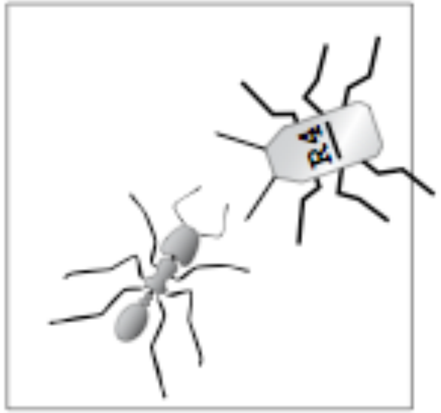
Evolution of Neurocontrollers

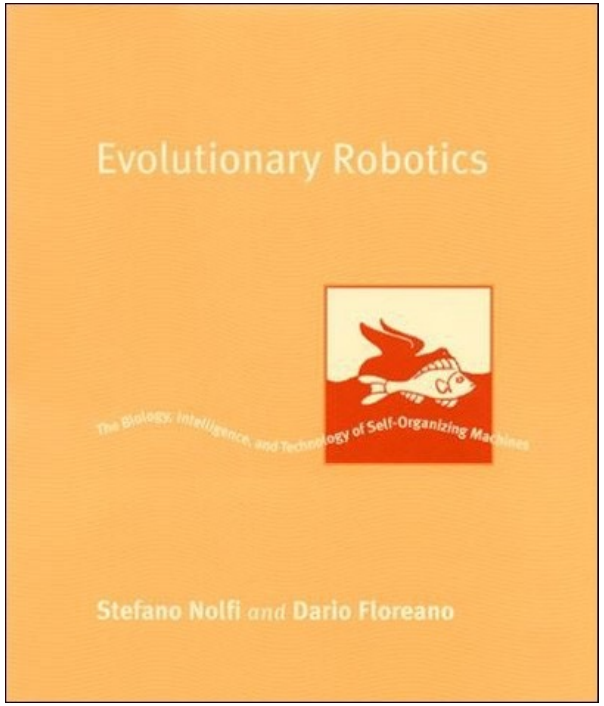


What you will learn in this class

- What is Evolutionary Robotics used for
- Genetic encodings of neural controllers
- Set up, carry out, and analyze a robotic experiment
- Evolution of vision-based neuro-controllers
- Analysis of evolved spiking neural networks
- Feature detection and active vision for neural controllers
- Comparing fitness functions: The Fitness Design Space
- Evolutionary control vs Reinforcement Learning

Evolutionary Robotics

Evolutionary Robotics is the automated generation of robot control systems* and morphologies by means of artificial evolution (Nolfi & Floreano, MIT Press, 2000)



Two motivations

Engineering: a tool to investigate the space of possible control strategies and body design

Biology: A *synthetic* (as opposed to *analytic*) approach to the study of mechanisms of adaptive behavior in machines and animals (Braitenberg, 1984)

**The control systems are often neural networks*

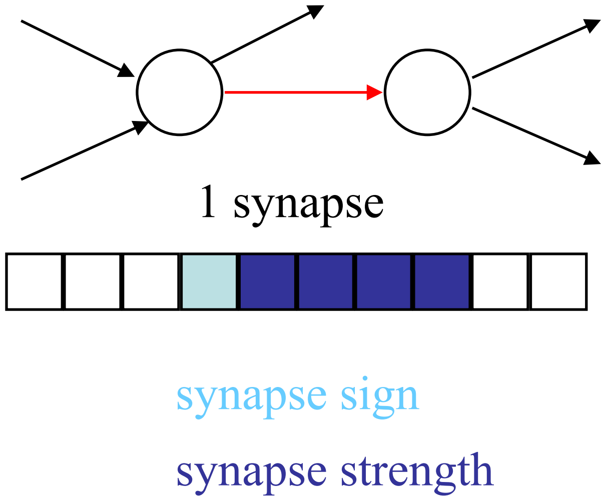
Genome can encode

1. <u>Connection Weights</u>
   a. pre-defined neural network architecture
   b. binary or real-valued representation of connection weights
   c. fixed-length genotype

2. <u>Learning Rules</u>
   a. pre-defined neural network architecture
   b. Binary or real-valued representation of learning rule
   c. Fixed-length genotype

3. <u>Topology</u>
   a. Neural network architecture created at birth
   b. Genotype encodes the parameters of a generative algorithm (program, L-System, neural network)
   c. Fixed-length or variable-length genotype

Evolution of connection weights

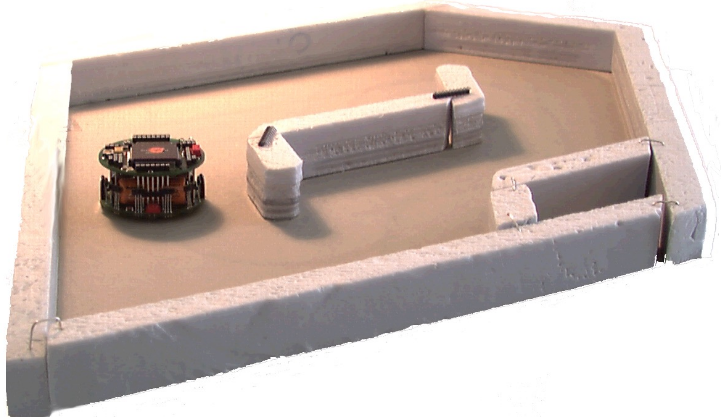


Binary encoding

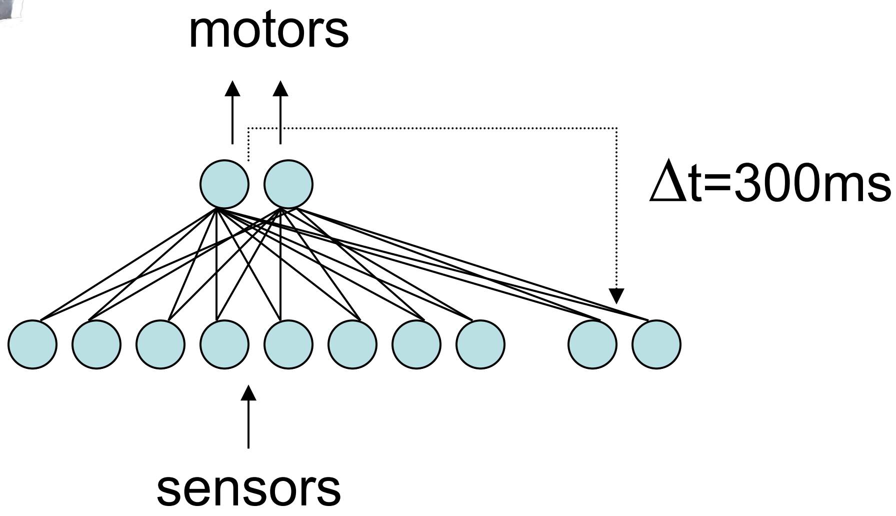
Fitness function is a measure of the robot behavior

Can be combined with neural network learning:
- learning starts from genetically encoded weights
- fitness measures performance of network after training
- learned weights are <u>not</u> written back into genome

