EVOLVING BIOMIMETIC CONTROL SYSTEMS

Gui Cavalcanti, Carl Herrmann

Computer Architecture, Fall 2008

12/18/2008

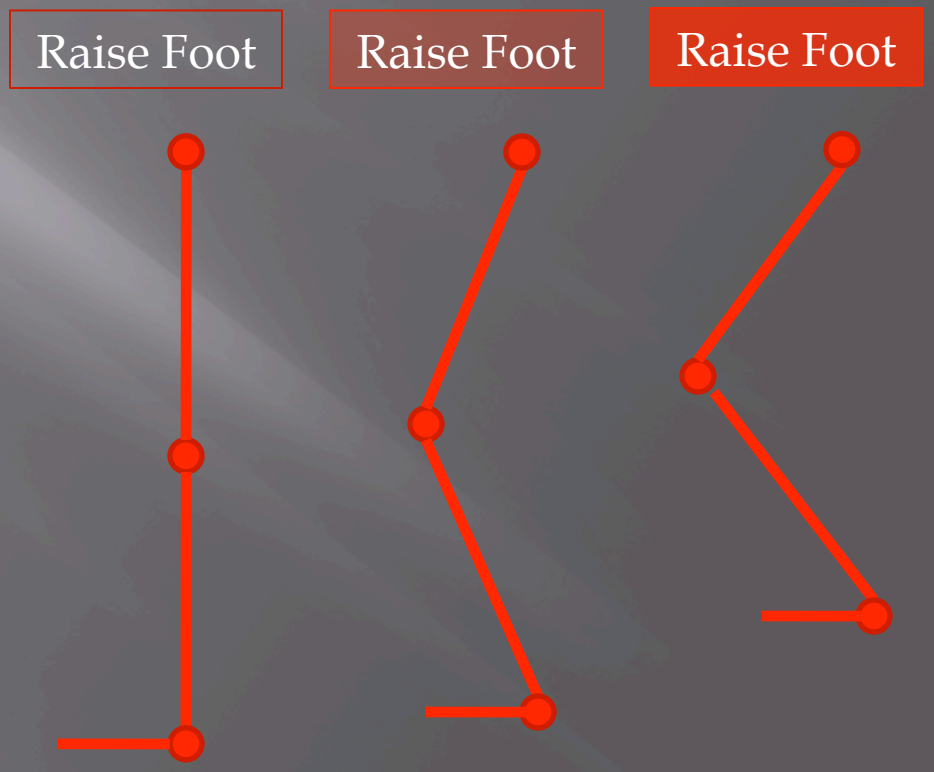
Proposal

- Current biological research suggests certain simple neuronal architectures are responsible for complex organism behaviors
- Such architectures lend themselves easily to sophisticated control via genetic algorithms
- Therefore, we decided to implement genetic algorithms on a simple simulated robot with a biomimetic neuronal control system

