Machine Learning for Fast Quadrupedal Locomotion

Nate Kohl and Peter Stone

Department of Computer Sciences
The University of Texas at Austin

Overview

The goal: Enable an Aibo to walk as fast as possible

Challenges:

- **No simulator** available
  - Learn entirely on robots
  - Minimal human intervention

- Which **learning algorithm** to use?

Motivation

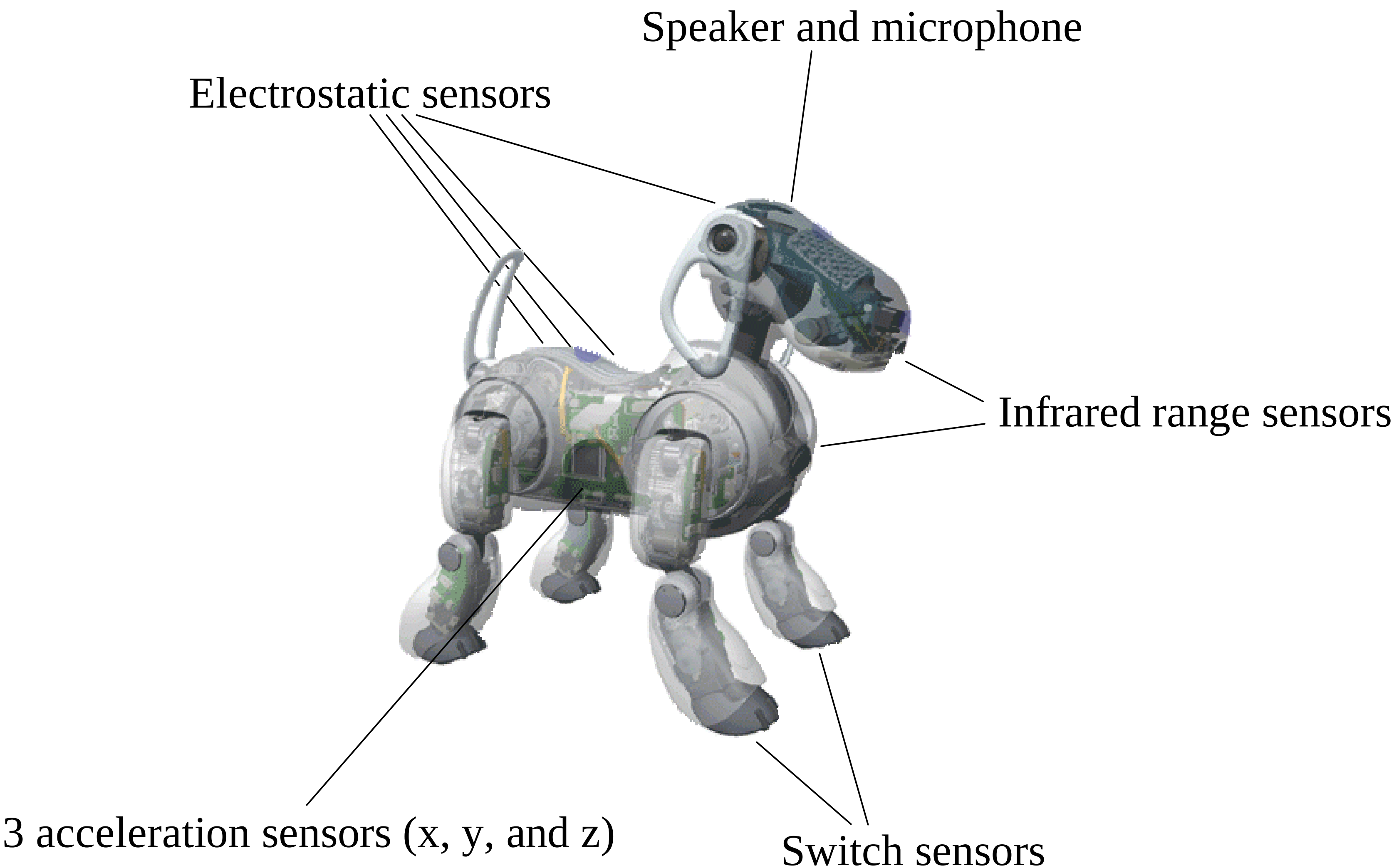
- Walks that “come with” Aibo are **slow**

- RoboCup soccer: 25+ Aibo teams internationally
  - **Motivates faster walks**

QuickTime™ and a
YUV420 codec decompressor
are needed to see this picture.

Hand-tuned gaits (2003)			Learned gaits		
German Team	UT Austin Villa	UNSW	Hornby et al. *(1999)*	Kim & Uther *(2003)*	Quinlan et al. *(2003)*
230 mm/s	**245**	**254**	**170**	**270**	**296**

The Robot: Sony Aibo (ERS-210A and ERS-7)



The Robot: Sony Aibo (ERS-210A and ERS-7)

Color camera

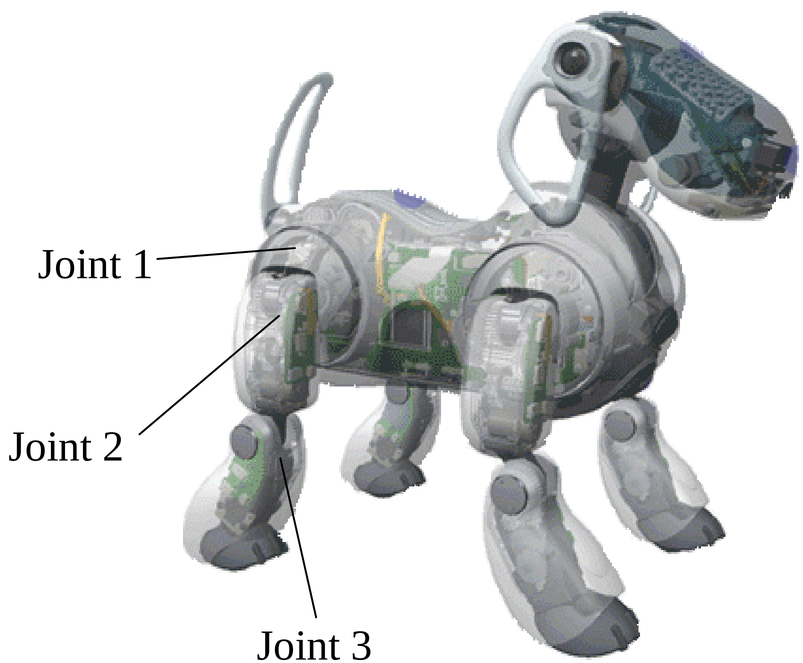
- Resolution: 208 x 160
- 30 frames per second