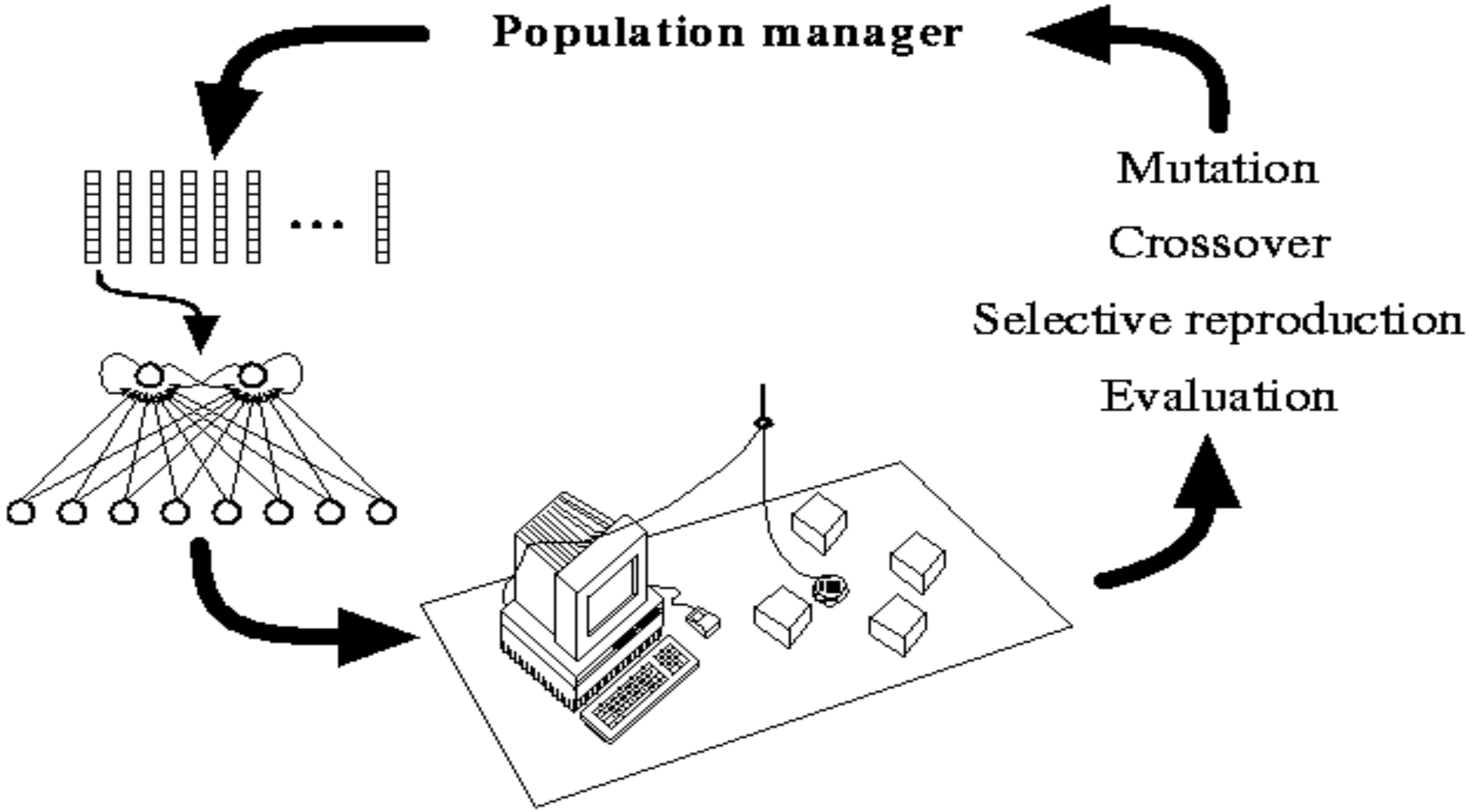
Collision-free Navigation

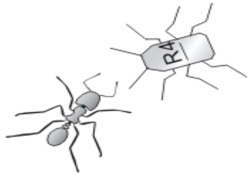


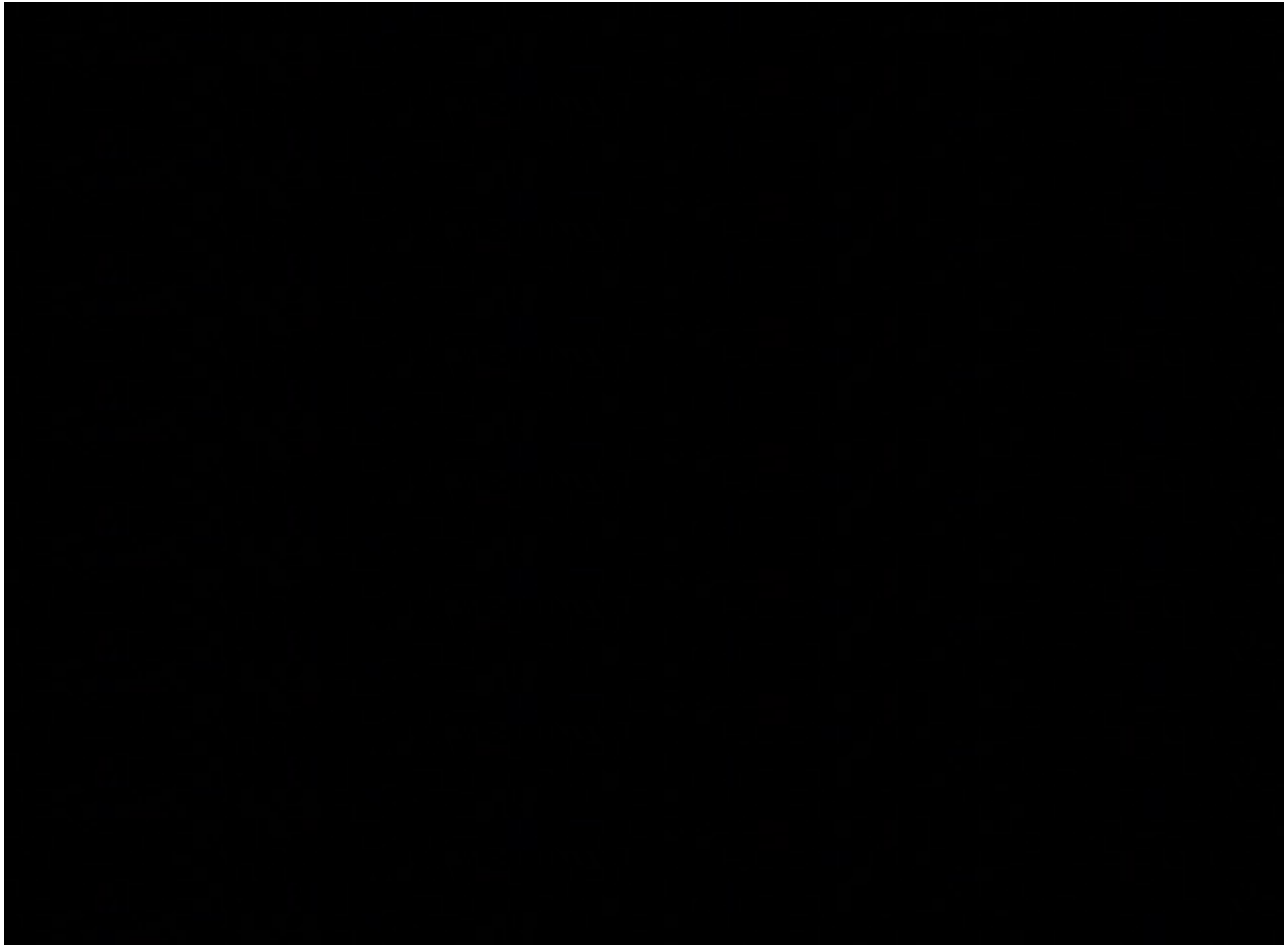
$$\text{Fitness} = V \times \Delta v \times (1-s)$$



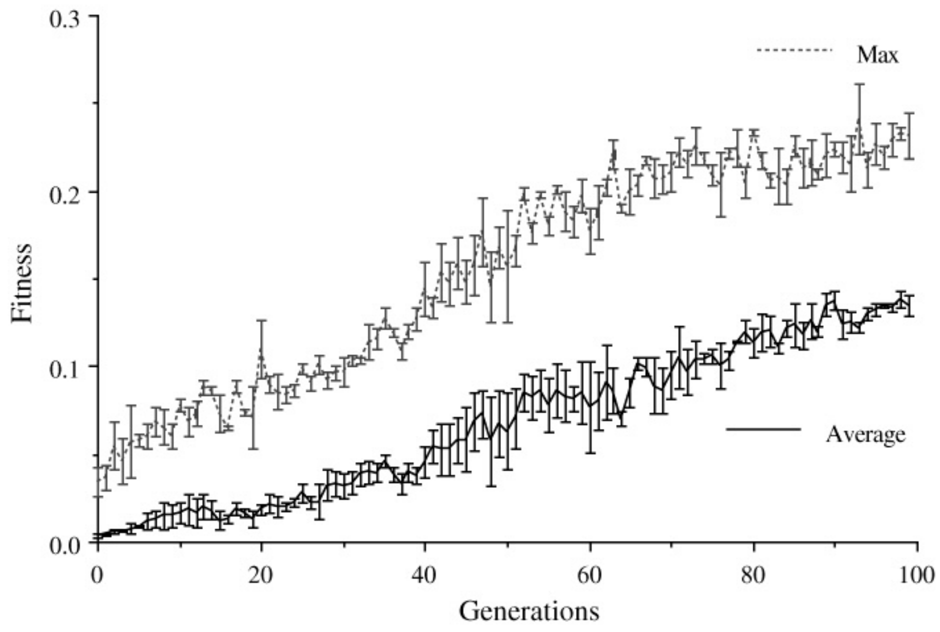
Methodology

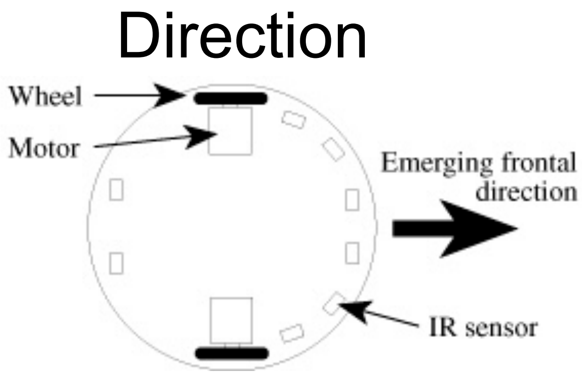






Results



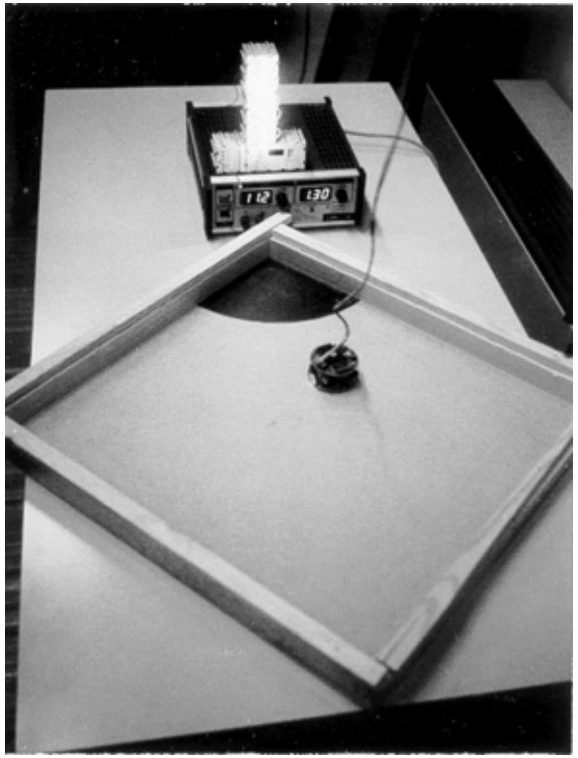


The average and best population fitness are typical measures of performance.

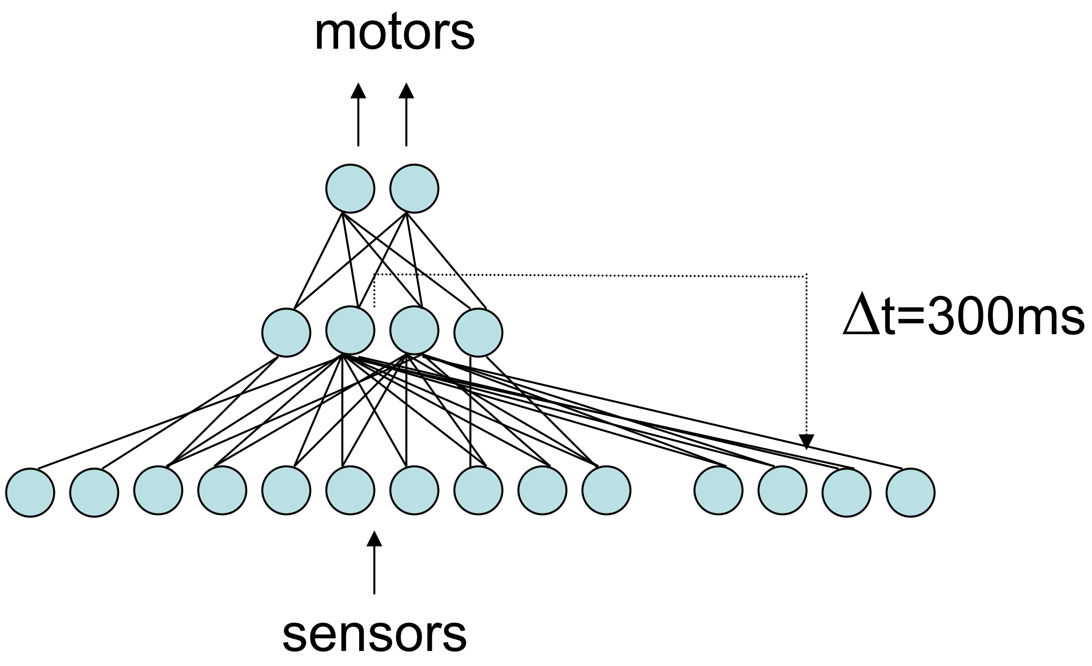
Evolved robots always have a preferential direction of motion and speed.

Homing for Battery Charge

Let us now put the robot in a more complex environment and make the fitness function even simpler. The robot is equipped with a battery that lasts only 20 s and there is a battery charger in the arena.

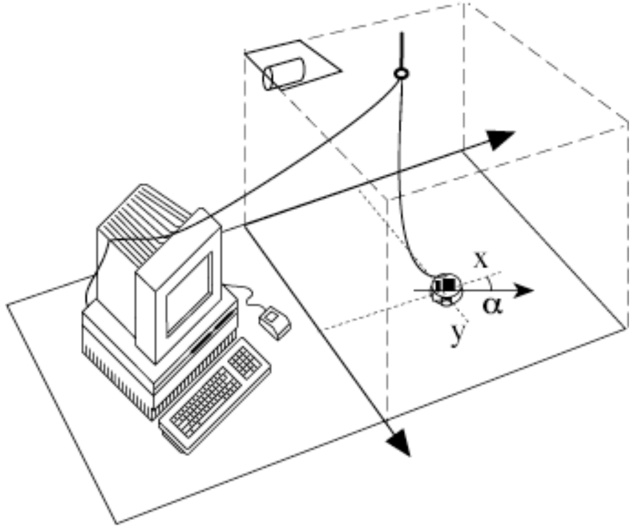


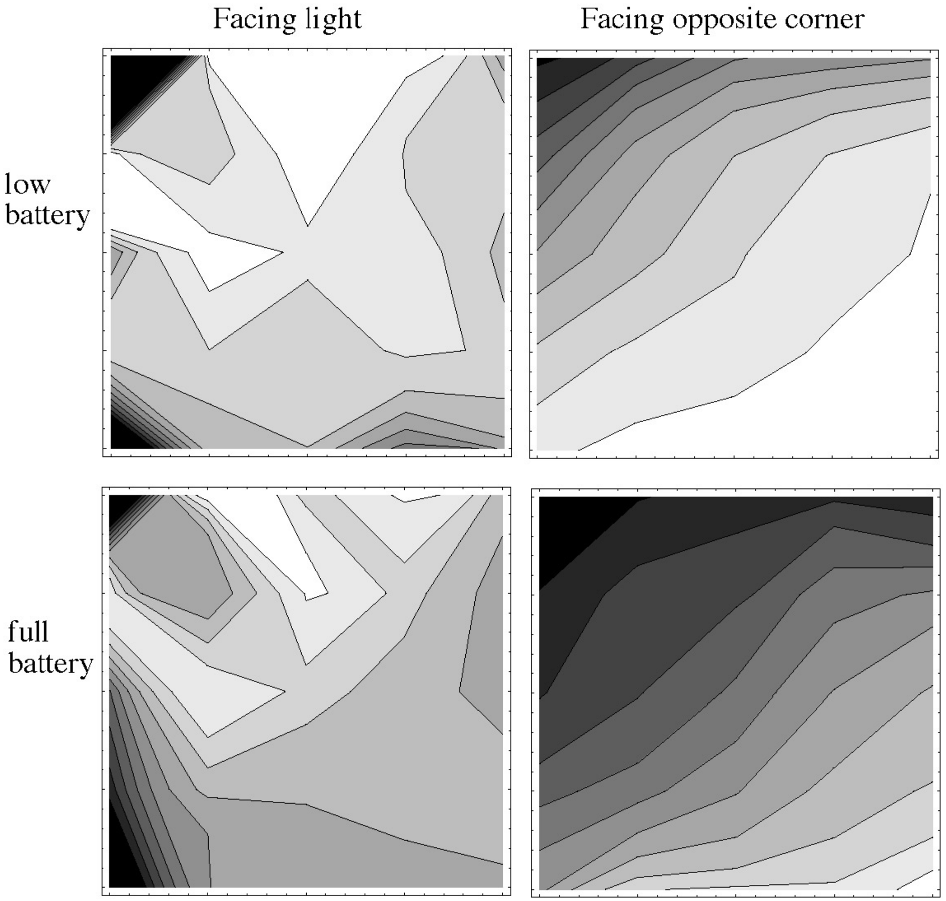
$$\text{Fitness} = V \times (1-s)$$



Machine Neuro-Ethology

Best evolved robots go to recharge with only 10% residual energy. Why and how?