Initial Research

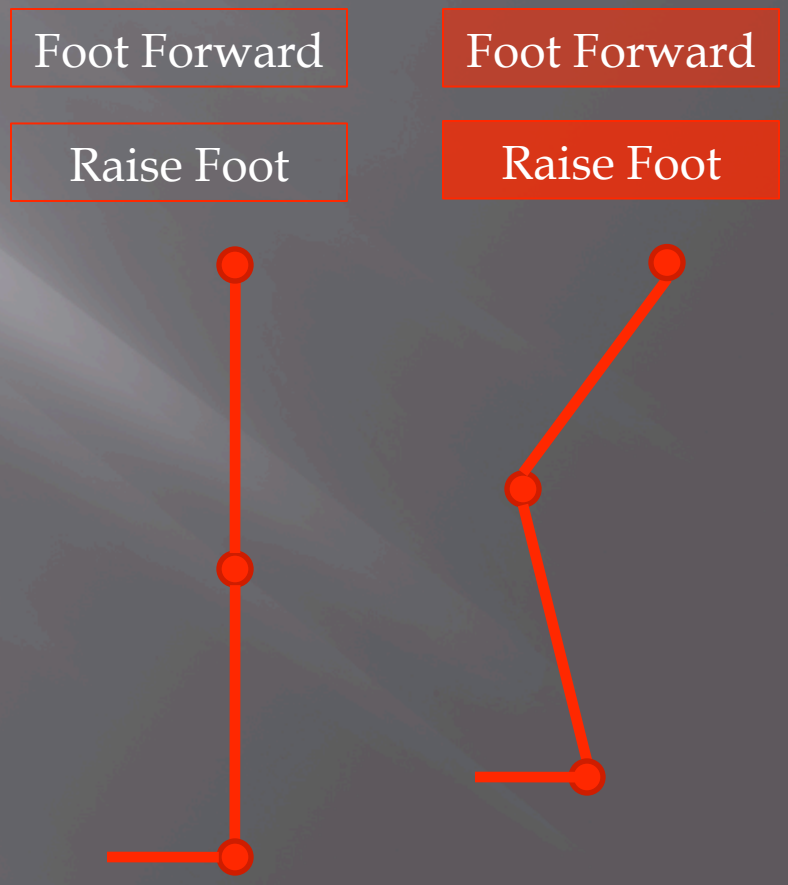
Biomimetic Research: Motor Neurons

- Motor neurons are the closest neurons to muscle groups along the neural path
- They directly or indirectly control one or more muscles at a time
- They tend to control muscles in such a way as to individually command basic actions such as “lift foot” (as opposed to controlling “twitch this one muscle”)
- They are inherently analog, with the command signal sent to them being amplified by them proportionally



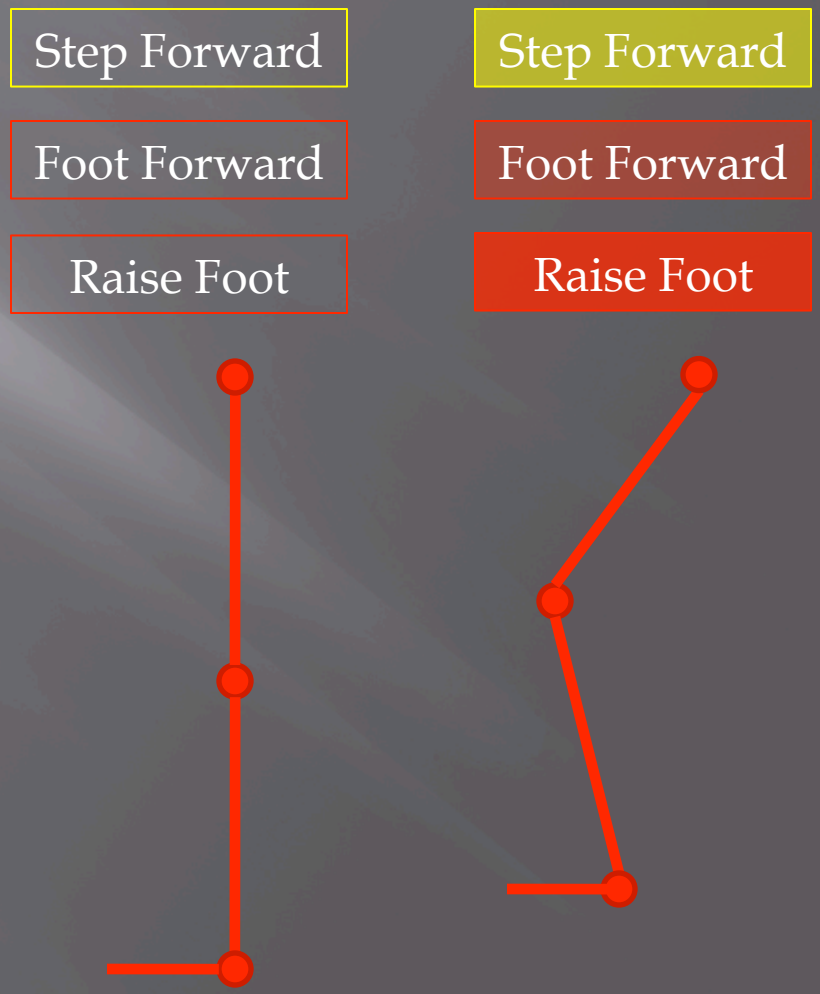
Biomimetic Research: Motor Neurons

- Multiple motor neuron signals can be superimposed on the same muscle groups
- Superimposition leads to complex motions



Biomimetic Research: Command Neurons

- Command neurons control multiple motor neurons to generate complex actions
- Different amplitudes of control signal to command neurons produces different sequences of motor neuron action
- Command neurons can control other command neurons, leading to extremely complex actions



Biomimetic Research: Central Pattern Generators

- Central pattern generators are essentially an organism's "clock" signal, a sinusoidal wave of a given frequency
- This sinusoidal pattern is selectively fed into command neurons to generate reciprocating actions such as walking
- Command neurons can be superimposed
  - Walk sideways + Walk forwards = Walk diagonally

Computational Research: Genetic Algorithms

- GAs are algorithms that mimic evolutionary processes to produce improved function performance given a set of parameters to modify and an output metric to compare against
- They include biological functions such as selection, breeding and mutation.
- They're used for optimization where analytical understanding of a system is difficult
- Require extensive computing power, but always produce a "good" result, and always get "better" with more computational power