Wireless ethernet
(802.11b)

- **On-board processor**
    - 576 MHz
    - 64 MB RAM
- OS: Aperios + Open-R
- Programming Language: C++

The Robot: Sony Aibo (ERS-210A and ERS-7)

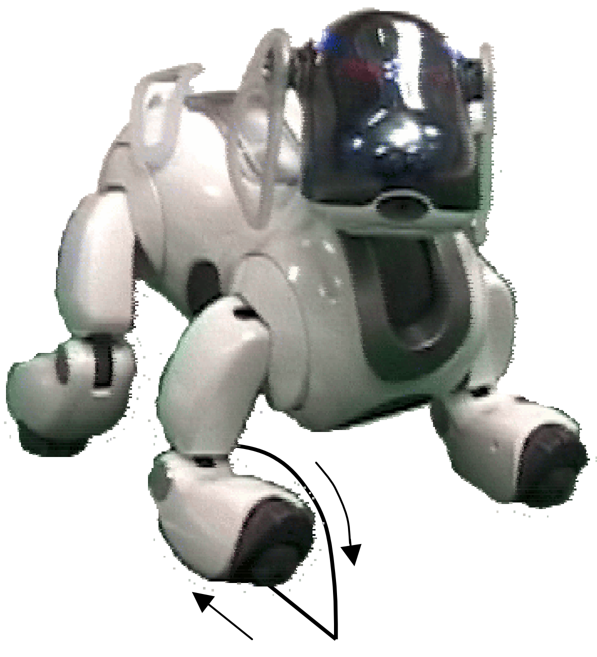
20 degrees of freedom

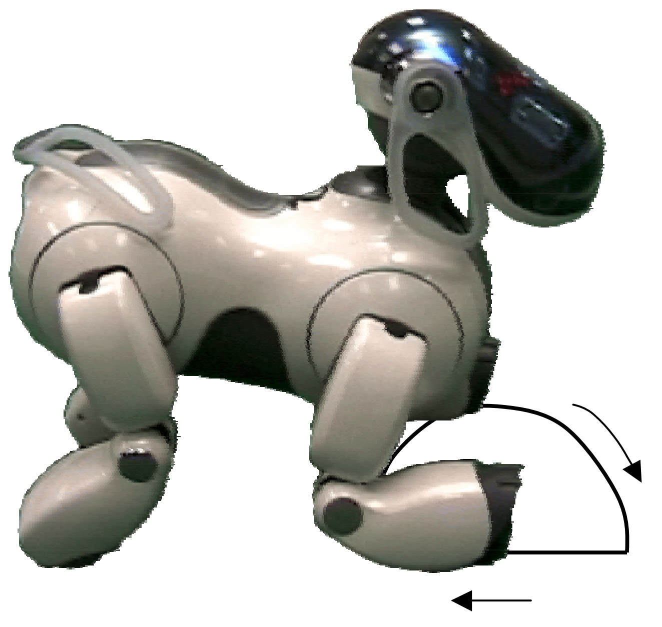


- head: 3 neck, 2 ears, 1 mouth
- **4 legs: 3 joints each**
- tail: 2 DOF

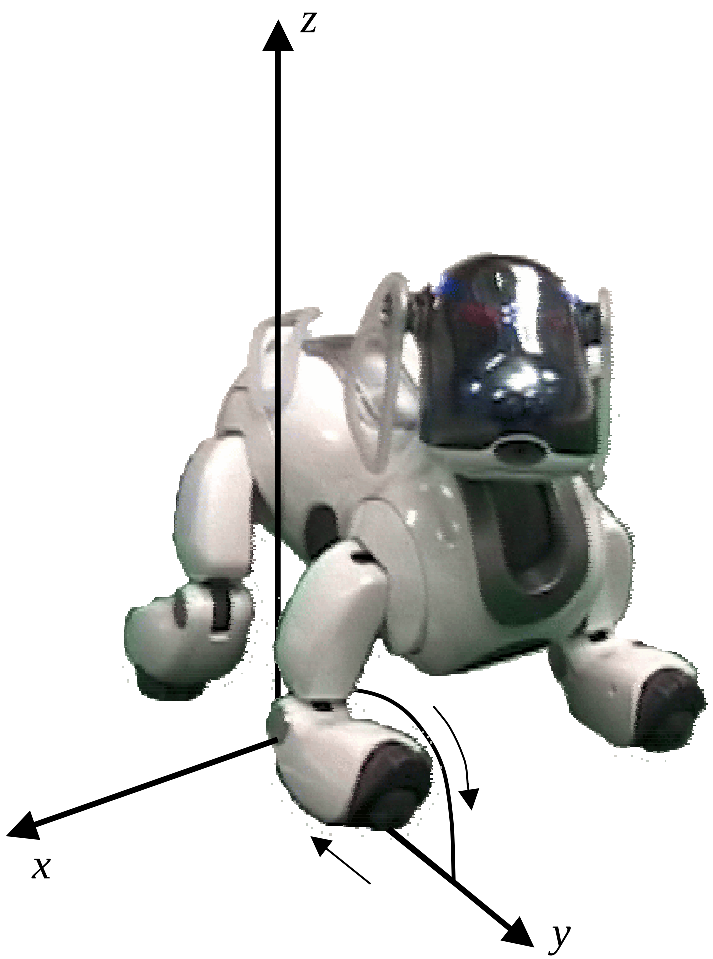
A Parameterized Walk

- Developed **from scratch** as part of UT Austin Villa 2003
- **Trot gait** with half-elliptical locus for each leg





