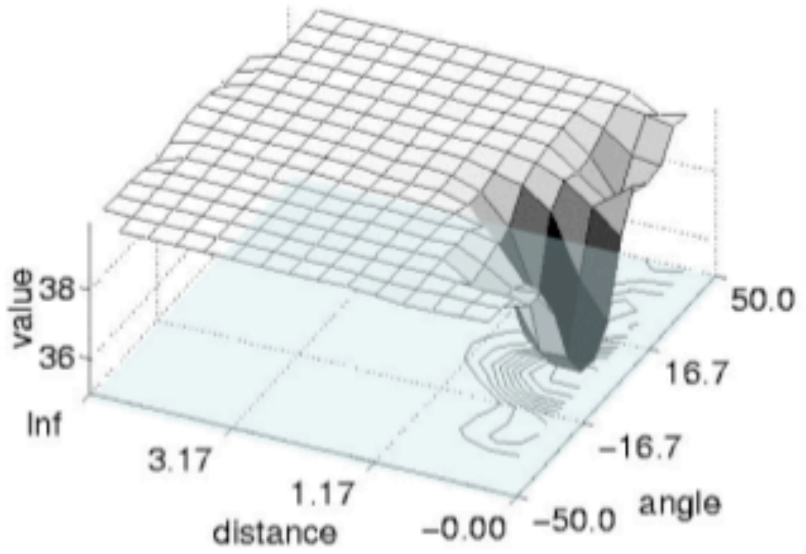


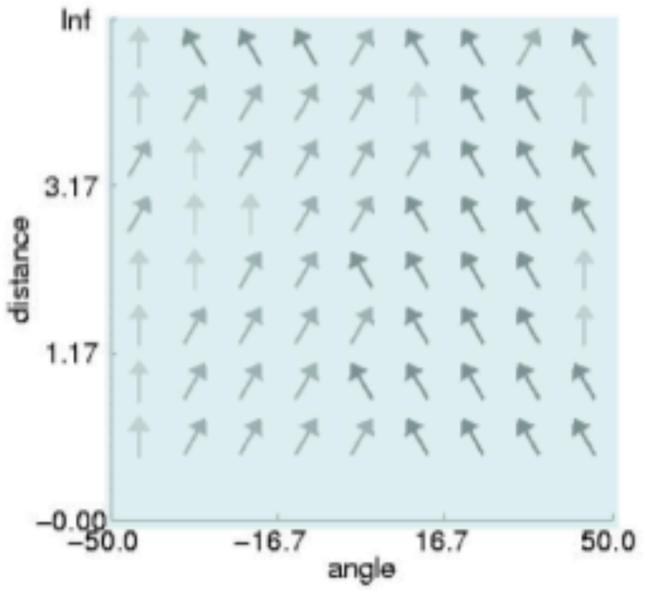
Reinforcement Learning Primer

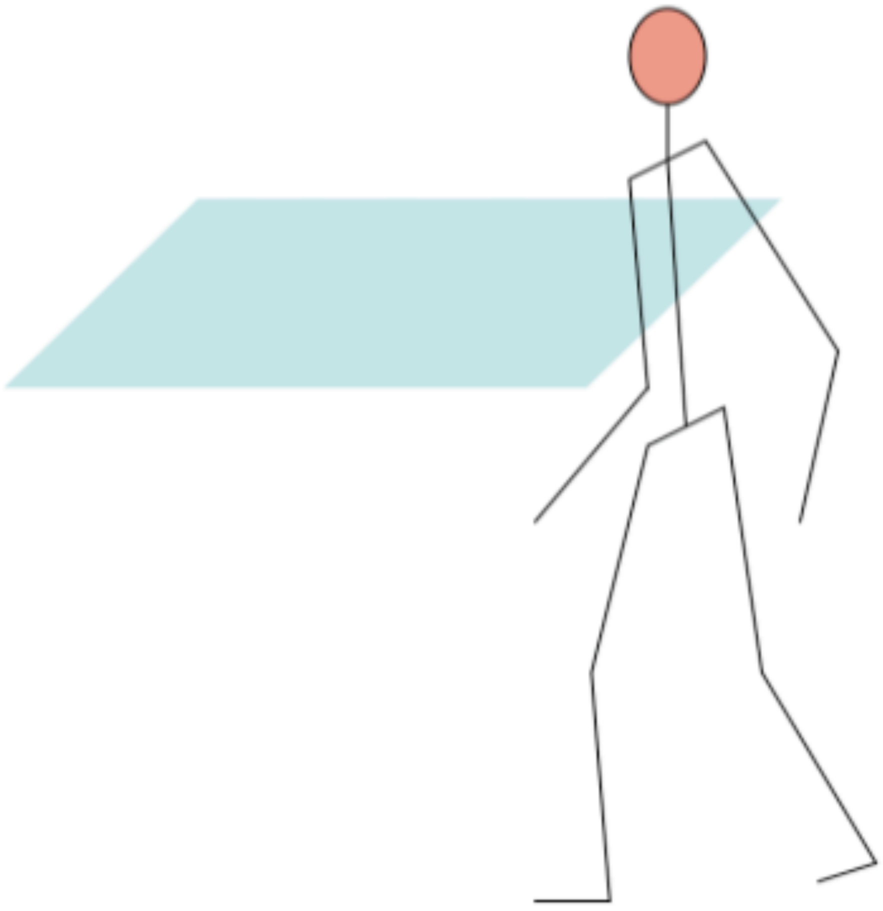

value
policy
$r + \gamma R$
R
$r + \gamma(r + \gamma(r + \gamma R))$
$r + \gamma R$
$Q(s,a) = r + \gamma(r + \gamma R)$

By trying different actions from different starting points, we gradually learn the expected reward value from any starting point

