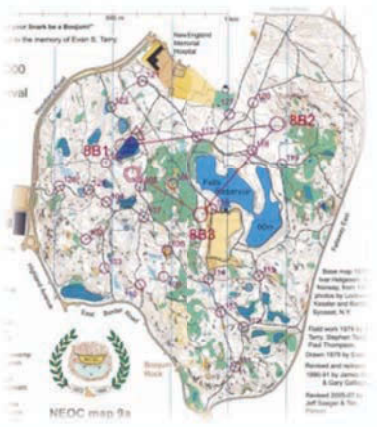
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
16.412J / 6.834J – COGNITIVE ROBOTICS
SPRING 2016

GRAND CHALLENGE

INTRODUCTION



As the culmination of Cognitive Robotics this semester, you'll complete a Grand Challenge: a showcase of the technology you've developed and taught in your Advanced Lecture groups, but applied to a real robotic platform. This will be the final project for this course, and we hope you have **fun** with it!

This project will be somewhat **flexible**, like the advanced lectures, in that you'll have some creative liberty on how you want to implement apply your work to the grand challenge (though some suggestions from the course staff are provided later in this document).

SCHEDULE & GRADING

The grand challenge itself will be on Wednesday, May 11th during the normal class period, from 10:30am – 12pm. There will be a grand challenge practice / dry run on the preceding Monday May 9th, also during class time. As such, there won't be any formal lecture during the last week of classes (and of course, there is no final exam). Additionally, many groups will need plenty of lab time with the robot. This will be arranged on an as-needed basis; just email the course staff.

Your final implementation will be due on Wednesday, May 11th by the time class starts, so that it can be run on the hardware. A short team writeup will be due the next day on Thursday, May 12th.

While the original course syllabus stated that the grand challenge would be 20% of your final grade, this figure has been revised due to course load and time considerations; it will now be 10-15% of your final grade instead.

You'll be working in the same teams as your advanced lectures. All team members should divide up the work amongst themselves, and each must contribute equally to the effort.

THE CHALLENGE

As has been noted many times throughout the semester, the grand challenge will follow a **modified Orienteering theme**[1]. Namely, the challenge will consist of driving a robot autonomously to several "challenge" stations in an environment, and must complete as many challenges as possible and as quickly as possible.

The robot we'll be using for this project is pictured. It's a Summit X platform (originally an X-WAM with a robotic arm that's been

[1] Your TA was ***extremely*** tempted to make this Star Trek-themed as well, but alas, acquiring reliable 24th century technology proved difficult.

removed), who has lovingly been named Thor. This mobile platform is rather powerful, and has four mechanum wheels – making it omnidirectional (it can move sideways in addition to forward and back). Additional capabilities and sensing on this robot include a Hokuyo lidar, an onboard camera with point, tilt, and zoom capabilities, a Kinect 3D camera, the ability to lift it's top up and down vertically via a scissor lift mechanism, and an onboard quad-core computer with wireless networking. We'll be running our software on this robot, and also on a base station computer that can wirelessly communicate to the onboard computer.

The grand challenge will take place in the MERS lab (32-226). The course has been constructed wth giant Lego walls, with certain areas being marked as **challenge stations**. At each challenge station, the robot will be required to solve a computational puzzle. The goal is to drive around to the various regions and solve as many computational puzzles as possible, in as short a time as possible.

PROPOSED TEAM PROJECTS

Below are some proposed team projects. Note that these are flexible and up for discussion. Please email the course staff **immediately** with any concerns you have, or if you have another idea of how you'd rather apply your advanced lecture implementation to the grand challenge. More information for each team is in a later section of this document.

Team	Task
Incremental Path Planning	Integrate D* lite implementation with ROS navigation stack
Semantic Localization	Port semantic localization implementation to ROS; run on real robot & compare with geometric localization
Visual Classification through Deep Learning	Identify the presence / absence of to-be-chosen fiducials placed around the area from ROS camera images
MCTS	**Challenge station:** Solve a game / puzzle against a human OR High-level planning for robot
Reachability	**Challenge station:** Solve a reachability puzzle
Planning with Temporal Logic	PDDL modeling of the challenge (with Büchi automaton), solving with planner for high-level robot control
Infinite Horizon Probabilistic Planning	Model challenge as MDP, find policy via LAO* for high-level robot control

Note that throughout much of the above, you'll need to interact with the course staff. This is as much a project for them as it is for you; as their work in running this challenge will heavily depend on exactly what each team does. So please definitely keep them posted about your thoughts and progress.