A Parameterized Walk



Locus Parameters:

1. Ellipse length
2. Ellipse height
3. Position on the x axis
4. Position on the y axis

12 continuous parameters

Experimental Setup

- Training Scenario
  - Robots **time themselves** while traversing a fixed distance
  - Off-board computer **collects results, assigns policies**
  - Multiple traversals (3) per policy to account for **noise**
  - **Multiple robots** evaluate policies simultaneously

No human intervention except battery changes

QuickTime™ and a
YUV420 codec decompressor
are needed to see this picture.

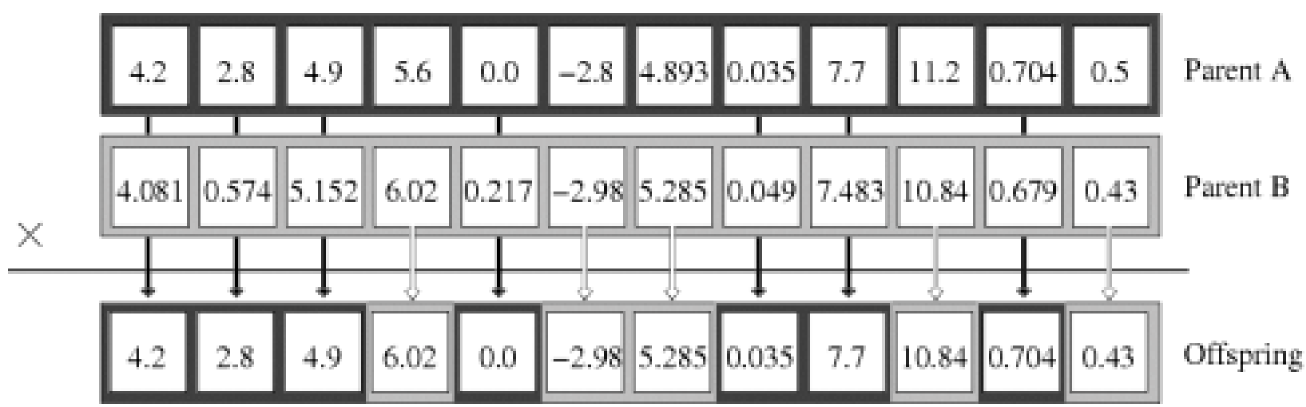
Learning Algorithms

How to find a good policy?

- Genetic Algorithm
- Downhill Simplex Method
- Hill Climbing Algorithm
- Policy Gradient Algorithm

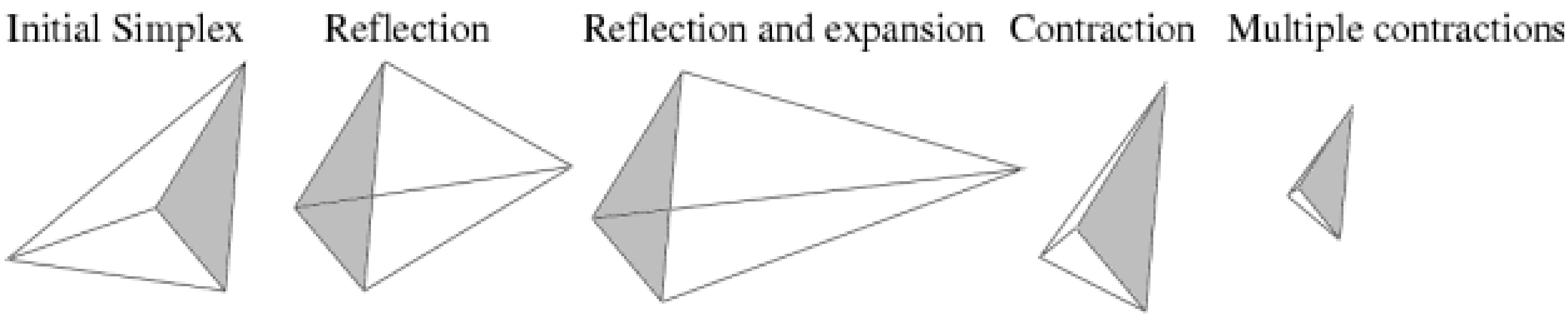
Genetic Algorithm

- Maintain a population of *t* policies
- Genetic operators of **mutation** and **crossover** explore policy space
- Offspring of good policies **replace bad policies**