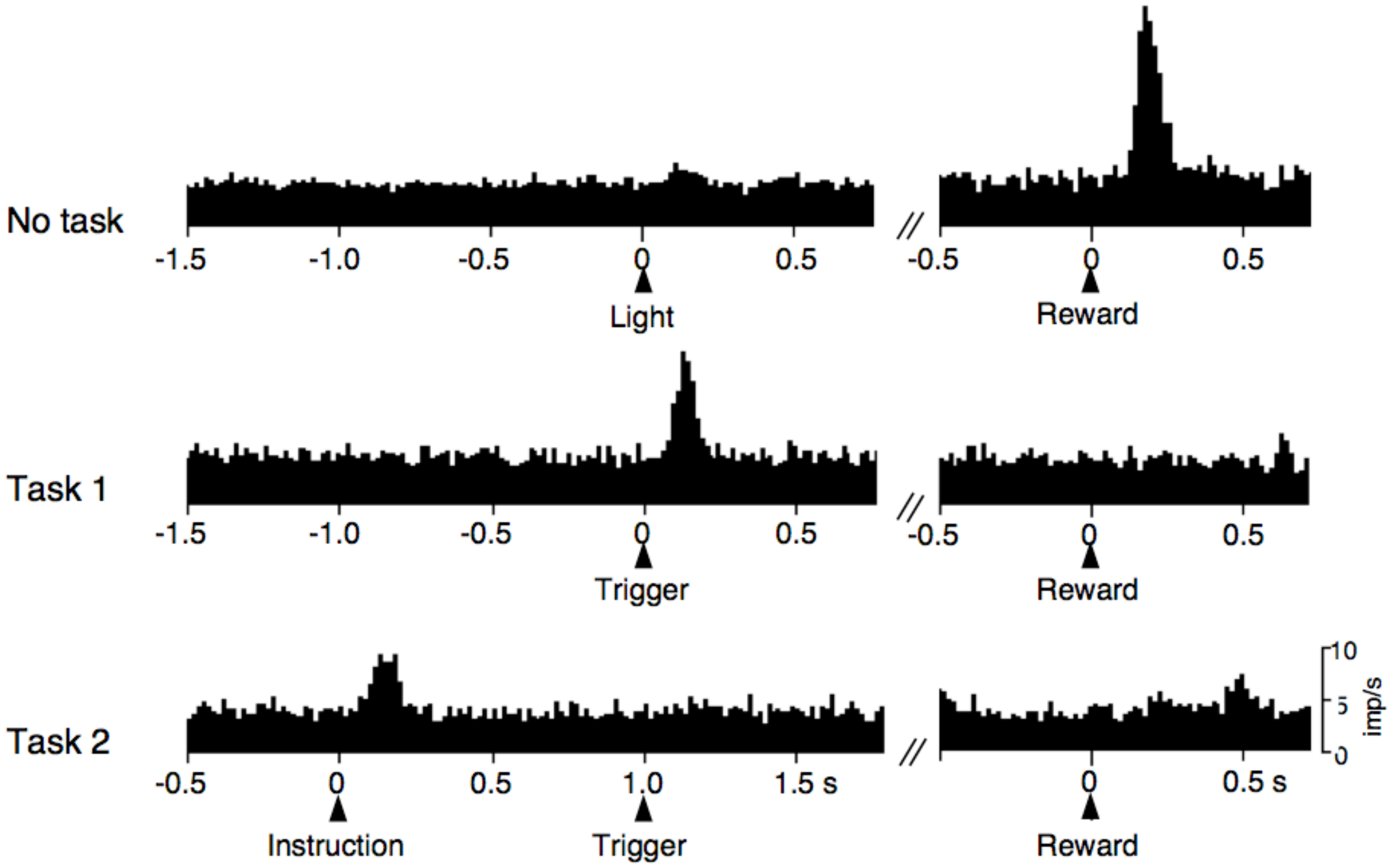
State Space


value
38
36
Inf
3.17
1.17
distance
-0.00
-50.0
-16.7
16.7
50.0
angle

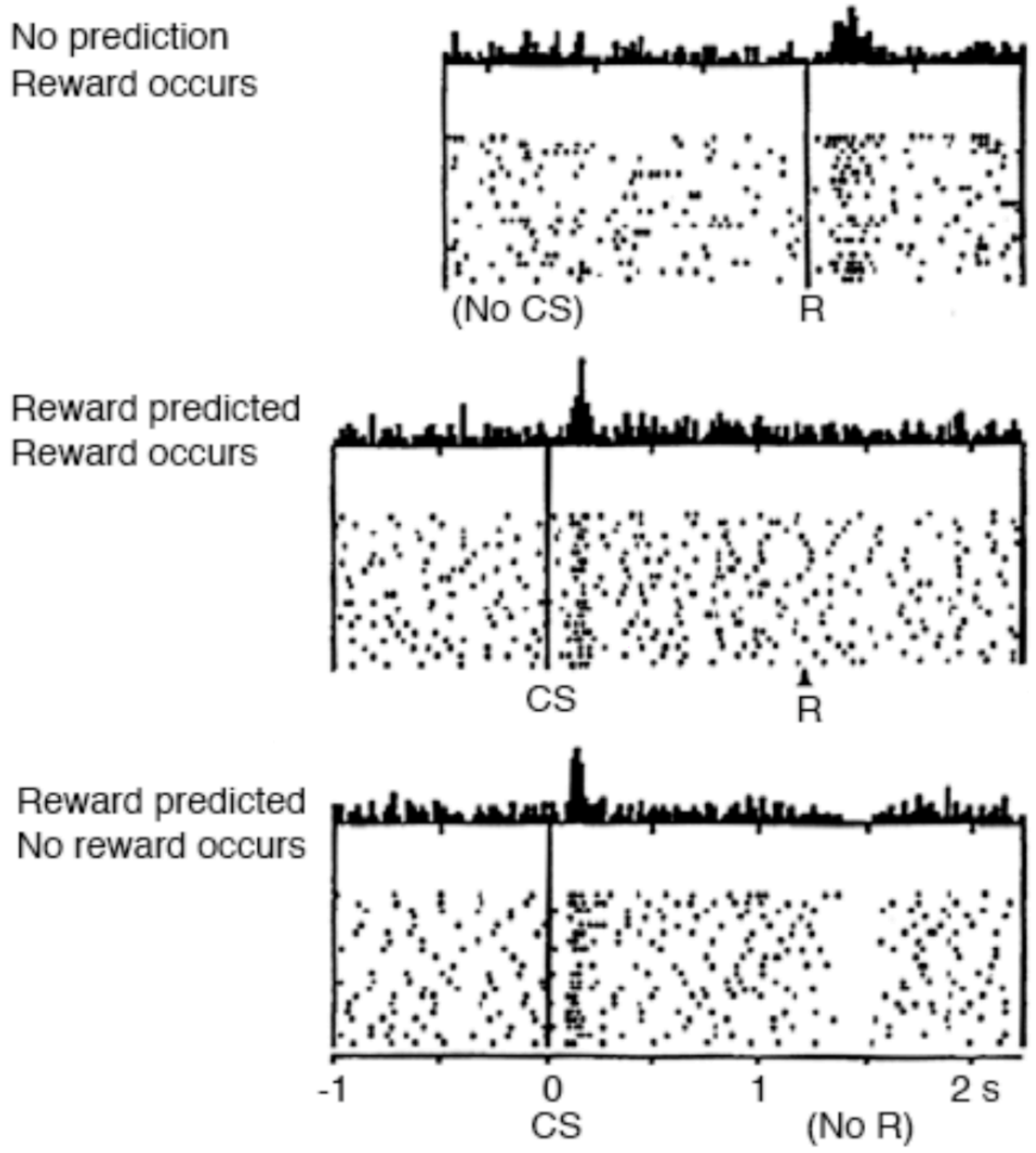

Inf
3.17
1.17
-0.00
distance
-50.0
-16.7
16.7
50.0
angle



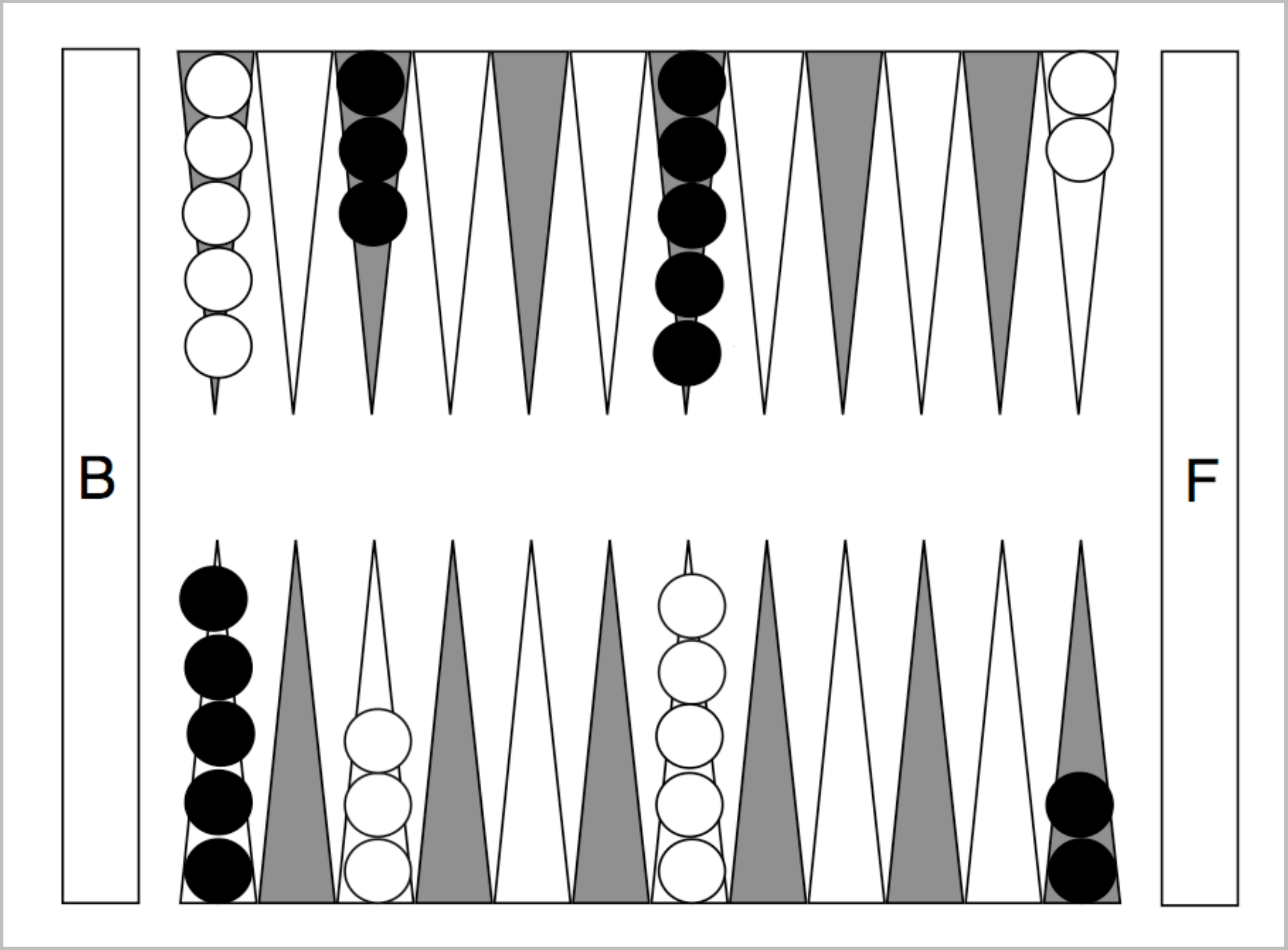
A Monkey uses Secondary Reward



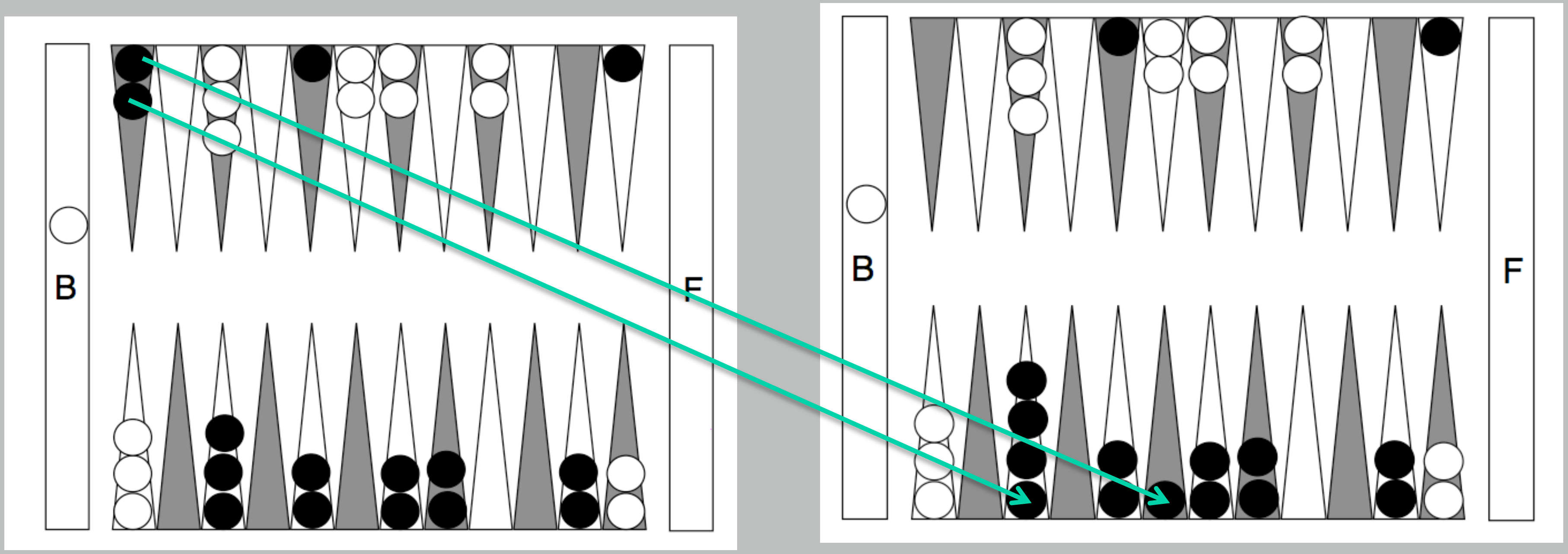
Do dopamine neurons report an erro□ in the prediction of reward?