Computational Research: Genetic Algorithms

- Genomes
  - A genome is a term for parameters the genetic algorithm modifies, that are inputs to the simulation
  - In any one iteration of a genetic algorithm, multiple genomes are run and their performance is compared
- Selection
  - Simulations provide outputs given genome inputs to compare against a selection metric, for each set of generated parameters
  - Selection algorithm then determines which genomes are "good" and which are "bad", and selects the top performers to continue forward
- Breeding
  - Good genomes live on to the next generation
  - Empty spaces in the iteration docket are filled with "children", crossed genomes of well-performing parents
- Mutations
  - Some genome attributes are randomized within a restricted range

The Project

Project Description

- Create a robot model whose movement is dictated by a set of simple motor neurons, and evolve several command neurons that control simple coordinated actions

Project Requirements

- Simple simulator
  - Written in Matlab due to previous student experience and ease of data display
  - Planar world with high viscous drag
- Simple robot model
  - Robot is modeled as a disc with 6 massless legs
  - Each leg can generate a force in any planar direction
  - Robot has linear and rotational inertia
- Simple genome
  - Each robot has 6 leg genes
  - Each gene corresponds to a command neuron that responds to the central pattern generator
  - Total genome represents a command neuron at the level of "walk forward," "strafe," or "turn in place."
- Genetic algorithm
  - Selection, breeding and mutation algorithms

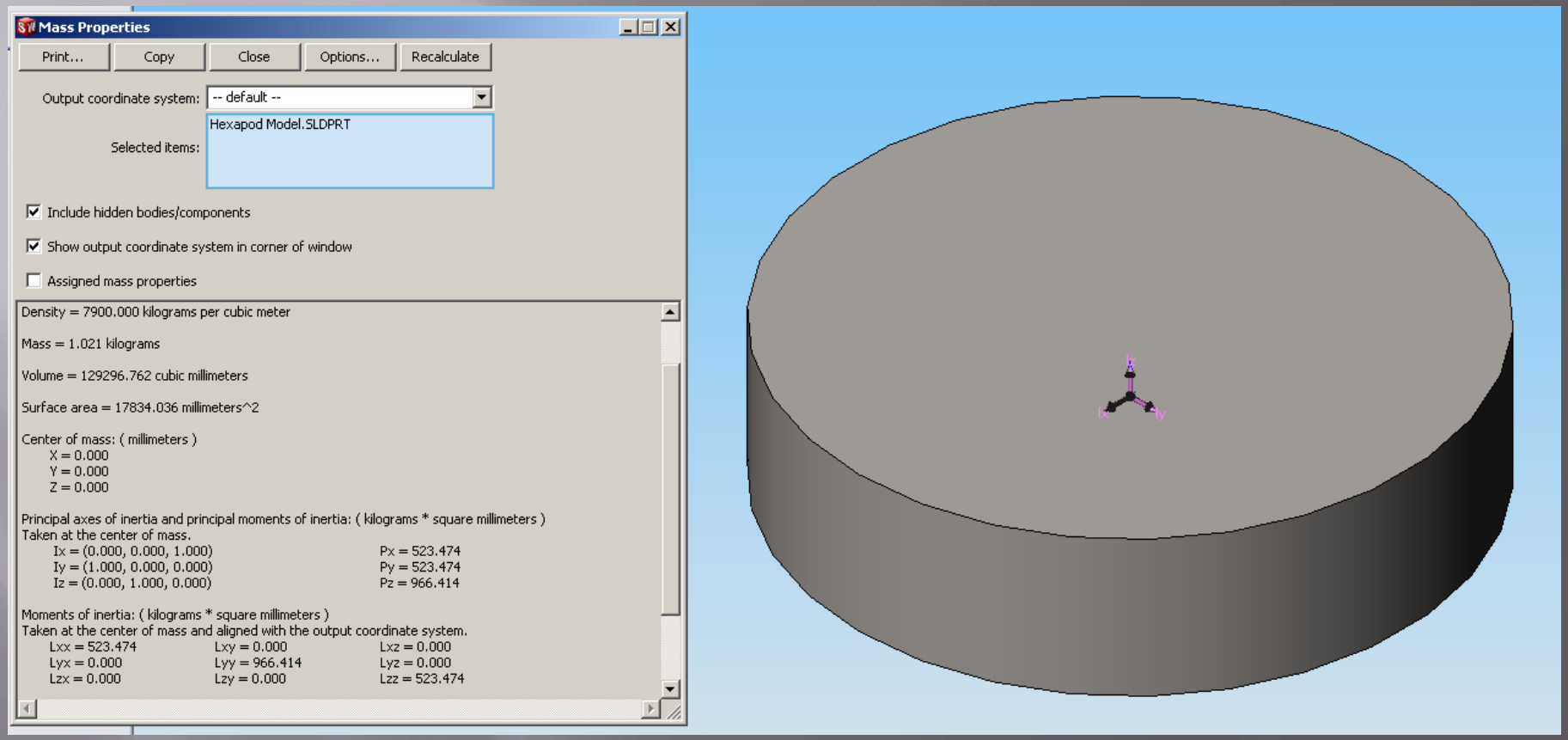
Simulation

- Uses a planar rigid body kinematics solver
- Solves for robot position and velocity given genomes, using ode45
- Includes viscous drag intended to stop robot motion within three periods of the Central Pattern generator (CPG)
  - -50 Ns/m linear drag
  - -.0358 Nms/rad rotational drag
- Written in Matlab