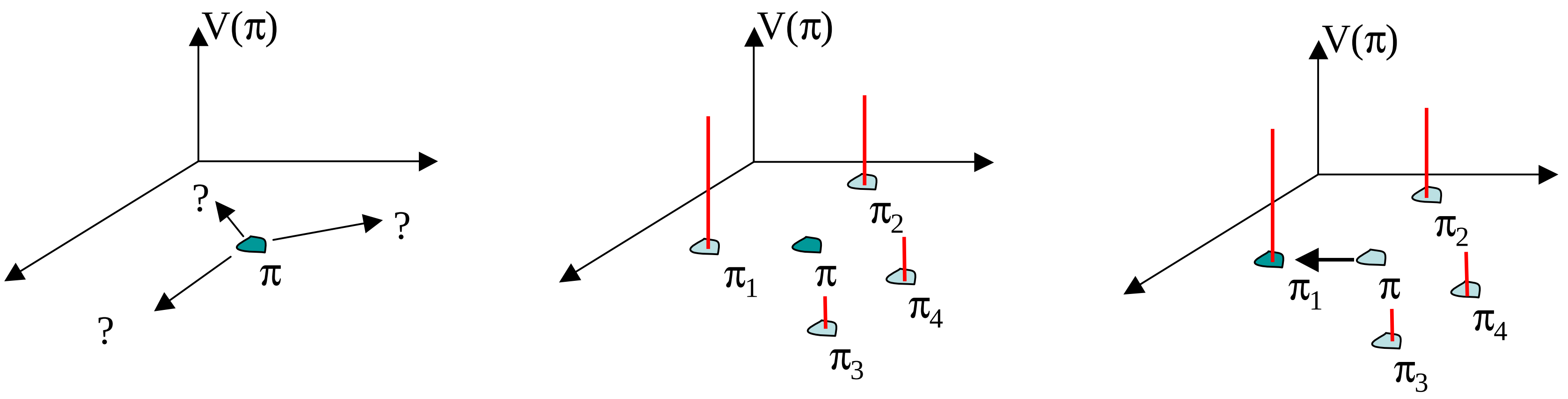
Downhill Simplex Method

- Maintain a simplex of *N+1* policies
- Different **transformations** move the simplex through policy space
- When the simplex becomes too small, **expand it**



Hill Climbing Algorithm

- Policy $\pi = \{\theta_1, \ldots, \theta_{12}\}$, $V(\pi)$ = **walk speed** when using π
- Evaluate t (15) policies in the neighborhood of π
- From π, move towards the best neighboring policy

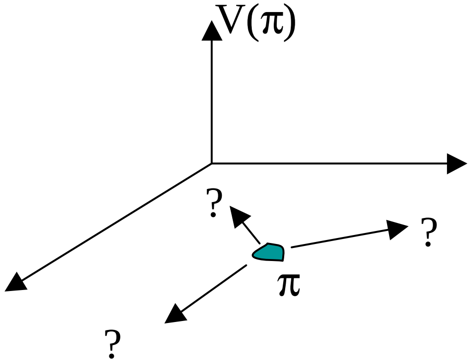


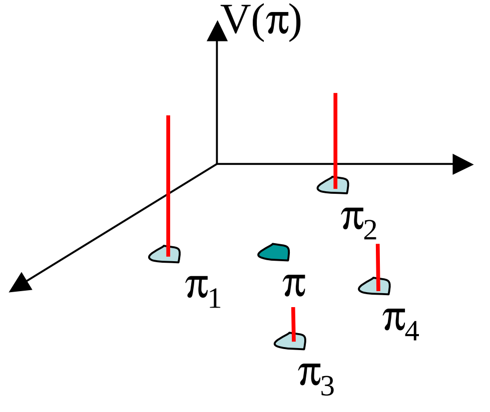
Policy Gradient RL

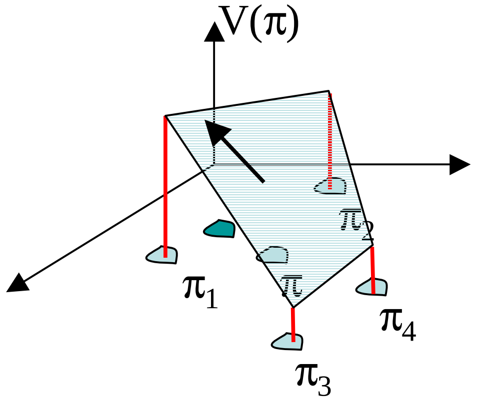
- Policy $\pi = \{\theta_1, \ldots, \theta_{12}\}$, $V(\pi)$ = **walk speed** when using π

- From π, move in the direction of the gradient of $V(\pi)$
  - Can't compute gradient directly: **estimate** empirically

- Evaluate neighboring policies to estimate gradient







Policy Gradient RL

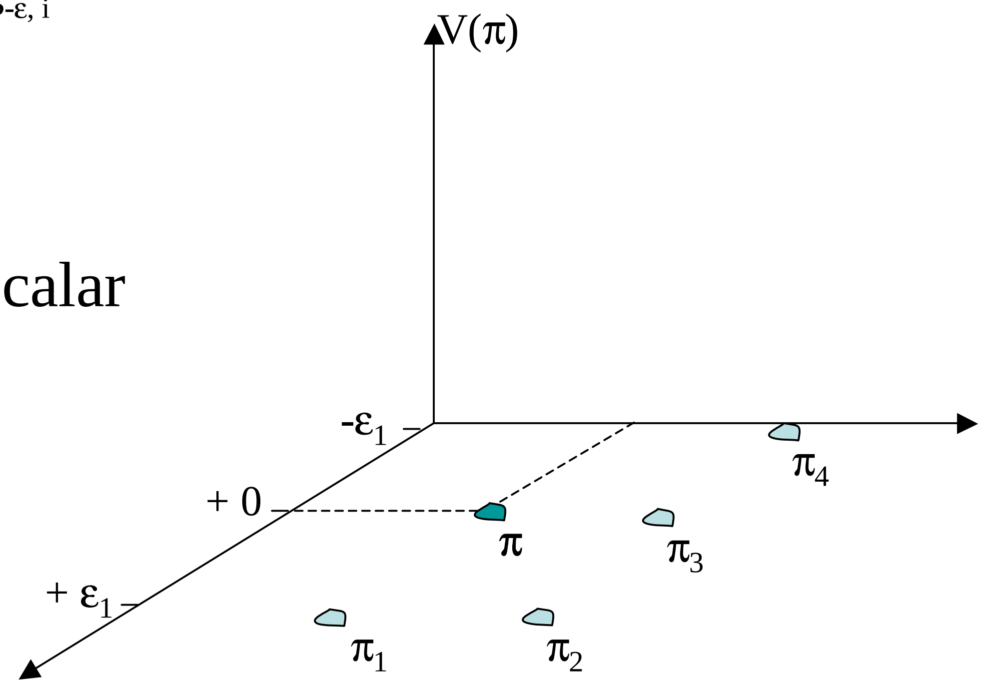
- Determine **3 average values** for each dimension