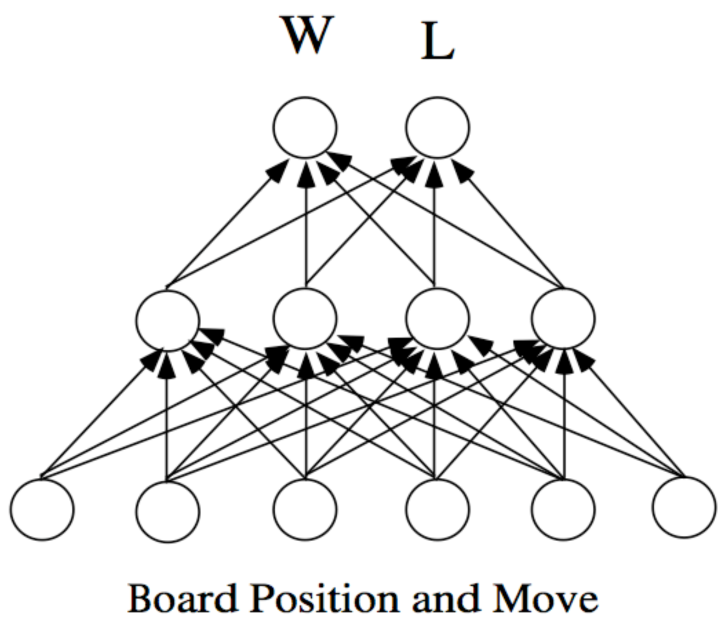
Backgammon

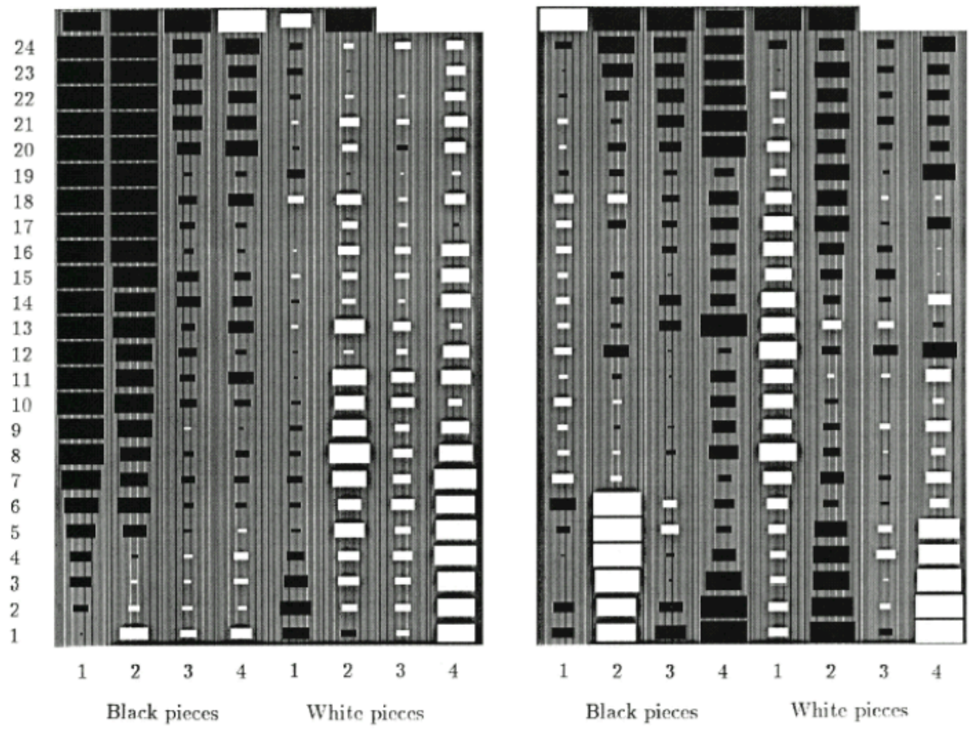


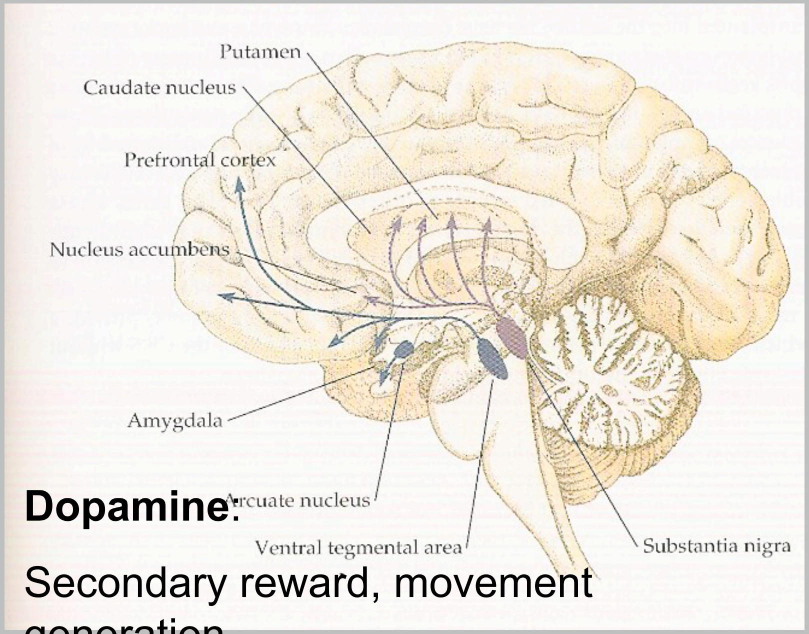
A move in Backgammon



Backgammon played with RL and Backpropagation

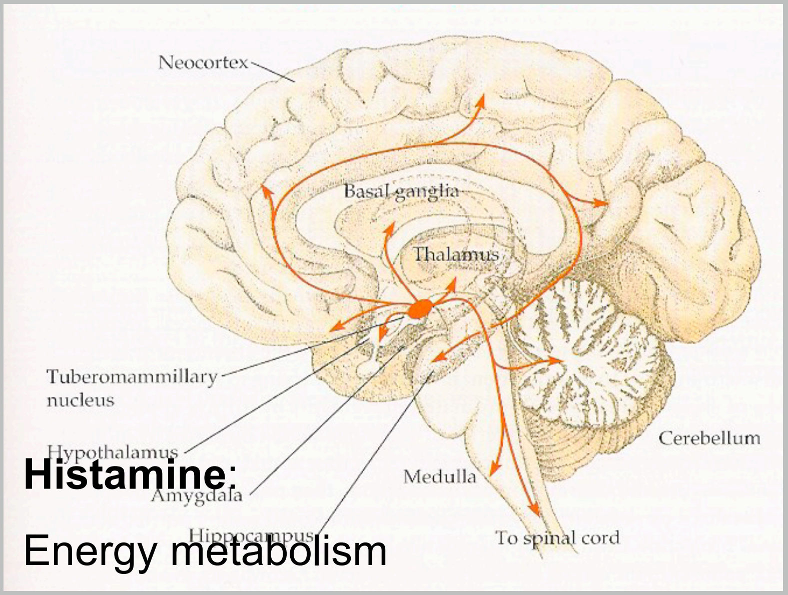






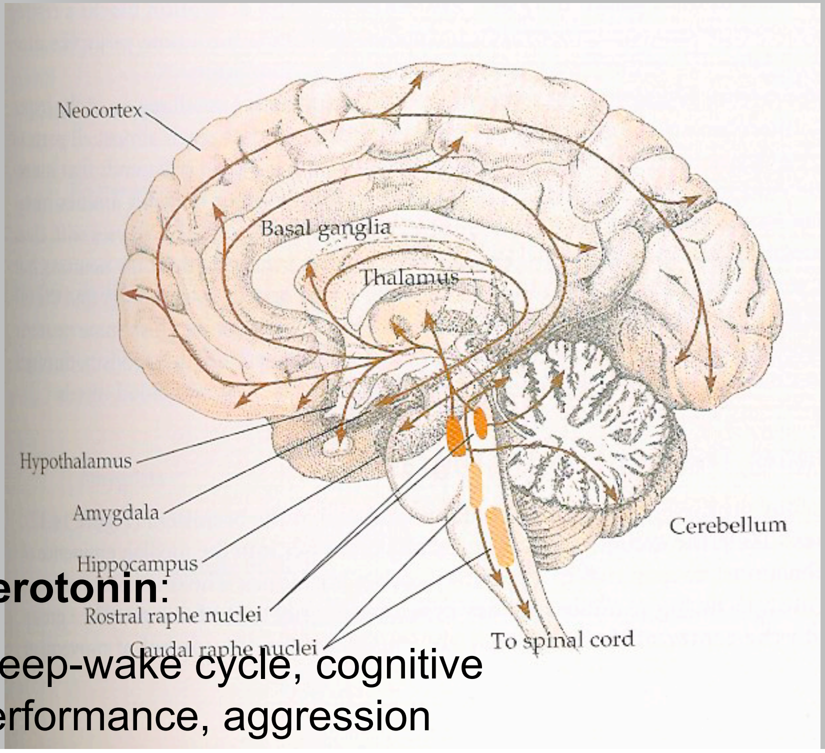
Dopamine:

Secondary reward, movement generation



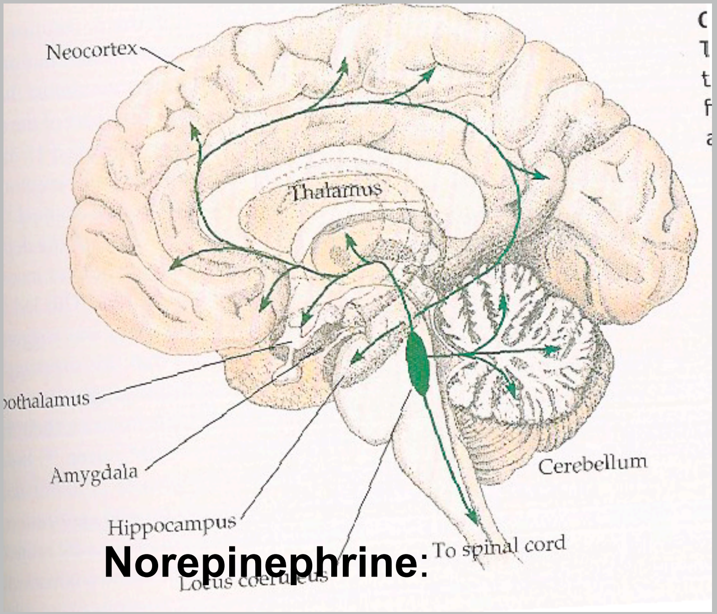
Histamine:

Energy metabolism



Serotonin:

Sleep-wake cycle, cognitive performance, aggression

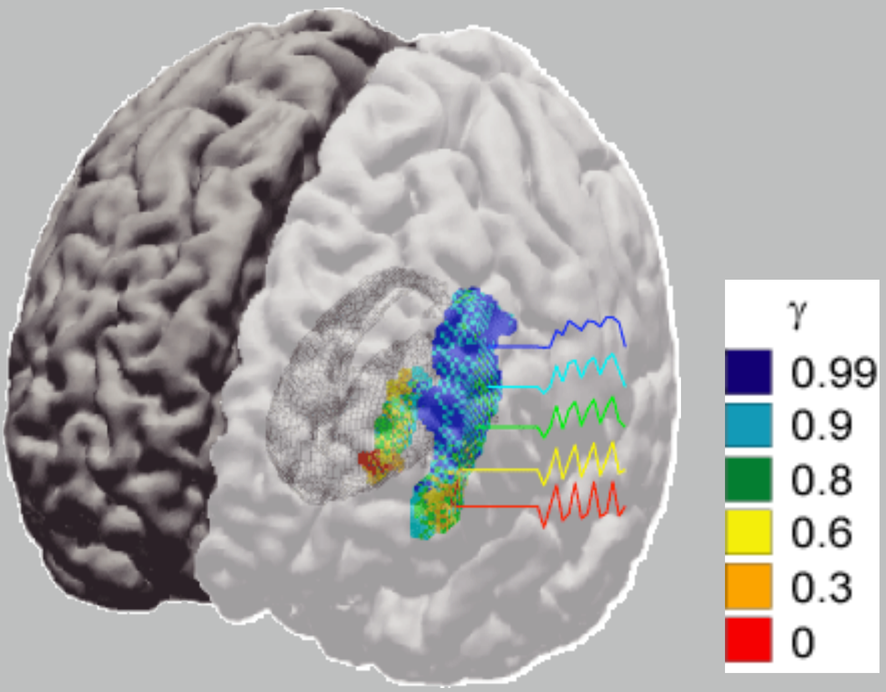


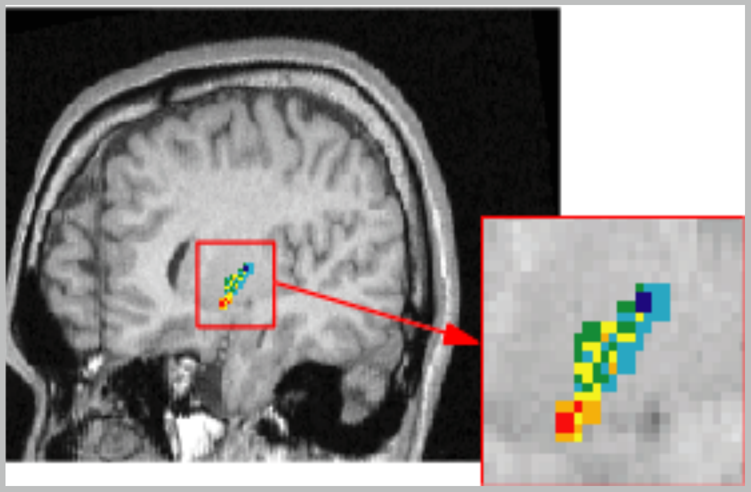
Norepinephrine:

Attention, arousal, circadian rhythms

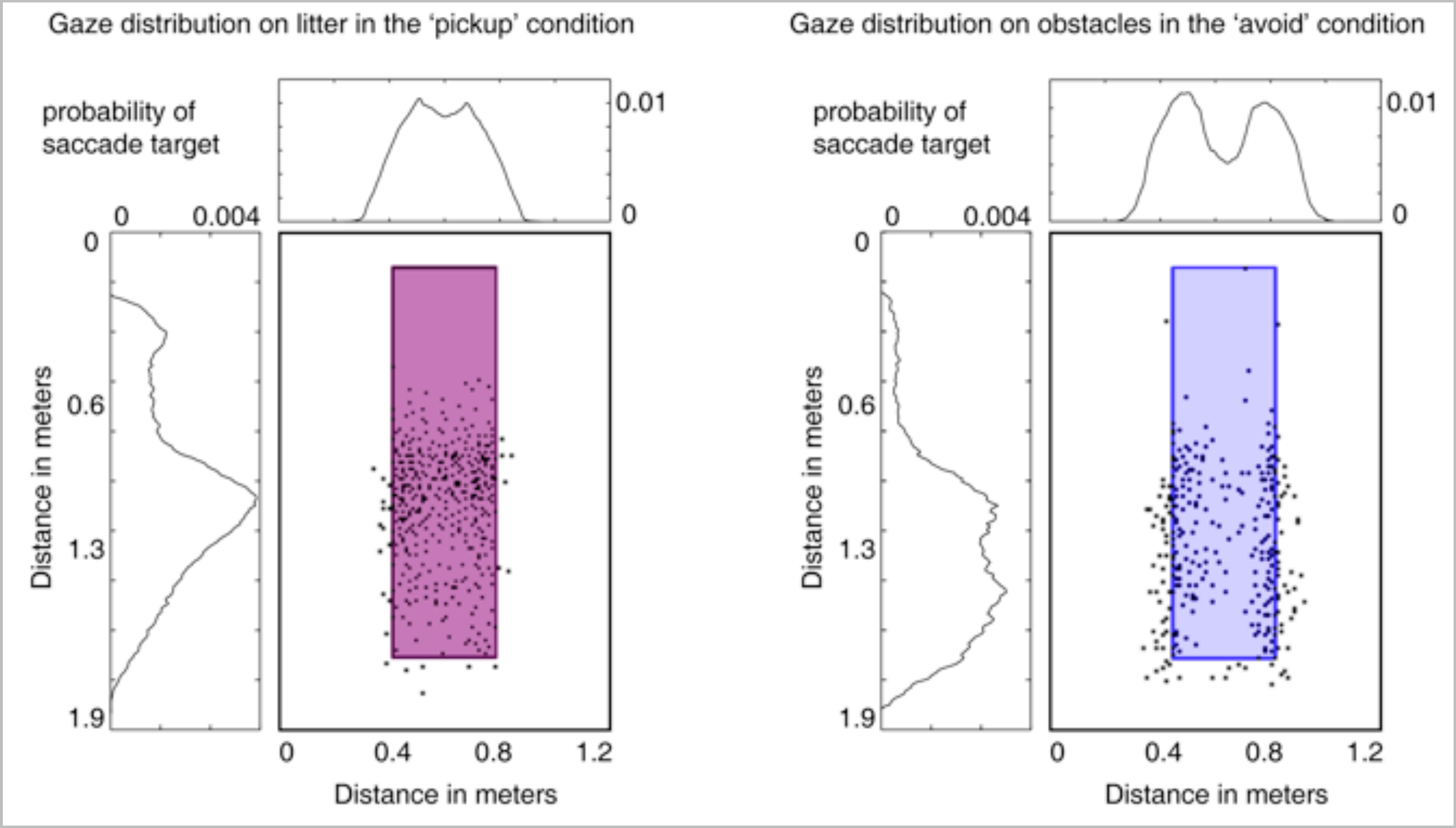
Map of Temporal Discounting
(Tanaka et al., 2004 (Kenji Doya))

- Markov decision task with delayed rewards
- Regression by values and TD errors
  - with different discounting factors g