Robot Model

- We needed a physical model that gave us reasonable physical values to use in the simulator
- We modeled the robot with the following physical parameters
  - 1 kg steel disc
  - 87 mm x 22 mm
    - Geometry was a function of desired weight
  - 9.66e-4 kg*m^3 rotational inertia
    - Inertia was a function of geometry and density

Robot Model


Mass Properties
Print...
Copy
Close
Options...
Recalculate
Output coordinate system: -- default --
Selected items: Hexapod Model.SLDPRT
Include hidden bodies/components
Show output coordinate system in corner of window
Assigned mass properties
Density = 7900.000 kilograms per cubic meter
Mass = 1.021 kilograms
Volume = 129296.762 cubic millimeters
Surface area = 17834.036 millimeters^2
Center of mass: (millimeters)
X = 0.000
Y = 0.000
Z = 0.000
Principal axes of inertia and principal moments of inertia: (kilograms * square millimeters)
Taken at the center of mass.
Ix = (0.000, 0.000, 1.000) Px = 523.474
Iy = (1.000, 0.000, 0.000) Py = 523.474
Iz = (0.000, 1.000, 0.000) Pz = 966.414
Moments of inertia: (kilograms * square millimeters)
Taken at the center of mass and aligned with the output coordinate system.
Lxx = 523.474 Lxy = 0.000 Lxz = 0.000
Lyx = 0.000 Lyy = 966.414 Lyz = 0.000
Lzx = 0.000 Lzy = 0.000 Lzz = 523.474

Robot Model/Genome

- Robot legs have the ability to produce forces in a given direction, whose amplitudes are linear functions of the CPG
- The robot genome controls 5 parameters
  - Leg force amplitude with respect to CPG
  - Direction of force with respect to body
  - Phase of leg force oscillation with respect to CPG phase
  - Phase of leg liftoff with respect to CPG phase
  - Duty cycle of leg liftoff with respect to CPG period
- Leg liftoff is a binary value; either the leg is producing force with respect to the CPG, or it is "off the ground" and unable to produce force
  - This allows legs to produce net forces in single directions without producing negative forces due to the CPG oscillation

Genetic Algorithm: Selection

- Our selection process first removes certain unfit genomes from the potential pool
  - Examples include genomes which move backwards when instructed to move forwards
- The process then picks the top 20% of the remaining pool and moves them into the next iteration as-is

Genetic Algorithm: Breeding

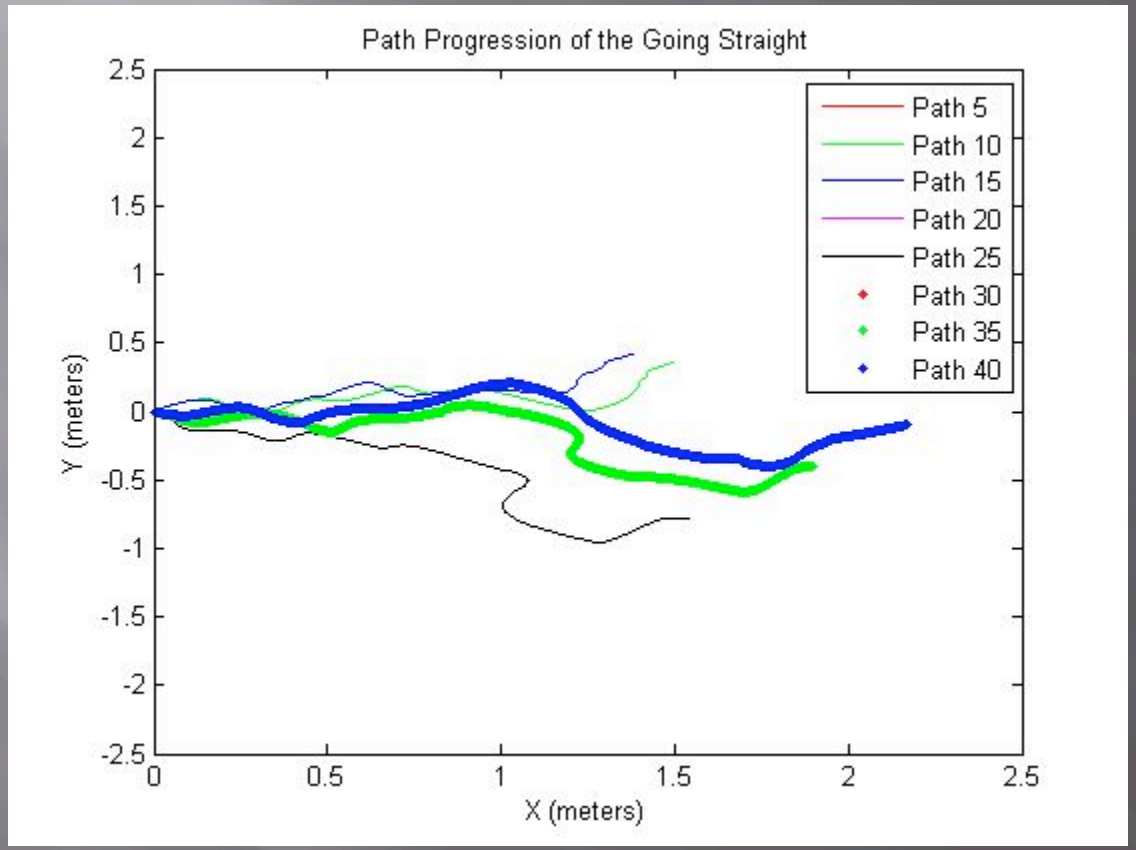
- Our breeding process fills the remaining spaces in the iteration dockets with children of the remaining genomes
- A random half of the parameters of one parent are mixed half of the parameters of another parent
- Mutations are then applied