- Compute an adjustment vector *A*:

$$A_i = \begin{cases} 0 & \text{If } Avg_{+0,\, i} > Avg_{+\varepsilon,\, i} \text{ and } Avg_{+0,\, i} > Avg_{-\varepsilon,\, i} \\ Avg_{+\varepsilon,\, i} - Avg_{-\varepsilon,\, i} & \text{otherwise} \end{cases}$$

- **Normalize** *A*, multiply by a scalar step size η

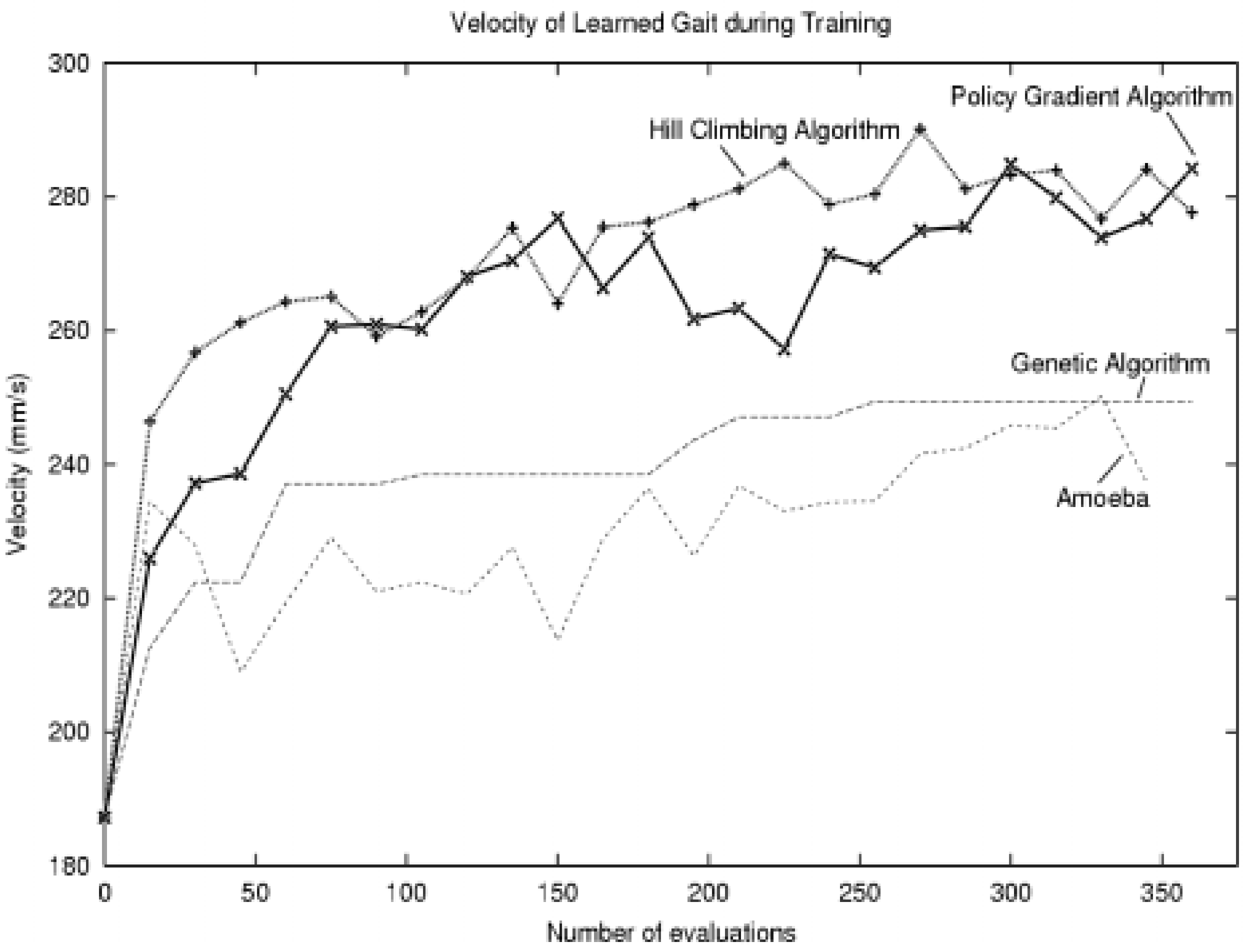
- $\pi = \pi + \eta A$



Results

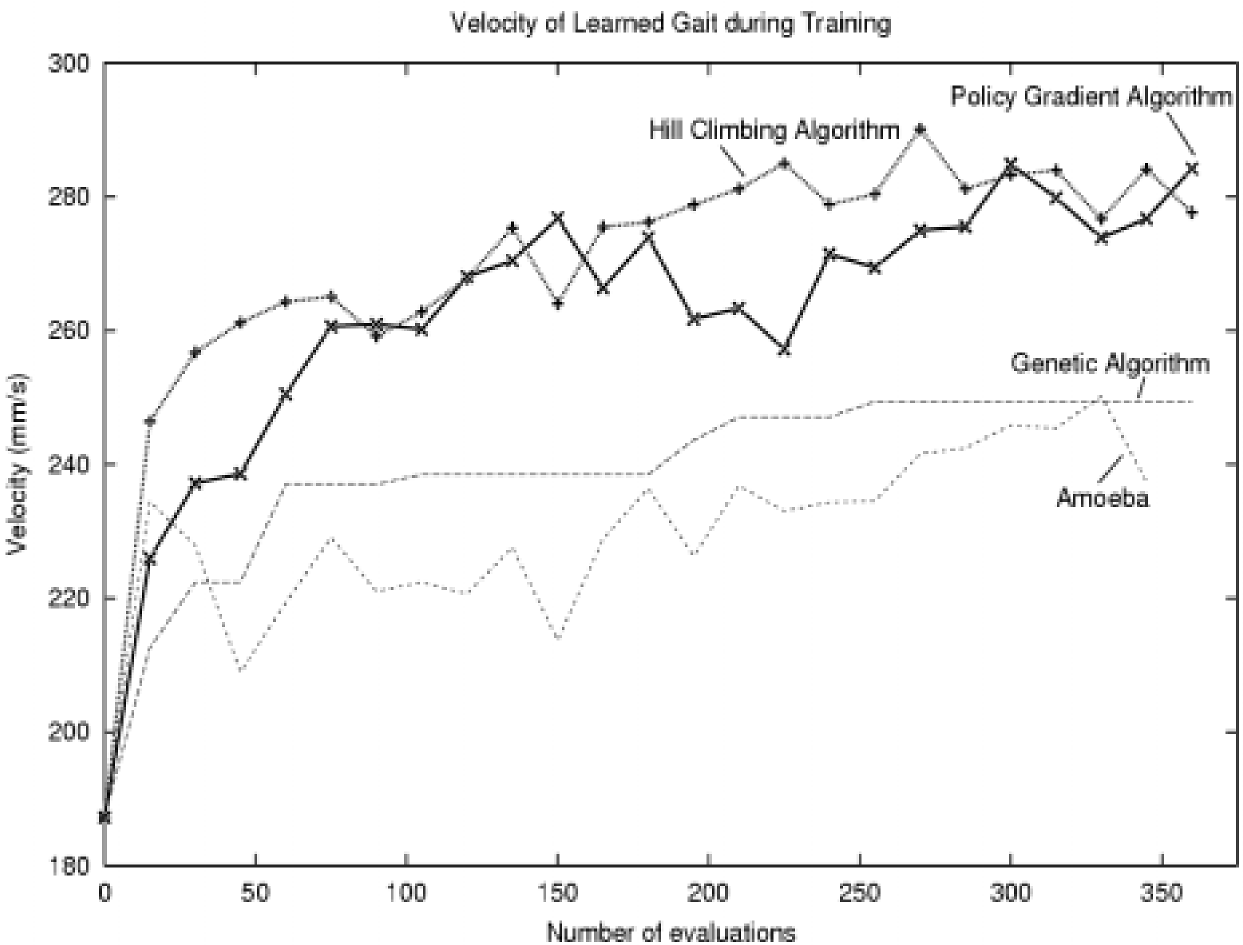
	ERS-210	ERS-7
