- **Distributed**: each player a separate client
- Server models dynamics and kinematics
- Clients receive **sensations**, send **actions**

![Diagram](image-url)

- Parametric actions: *dash, turn, kick, say*
- **Abstract, noisy** sensors, hidden state
  - *Hear* sounds from limited distance
  - *See* relative distance, angle to objects ahead
- $> 10^{923}$ states
- **Limited resources**: stamina
- Play occurs in **real time** ($\approx$ human parameters)
3 vs. 2 Keepaway

- Play in a **small area** (20m × 20m)

- **Keepers** try to keep the ball

- **Takers** try to get the ball

- **Episode:**
  - Players and ball reset randomly
  - Ball starts near a keeper
  - Ends when taker gets the ball or ball goes out

- Performance measure: **average possession duration**

- Use **CMUnited-99 skills:**
  - HoldBall, PassBall(k), GoToBall, GetOpen
The Keepers’ Policy Space

Example Policies

**Random:** HoldBall or PassBall\((k)\) randomly

**Hold:** Always HoldBall

**Hand-coded:**
- If no taker within 10m: HoldBall
- Else If there’s a good pass: PassBall\((k)\)
- Else HoldBall
Mapping Keepaway to RL

Discrete-time, episodic, distributed RL

- Simulator operates in discrete time steps, $t = 0, 1, 2, \ldots$, each representing 100 msec

- Episode:
  $s_0, a_0, r_1, s_1, \ldots, s_t, a_t, r_{t+1}, s_{t+1}, \ldots, r_T, s_T$

- $a_t \in \{\text{HoldBall}, \text{PassBall}(k), \text{GoToBall}, \text{GetOpen}\}$

- $r_t = 1$

- $V^\pi(s) = E\{T \mid s_0 = s\}$

- Goal: Find $\pi^*$ that maximizes $V$ for all $s$
Representation

Full soccer state

Sparse, coarse, tile coding

Few continuous state variables

(13)

Huge binary feature vector

(about 400 1’s and 40,000 0’s)

Linear map

Action values
13 Continuous State Variables

- 11 distances among players, ball, and center
- 2 angles to takers along passing lanes
Function Approximation: Tile Coding

- Form of sparse, coarse coding based on CMACS [Albus, 1981]

- Tiled state variables individually (13)
Policy Learning

- Learn $Q^\pi(s, a)$: Expected possession time

- Linear Sarsa($\lambda$) — each agent learns independently
  - On-policy method: advantages over e.g. Q-learning
  - Not known to converge, but works (e.g. [Sutton, 1996])

- Only update when ball is kickable for someone:
  Semi-Markov Decision Process

---

Kick: k1 k1 k1 k2 k2 k3 k3
Update: ● ● ● ● ● ● ●

TIME

Peter Stone
Main Result

![Graph showing episode duration vs. hours of training time]

- **Episode Duration (seconds)**
- **Hours of Training Time (bins of 1000 episodes)**

**Handcoded** vs **Random**

1 hour = 720 5-second episodes

Peter Stone
4 vs. 3 Keeper Learning

- Preliminary: taker learning successful as well
- Also tried varying field sizes