Genetic Algorithm: Mutations

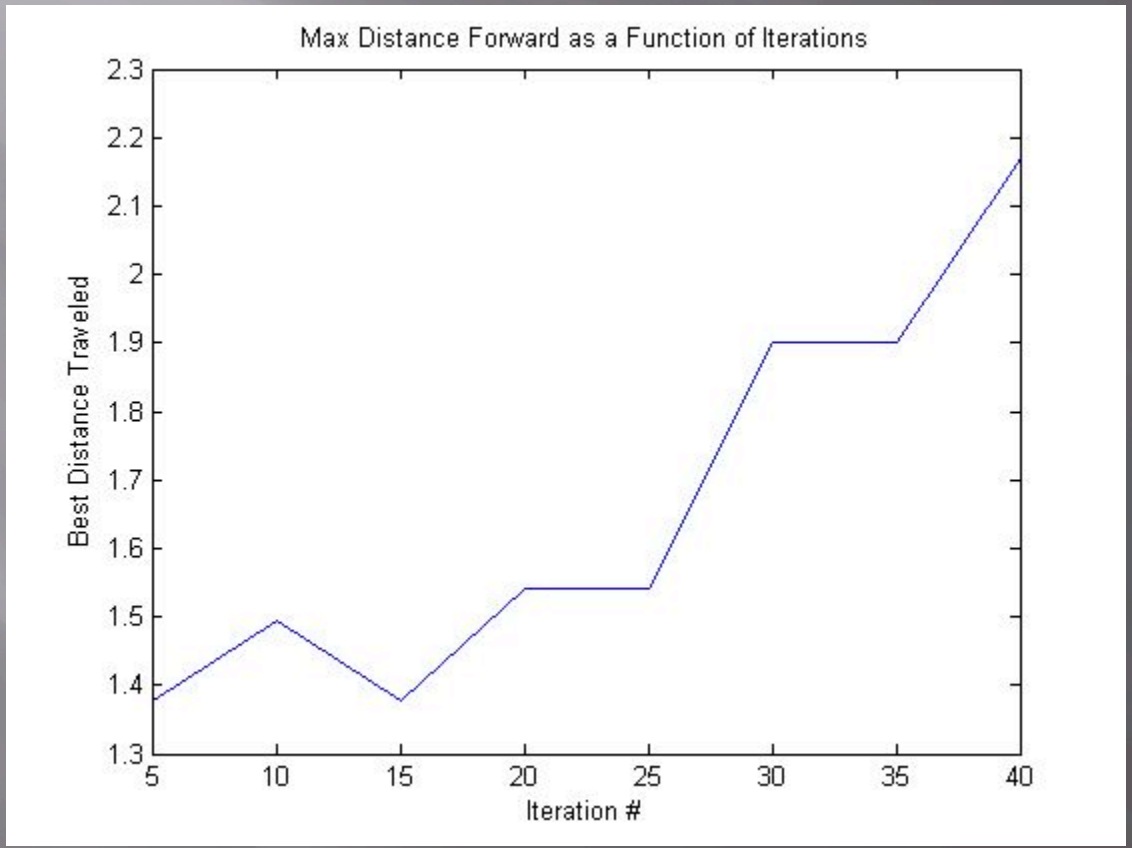
- Our mutations happen within a restricted range of potential values of the genome
- A point value mutation happens about once per child

