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 1\textsuperscript{st} but apply now!

Funding is available for 4-5 students per FRI stream
Semester Schedule

C++ and Robot Operating System (ROS)
Learning to use our robots
Computational Perception
Developmental Robotics
Human-Robot Interaction

You are here
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:
  
Today

• Reading Discussion

• ROS Services II

• Homework 4
"A robust layered control system..."

**Figure 2.** A decomposition of a mobile robot control system based on task achieving behaviors.
“A robust layered control system...”

Functional decomposition:

- Sensors
  - Perception
  - Modelling
  - Task execution
  - Planning
  - Motor control
  - Actuators

Behavioral decomposition:

- Sensors
  - Reasoning about behavior of objects
  - Planning changes to the world
  - Object identification
  - Change detection
  - Map building
  - Exploration
  - Wandering
  - Obstacle avoidance
  - Actuators
"A robust layered control system..."

Figure 3. Control is layered with higher level layers subsuming the roles of lower level layers when they wish to take control. The system can be partitioned at any level, and the layers below form a complete operational control system.
"A robust layered control system..."

Figure 4. A module has input and output lines. Input signals can be suppressed and replaced with the suppressing signal. Output signals can be inhibited. A module can also be reset to state NIL.
“A robust layered control system...”

“I am interested in learning more about the sonar system that the robot implements in order to create a map of its surroundings. I am curious about some of the technicalities of this system, as well as how accurate it truly is in practice, especially in a room filled with constantly moving beings/objects. “

- Zara
“A robust layered control system...”

“How will complex decisions based on input from various “layers” be made without some central [module] driving it all?”

- Saket
“A robust layered control system...”

“They said their work was limited by the hardware to teach robots learning needing a "new sort of video camera and high-speed low-power processing box to run specially developed vision algorithms at 10 frames per second" to make progress". With hardware rapidly getting better, has this been made already and if so how far sway are they have they progressed?”

- Nathan
“Intelligence without representation”

(CAN (SIT-ON PERSON CHAIR)), (CAN (STAND-ON PERSON CHAIR)), 
Fig. 2. We wire finite state machines together into layers of control. Each layer is built on top of existing layers. Lower level layers never rely on the existence of higher level layers.
“A robust layered control system…”

“Some questions I had were about the layering system. What would happen if say, there were contradictions in the desires with the same importance? How would the robot then respond to this situation?

- Justin
“A robust layered control system…”

“Is it still believed that it is impossible to create a robot with human intelligence?

Just curious, what is the debate surrounding animal intelligence? How does it apply to Artificial Intelligence?

Is this approach still used in practice? How many layers have been achieved now?

-Amrutha
Readings for this week


ROS Services
Messages vs. Services
Messages vs. Services
Messages vs. Services

Publisher

Subscriber

topic

t_1

Client

Server

request

response

t_2

t_3
Useful command line tools

rosservice list : list all available services
rosservice info <name> : print information about service to the console
rosservice call <name> <request> : call a service from the command line
Calling a service from code

• Find out what the name of the service is
• Find out what the type of the srv is (i.e., what package it is declared in and the name of the .srv file)
• Add the package where the srv is declared to your dependencies in package.xml and CMakeLists.txt
Calling a service from code (con't)

• In your ROS node code, include the header file for the srv
  – e.g., if the srv is turtlesim/Spawn.srv then “#include "turtlesim/Spawn.h"

• Create the client object as described in Tutorial #14:
  http://wiki.ros.org/ROS/Tutorials
Homework 4: Multi-Agent System
Reactive Paradigm Example
Reactive Paradigm Example
Reactive Paradigm Example
Reactive Paradigm Example
Reactive Paradigm Example
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
Homework 4: Part 1

• 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
THE END