Before:	QuickTime™ and a YUV420 codec decompressor are needed to see this picture.	QuickTime™ and a decompressor are needed to see this picture.
After:	QuickTime™ and a YUV420 codec decompressor are needed to see this picture.	QuickTime™ and a decompressor are needed to see this picture.

Results



- 24 iterations = **1080 field traversals ≈ 3 hours**
- Additional iterations didn't help

Results

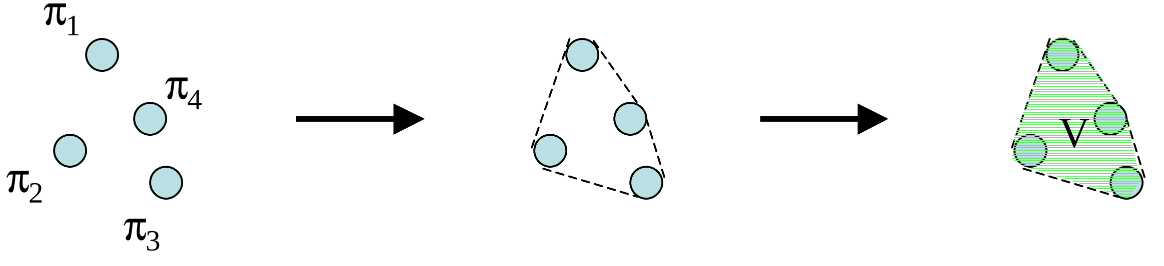


Why do the simpler algorithms do better?