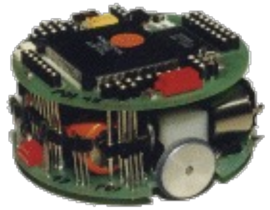




Activity of an internal neuron

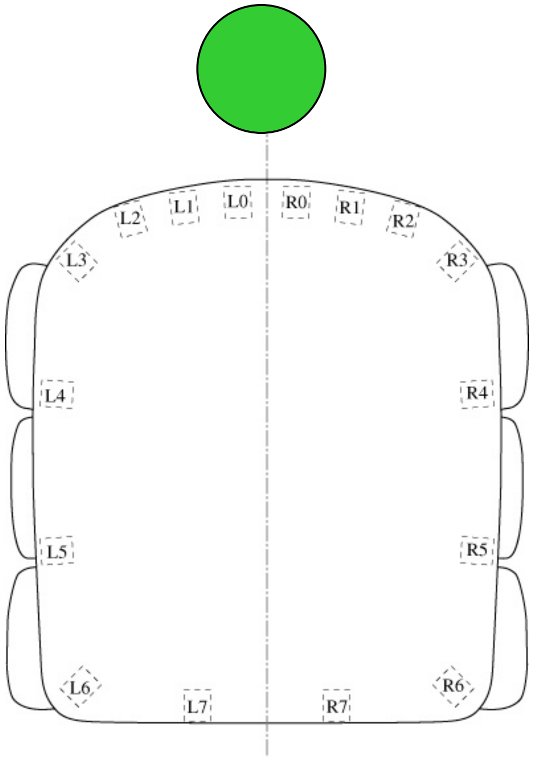
Evolution of complex robots

It is difficult to evolve from scratch large and complex robots because of:
- hardware robustness
- *bootstrap problem:* zero-fitness of all individuals of the initial generation



Khepera
robot

Koala
robot

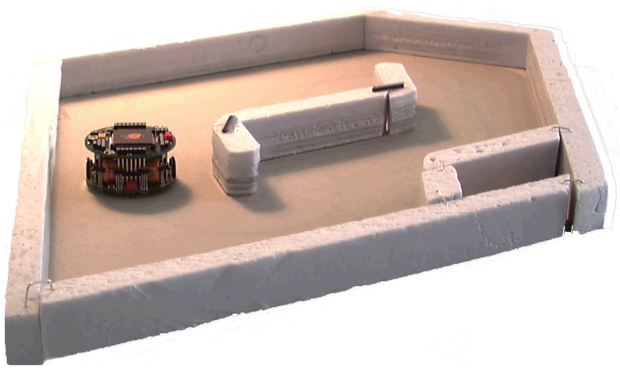


Incremental evolution (a.k.a robot shaping)

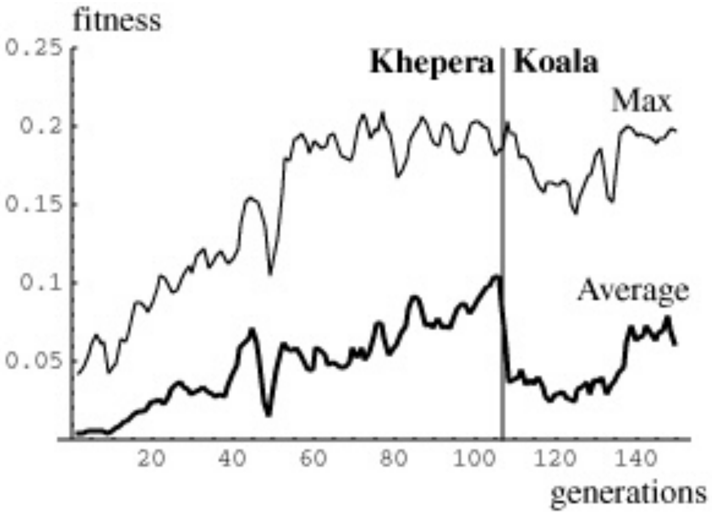
simulation

Fitness=$V \times \Delta v \times (1-s)$

real robot (Khepera)



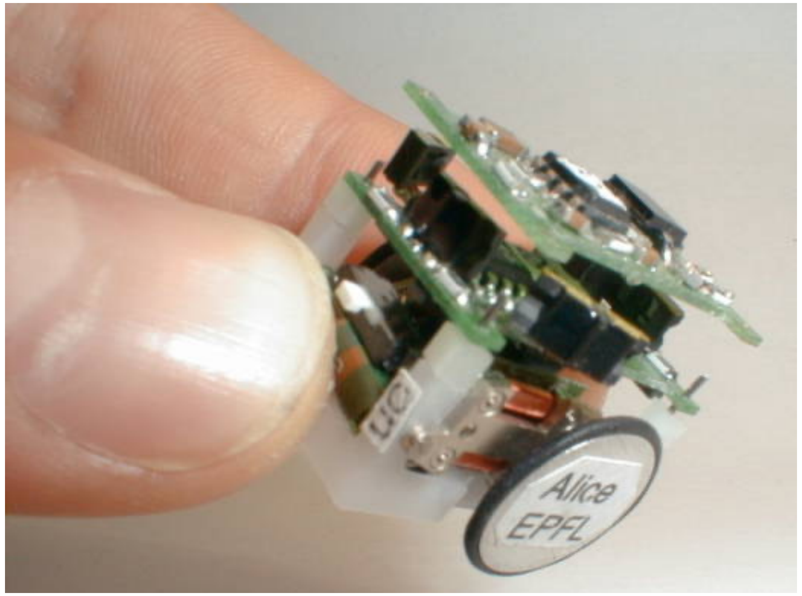
Fitness=$V \times \Delta v \times (1-s)$

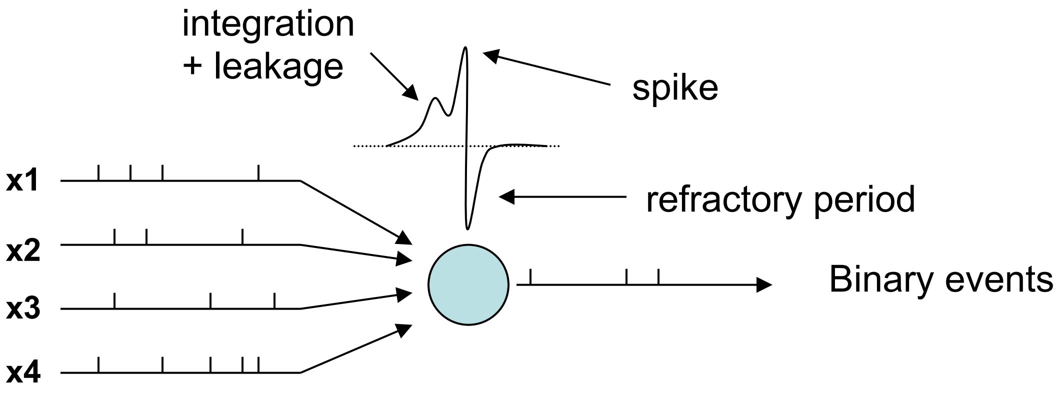


different robot (Koala)

Fitness=$V \times \Delta v \times (1-s)$

Evolution of spiking neural controllers



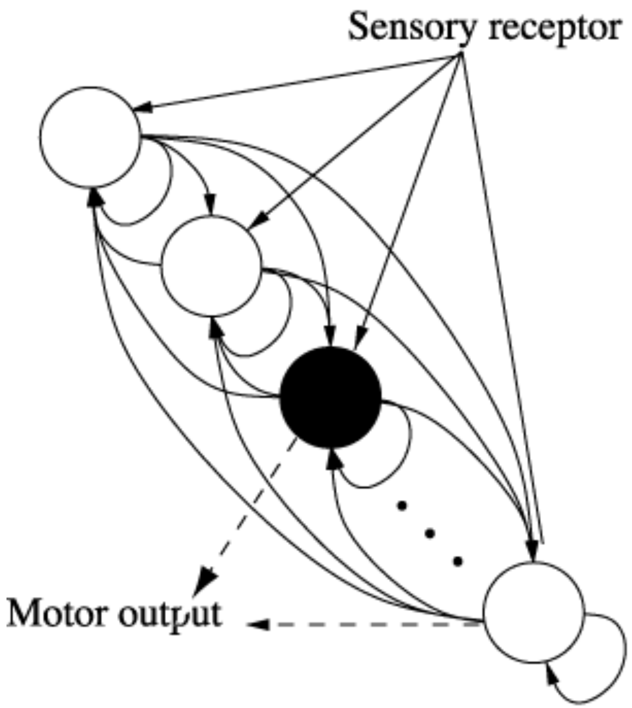


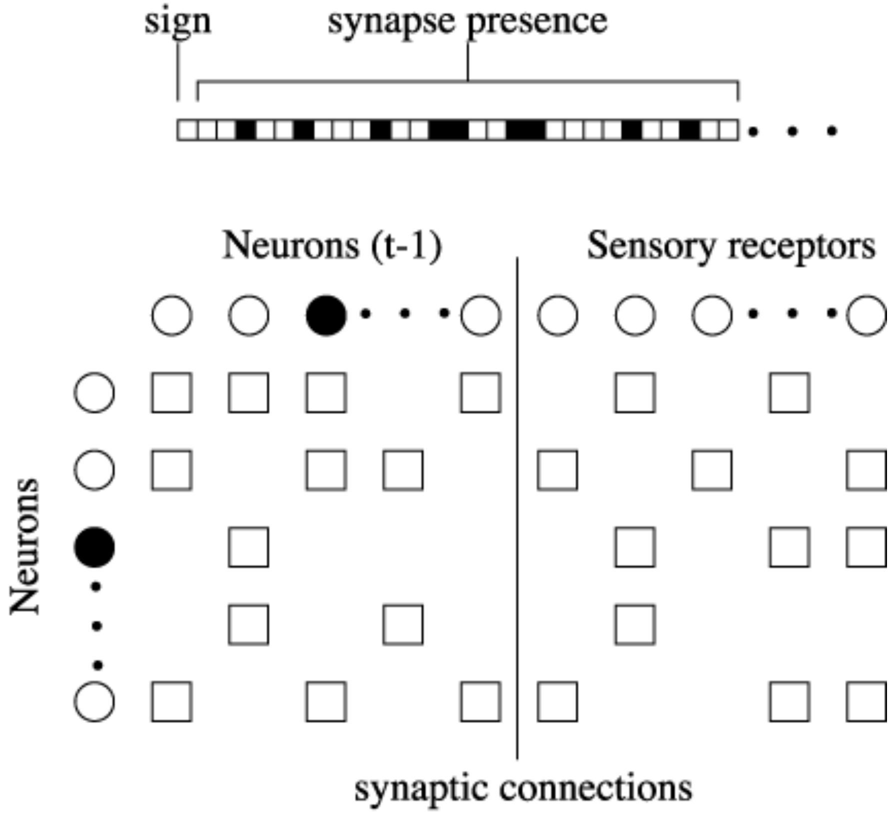
EPFL Microrobot
- 4 proximity sensors
- 2 Swatch motors
- 10 hours autonomy