In order to integrate with our planning & execution framework, your software will need to use ROS (the Robot Operating System). ROS is a framework that is well established in the robotics community. At it's core, it provides message passing. Programs that can communicate with ROS (called ROS nodes) can send

messages to each other over "topics." Additionally, nodes can call special functions in other nodes called "services," and can also call longer-running actions in other nodes called "actions." Please refer to the ROS documentation for more information; there are great tutorials online. **For this assignment, please be sure to use ROS indigo**. (There are different versions of ROS). Note that ROS is only supported on Linux (and best supported on Ubuntu) – and it should be easily installed in the course VM via the instructions on the ROS website.

GROUP-SPECIFIC CONSIDERATIONS

Since each group will be doing a very different project, here are some specific thoughts for each group.

INCREMENTAL PATH PLANNING

It would probably be best to implement a ROS package that interfaces with the ROS navigation stack. Specifically, one approach would be to subscribe to ROS messages for the map, the pose of the robot within that map, the costmap, etc. It could then call D* lite to get a path, which will be a sequence of waypoints forming a trajectory. These waypoints could then be sent, one by one, to the local / base planner of the ROS navigation stack for execution. Incremental replanning could be re-called after each point waypoint. The local costmap could also be used to update the weights of edges in your graph.

What the course staff could provide you with:

- Plenty of testing time with the robot
- Our existing code for calling the navigation stack, for use as a template to insert the code for calling your code instead

What you would deliver:

- ROS node properly integrated with the navigation stack

SEMANTIC LOCALIZATION

It is likely possible to wrap your semantic localization implementation around ROS, possibly with some scaling / speed considerations, to make it run with input from the Visual Classification group (please read their section below). Your team should help them choose appropriate objects to recognize and their placement in the course, such that you think it will allow for good localization. This will likely involve a joint meeting together in our lab space.

What the course staff could provide you with:

- A ROS message type listing the objects currently visible. This would be the output of the visual classification group

What you would deliver:

- A ROS node that publishes the current estimated pose of the robot, as a ROS transform /tf message and as a Marker message (viewable in ROS's rviz)

VISUAL CLASSIFICATION THROUGH DEEP LEARNING

The XWAM has an onboard camera which would be perfect for doing object detection. What you could provide is a ROS node that reads in camera images from this onboard camera, and publishes a message saying what objects (if any) are recognized in that image. This message would be the input to the semantic localization node, which would use it as a basis to perform localization.

There are several tasks that need to be done here, likely in collaboration with the semantic localization group:

1. Choose what physical objects / images you want to try and recognize
2. Physically place those objects / images around the test course at pre-defined locations (which will be recorded in a map)
3. Record several video sets of the robot driving around within the test course. Different frames of those videos will have different objects visible in them
4. Train your CNN / classifier with some of those video sets, and possibly use any extras to validate against
5. Make a ROS node to recognize several objects in scenes, ideally in real time (but it's fine if it's a bit slower)

The course staff has some thoughts on each of the above. Task 1 should be chosen in collaboration with the semantic localization group. From their lecture, it seems that it may be better to choose fewer classes of objects, and have many instances of objects from those classes around the testbed, **but you should check with them to be sure**. Feel free to use small physical objects, if you can find / have access to something appropriate. Otherwise, a likely easier logistically approach would be to print out images of certain things on paper, and attach those papers at certain locations around the test course.

Tasks 2 and 3 would likely require a significant amount of time in our lab space for setup and for recording training data with our robot, so it may be best to book several hours one afternoon and also have some someone present from the semantic localization group the part of that meeting when objects are placed. For recording the videos, we recommend using ROS's rosbag capability, which can record a "bag" of topic messages and allow you to "play them back" later in simulation on your computer. This could allow to test your program at home, by making it think that it's seeing live input from the robot without actually being in lab – it'll be playing back the recorded images in real-time.

Task 4 is probably similar in some ways to the CNN mini problem set, but possibly with some scaling considerations. You may also need to use OpenCV or another API to convert the images from ROS to an appropriate format for tensorflow (or any other library you choose to use). A possible concern: I'm not sure exactly how many images you'd need to train with, but as CNN's can sometimes require a lot, you'd need to make sure you record enough data.

Task 5 would output a ROS message with a common type, for use by the semantic localization group.

