

Back to Evolutionary Robotics

Blackboard Demo

- Fitness functions for obstacle avoidance behaviour for the Khepera robot
- Typical “trap” behaviours requiring refinement of fitness func:
 - staying still
 - jittering on the spot
 - only ever turn in one direction



Fitness Evaluation is a problem!

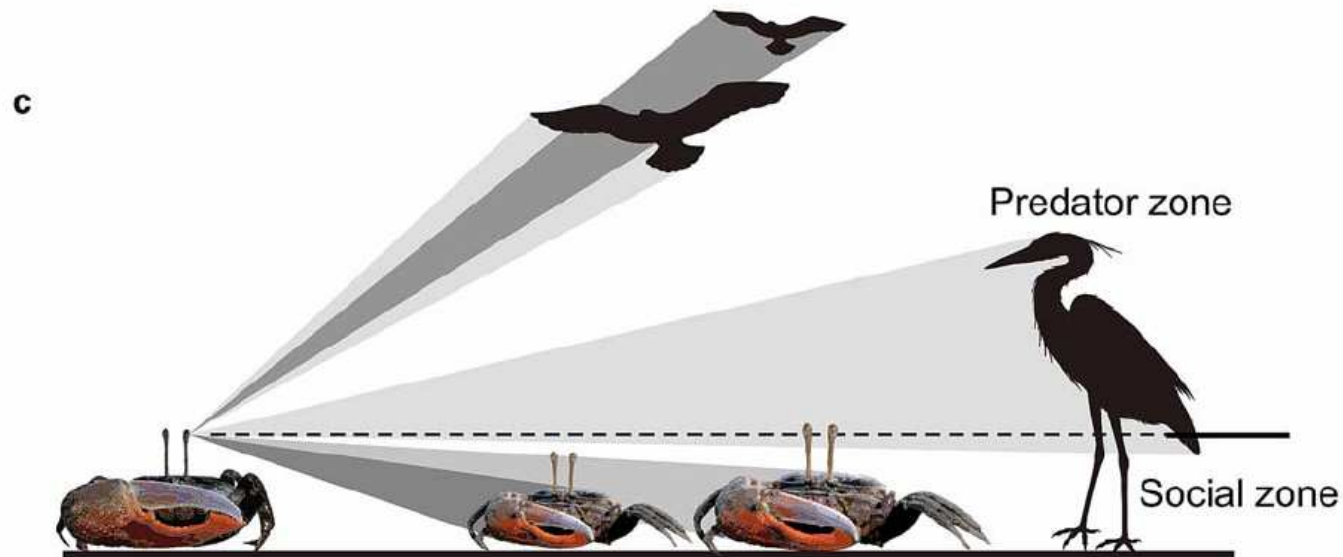
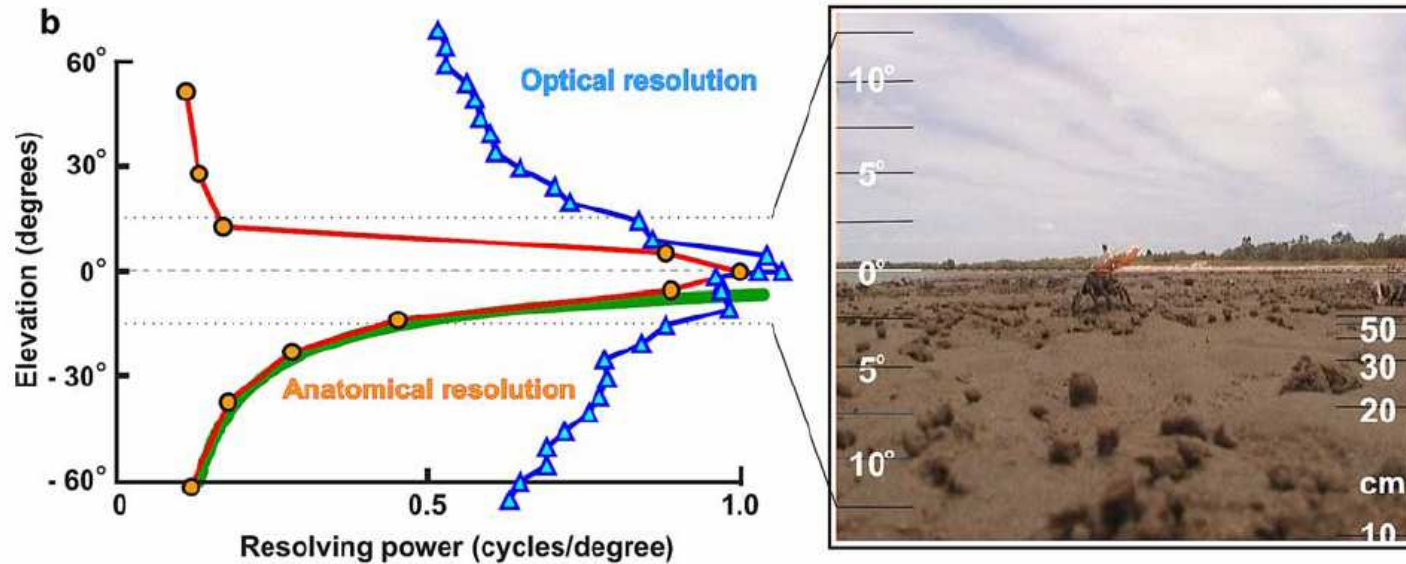
- Designing an adequate test environment and measurement protocol that guides evolution to what you had in mind is difficult. Often evolution:
 - gets stuck because there is no path of variations incurring non-decreasing fitness that leads to what you want
 - always falls into a simple way of getting mediocre fitness
 - finds an unanticipated way of getting good fitness
 - produces a good behaviour in the test environment(s) that doesn't generalise to others