Microcontroller PIC16F84, (Microchip, 2001)
1024 words of program memory
68 bytes of RAM
64 bytes of EEPROM

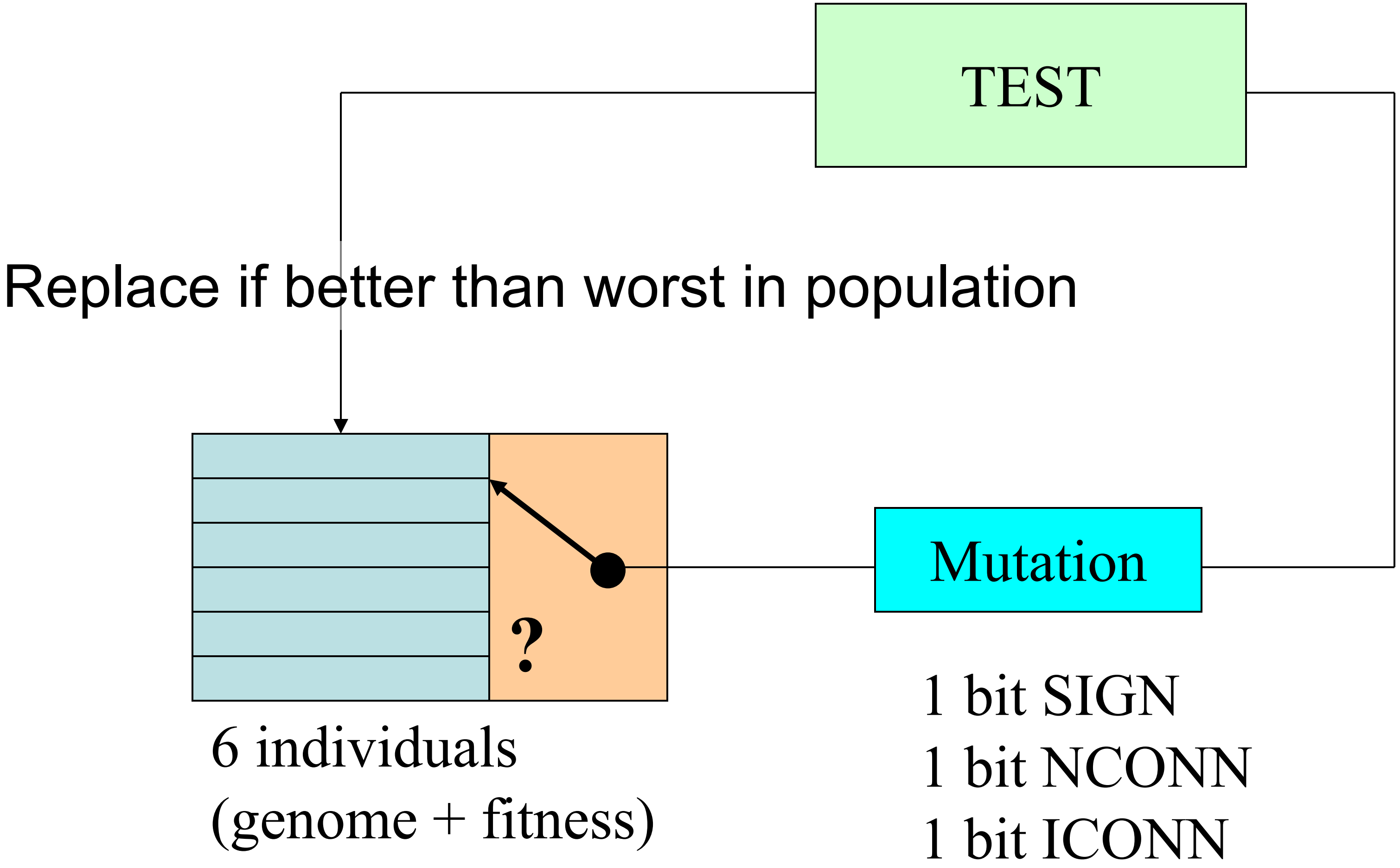
Representation and encoding of neural architecture

Each neuron has a binary gene: projection sign and pattern of incoming connections
Genotype of individual = concatenation of 8 neuronal genes
Weight of existing incoming connection = 1 (no learning)





Steady-state evolutionary algorithm

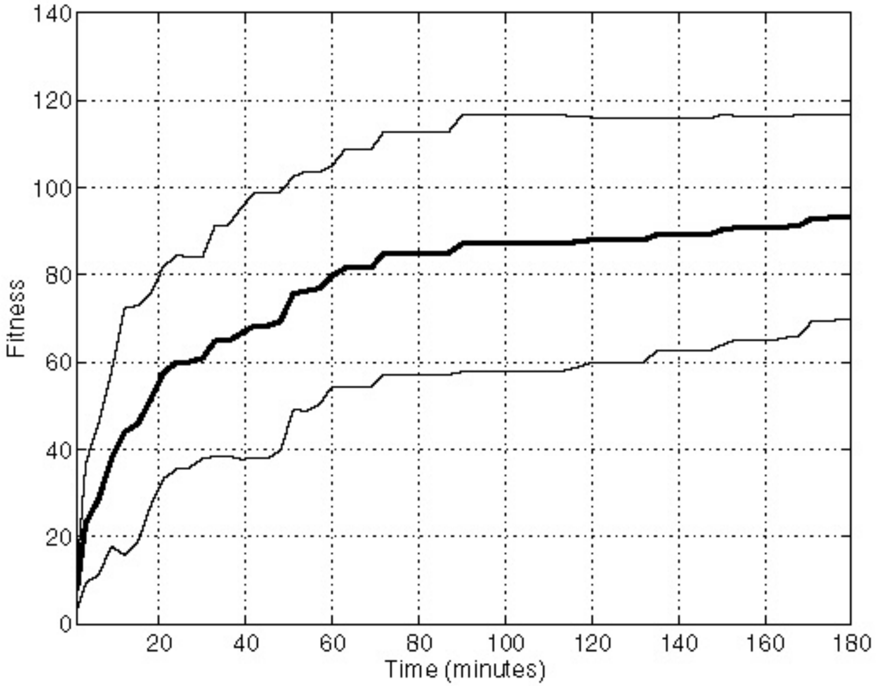


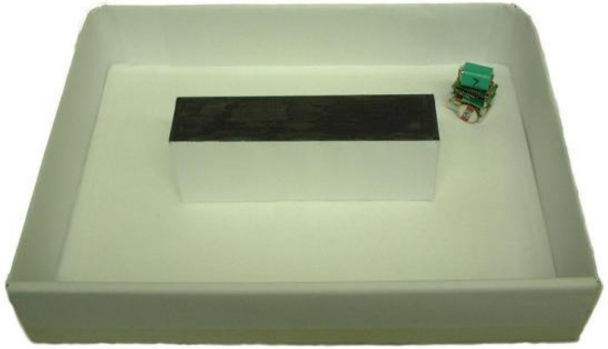
Companion slides for the book *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies* by Dario Floreano and Claudio Mattiussi, MIT Press

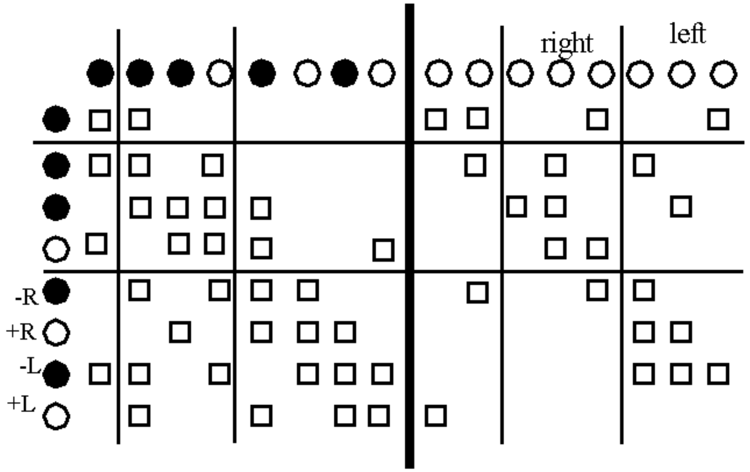
Forward navigation with obstacle avoidance

