

# Applying Evolutionary Computation to Robotics

Adrian Thomas Schiller

Division of Science and Mathematics  
University of Minnesota, Morris  
Morris, Minnesota, USA

28 April 2014

# The Big Picture

- **Problem:** A robot is faced with a problem where the solution is not immediately obvious
- **Potential Solution:** Evolutionary computation (EC) is a process which can solve difficult problems in programming
- **Issue:** Since a robot interacts with the physical world, EC is slower by many magnitudes
- **Solution:** By using simulation and applicable evolutionary strategies, it is possible to use EC to evolve robots

# Outline

## 1 Research Cases

- Station Keeping Robot
- Walking Robot
- Coordinate Tracking Robot
- Artificial Neural Networks

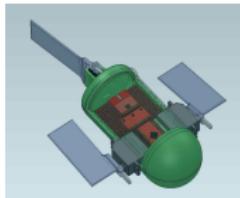
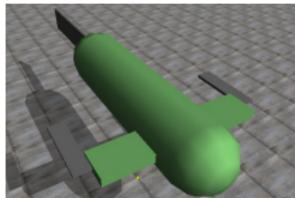
## 2 Background

## 3 Simulation

## 4 Evolutionary Process

## 5 Results

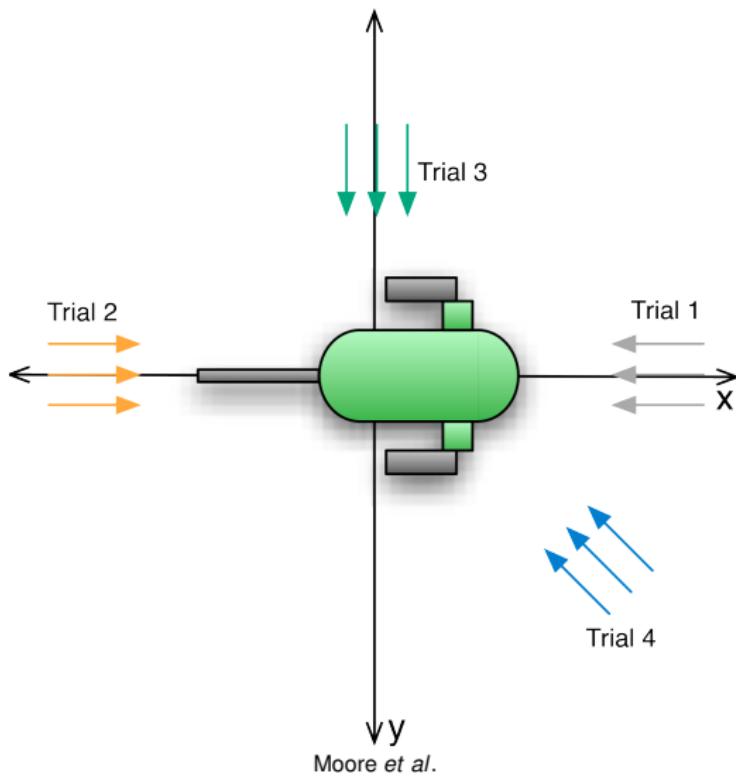
# Station Keeping Robot



Moore *et al.*

- Moore *et al.* developed the station keeping robot
- Goal: to maintain position in a body of water