Fitness Evaluation is a problem! (contd)

- Usually need several test environments/starting positions: take the average score, or the worst, or some other function of the individual performances
- Actually, we'd rather not say exactly what the behaviour should be: can we just measure something like "survival"?

Eg. Specialised vision: fiddler crabs





From Zeil&Hemmi, J Comp Physiol A (2006) 192: 1–25 “The eyes of fiddler crabs have clearly not evolved to efficiently relay all available image information, as is frequently assumed... but to detect biologically significant events“

Eg. The Sussex Δ/\blacksquare Robot (seminar 4)

- Genotype specifies DRNN, and which patches of pixels from the camera were averaged to form inputs to the network.

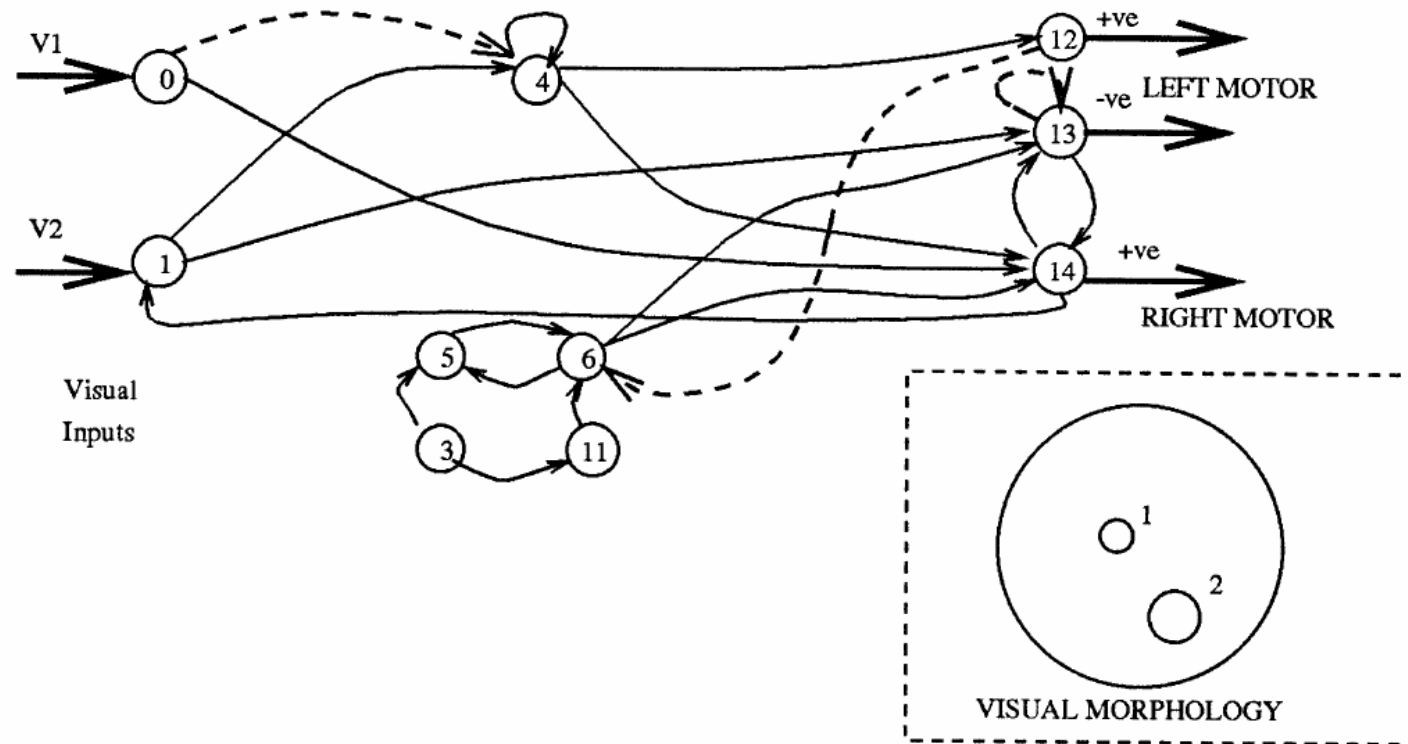
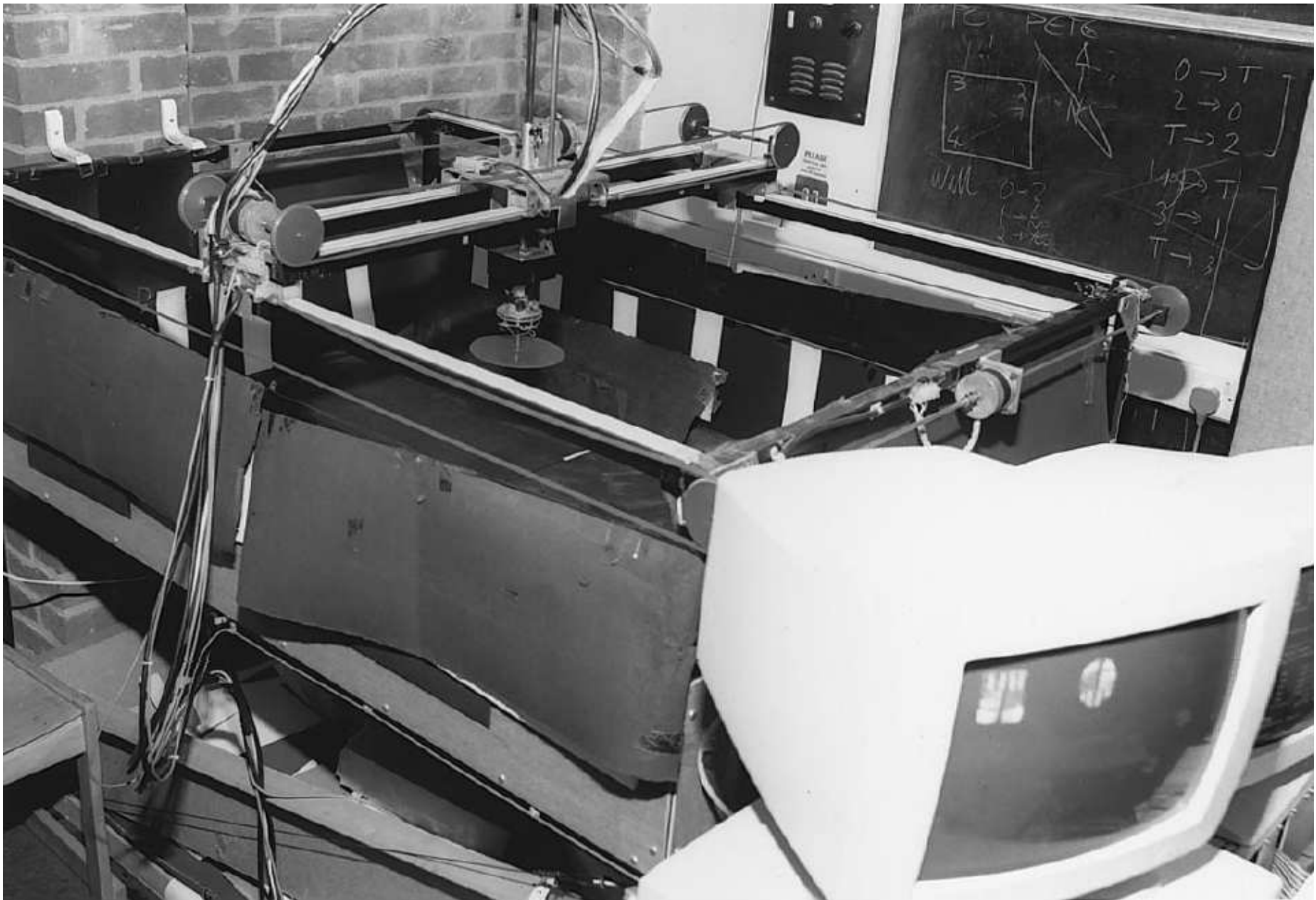
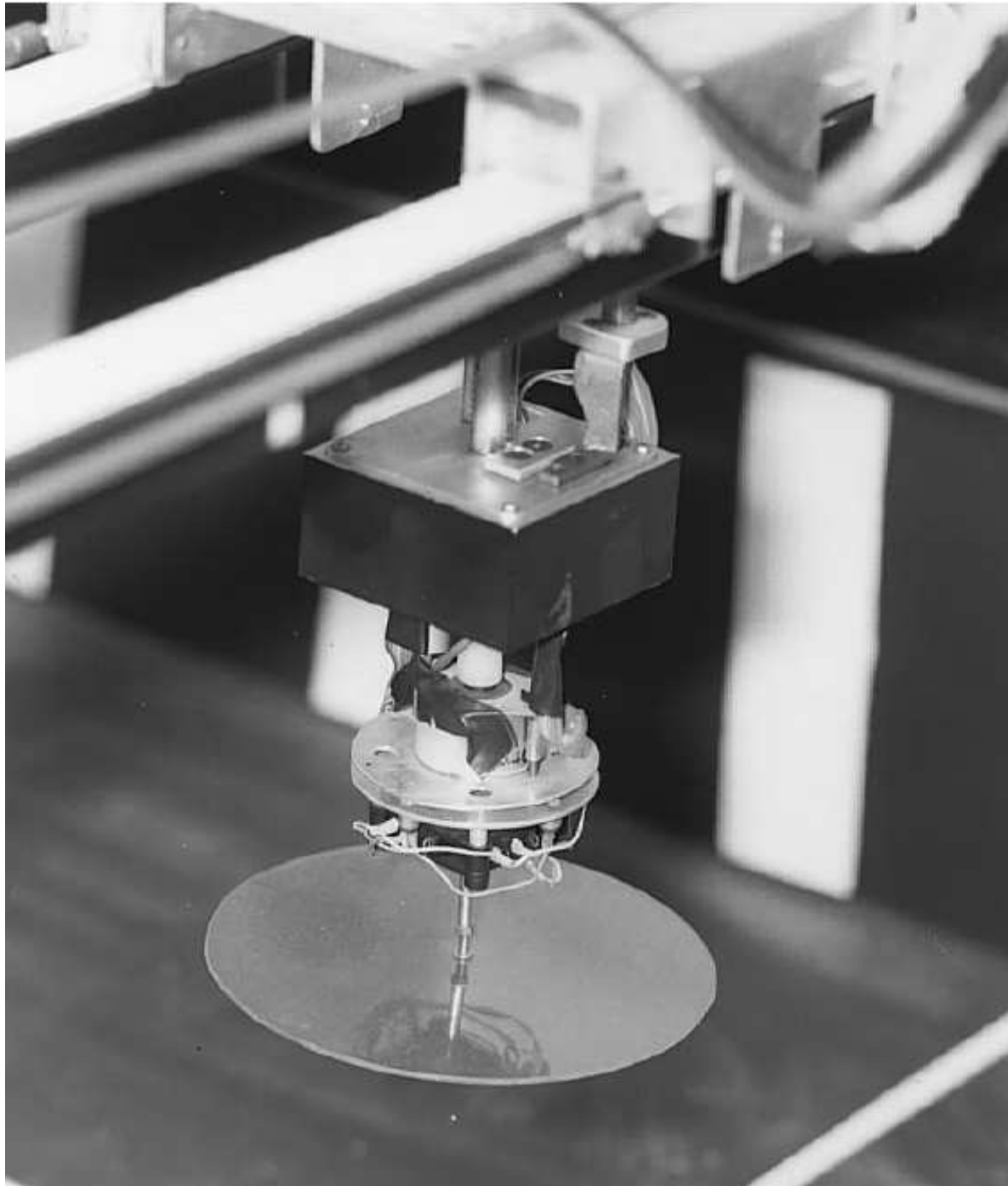