What the course staff could provide you with:

- Access to hardware and the robot's camera, for recording test & validation video
- The ROS message type for your output

What you would deliver:

- ROS node that reads in camera images published via ROS, and outputs message for what is in the scene

MCTS

One possibility for incorporating MCTS into the grand challenge would be to implement an interactive game at one of the challenge stations. This game could be against an advanced adversary, or even cooler, be against a human player. Your MCTS implementation could be applied to a different game, and you could also develop a GUI for a human to play against your implementation.

What the course staff could provide you with:

- A ROS template for integrating your game into our larger planning & execution architecture

What you could provide:

- A ROS node that implements game play, likely including a game GUI

Another possibility, brought up by one of your teammates after class, would be to use MCTS to direct the robot's high level actions. This is also a possibility, and the course staff leaves it to you to figure out how or if you'd like to do this. Please talk amongst your team and keep the course staff posted as to your thoughts.

REACHABILITY

One possibility for incorporating reachability into the grand challenge would be to implement it as a challenge station. A scenario could be that the robot discovers an underground cave, and must quickly explore that cave via a (simulated) detachable scout robot that must see if a certain object of interest (such as gold, or extra batteries, etc.) is reachable from the labyrinth of underground caverns.

This could involve several things:

- Updating / augmenting the models you used for your mini problem set implementation to suit this scenario
- Modeling a few complicated domains and performing reachability tests with them

The course staff could provide you with:

- A few new test scenarios, similar to the domain that you model, on the day of the grand challenge itself. This would be the one used during the competition to test for reachability
- A ROS template for integrating this challenge into our larger planning & execution architecture

What you could provide:

- A ROS wrapper around your reachability implementation
- An example problem

There may be other ways to integrate the notion of reachability into the grand challenge. The course staff asks that you talk amongst yourselves in the next day or so, and if you have any different ideas to please run them by us.

PLANNING WITH TEMPORAL LOGIC

Since our robot will need to have a plan to decide what to do, applying your advanced lecture implementation to controlling the high-level behavior of the robot seems like it would be a natural fit. Namely, you could in essence, generate high-level plans for the robot to achieve its goal.

There would be a few steps to this:

1.) Model the grand challenge in PDDL, and model any LTL goals with a Büchi automaton
2.) Use your implementation to transform this PDDL and Büchi automaton to a new PDDL problem that can be solved with a classical planner. Call a classical planner to get the plan.
3.) The above step effectively creates a planner with time-evolved goals. Wrap it with ROS and integrate it with our existing planning & execution system.

What the course staff could provide you with:

- Pointer on how to integrate with our existing planning & execution system
- The names and parameters of PDDL operators, for use in your modeling

What you would deliver:

- A PDDL model of the grand challenge
- A ROS wrapper around a planner that can be integrated with our existing planning & execution system

INFINITE HORIZON PROBABILISTIC PLANNING

Similar to the planning with temporal logic group, your advanced lecture implementation may work well in controlling the high-level behavior of the robot. Specifically, you could model the grand challenge as an MDP, and then call your LAO* implementation on it to get a policy.

Once you have a policy, you'd need to execute it. For this, you could write a ROS node to execute the policy; it would determine the current state (we have some ROS infrastructure for this in our current planning & execution framework), select the appropriate action from your policy, and then execute it. Furthermore, you could also add an optional flag to simulate the stochasticity from the MDP. For instance, supposing your action is "move forward 1 cell", you could instead with 0.25 probability move left instead. (This trick would allow you to simulate stochasticity in the dynamics!).

What the course staff could provide you with:

- The names of actions that can be dispatched to the hardware (things like "move up 1 cell", etc.), and the ROS API for dispatching commands to the hardware so that you can actually move the robot (those calls will end up calling a motion planner for instance)
- The API for our existing planning & execution system that publishes the state. This would allow you to make observations, know your current state, and execute the appropriate action from your policy.

What you would deliver:

- An MDP model of the grand challenge
- A ROS node that, given an MDP model, solves it using LAO* and then executes that policy, with optional noise.

DELIVERABLES

Each team will deliver their implementation, and any modeling as appropriate.

Additionally, each team should submit a short (1-2 page) report about their implementation and it's application to the grand challenge. This report should talk about the details of your implementation, any considerations made given that it's being executed on real hardware, any surprises that came up, and lessons learned.

16.412J / 6.834J Cognitive Robotics

Spring 2016