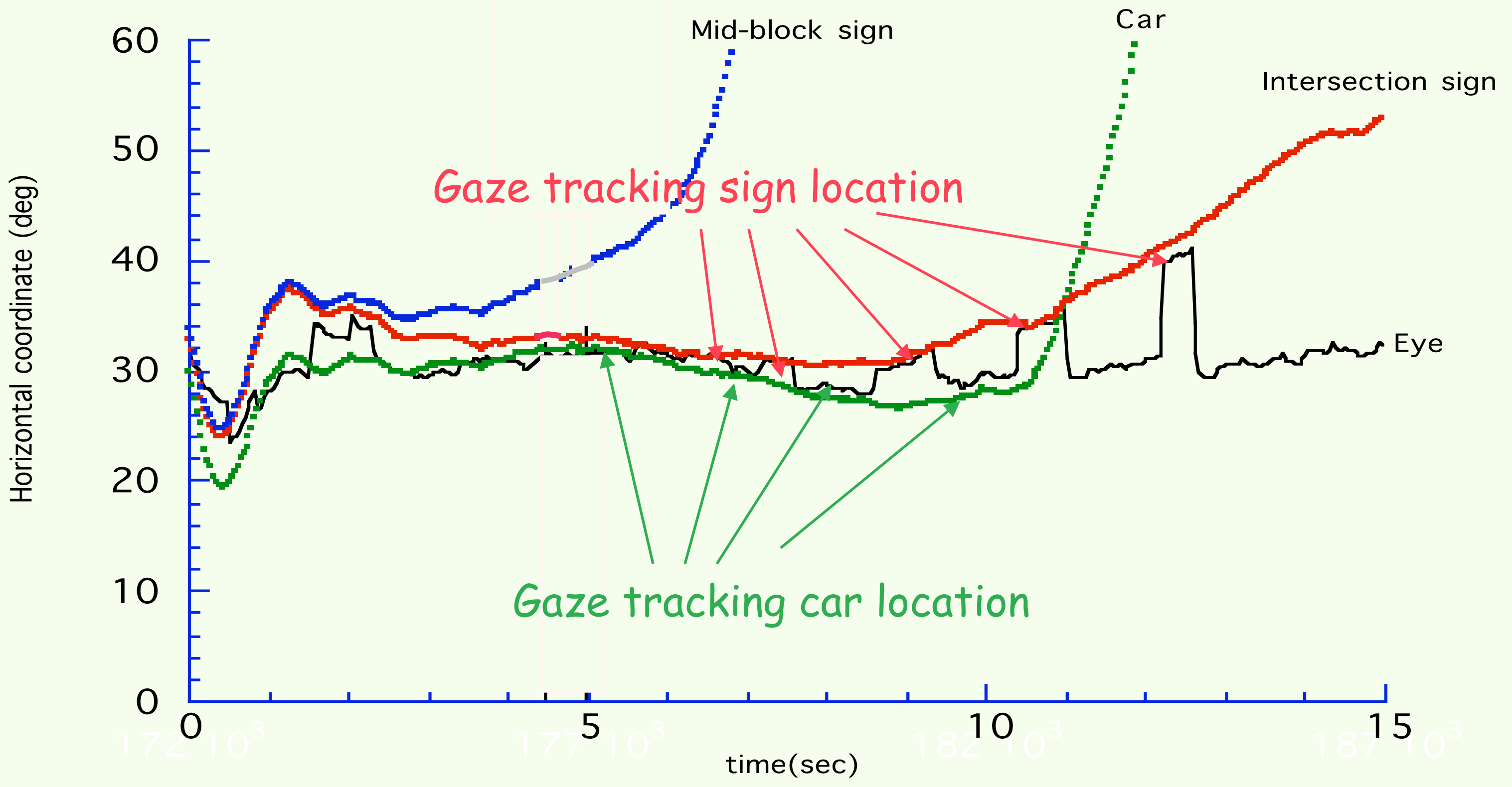


Task determines fixation point



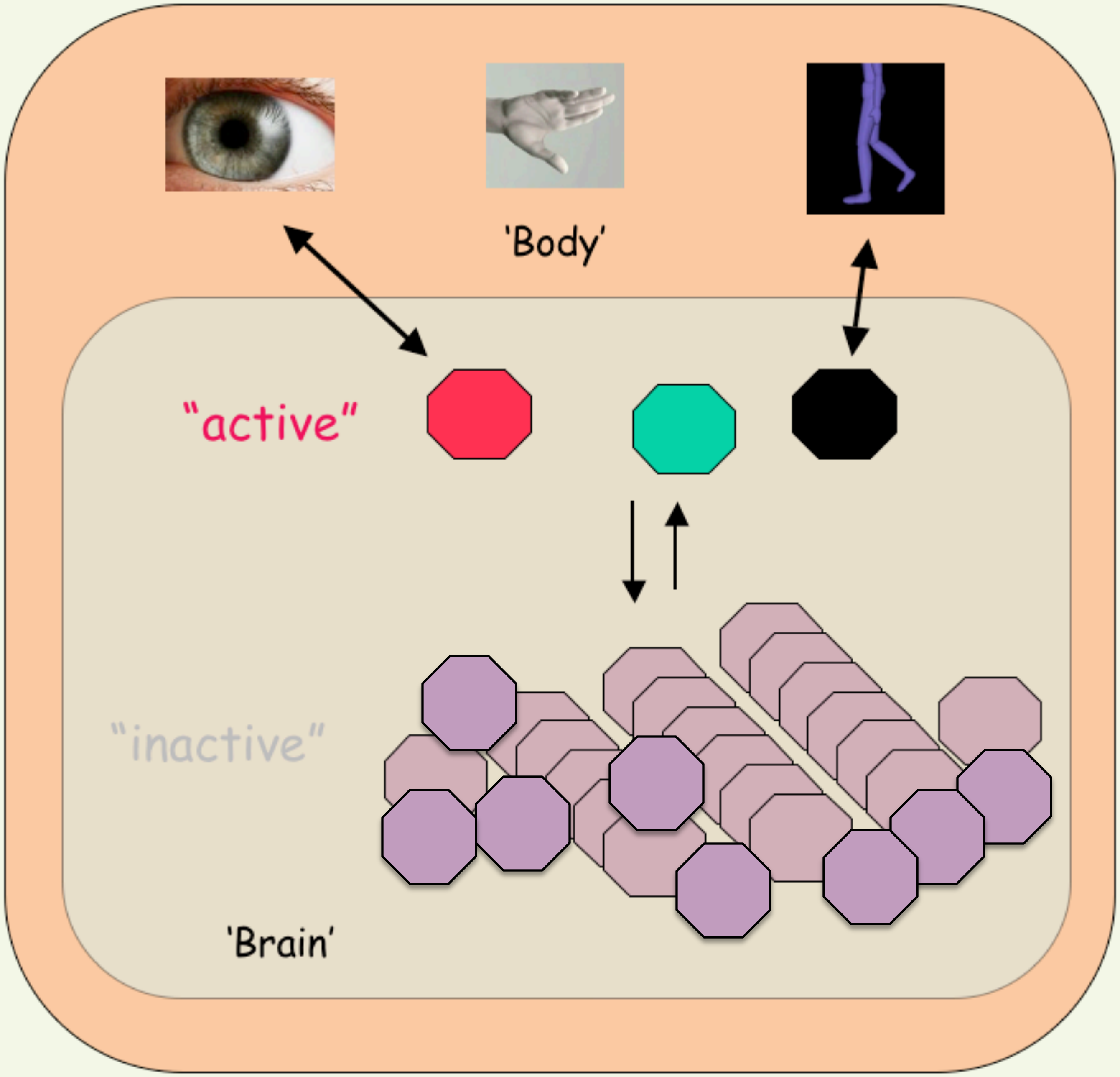
Rothkopf and Ballard Journal of Vision (2005) 5, 1-3