FIG. 10. Active part of the control system that generated fit behavior for the rectangle and triangle experiment. Visual morphology shown in the inset.





Eg. The Sussex Δ/\blacksquare Robot (seminar 4)

- Real vision, simulated body, actuators & brain
- **Incremental** evolution. Fitness function changed in the sequence:
 1. Forward movement
 2. Movement towards large white target
 3. Movement towards small white target
 4. Movement towards white triangle *not* white square



Fitness functions for Δ/\blacksquare

Incremental evolution: fitness function was changed in the sequence:

- Forward movement – randomly generated individuals were informally observed and some picked to seed the initial population.
- Big then small targets:

$$\epsilon_2 = \sum_{i=1}^{i=20} (-d_i),$$

where d_i gives the distance to the target at 20 evenly-spaced observations over a 25 second trial. Repeat from four different starting orientations and take the worst score.

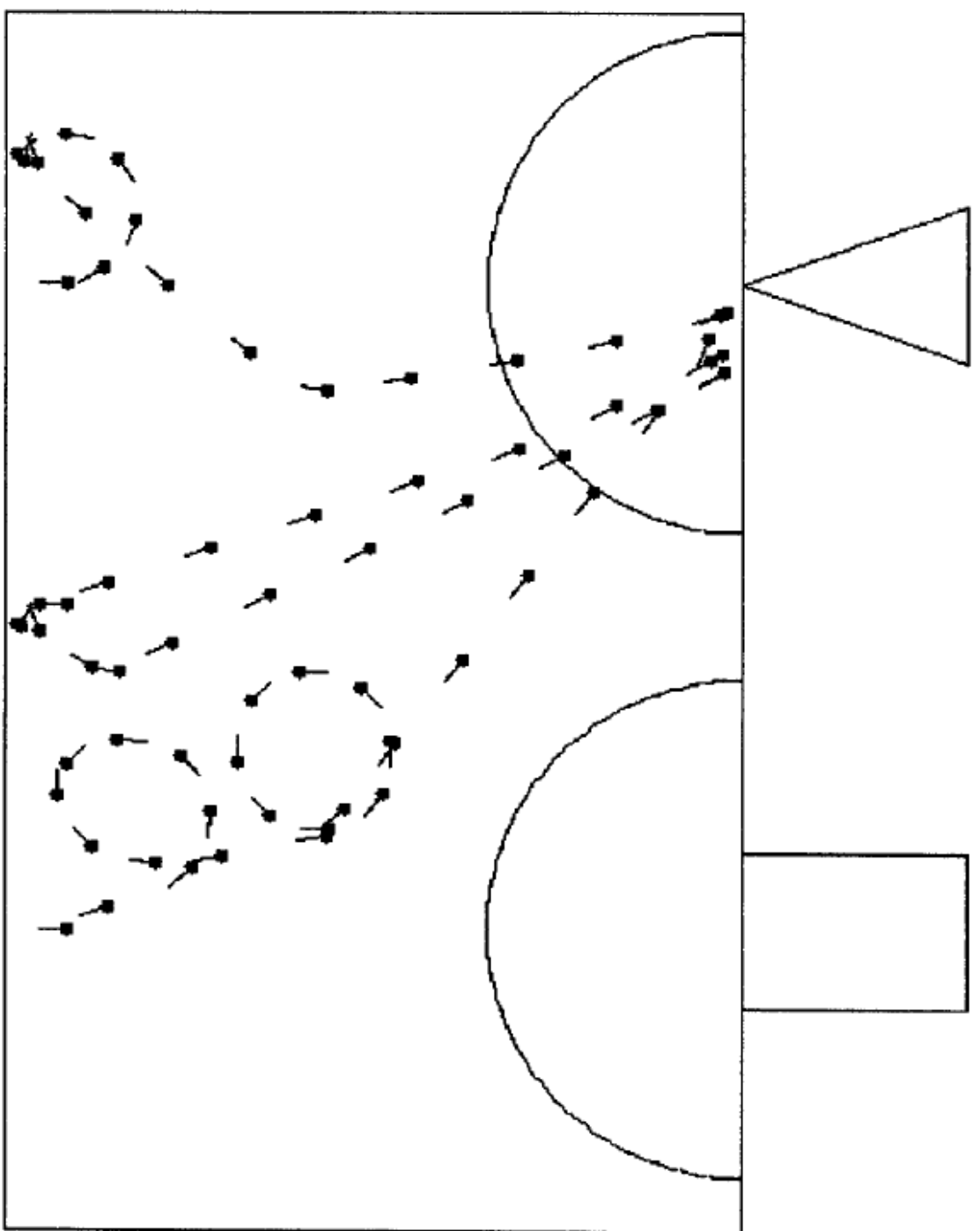
- Triangle not square :