# Station Keeping Robot



# Walking Robot

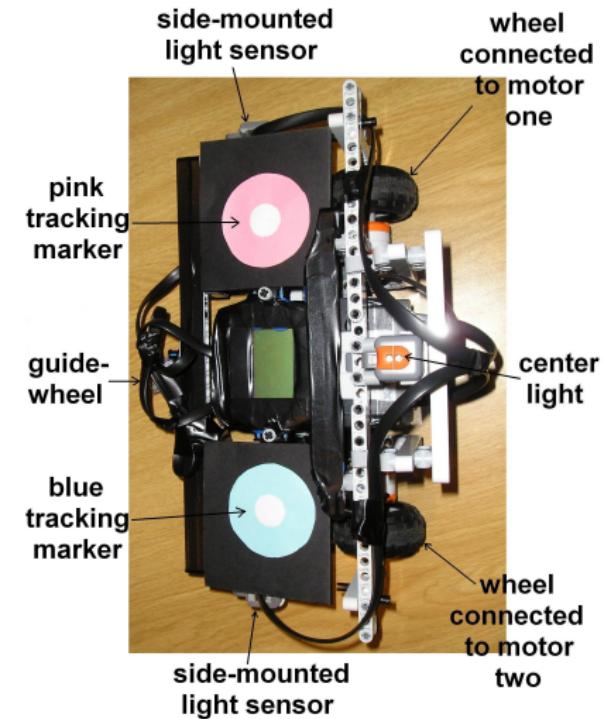
- Farchy *et al.* modified the code of the Aldebaran Nao robot
- Goal: to increase walking speed



Farchy *et al.*

# Coordinate Tracking Robot

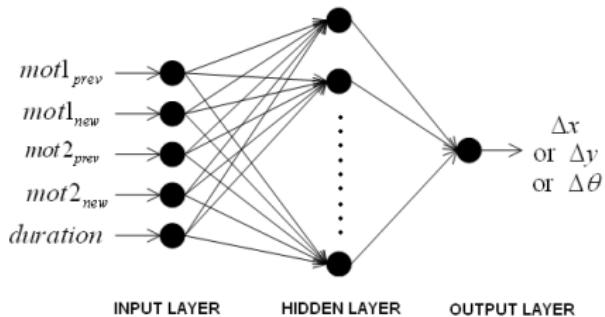
- Pretorius *et al.* created a Lego Mindstorms robot
- Goal: to evolve an internal navigation controller



Pretorius *et al.*

# Artificial Neural Networks (ANN)

- ANNs are a collection of nodes with weighted edges.
- The purpose of the network is to develop a functional relationship from the input to the output



Pretorius *et al.*

# Outline

1 Research Cases

2 Background

- Evolutionary Computation

3 Simulation

4 Evolutionary Process

5 Results

6 Conclusion

# Evolutionary Computation

- Evolutionary Computation (EC) is a problem solving technique which mimics natural selection
- EC requires:
  - A candidate representation of a potential solution
  - A population of randomly generated candidates
  - A fitness function

# Evolutionary Computation: Process

- Candidates are evaluated
- The best performing candidates are selected
- Selected candidates undergo transformations to repopulate the population
- Process repeats until some limit is reached

# Outline

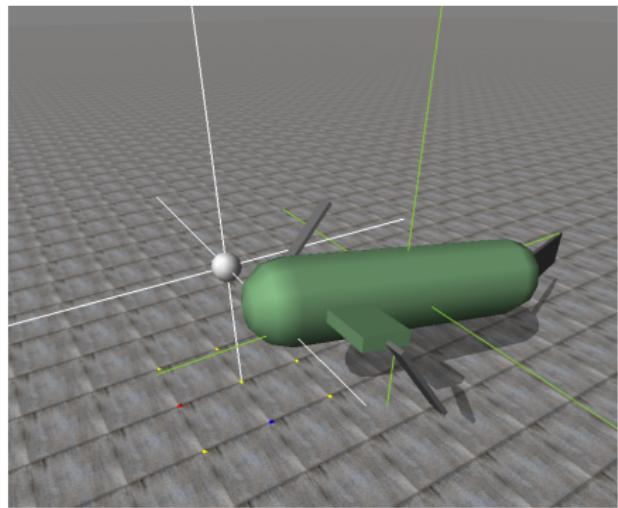
- 1 Research Cases
- 2 Background
- 3 Simulation
- 4 Evolutionary Process
- 5 Results
- 6 Conclusion

# Simulation

- Defined as representing the characteristics or behaviors of one system through the use of another
- Error caused from inaccuracies of simulation is known as transitivity