Multi-tasking
revealed by gaze sharing in human data



Shinoda & Hayhoe, Vision Research 2001

Scheduling - Modules - Routines

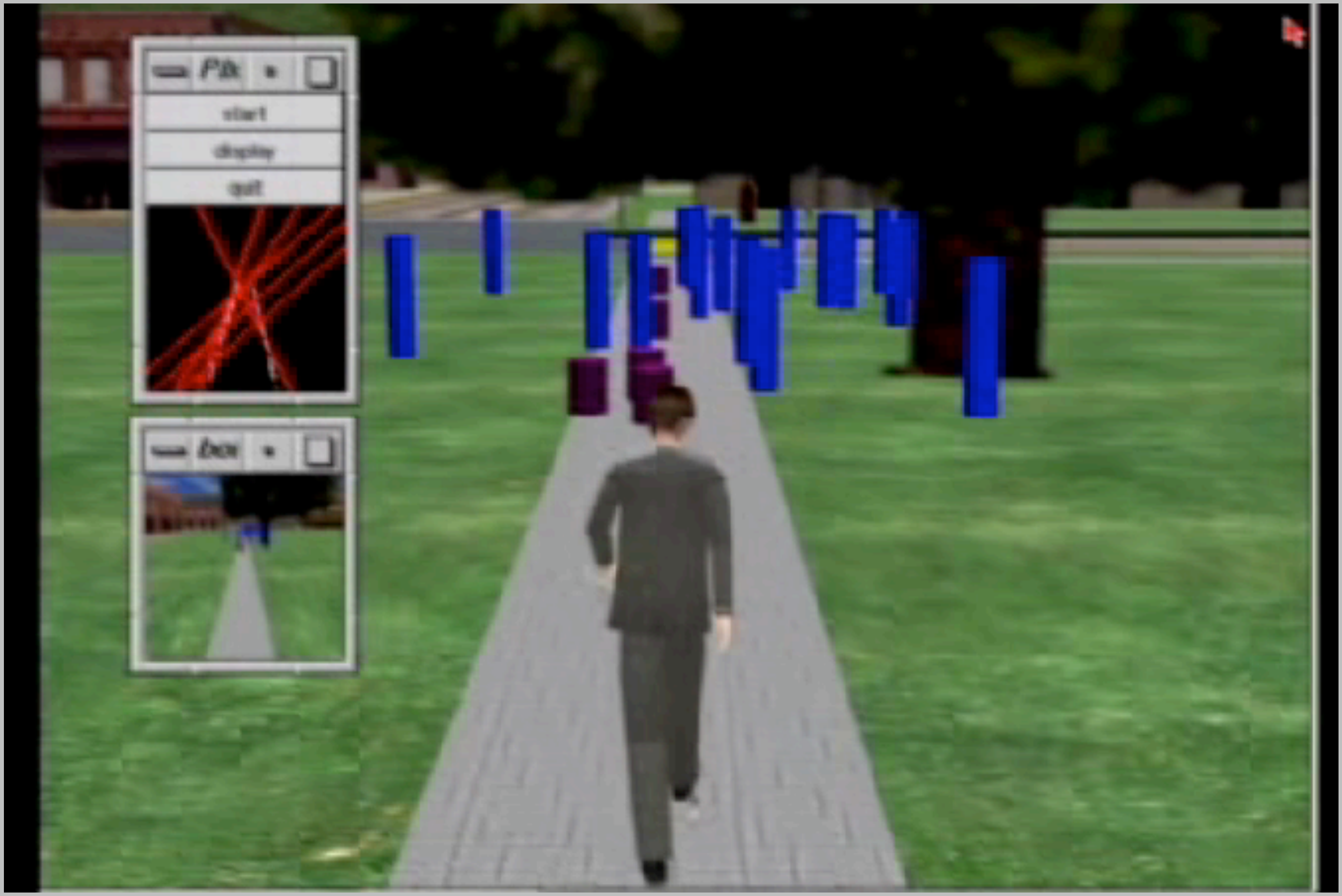


Routines
These behaviors contain instructions of how to use the body

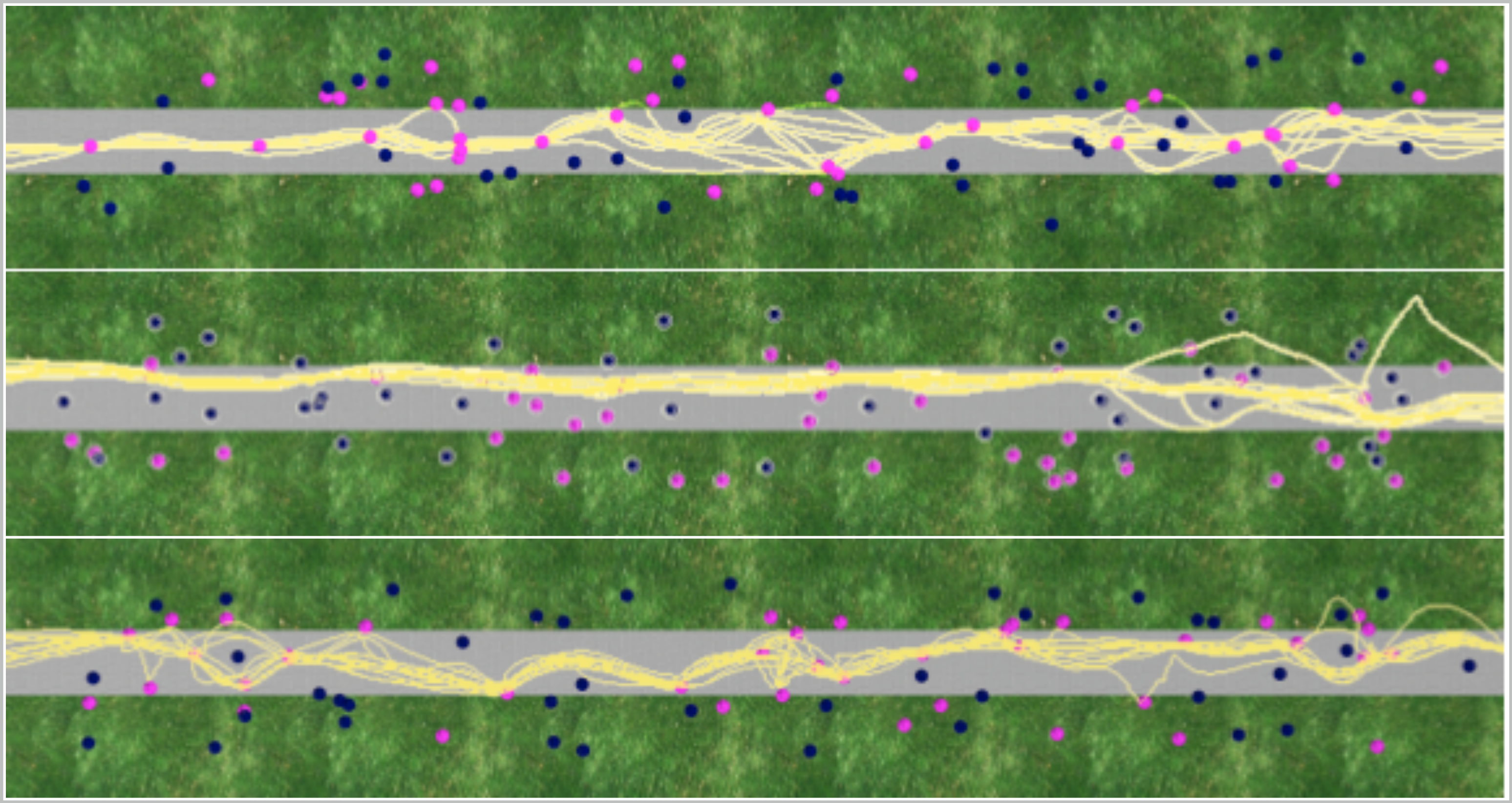
Scheduling
At any moment only a small subset are being used

Modules
There is an enormous library of special purpose behaviors

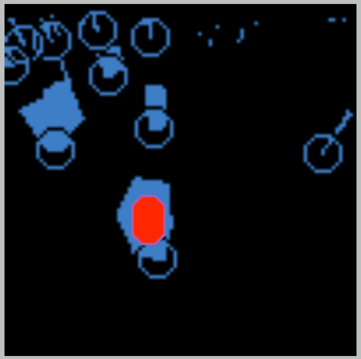
Humanoid models are used to study multi-tasking