$$\text{Fitness} = V \times \Delta v \times (1-s)$$

Steady-state evolution

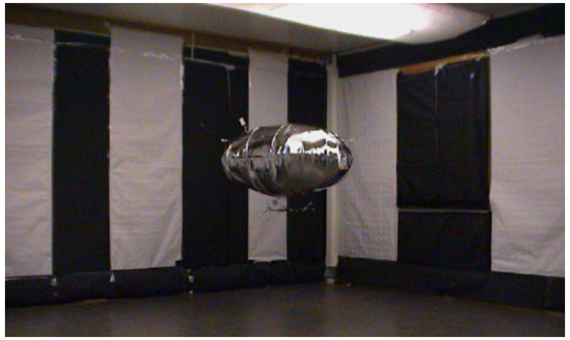




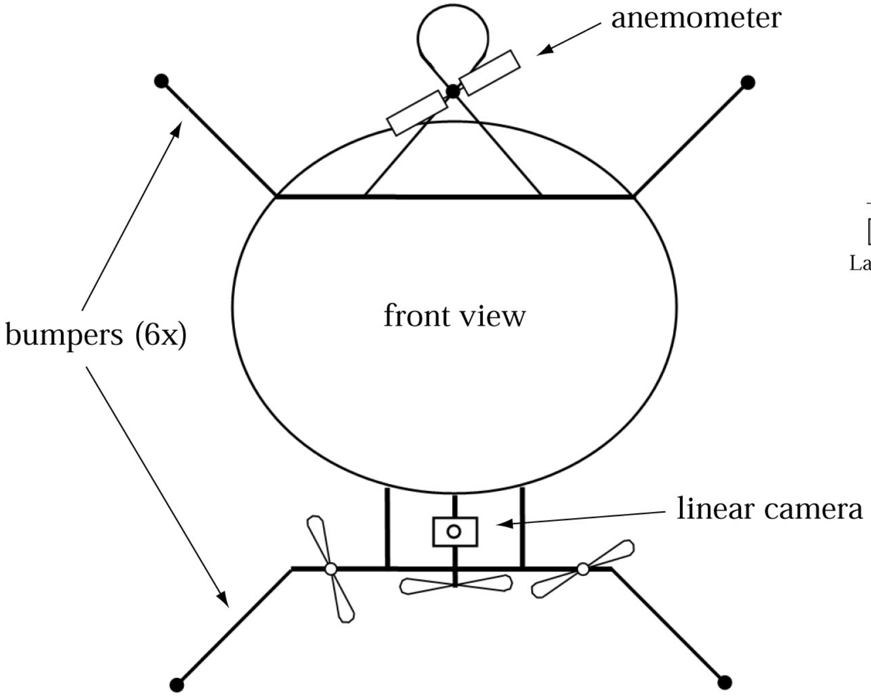


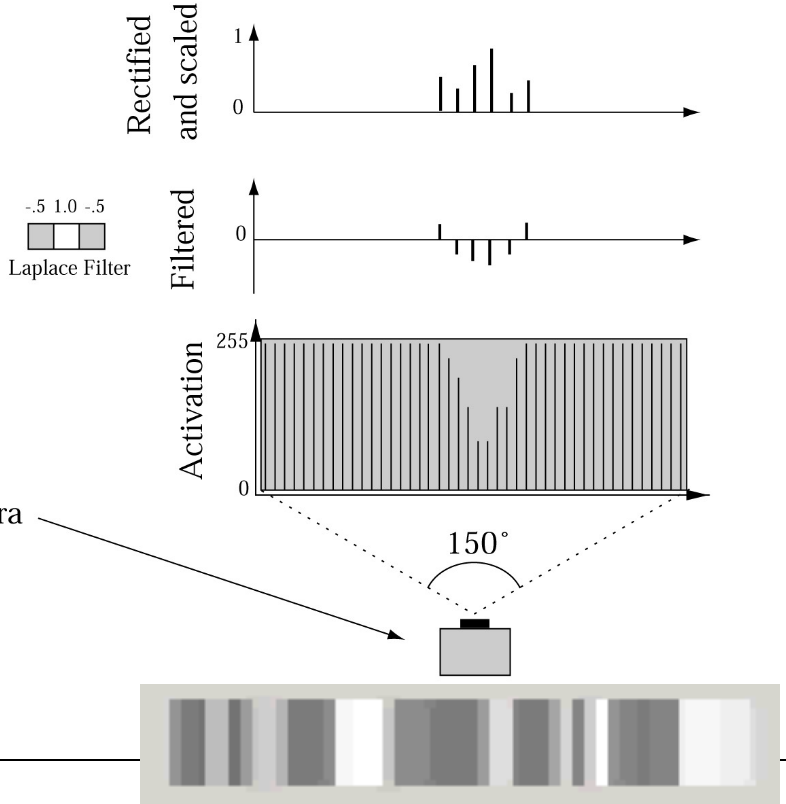
- bias: ↻
- IR Right: ↺
- IR Left: ↻

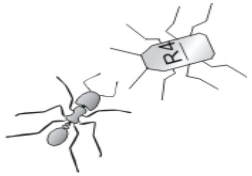
Vision-based flight of a blimp



- 5 x 5 room, random size stripes
- Fitness = forward motion (anemometer)
- 2 trials, 2 minutes each
- Evolution + network activation on PC
- Sensory pre-processing on microcontroller

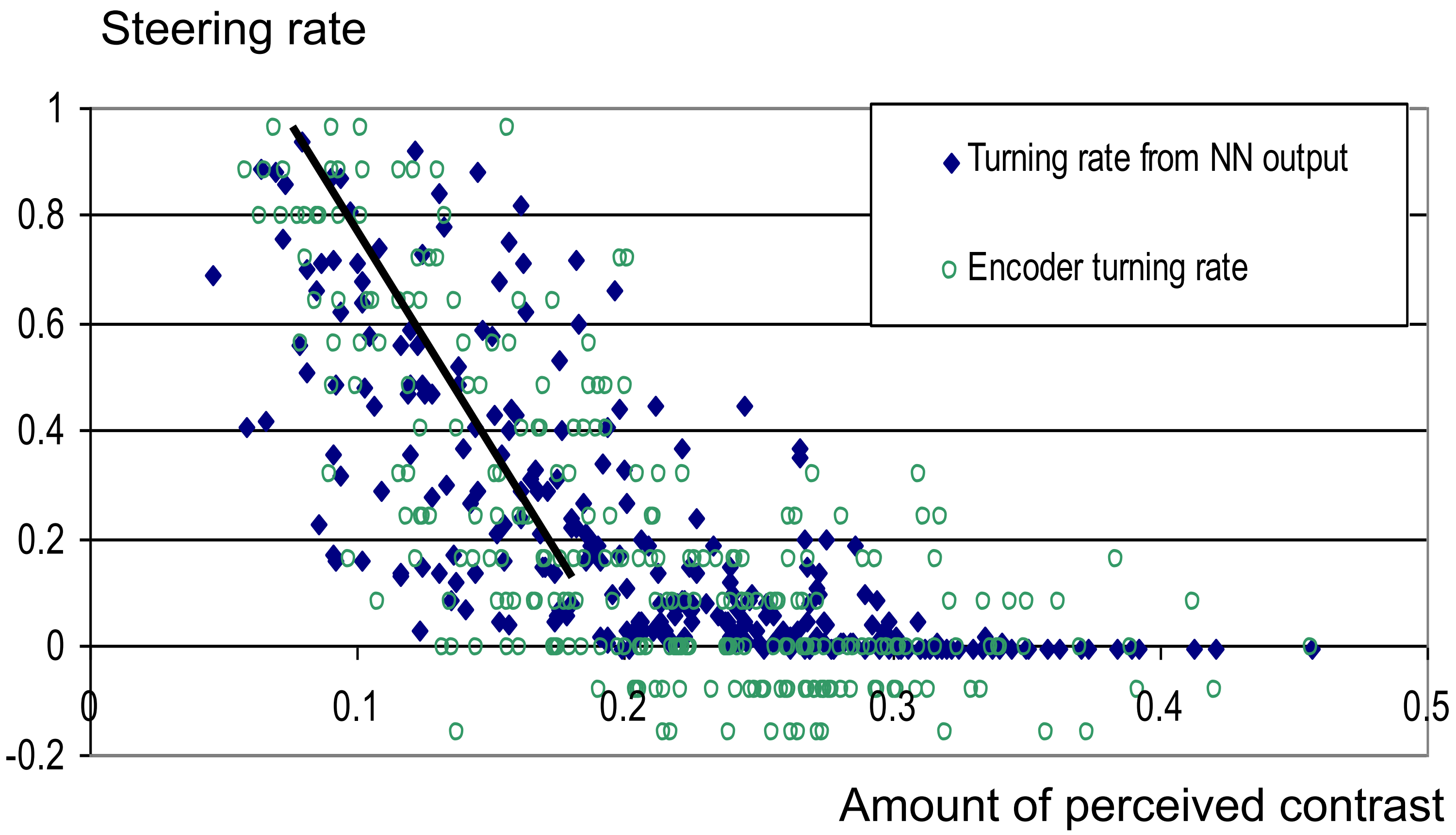


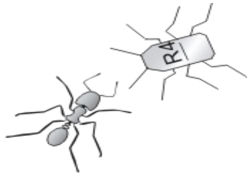




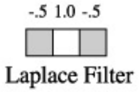
After 50 generations on the real blimp

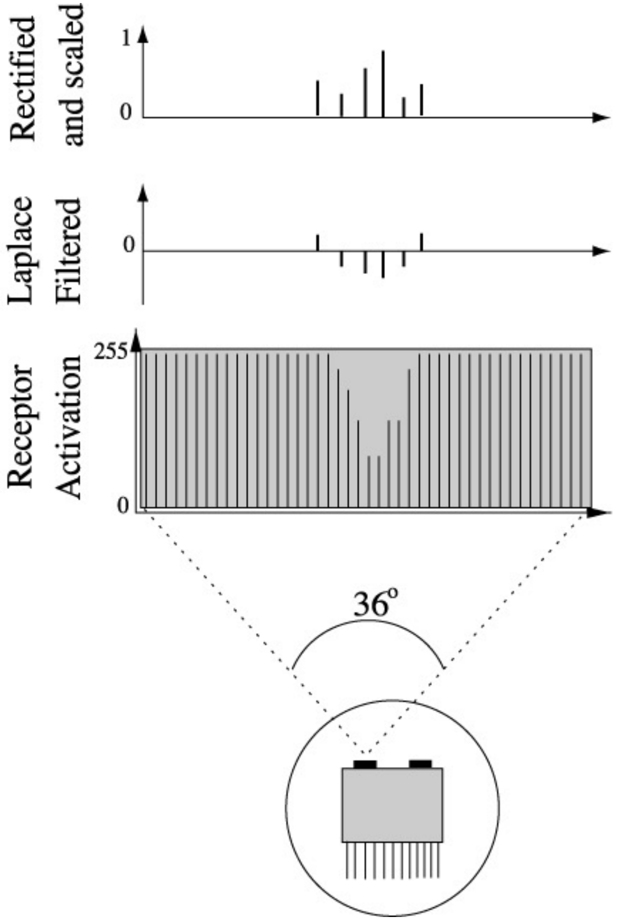
Evolution is opportunistic!





Vision-based navigation with spiking neurons





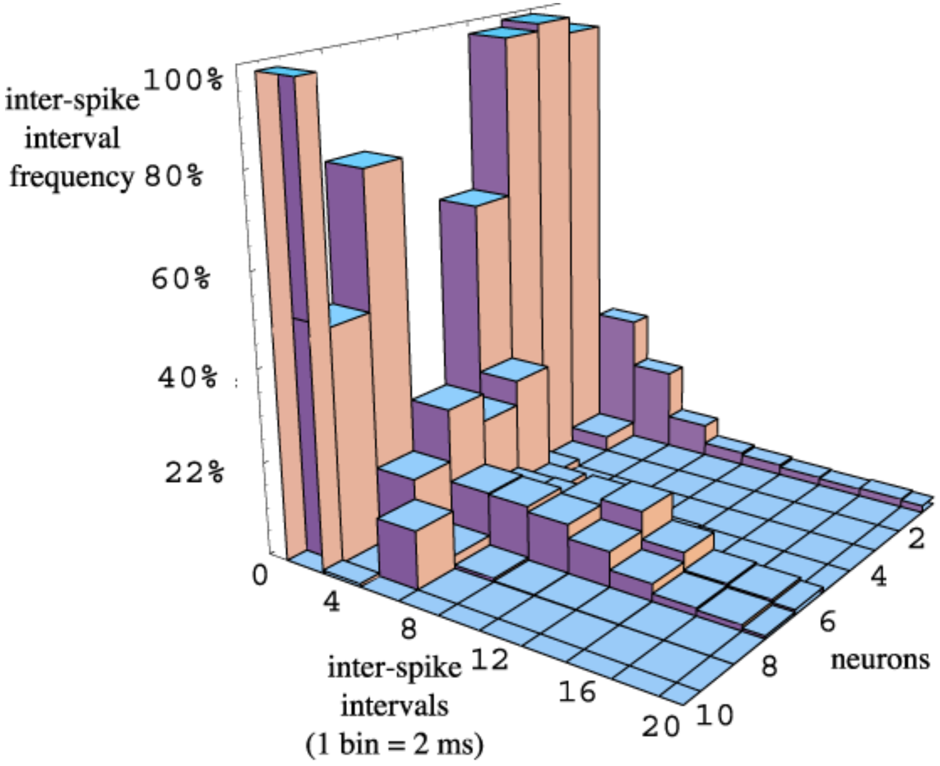
Fitness proportional to amount of forward translation over 2 mins

After 30 generations

Firing rate or firing time?

