CS 378: Autonomous Intelligent Robotics

Instructor: Jivko Sinapov

http://www.cs.utexas.edu/~jsinapov/teaching/cs378/
Announcements

FRI Summer Research Fellowships:
https://cns.utexas.edu/fri/beyond-the-freshman-lab/fellowships

Applications are due March 1st but apply now!

Funding is available for 4-5 students per FRI stream
Progression

2D simulation

3D simulation

Real World
The Gazebo 3D simulator

- Install gazebo_ros package:
  
  `sudo apt-get install ros-indigo-gazebo-ros`

- Run the simulator:
  
  `roslaunch gazebo_ros rubble_world.launch`

- Guide for installing the gazebo simulator on Mac OS:
  
Readings for this week


Today

• Overview of Homework 3 solution

• ROS launch files example

• Discussion on Homework 4: Multi-Agent System
ROS Launch Files

• Example with turtlesim
• Using waitForService(“…”) when launching multiple nodes at once
• A few things about roslaunch files:
  – A launch file may include another launch file, even from a different package
  – To start a launch file:
    roslaunch <package_name> <roslaunch_filename>
  – No need to start a roscore
Homework 4: Multi-Agent System
Homework 4: Multi-Agent System

- How should we break down the problem?
- What should each agent “sense” about the environment?
- How should each agent make a decision about its linear or angular velocity at each time step?
Reactive Paradigm Example
Breaking the problem down

• What dependencies should the package have?
• How many nodes / launch files do I need to write?
• How should I modify the CMakeLists.txt file?
• What part is easy and what part is hard? Where should I start?
What should go in your ROS package
What is easy and what is hard?

Easy / Simple          Hard / Complex
Implementing a random walk
Implementing a following behavior

• What should the agent know about itself and the target?
Following Behavior

\[
(x_a, y_a) \quad \theta_a
\]

\[
(x_t, y_t) \quad \theta_t
\]
Following Behavior

\[(x_a, y_a)\]
Computing the angle

In quadrant II, \( R_x < 0, R_y > 0 \), and the arctangent gives angle \( \phi \) with a negative sign.

\[
\theta = \text{atan} \left( \frac{R_y}{R_x} \right) + 180^\circ
\]

In quadrant III, \( R_x < 0, R_y < 0 \) and the arctangent gives angle \( \alpha \) with a positive sign.

\[
\theta = \text{atan} \left( \frac{R_y}{R_x} \right) + 180^\circ
\]

In quadrant IV, \( R_x > 0, R_y < 0 \) and the arctangent gives angle \( \beta \) with a negative sign.

\[
\theta = \text{atan} \left( \frac{R_y}{R_x} \right) + 360^\circ
\]

The "standard angle" is taken to be the angle counterclockwise from the positive x-axis. It is a positive number between 0° and 360°.
Computing the angle

\[
\begin{align*}
(-2, 3) & \quad \text{atan2}(3, -2) = 123.7^\circ \\
(2, 3) & \quad \text{atan2}(3, 2) = 56.3^\circ \\
(-2, -3) & \quad \text{atan2}(-3, -2) = -123.7^\circ \\
(2, -3) & \quad \text{atan2}(-3, 2) = -56.3^\circ
\end{align*}
\]

[http://i.stack.imgur.com/xQjWG.png]
Computing the relative angle

\[ \theta_r = \text{atan2}(1.0, 1.0) = \pi/4 \]

\[ x_t - x_a = 1.0 \]
\[ y_t - y_a = 1.0 \]
Computing the relative angle

\[ \theta_r = \text{atan2}(-1,1) = -\pi/4 \]

\[ x_t - x_a = 1.0 \]
\[ y_t - y_a = -1.0 \]
Computing the relative angle

\[ \theta_r = \text{atan2}(-1,-1) = -3\pi/4 \]

\[ x_t - x_a = -1.0 \]
\[ y_t - y_a = -1.0 \]
Following vs Avoid Behavior

$\theta_b$
How should we balance following the “fish” with avoiding the “shark”? 
One solution: A finite state machine

Patrol → Follow

Follow → Attack

Patrol → Attack

Player dies

Sees Player → Player within range

Player disappear from sight → Player get out of range
One solution: A finite state machine
What would this look in code?
Any alternatives?
Homework 4: Prerequisites

• ROS tutorial on launch files (#8):

• ROS tutorial on services (#14)

• Turtlesim video tutorial:
  http://wiki.ros.org/turtlesim/Tutorials#Video_Tutorials
Create a new package called “cs378_<eid>_hw4”
The package's dependencies should include the turtlesim package
Homework 4: Part 1

- For part 1, the task is to write a ROS node which adds a new turtle to the simulator.
- After adding the new turtle, it should follow turtle1.
- Include a launch file called “hw4_part1.launch” which should launch the simulator, your node and the keyboard teleop node to control turtle1.
Homework 4: Part 2
Homework 4: Part 2

• For Part 2, you should implement three different ROS nodes, with each corresponding to the “turtle”, the “shark”, and the “fish”.

• Behavior:
  – “fish” should move randomly with low velocity
  – “shark” should follow the turtle
  – “turtle” should avoid the shark but try to get to the fish
Homework 4: Part 2

- For Part 2, you should implement three different ROS nodes, with each corresponding to the “turtle”, the “shark”, and the “fish”.

- Behavior:
  - “fish” should move randomly with low velocity
  - “shark” should follow the turtle
  - “turtle” should avoid the shark but try to get to the fish
Homework 4: Part 2

• A single launch titled “hw4_part2.launch” should launch all 3 nodes along with the turtlesim simulator
• 2 of the 3 nodes, the “fish”, and the “shark” should make a client call to the simulator to add a turtle that will represent them
Homework 4: Part 2

• Due Friday March 4th
• What to turn in:
  – A zip of your package as it is in the catkin_ws/src folder
  – A README file inside the package describing how you solved the problem and whether any extra credit was completed
THE END