Human performance data shows unexpected regularities



Module for Litter Cleanup

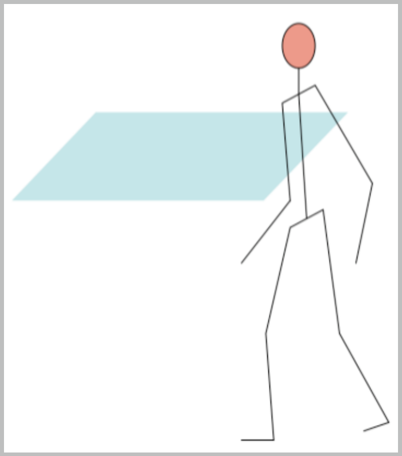


1. Visual Routine

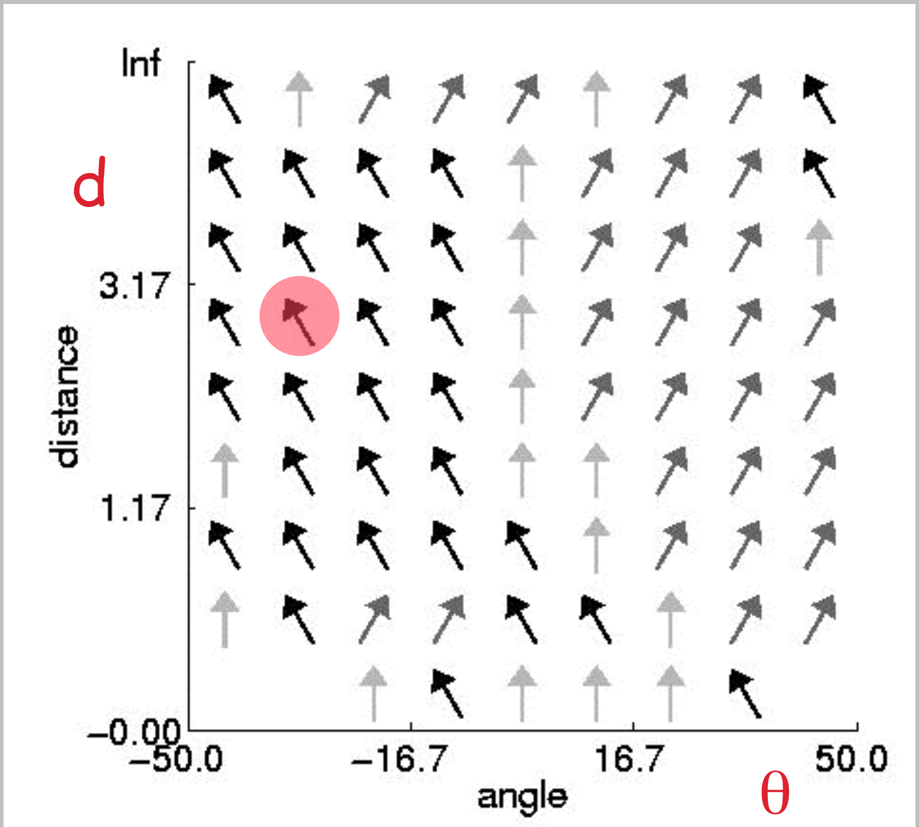
$\rightarrow \theta, d$

$\downarrow$

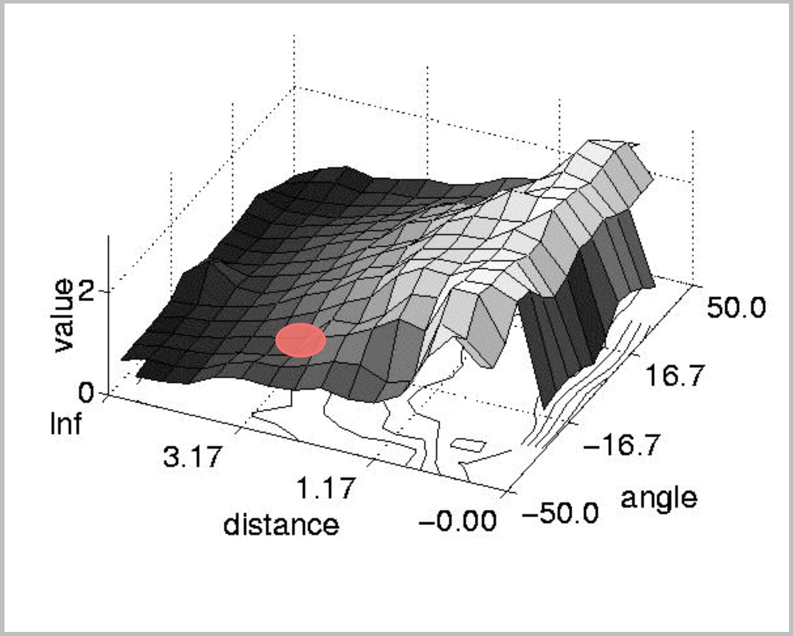
2a. Policy



2b. V is value of Policy



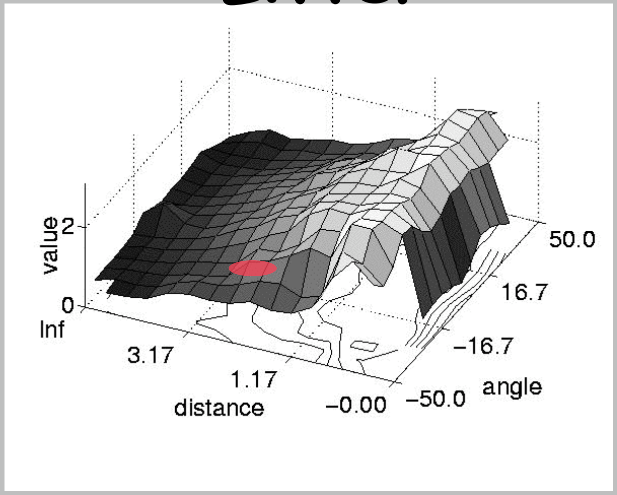
Heading from Walter's perspective

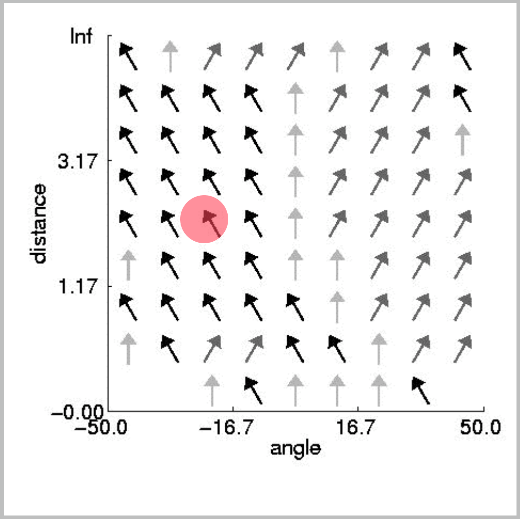


$V(s) = \max_a Q(s,a)$

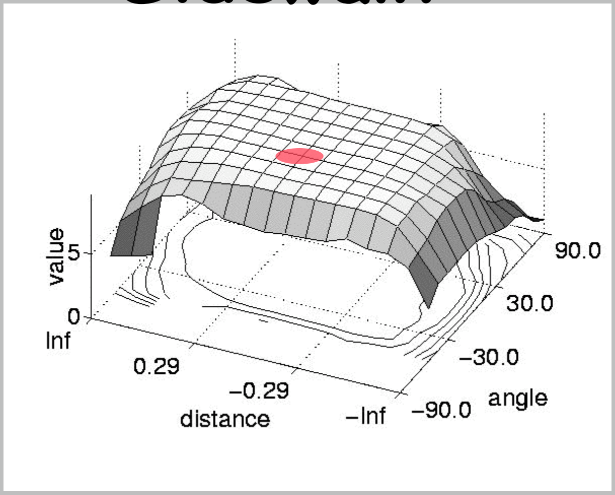
Learned Microbehaviors

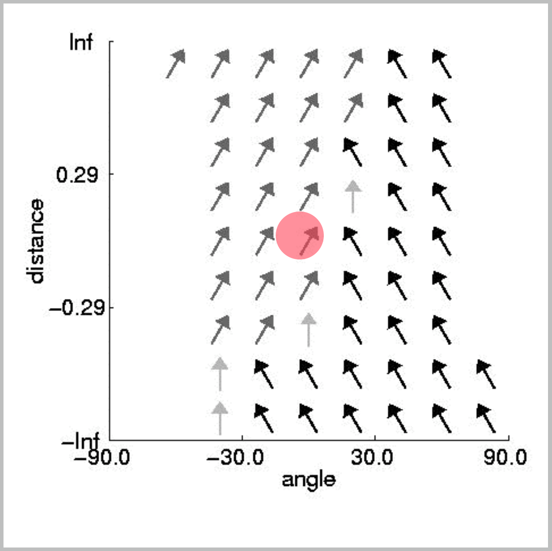
Litter



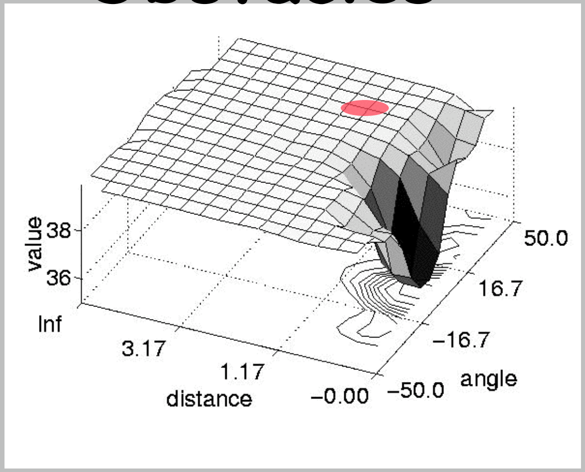


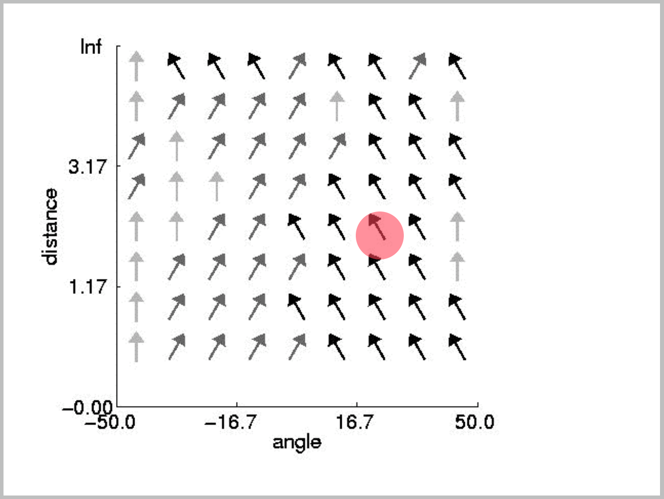
Sidewalk



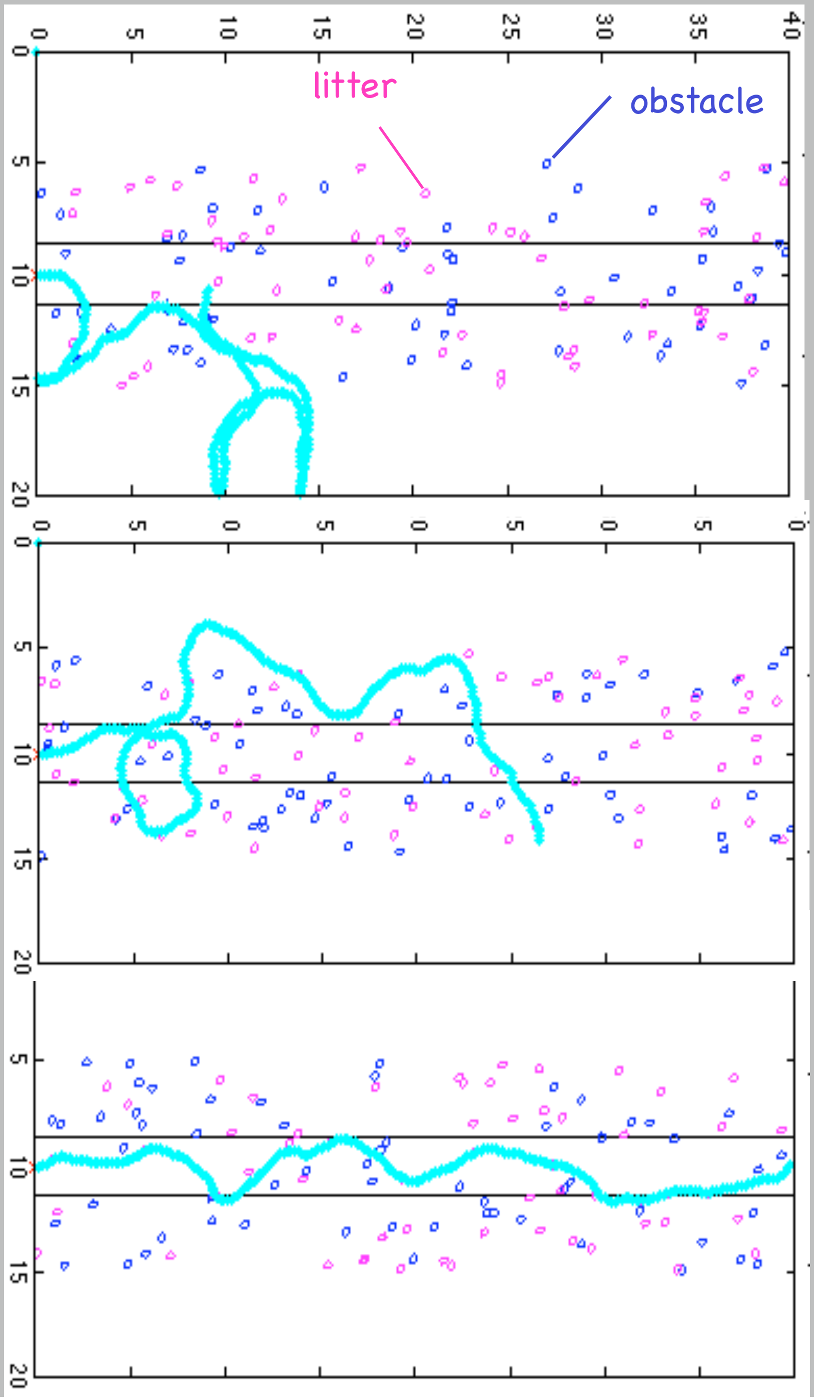


Obstacles





Overhead view of trajectory



Initial performance

After 100 iterations

After 150 iterations

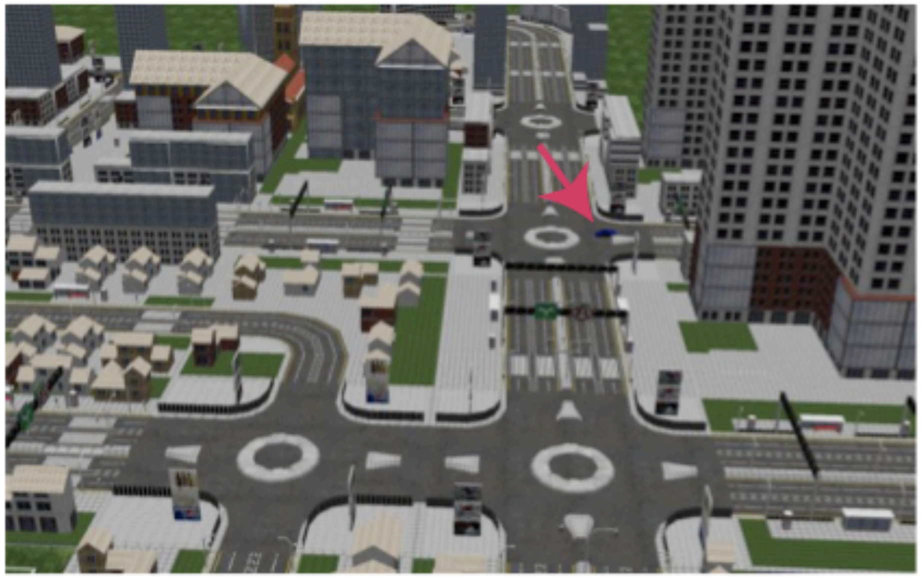
The basic RL update for i-th module:

$$Q_i(s_t^{(i)}, a_t^{(i)}) \leftarrow Q_i(s_t^{(i)}, a_t^{(i)}) + \alpha\delta_{Qi}$$

where δ_{Qi} is given by:

$$\delta_{Q_i} = \hat{r}_t^{(i)} + \gamma Q_i(s_{t+1}^{(i)}, a_{t+1}^{(i)}) - Q_i(s_t^{(i)}, a_t^{(i)})$$

Driving Simulator



Markov Decision Processes

Problems with delayed reinforcement are well modeled as *Markov decision processes* (MDPs). An MDP consists of

- a set of states $\mathcal{S}$,
- a set of actions $\mathcal{A}$,
- a reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \Re$, and
- a state transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$, where a member of $\Pi(\mathcal{S})$ is a probability distribution over the set $\mathcal{S}$ (i.e. it maps states to probabilities). We write $T(s,a,s')$ for the probability of making a transition from state s to state s' using action a.

The state transition function probabilistically specifies the next state of the environment as a function of its current state and the agent's action. The reward function specifies expected instantaneous reward as a function of the current state and action. The model is *Markov* if the state transitions are independent of any previous environment states or agent actions. There are many good references to MDP models [10, 13, 48, 90].

The basic RL algorithm w Model

We will speak of the optimal *value* of a state--it is the expected infinite discounted sum of reward that the agent will gain if it starts in that state and executes the optimal policy. Using π as a complete decision policy, it is written

$$V^*(s) = \max_{\pi} E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \quad .$$