Analysis

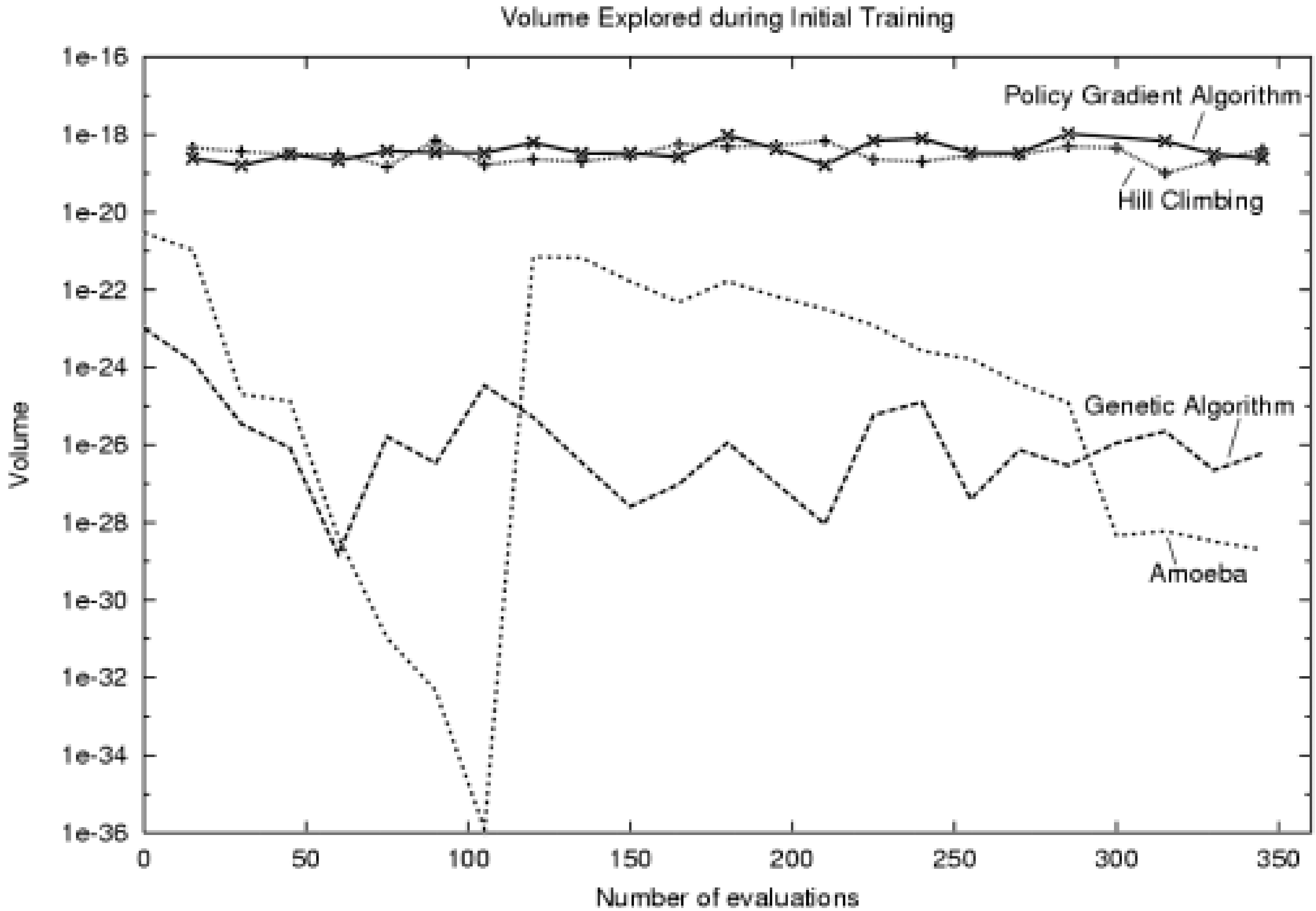
Why do the simpler algorithms do better?

- Rate of exploration
  - Analyze how much of the policy space was explored



  - How does V change over time?

Analysis - rate of exploration



Simpler methods do more exploration

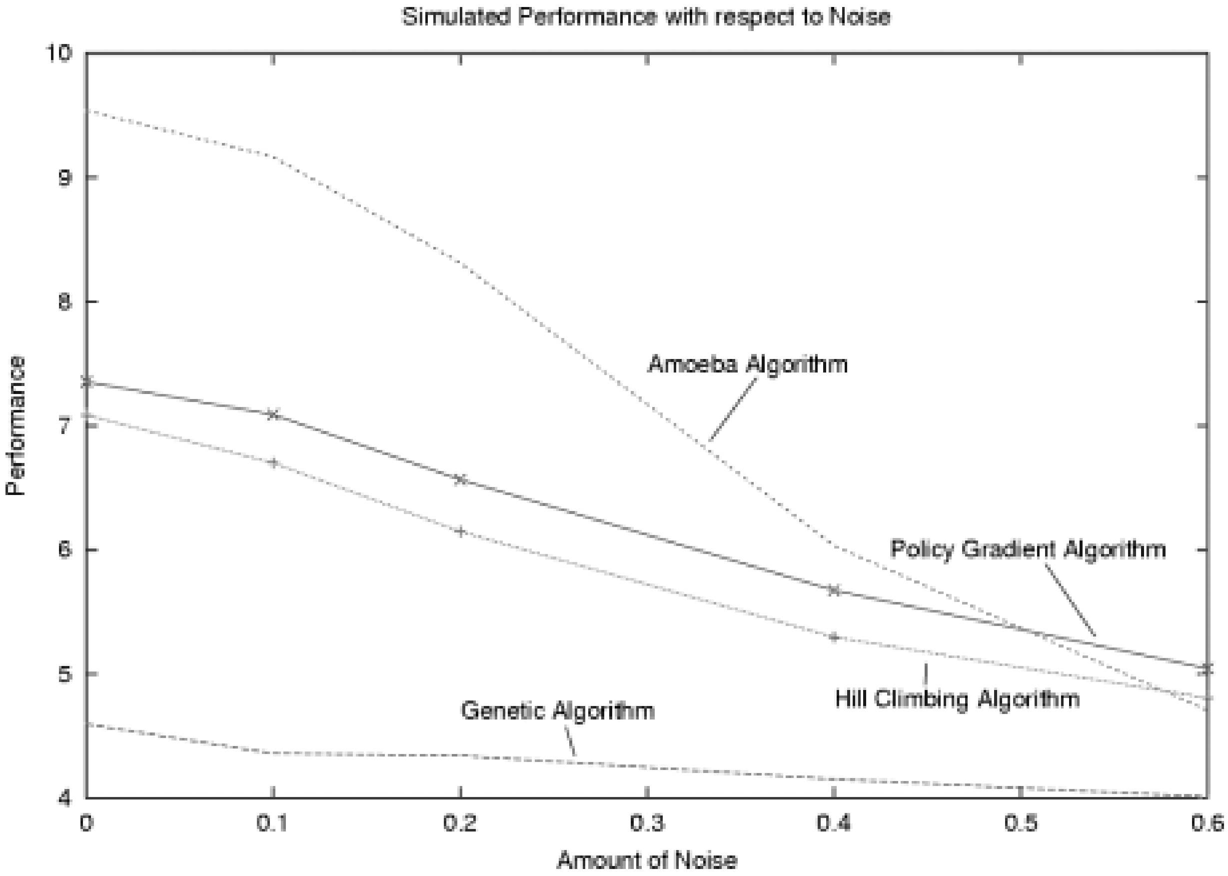
Analysis

Why do the simpler algorithms do better?