$$\epsilon_3 = \sum_{i=1}^{i=20} [\beta(D_{1_i} - d_{1_i}) - \sigma(D_{2_i}, d_{2_i})]$$

Where D_1 , D_2 are the distances of the robot's starting position from the triangle and square (resp); d_1 and d_2 are the distances of the robot from the triangle and square (resp).

$\beta = (D_1 - d_1)$ unless d_1 is less than some threshold T , in which case $\beta = 3(D_1 - d_1)$

Penalty function $\sigma = 0$ unless $d_2 < T$, in which case $\sigma = I - (D_2 - d_2)$ where I is the distance between the targets.



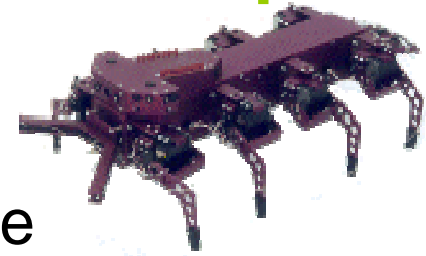
Note

- Fitness functions typically use information not available to the robot control system.
- Fitness functions are typically developed in response to several unsuccessful evolutionary runs – you wouldn't think of the previous slides as your first idea!
- Need *generalisation* over a range of conditions. Eg. Evolved a system that only worked in the morning: ended up having to build disco lights to give variable lighting conditions during evolution!

The simulation/reality dilemma

- Simulations can be faster, more flexible (eg. allowing full body morphology evolvability), and won't catch fire overnight.
- Unless special steps are taken, systems evolved in simulation won't work in reality (a kind of generalisation problem)
- Just adding noise in the simulation won't work: the systems can evolve to rely on the noise!
- Need to carefully perform multiple simulated trials with varying noise and parameters to force the evolved system to use only the properties you want: needs human analysis of the task, and brings unwanted constraints.
 - The extreme is Nick Jakobi's "Minimal Simulation" technique ..\Resources\jakobi_octopod.mpg
 - There is still a lot to be said for evolutionary trials in physical reality