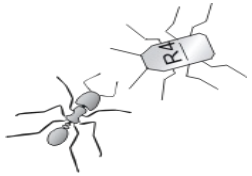
Neuron #

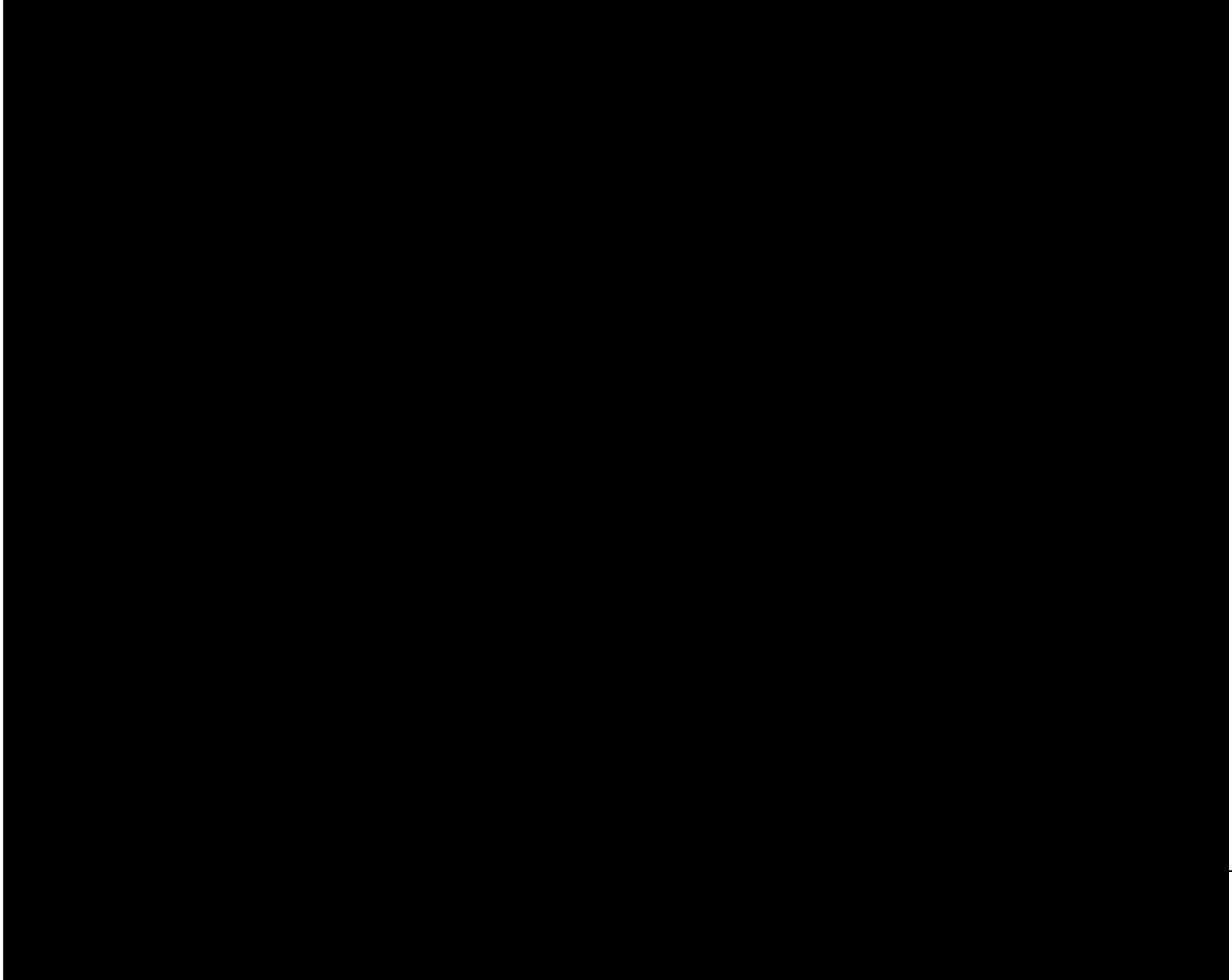
	1	2	3	4	5	6	7	8	9	10
spikes/s	9	445	453	450	330	40	129	363	0	452



- Removing any single neuron (except # 9) decreases the navigation performance
- Removing any pair of neurons decreases even further navigation performance
- Removing neurons 1, 5, 6 has no effect on performance

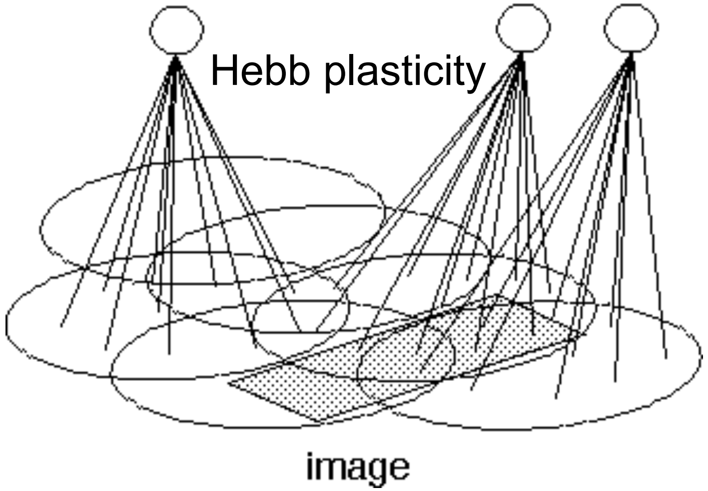
we infer that evolved neurons use time difference of incoming signals, not only total signal intensity

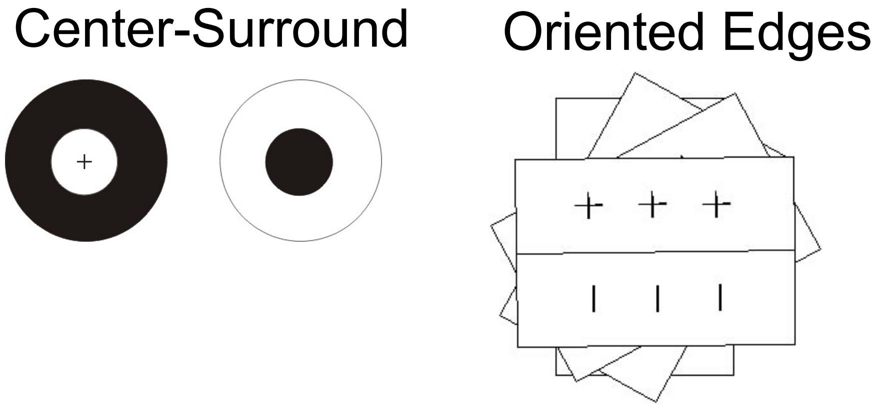


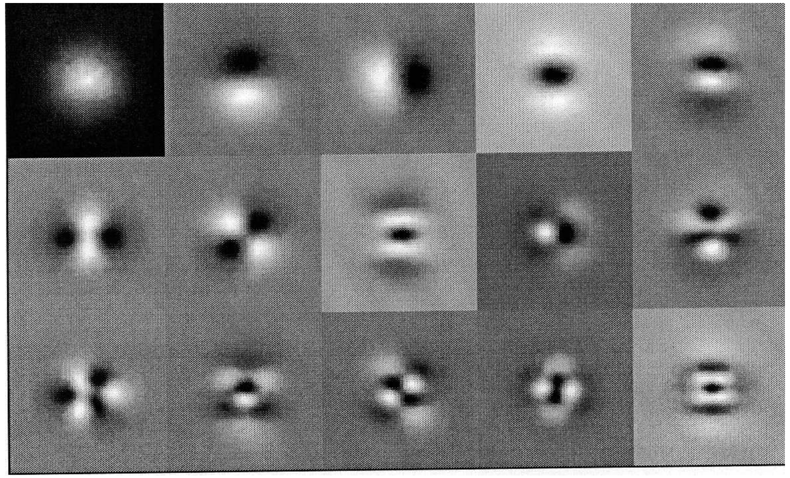


Visual feature detection

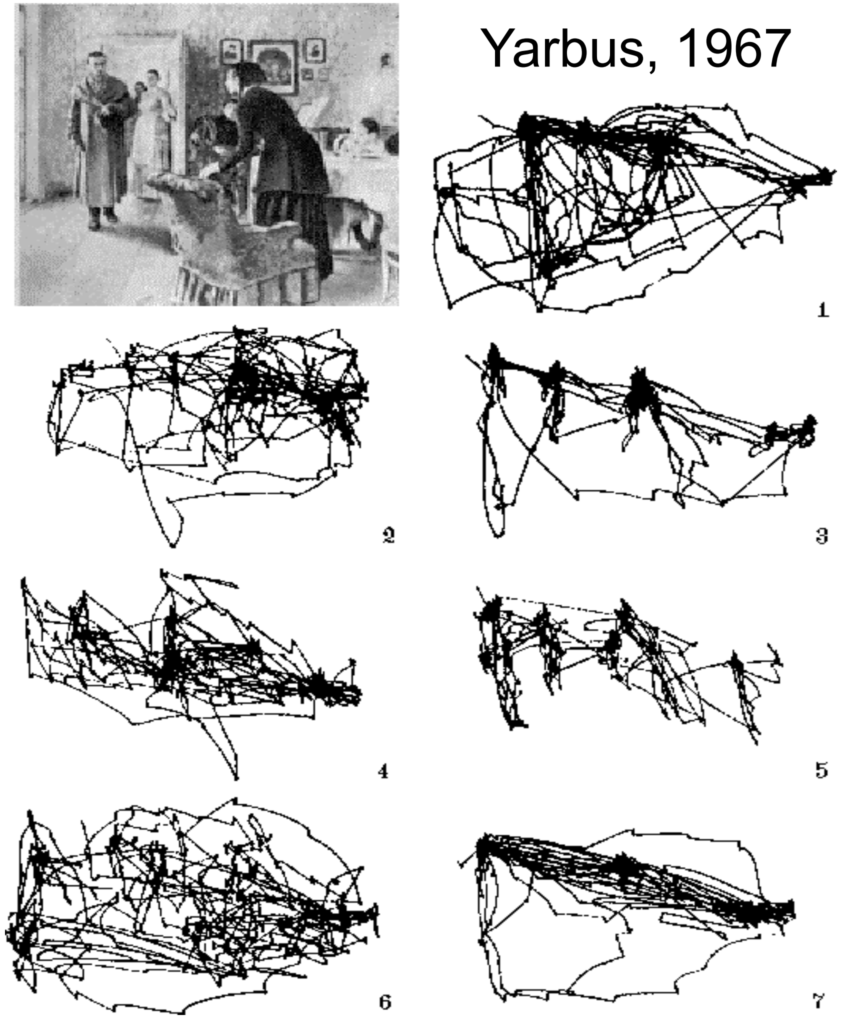
Process whereby visual neurons become sensitive to certain sensory patterns (features) during the developmental process (Hubel & Wiesel, 1959)