# Station Keeping Robot: Simulation

- Used ODE to replicate the robot
- No fluid dynamics



Moore *et al.*

# Walking Robot: Simulation

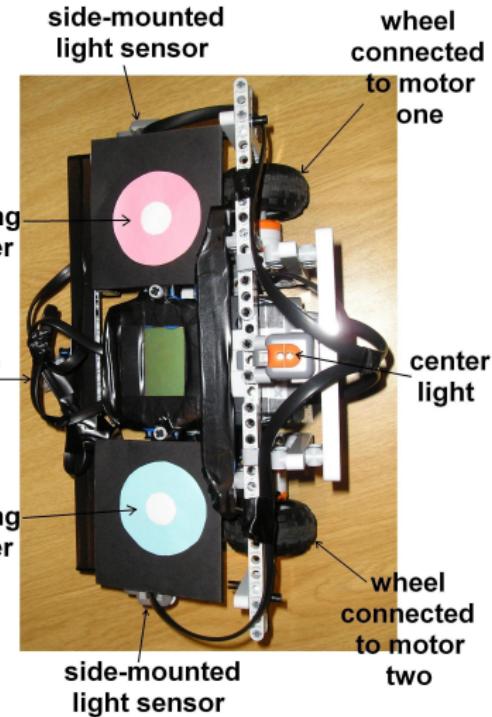
- Uses SimSpark (also ODE)
- Not a perfect representation



Farchy *et al.*

# Coordinate Tracking Robot: Simulation

- An overhead camera captured heading/orientation of robot from arbitrary motor commands
- A testbed of 5,000 commands were sent to the robot and captured by the camera, creating map of command and position



Pretorius et al.

# Outline

- 1 Research Cases
- 2 Background
- 3 Simulation
- 4 Evolutionary Process
- 5 Results
- 6 Conclusion

# Station Keeping Robot

- Evolved a separate candidate for each of the trials
- Population size of 100 candidates
- Evolved for 2,000 generations
- The entire process was repeated 25 times for each of the four trials
- Total: 20,000,000 runs

# Station Keeping Robot: Neural Network

- Input:
  - Current 3D coordinates,  $(x, y, z)$
  - The difference between current and desired coordinate  $(x, y, z)$
  - The output of the previous output (servo speeds and oscillations)
- Output:
  - oscillation of the rear fin
  - speed of the left flipper
  - speed of the right flipper

# Station Keeping Robot: Fitness Function

$$\text{fitness} = \sum_t (10 - d_t(x, y, z))$$

where

$$d_t(x, y, z) = \begin{cases} 10, & \text{if } \text{distance}_t(x, y, z) > 10 \\ \text{distance}_t(x, y, z), & \text{otherwise} \end{cases}$$

# Walking Robot: Parameter optimization

- Farchy *et al.* wanted to optimize several parameters to increase speed

Parameter	Description
<i>stepPeriod</i>	Number of frames to take two steps.
<i>amp<sub>swing</sub></i>	Amplitude of the swing calculation.
<i>knee</i>	Base of the leg lifting calculation.
<i>startLength</i>	Used in calculating initial ramp up.
<i>v<sub>short</sub></i>	Factor for the leg lifting calculation.
<i>a<sub>short</sub></i>	Amplitude of the leg lifting calculation.
<i>ϕ<sub>short</sub></i>	Offset of the leg lifting calculation.
<i>v<sub>swing</sub></i>	Factor for the swing calculation.
<i>ϕ<sub>swing</sub></i>	Offset for the swing calculation.
<i>gyro<sub>hipPitch</sub></i>	Body pitch factor for calculating hip pitch.
<i>gyro<sub>kneePitch</sub></i>	Body pitch factor for calculating knee pitch.
<i>gyro<sub>hipRoll</sub></i>	Body roll factor for calculating hip roll.
<i>gyro<sub>ankleRoll</sub></i>	Body roll factor for calculating ankle roll.
<i>scale<sub>roll</sub></i>	Scale for sensor value of body roll.
<i>offset<sub>pitch</sub></i>	Offset for sensor value of body pitch.
<i>scale<sub>pitch</sub></i>	Scale for sensor value of body pitch.
<i>fwdOffset</i>	Offset to have the robot walk in place.