- Robustness to noise
  - Examine a problem with different amounts of noise

1) Replace objective function with set of 10 mathematical functions
2) Add a variable amount of noise

Analysis - performance with varying noise



Amoeba does better with less noise

Learned Parameters

Parameter	Initial Value	ε	Best Value
Front ellipse:			
(height)	4.2	0.35	4.081
(x offset)	2.8	0.35	0.574
(y offset)	4.9	0.35	5.152
Rear ellipse:			
(height)	5.6	0.35	6.02
(x offset)	0.0	0.35	0.217
(y offset)	-2.8	0.35	-2.982
Ellipse length	4.893	0.35	5.285
Ellipse skew multiplier	0.035	0.175	0.049
Front height	7.7	0.35	7.483
Rear height	11.2	0.35	10.843
Time to move through locus	0.704	0.016	0.679
Time on ground	0.5	0.05	**0.430**

Practical Questions

- Can it apply directly to **omnidirectional gaits**?
- Does **individualizing** per robot help?
- Can we optimize for **stability** too?
- How well will it work on **other platforms**?
- Can it work **out of the lab**?

Related Work

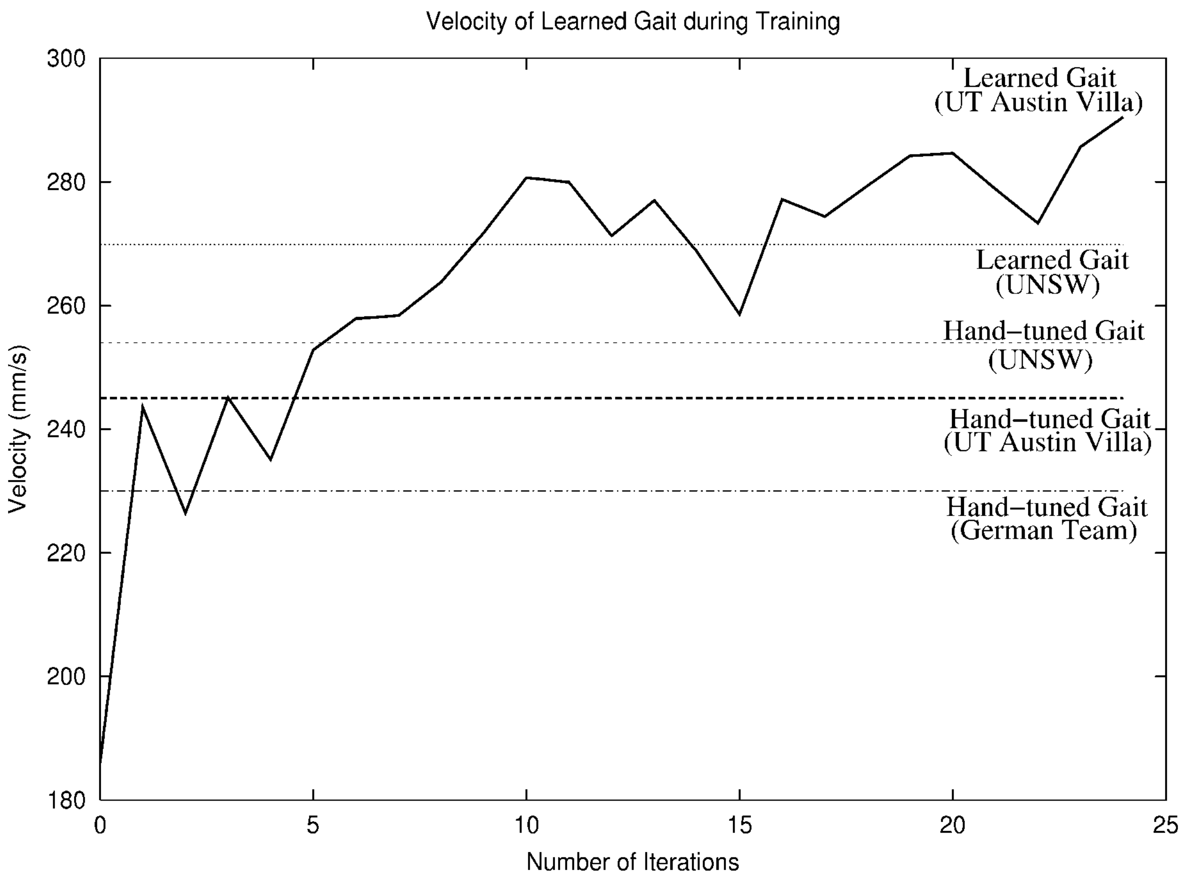
- Learning gaits for the Aibo: Hornby et al (2000), Kim & Uther (2003), Quinlan et al (2003)
- Helicopter flight: Ng et al (2004), Bagnell & Schneider (2001)
- EA for a biped robot: Zhang and Vadakkepat (2003)

Summary

- Used machine learning to **generate fast Aibo walk**
- Compared four ML algorithms
- All learning done **on real robots**
- **No human intervention** (except battery changes)

http://www.cs.utexas.edu/users/AustinVilla/legged/learned-walk/

Results



Velocity of Learned Gait during Training
Velocity (mm/s)
Number of Iterations
300
280
260
240
220
200
180
0
5
10
15
20
25
Learned Gait (UT Austin Villa)
Learned Gait (UNSW)
Hand−tuned Gait (UNSW)
Hand−tuned Gait (UT Austin Villa)
Hand−tuned Gait (German Team)

Experiments

- Started from **stable**, but fairly slow gait
  - Used small ε's, η = 2.0

- Used **3 robots** simultaneously
  - Can be **distributed** if share knowledge of t, ε's, η
  - Each robot picks own random policies to evaluate

- Each iteration takes 45 traversals, about 7 minutes