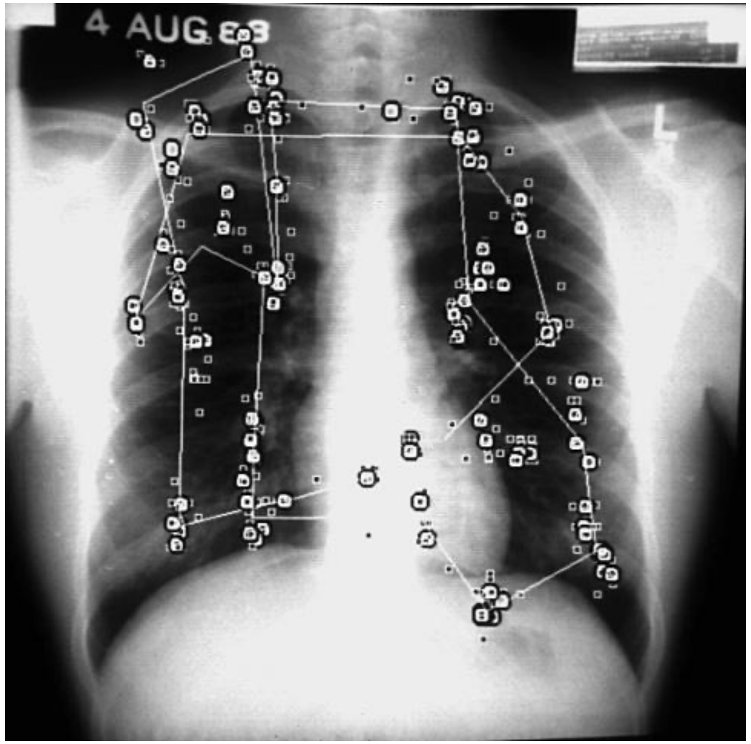


Active vision



Yarbus, 1967

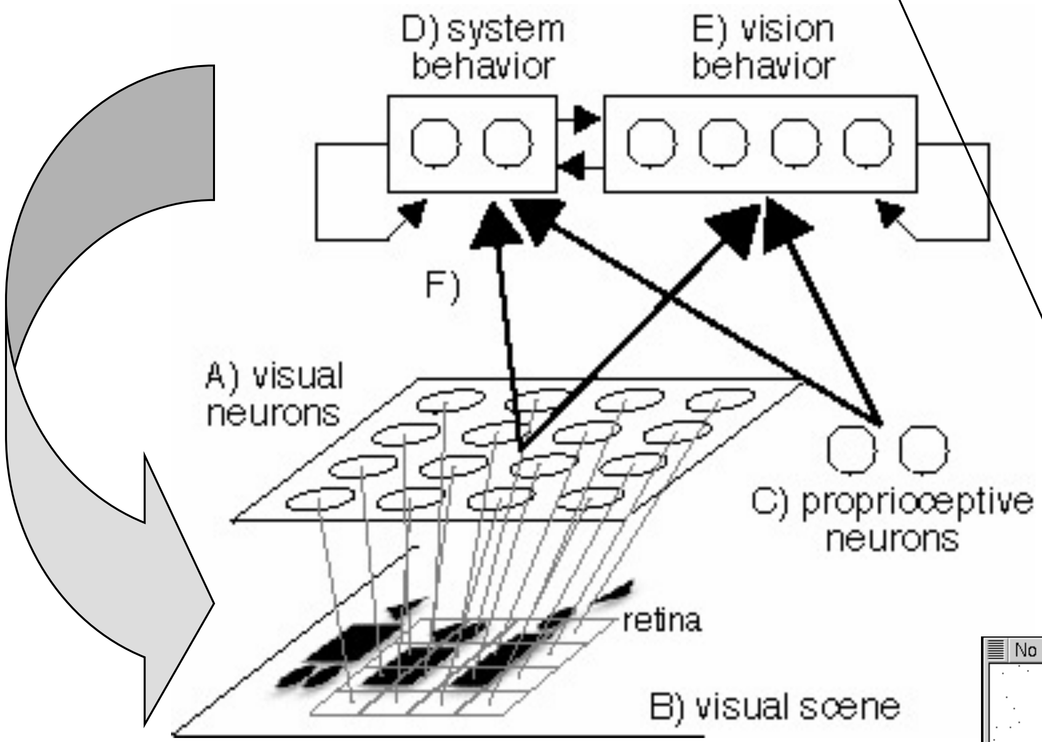
Process of selecting by motor actions sensory patterns (features) that make discrimination easier (Bajcsy, 1988)

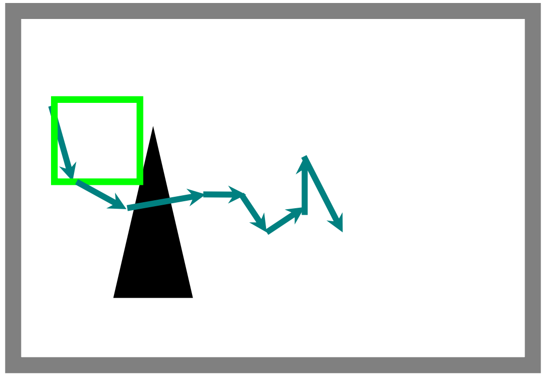


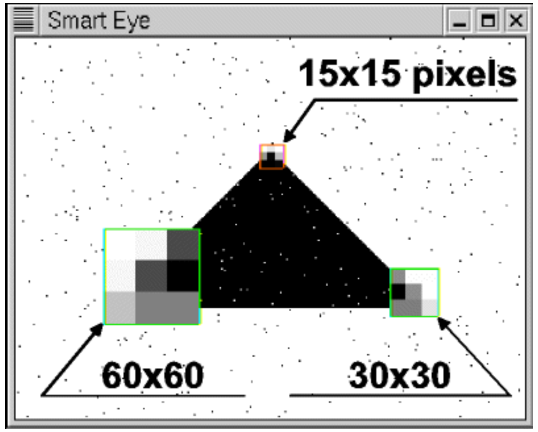
Neural architecture for active vision

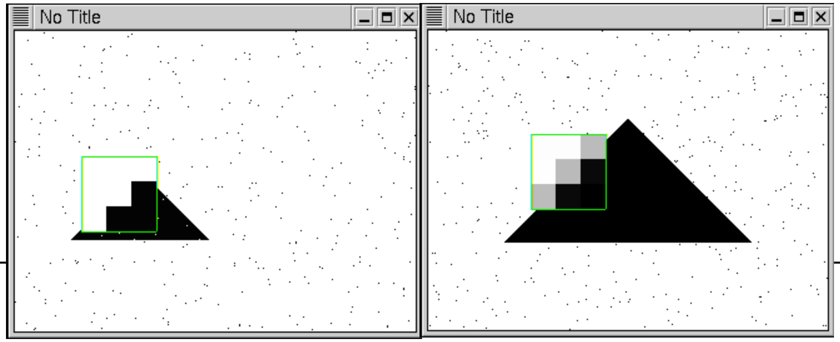
shape discrimination
robot control
car driving

retina movement
zooming factor
filter type





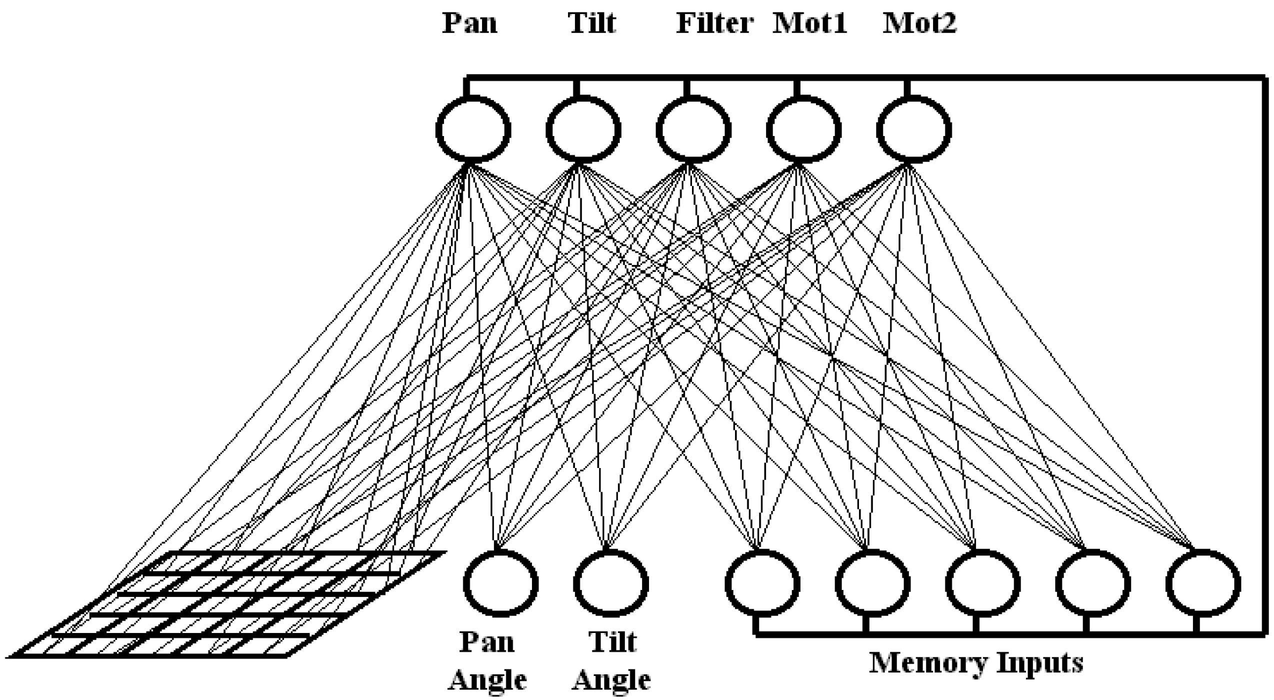




Robot navigation with active vision architecture

Goal: Evolve collision-free navigation using only vision information from a pan/tilt camera.

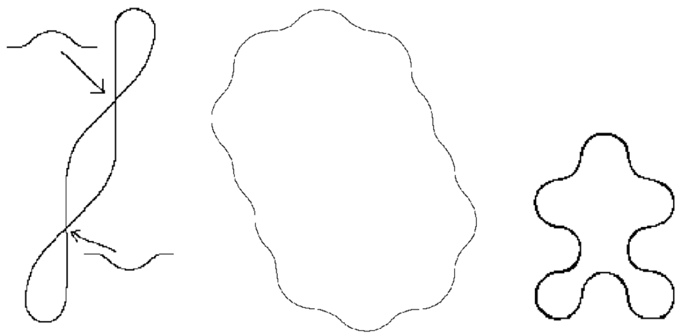
Output of vision system is movement of camera (pan/tilt) and of robot wheels (mot1/mot2). Filter as before.



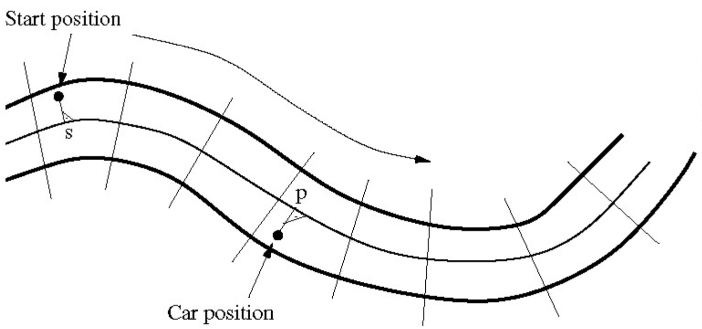

Environment

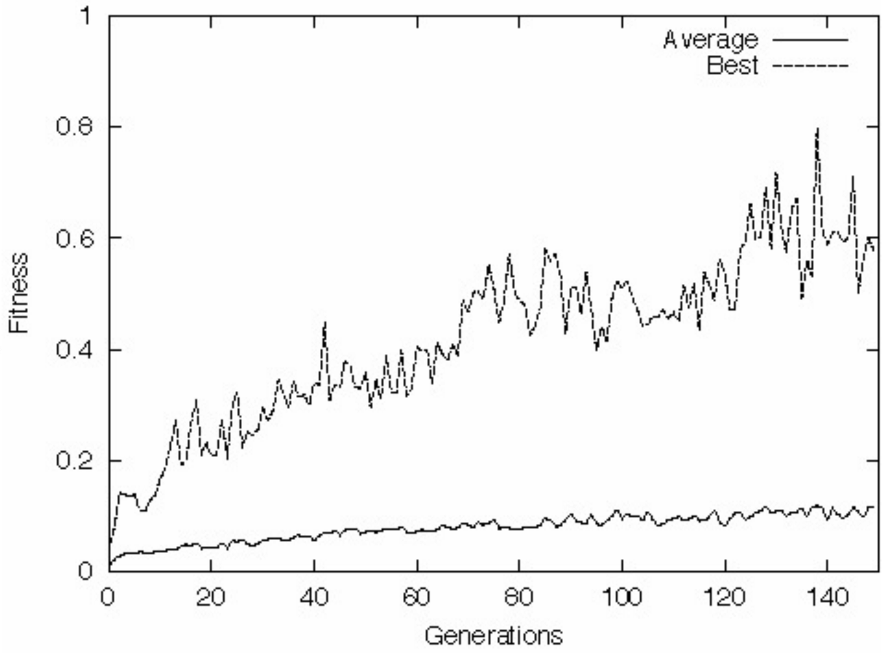
Active Vision for Car Driving

Fitness = percentage of covered distance D in R races on M circuits (limited time for each race).