Farchy *et al.*

# Walking Robot: Fitness Functions

- Used two fitness functions for two separate runs
  - omniWalk*

$$\text{fitness} = \left( \sum_t (\text{DistanceTraveled}_t) \right) - \text{fallingPenalty}$$

- WalkFront*

$$\text{fitness} = \text{maxVelocity}() \text{ in 15 seconds}$$

# Walking Robot: Grounded Simulation Learning

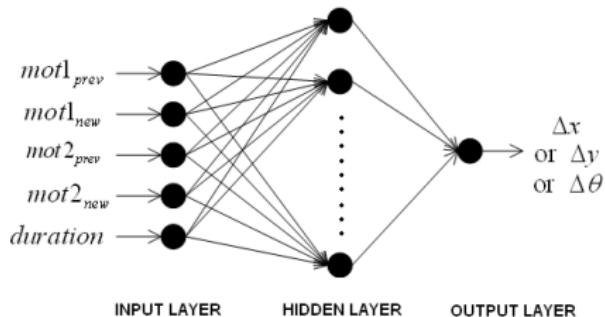
- Farchy *et al.* used Grounded Simulation Learning (GSL) when evolving candidates
- The point of GSL is to add human guidance in the evolution process
- This is done by examining the physical robot with an evolved candidate implementation, and isolating particular attributes

# Coordinate Tracking Robot

- Population of 250 candidates
- Evolved for 15,000 generations
- Process repeated three times for each ANN
- Total: 11,250,000 runs

# Coordinate Tracking Robot: Artificial Neural Network

- Inputs of the ANNs:
  - Current Motor speeds
  - Current length of time
  - Previous Motor speeds
- The ANN output was either:
  - The x-coordinate,
  - The y-coordinate,
  - And the angle



Pretorius *et al.*

# Coordinate Tracking Robot: Fitness Function

- Used the Mean Squared Error (MSE) as the fitness function

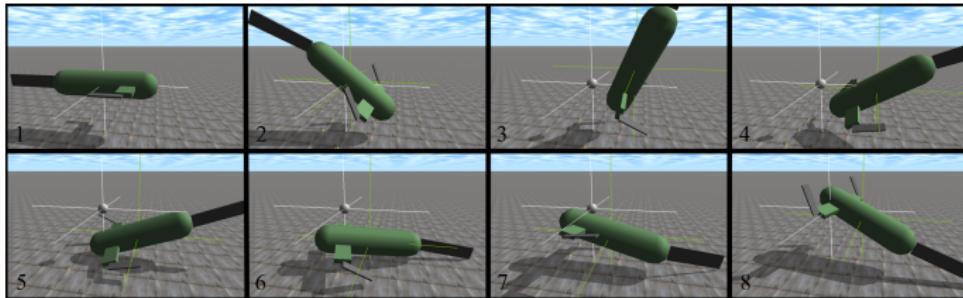
$$\text{fitness} = \frac{1}{N} \sum_{p=1}^N \sum_{i=1}^O (t_{pi} - a_{pi})^2,$$

- N is the size of the testbed (5,000)
- O is the ANN (1,2,3)
- t is the expected output (computed by ANN)
- a is the actual (testbed value)

# Outline

- 1 Research Cases
- 2 Background
- 3 Simulation
- 4 Evolutionary Process
- 5 Results
- 6 Conclusion