This optimal value function is unique and can be defined as the solution to the simultaneous equations

$$V^*(s) = \max_{a}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')\right), \forall s \in S \quad , \qquad (1)$$

which assert that the value of a state *s* is the expected instantaneous reward plus the expected discounted value of the next state, using the best available action. Given the optimal value function, we can specify the optimal policy as

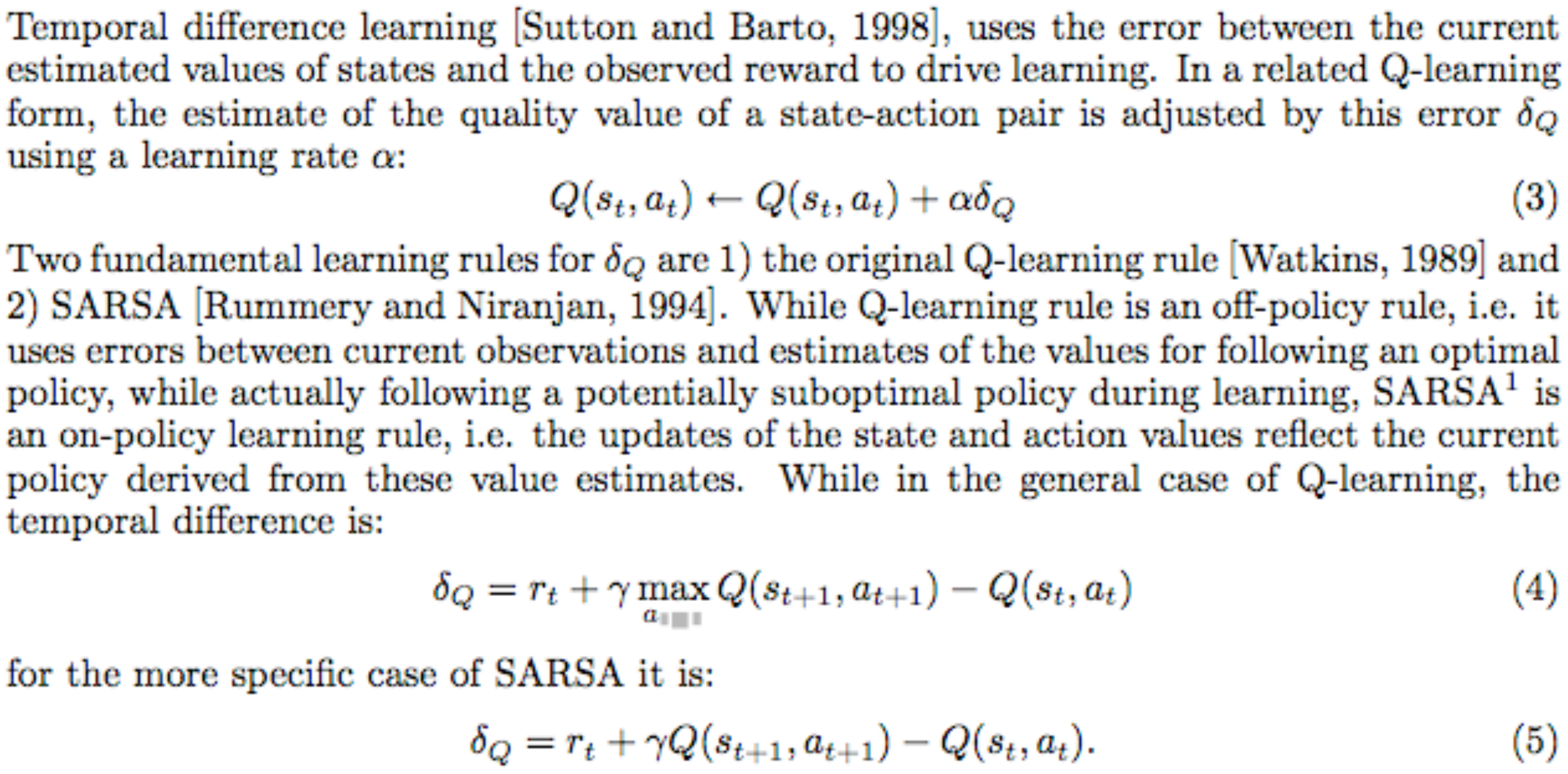
$$\pi^*(s) = \arg\max_{a}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')\right) \quad .$$

Value Iteration

```
initialize V(s) arbitrarily
loop until policy good enough
        loop for $s \in S$
                loop for $a \in \mathcal{A}$
                        $Q(s,a) := R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s')$
                $V(s) := \max_a Q(s,a)$
        end loop
end loop
```

Temporal Difference Learning

Q - Learning



Temporal difference learning [Sutton and Barto, 1998], uses the error between the current estimated values of states and the observed reward to drive learning. In a related Q-learning form, the estimate of the quality value of a state-action pair is adjusted by this error δ_Q using a learning rate α:

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

Two fundamental learning rules for δ_Q are 1) the original Q-learning rule [Watkins, 1989] and 2) SARSA [Rummery and Niranjan, 1994]. While Q-learning rule is an off-policy rule, i.e. it uses errors between current observations and estimates of the values for following an optimal policy, while actually following a potentially suboptimal policy during learning, SARSA[1] is an on-policy learning rule, i.e. the updates of the state and action values reflect the current policy derived from these value estimates. While in the general case of Q-learning, the temporal difference is:

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

for the more specific case of SARSA it is:

$$\delta_Q = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t). \tag{5}$$

Policy Iteration

```
choose an arbitrary policy $\pi'$

loop

          $\pi := \pi'$

          compute the value function of policy $\pi$ :

                    solve the linear equations

                              $V_\pi(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} T(s,\pi(s),s') V_\pi(s')$

          improve the policy at each state:

                    $\pi'(s) := \arg\max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_\pi(s')\right)$

until $\pi = \pi'$
```