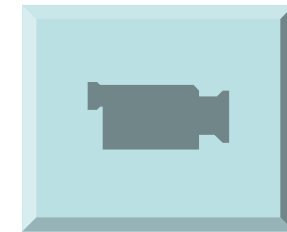
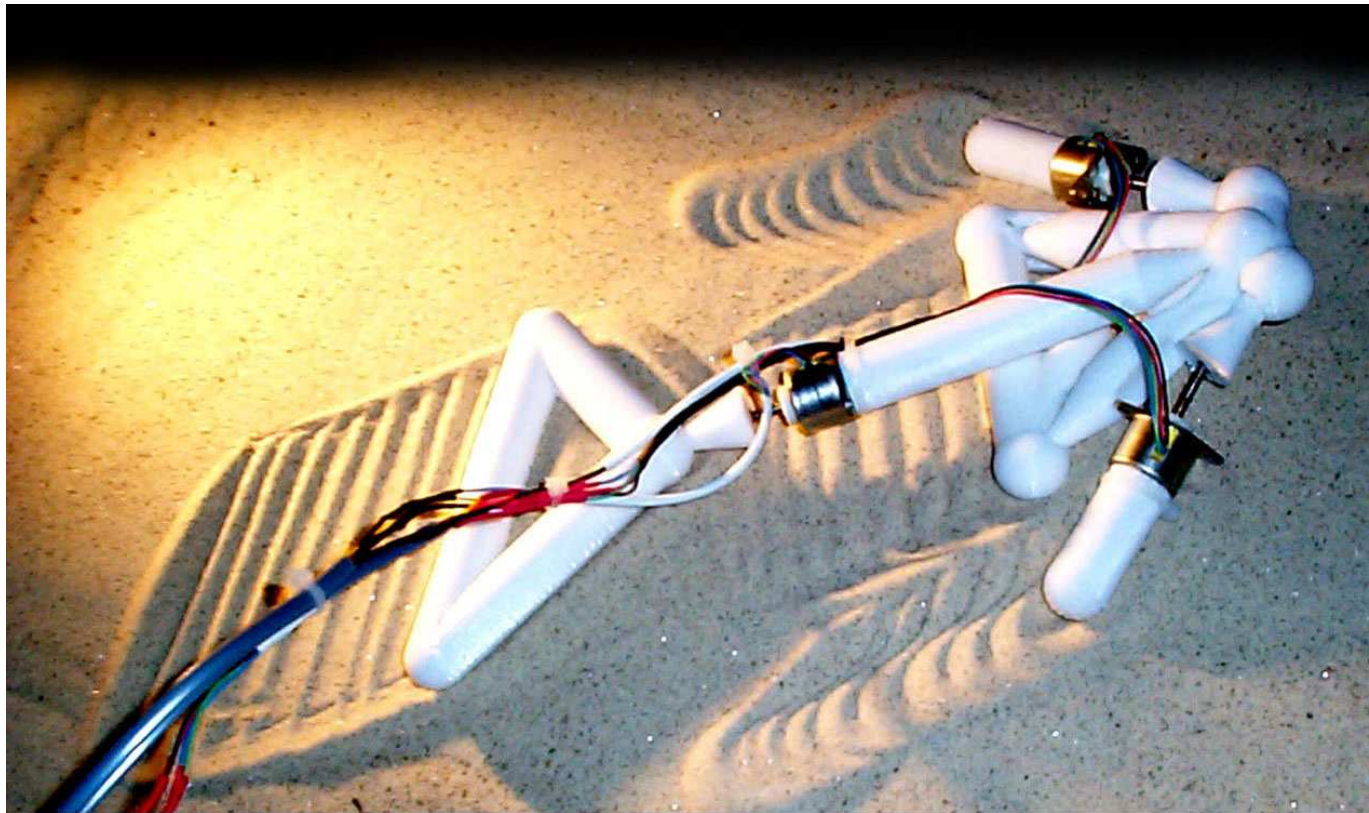
Nick Jakobi's Octopod Example



- “Minimal Simulation”
- Simulation is like swimming through treacle
- If any leg is moving backwards/forwards, it contributes to pushing that side of the robot forwards/backwards
- The size of the push is stronger the lower the foot is
- Only “optimal” treacle-swimming will give walking in reality: everything else is an unrealistic sim. to guide evolution towards this goal
 - it’s only the final robot we test in the real world
 - sim. is task-specific, and designing it needs human analysis
 - actually, this robot is too aggressive on its motors and damages them: fitness function needs something to reduce stress on motors somehow

Genetically Organized Lifelike Electro Mechanics (GOLEM)

Hod Lipson (Cornell) & Jordan Pollack (Brandeis)



H. Lipson and J. B. Pollack (2000), "Automatic design and Manufacture of Robotic Lifeforms", *Nature* 406, pp. 974-978

Week 8 Seminar

- See NSAI web page
- Read extracts from “Artificial Evolution: A new path for Artificial Intelligence” based on early Sussted evolutionary robotics work
 - future potential? Limitations?
 - Questions? Ideas?
- **Plus:** get further help on GAs/Backprop