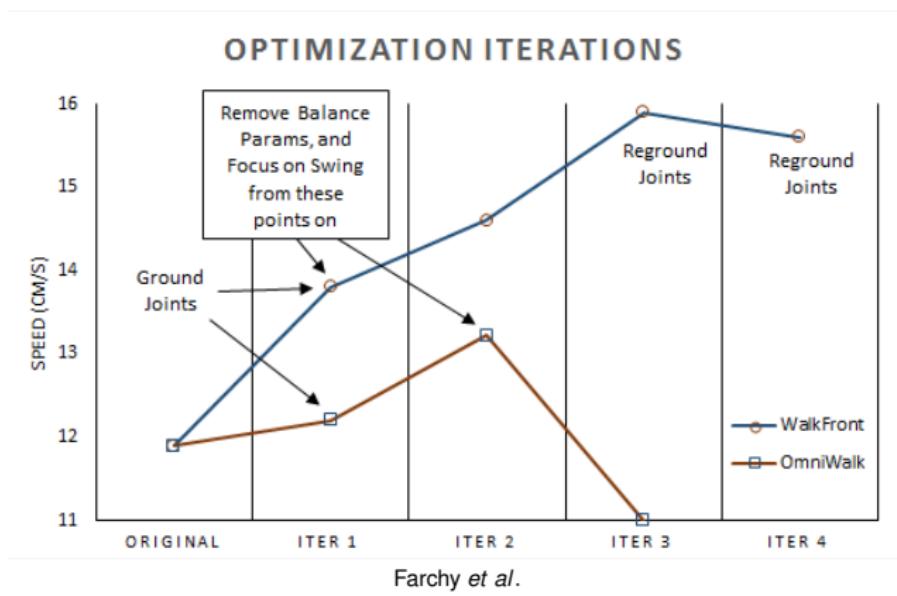
# Station Keeping Robot: Results



Moore et al.

- Each trial had a candidate which successfully maintained the position
- When the flow was coming from behind, the evolved candidate would flip end-over-head to orient itself  
(<http://y2u.be/UufbnEGFwV4>)

# Walking Robot: Results

Farchy *et al.*

# Coordinate Tracking Robot: Results

- Each of the ANNs evolved for 12 hours
- Pretorius *et al.* noted that the results were reasonably accurate

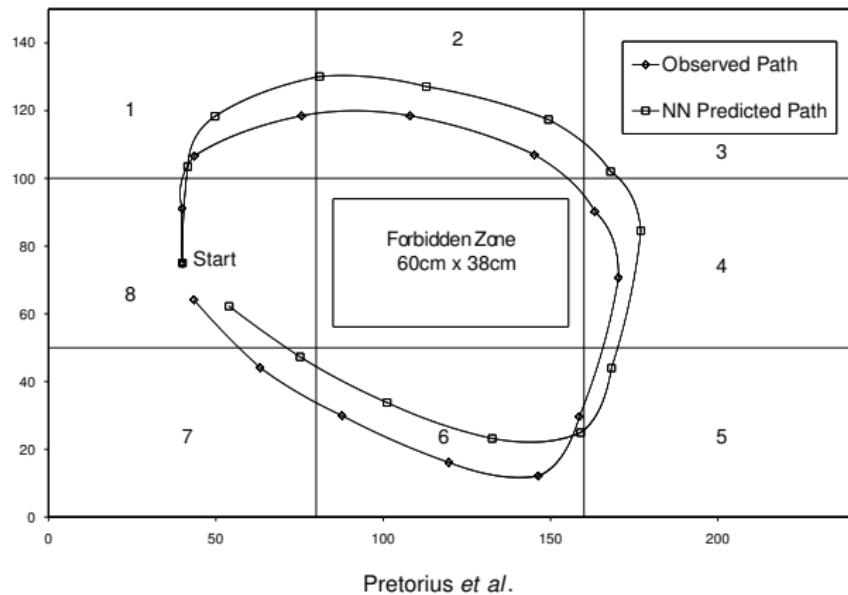
NN Simulator	Final MSE	Average absolute error
change in angle	26.412	3.585 degrees
change in y-coordinate	12.909	2.143 cm
change in x-coordinate	18.559	2.782 cm

Pretorius *et al.*

# Coordinate Tracking Robot: Navigation Test

- Using the evolved ANNs, a navigation test was made for a practical application
- The test was evolved to:
  - Drive the robot in a circle around a 3x3 grid,
  - Not leave the grid or touch the middle square

# Coordinate Tracking Robot: Results



# Outline

1 Research Cases

2 Background

3 Simulation

4 Evolutionary Process

5 Results

6 Conclusion

# Conclusion

- By using a simulation, the evolutionary process can occur at a significantly faster rate
- Evolutionary robotics could be applied if:
  - the robotics problem is well defined,
  - the robot and environment can be simulated,
  - and the simulated robot's success can be quantified

Thank you to Nic McPhee, Elena Machkasova, and Alex Jarvis  
Any Questions?

asdghdsa