Results

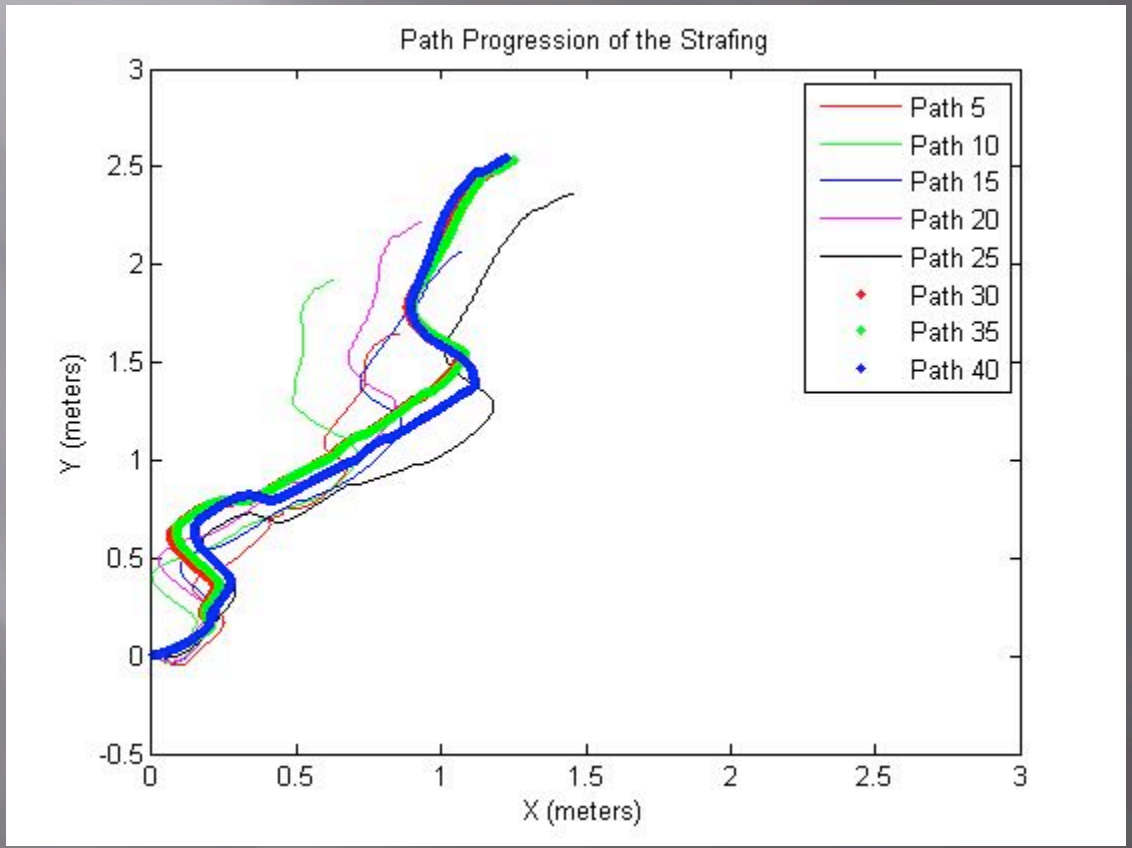
Data: Forward Path



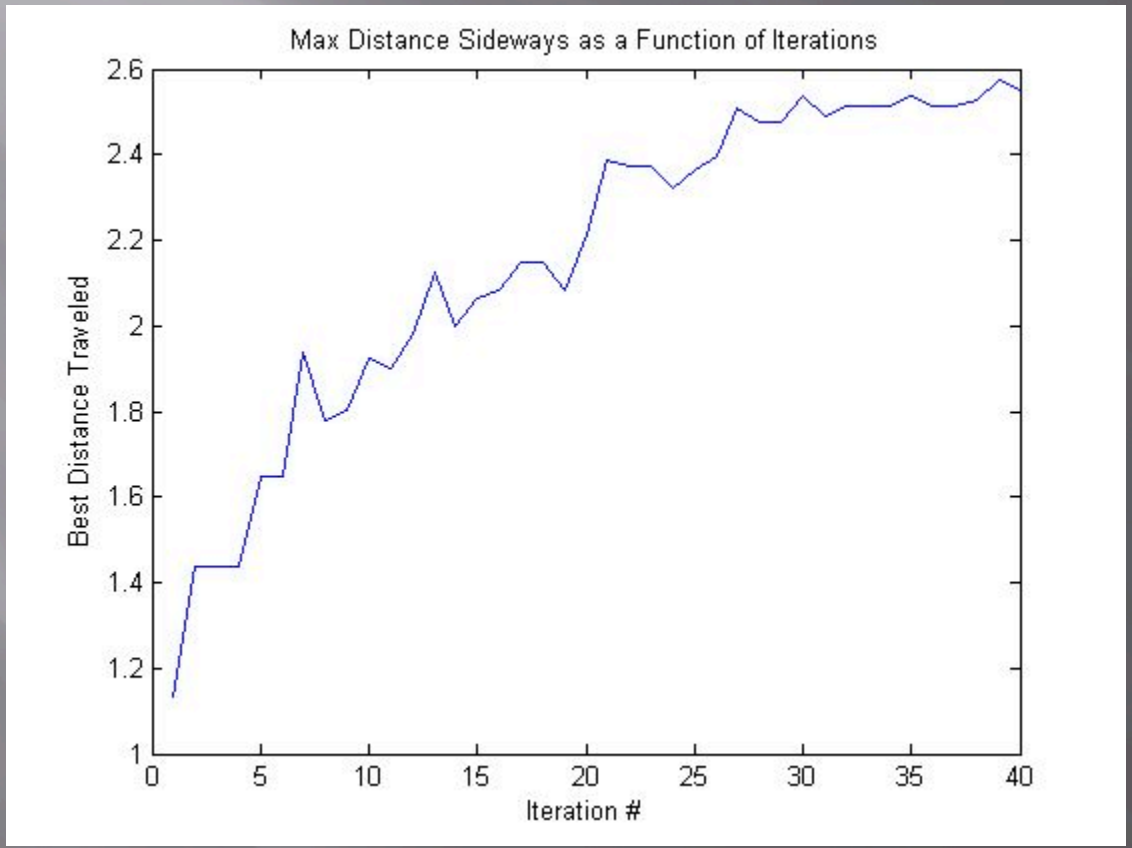
Data: Forward Path



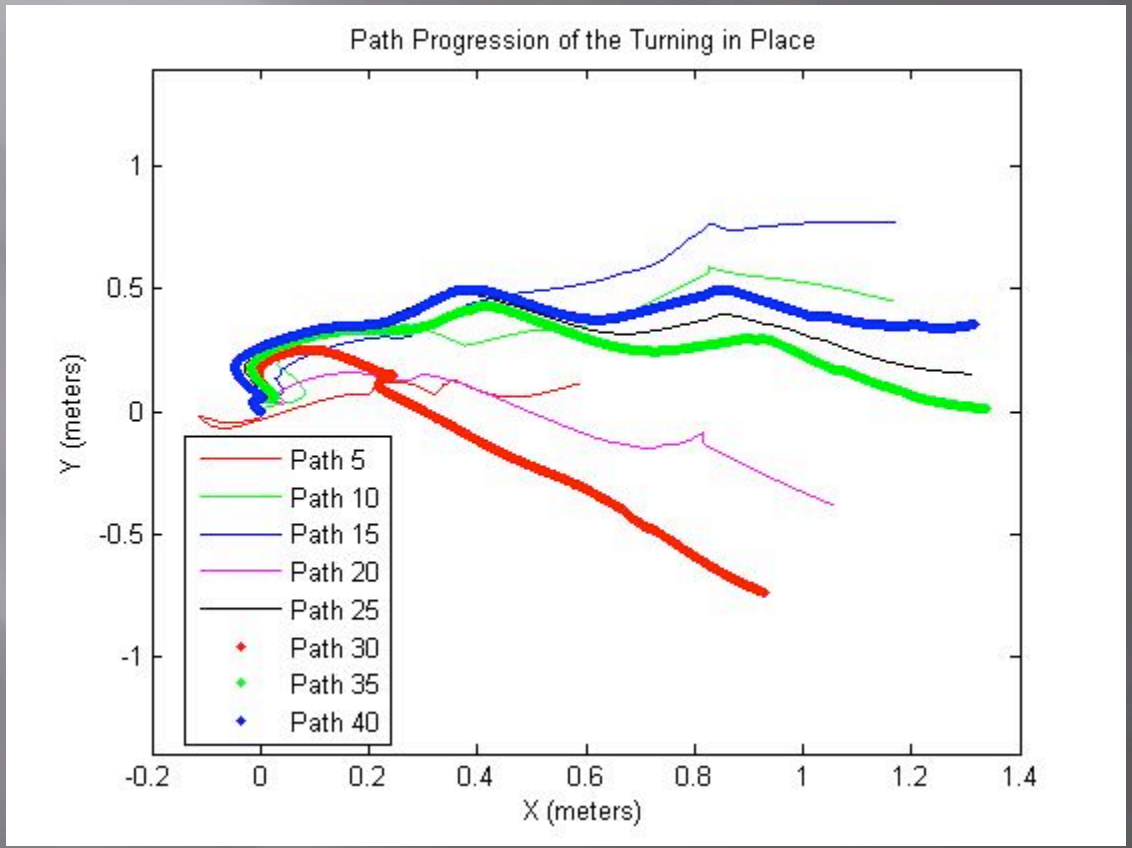
Data: Strafing



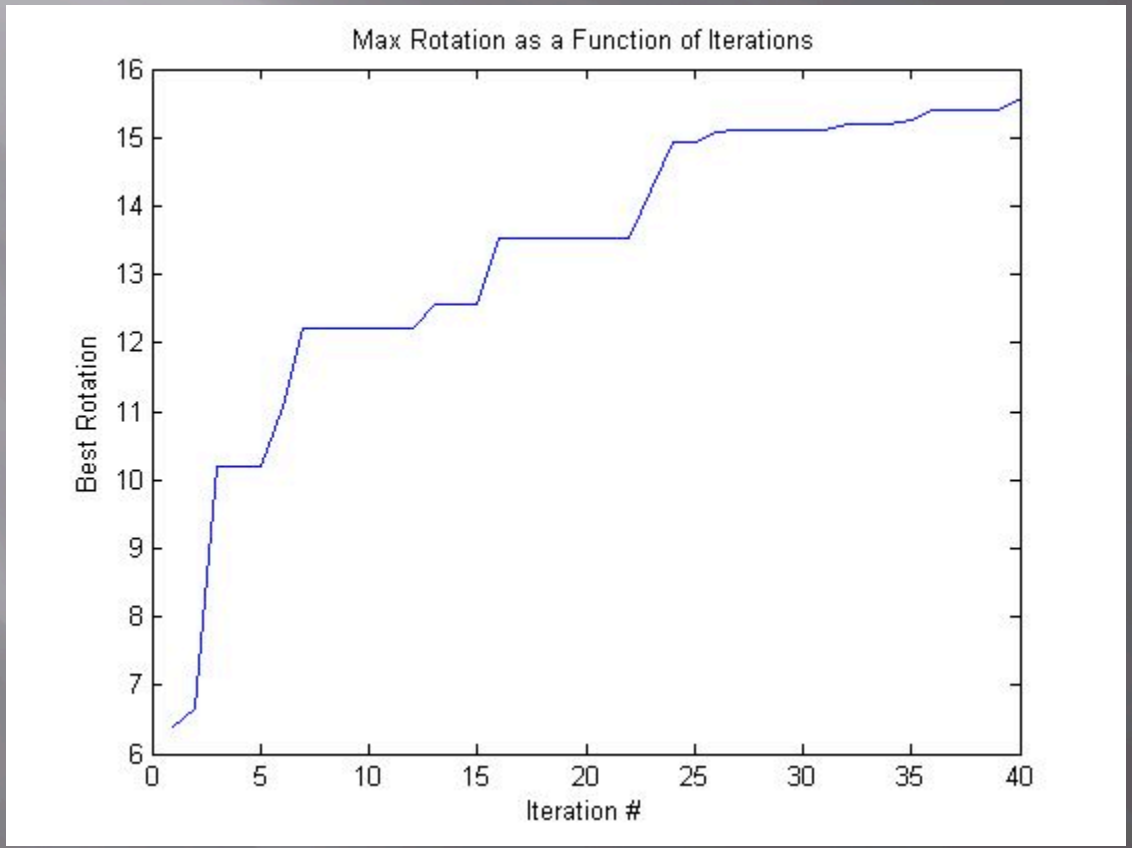
Data: Strafing



Data: Turning In Place



Data: Turning In Place



Data Reflections

- Results are indicative of benefits and pitfalls of genetic algorithms
- Results show that robots are improving, but are getting caught in local optima
- Paths are also not as constrained as we'd like them to be – optimizing for multiple variables (orientation and translation, for instance) is very hard
- Changing the selection, mutation, or breeding algorithms is difficult, and not analytical

Algorithm Reflections

- Selection is difficult
  - Deciding how many members to keep is hard
    - Keeping few members speeds iterations
    - Keeping more members maintains diversity
  - Optimizing for multiple variables is also hard
    - Fewer variables optimize well for those variables, but other aspects are completely uncontrolled (i.e., it walks forward but spins at the same time)
    - More variables increases solution space, leading to relative ineffectiveness of algorithm
- Escaping from local optima is difficult to do, especially after several iterations

Project Reflections

- This was an appropriately sized, interesting project for two people, for four weeks
- We learned a lot about how genetic algorithms work, what they're good for, and what they're not good for