16.401 – Robotics: Science and Systems
Syllabus

Introduction

The objective of the second semester of Robotics: Science and Systems is for the students to apply the lessons they have learned in the first semester to some real-world problem. Working in teams, the students must develop a robotic system (multiple robot systems?) capable of accomplishing a real world challenge.

The challenge should consist of the following components:

- Purposeful motion, e.g., map-based navigation, reactive navigation, *not* random wandering.
- Learning from sensor data. This will most likely mean map-building, but if the students are energized, it could be mean learning models of the robot dynamics, or even reinforcement learning of behaviours
- Manipulation of the environment.

Are there more components? Human-computer interaction?

The Challenge

The fundamental challenge is identical to the first semester of R:SS — build a shelter on Mars. However, this semester, the challenge is much closer to reality. There are large-scale objects in the MIT environment that must be collected and returned to the base location, and then assembled into a structure. This larger challenge is then broken down into a series of sub-tasks. Unlike the first semester, however, these tasks are parallelizable. Example tasks might be (and some of these are mutually exclusive, and some may not be necessary):

- Large-scale motion planning
- Vision-based localization
- Dynamic object detection
- Behaviour-based exploration for map learning
- Map learning
- Wireless network connectivity mapping
- Behaviour-based grasping
- Object manipulation planning

Let's say that we give the robot is an *a priori* spatial map of the environment, its initial position and the location of the objects in the map. For example, the robot could be given a map as in figure 1. Note that this map as given contains no precise metric information. The students could choose to measure building locations themselves, but this would still not precisely give the object positions. Or, the students could use the map to infer a topology of the environment, and then use reactive strategies to navigate as they correlate the topological structure to sensor measurements.

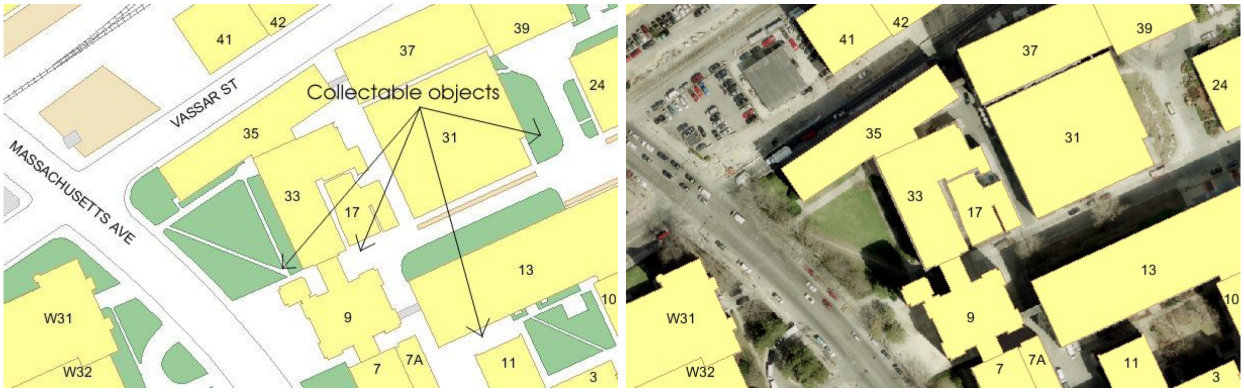

Figure 1: Example map and object locations.

Large-scale motion planning

Team 1 might extend their motion planner to account for a non-round robot, or perhaps a non-holonomic robot. They might also extend their planner so that the map can be updated with new sensor information, triggering replanning.

Vision-based localization

Team 4 might extend their vision-based localization algorithm to incorporate a wide variety of visual features.

Map learning / Wireless connectivity mapping

Team 2 might develop a set of behaviours that allow the robot to explore in a partially known environment, at the same time using an Atlas-like algorithm to build a better map using the output of the vision system built by Team 2.

Grasping

Team 5 might develop a set of behaviours that use visual features to determine how to position the robot with respect to the object to be grasped. This is particularly interesting if we have different kinds of objects to be grasped.

Constraints that I think we should maintain:

- Absence of perfect GPS
- Imperfect map
- Imperfect global networking

We should push on these constraints by regularly breaking the underlying assumptions at the beginning of the semester. For example, we (or at least Andrew) did **not** make an assumption about the presence of a network when he set up the hangar net. We ended up breaking that assumption by moving team members around and putting the repository in 33x. We could have fixed our failure had we made explicit the no-network constraint and then tested it at regular intervals.

In term II, we should make these constraints (and their underlying assumptions) explicit and then test regular. Let's use broken maps, let's take away the network periodically.

System Integration

The challenge is clearly too large a problem for any one team. Each team must choose one or two of the tasks and address these over the semester. The challenge will culminate with all teams running their tasks concurrently on a single robot.

In the first semester, the students each built a robot that had more or less the same capabilities. None of these robots will be sufficiently capable to perform the larger challenge. However, these smaller robots make a good platform for each team to test their particular deployments on. For instance, a team that is working on planning can test their planner in simulation and also on their small robot in the confines of the hangar. A team that is working on localization can teleoperate their robot to gather sensor data.

There is a clear issue of how to integrate the different systems cleanly. This will be addressed in several steps:

- The instructors must ensure that the tasks are sufficiently well-defined to allow clean modularity.
- Each team must define a Java API for their task(s). This API allows all other teams to interact with it. By the end of the first month, this API must be frozen. Stubbed out versions of the API must be provided by each team to all other teams, to allow testing.
- The TAs must work closely with each team to ensure that no changes are made, and that no conceptual errors in interface are made.
- The instructors have to use the system integration as a teaching opportunity. This means lectures in software engineering.

Components

Hardware:

We need a single, bigger, more capable robot. It should be able to travel outdoors, carry more computation.

- Bigger frame. 80/20?
- Additional sensors for the robot. Laser range finders?
- Additional computation on the robot. Give each team a laptop so that all laptops can be rack-mounted?
- On-board network?
- Display on the robot
- Better manipulator?

Software:

- Task manager.
- Additional sensor managers?
- Better manipulator controller?

Decisions to be made:

1. What are the objectives of the larger challenge?
2. What are the tasks of the larger challenge?
3. What kind of objects to recover?
4. What kind of manipulator to use?
5. How many teams?
6. What kind of hardware do we need to buy?
7. What additional **support** software do we need to write?
8. What to do about bad weather?

FWIW: Learning Objectives:

ero/Astro requires learning objectives for courses they offer. These are the ones we listed.

Students will be able to:

1. specify design requirements for an autonomous system performing some task
2. specify an integrated hardware and software design that satisfies the design requirements
3. use kinematics and control to design and implement controllers that satisfy the stated requirements
4. use knowledge of state estimation theory to design and implement algorithms and estimators that satisfy the stated requirements
5. use knowledge of planning and geometric reasoning to design and implement high-level planners that statisfy the stated requirements
6. critically evaluate different choices of design and architectures
7. design an integrated hardware and software system.
8. operate their system for an extended (on the order of hours) period of time