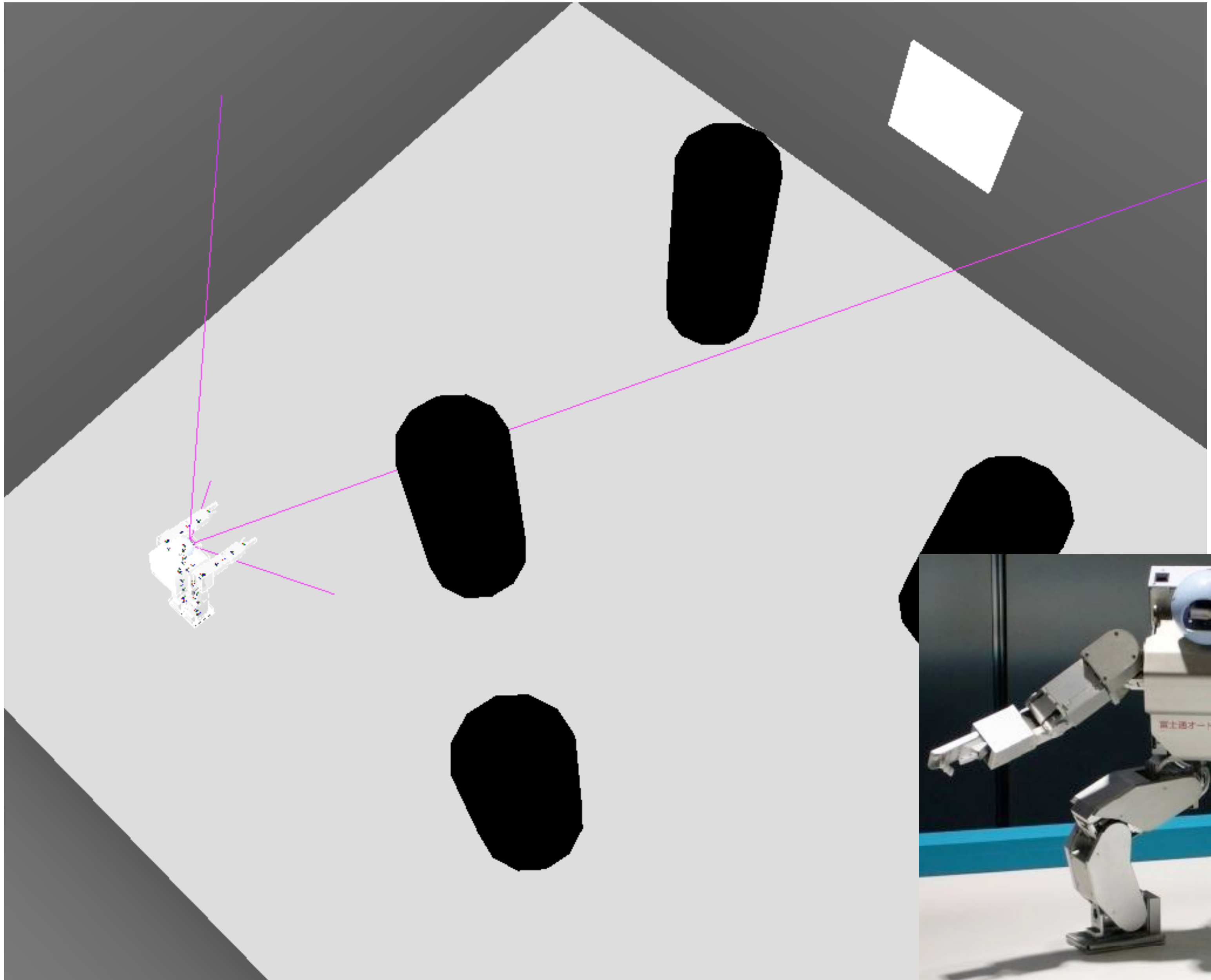
$$F = \frac{1}{R * M} \sum_{r=1}^{R} \sum_{m=1}^{M} D_{r,m}$$

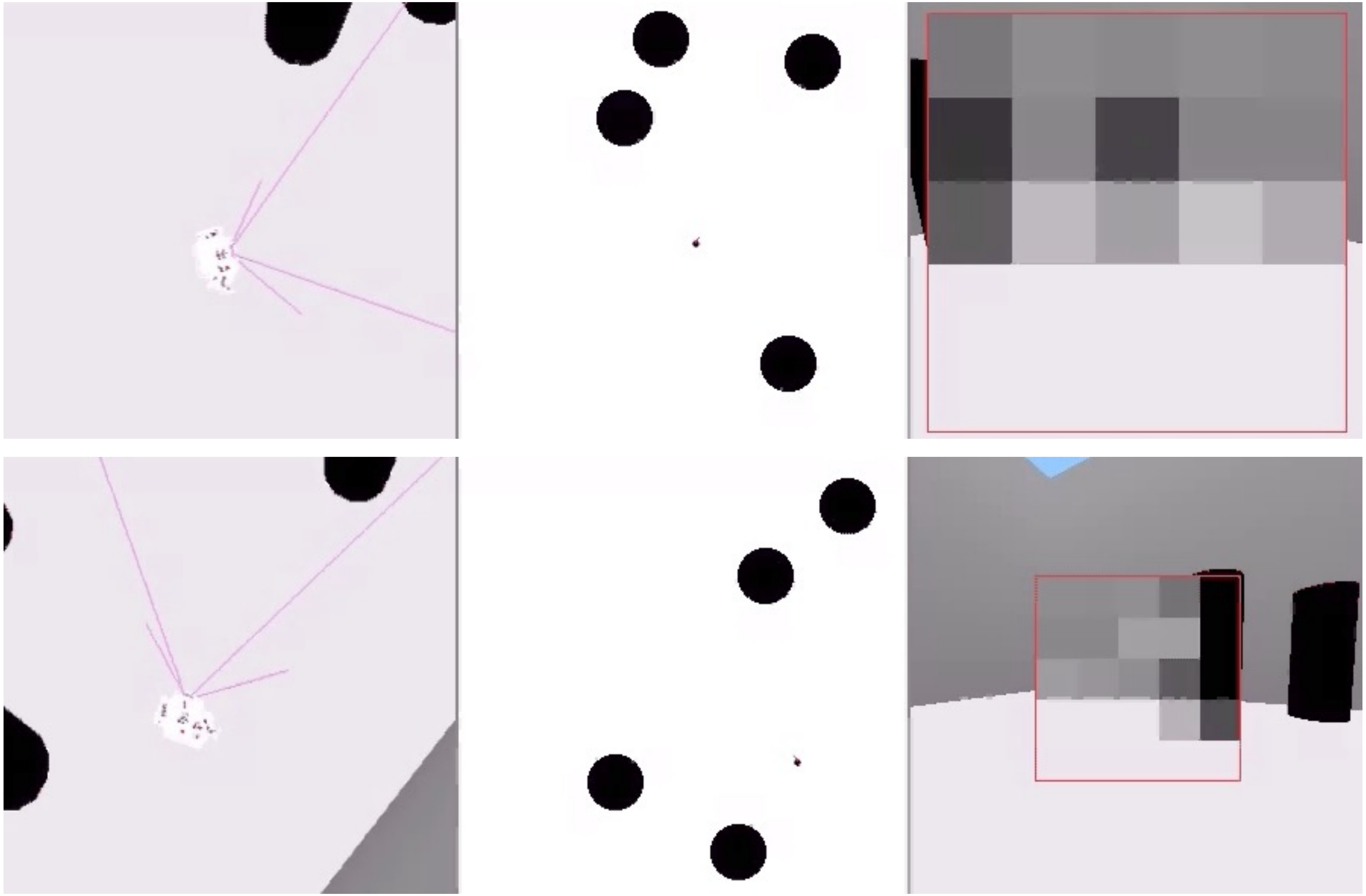




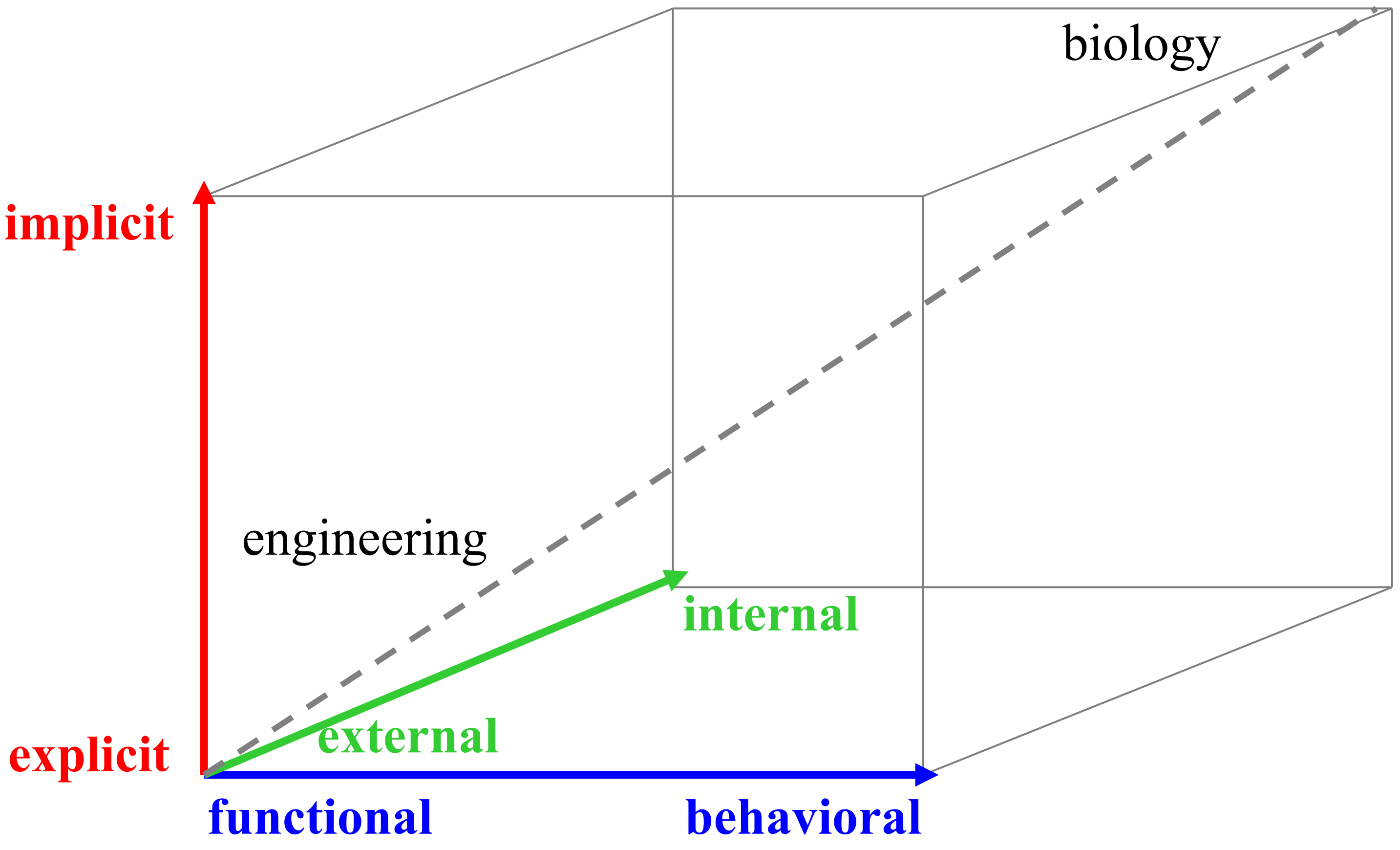

20.0fps

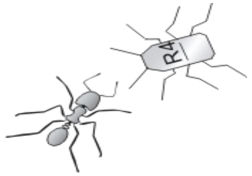
Active Vision for bipedal locomotion





Fitness design space: comparing fitness functions

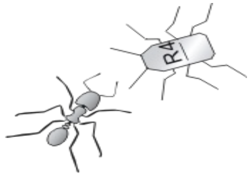




Reinforcement Learning - Evolutionary Computation

Two methods for learning behavioral policies from rewards

	Reinforcement Learning		Evolutionary Computation
-	Definition of Reinforcement Policy	-	Definition of Fitness Function
-	Gradient descent/ascent	+	No need of gradient
-	Lots of hyperparameters and ”tricks”	+	Comparatively less hyperparameters
+	Efficient search of state-action space	-	Random search after selection (but CMA-ES)
-	Difficult in long rollouts without reward	+	No problem with rollout length
-	Operates only on weights of neural network	+	Operates on weights, morphologies, learning
-	Requires many rollouts	-	Requires many rollouts
+	Has strong mathematical foundations	-	Some algorithms are rather empirical



See also https://openai.com/blog/evolution-strategies/