Massachusetts Institute of Technology

Robotics: Science and Systems – Spring 2012

Course Challenge: Construction and Object Search

Dry Run: 5/7/12, 3pm
Hardware Freeze: 5/9/12, 3pm
Final Challenge Run: 5/11/12, 3pm
Challenge Lab Report Due: 5/14/12, 5pm

Objectives and Lab Overview

Starting with this lab, your objective is to program your robot to solve the course challenge which is inspired by the grand challenge: “Build a Shelter on Mars” . Your robot will traverse an environment gathering components. A map of the environment will be provided but the environment will also include some unmapped components. Your robot’s goal is to deposit all the components at a specified location on the map, arranging them in the form of a structure. This lab will give you technical guidance toward integrating many robot modules to solve this complex task.

Time Accounting and Self-Assessment:

Make a dated entry called “Start of Object Search Lab” on your Wiki’s Self-Assessment page. Before doing any of the lab parts below, answer the following questions:

- **Programming**: How proficient are you at writing large multiple component programs in Java (as of the start of this lab)?
  (1=Not at all proficient; 2=slightly proficient; 3=reasonably proficient; 4=very proficient; 5=expert.)
- **Hardware**: How proficient are you at modifying the hardware of your robot?
  (1=Not at all proficient; 2=slightly proficient; 3=reasonably proficient; 4=very proficient; 5=expert.)
- **Robot Programming**: How proficient are you at developing integrated algorithms for your robot?
  (1=Not at all proficient; 2=slightly proficient; 3=reasonably proficient; 4=very proficient; 5=expert.)

To start the lab, you should have:

- This handout
- Your notes from the Planning, Localization, and Mapping lectures

Physical Units

We remind you to use MKS units (meters, kilograms, seconds, radians, watts, etc.) throughout the course and this lab. In particular, this means that *whenever you state a physical quantity, you must state its units*. Also show units in your intermediate calculations.

Course Challenge Description and Rules

The Grand Challenge is to "Build a Shelter on Mars." As a way of addressing the fundamental technical aspects of the grand challenge, each group will build a robot that can explore, search, gather materials, and build a structure in a partially-known environment.

- The robot will be given a map. The map will indicate obstacles, components, boundaries and fiducials (colored spheres to aid in navigation).
- As extra credit, you may optionally choose to have us place obstacles in the environment which are not known to the robot. Otherwise, all obstacles will be in the map.
- The robot's starting location will be known.
- The robot will have to search for prefabricated components of multiple types that can be used to create a structure at the construction site. Some specified portions of the environment will be known to contain these components. However, useful parts are also scattered throughout the environment so that your robots will have to use sensors to look for them.
- The components will be blocks of two different sizes, and a variety of colors.
- Your robots will be fully autonomous: they will be controlled entirely by your code running on the robot-mounted laptop and/or workstation. No human will intervene with the operation of this code for the duration of the run (though up to two physical "nudges" are allowed, see below).
- Your robot's task will be to collect as many components as it can and use them to assemble a very simple structure at a construction location.

Sub-Tasks

Your robot will need to perform the following tasks:

1. **Navigation**: Basic navigation from the initial location to the construction site; motion planning given a map and starting/goal locations; motion planning with unmapped obstacles. Optional: motion planning where the starting location is unknown (i.e. using features from the environment, the robot identifies its starting location on the map).
2. **Identifying Construction Site**: Your robot must be able to get to a construction site (and to know when it gets there), but you are free to define the location of the site. You may either specify it to your robot a-priori or the robot may choose the site autonomously.
3. **Find Objects**: Detect objects at known locations, detect objects at unknown locations (and identify the object type).
4. **Move Objects**: Pick up components, transport them, and deposit them at the construction site.
5. **Construction**: Create a simple structure (e.g., a wall) at the construction site. You are free to define this structure. For example, you may choose to place the components in a pile, to place them next to each other as a one-row wall, or to stack them on top of each other as a multi-row wall.

Depending on your approach to solving the overall challenge, you may wish to add to this basic skill list.

Your robot will have a gradation of skills within each task. You will be evaluated on the skill level of your robot. We strongly advise you to do build the control system of your robot according to the skill hierarchy. Start with the simplest skills, make sure they work, and then move on to the next level of difficulty/complexity. You get rewarded for every level of competence reached by your robot. Your robot is required to have the most basic level of each type of skill. Your challenge is to reach the skill level necessary to accomplish the task in its most basic form and then add as much additional smarts as possible your robot within the time given for development.

The Challenge Run

During the lab session on the due date for this lab, each group robot will be given 10 minutes to construct a simple structure at its construction site. You will have 2 minutes to setup your system. Each robot will be run separately.

We will set up an environment with walls, obstacles, construction objects, and fiducials in advance and give you a corresponding map. You will be given two weeks to test in this environment prior to the final run, but remember that there will be unmapped obstacles and construction objects, and these may be moved arbitrarily at the beginning of the challenge run (i.e. the robots cannot know their locations at the start of the run).

The robot will start in the position identified in the map file. For extra credit, you may allow the judges can start the robot anywhere they want in the environment. The robot will have to pick a construction site (e.g. the start location, or at a place you have pre-specified, or by its own autonomous reasoning), identify as many objects as possible, bring them to construction site, and build as much of a structure as possible.

The objective of the challenge is to complete as much of the task as possible. Your solutions should be reliable and robust. You will be expected to have basic solutions for each of the challenge sub-tasks. After the start of the round, no humans are allowed to interact with your robot's operating programs, which must all be running on the robot's laptop computer.

Budget

To encourage creativity, teams may spend up to $50 to purchase additional components used in their designs. You must document these components with receipts and you will get reimbursed. No single part may cost more that $10. To maintain fairness, we will assign a value to found or donated parts you may wish to incorporate in your design, and the sum total of all extra parts must stay below the $50 maximum.

Detailed Rules

Robot Hardware and Operation

1. The robots must be constructed (only) with parts from the kit and with up to $50 of extra parts, as described above, unless modifications have been discussed with the staff.
2. Any robot that appears to be a safety hazard will be disqualified. Liquids and explosives are specifically prohibited.
3. No parts or substances may be deliberately dumped, deposited, or otherwise left in the robot's space.
4. Your will not be allowed to modify your robot hardware after 5/9/12 (you will be allowed to do small repairs in the case of breakage). In the unlikely case of catastrophic failure after this date, contact your instructors.

The Map and Landmarks

We will provide a physical environment including walls, obstacles, and a set of fixed-position visual landmarks (colored spheres). We will also provide a map giving all of this information in a global coordinate frame, and code which parses this map. The fiducial objects will be specified in terms of their (x, y, z) locations, their radii, and their colors. See the code for the exact details of the map file format.

Construction Objects

1. There are 2 shapes of blocks: 5cm$\times$5cm$\times$5cm and 5cm$\times$10cm$\times$5cm. The weight of each object is no more than 100 grams.
2. The objects will have known colors (see the course Wiki for details).
3. There will be a known number of mapped components and a known number of unmapped components (see the course Wiki for details).

Time

Each robot will have a total of 15 minutes to find and retrieve as many components as possible and to create a structure as large as possible at the construction site.

Finding and Fetching Components

Robots will signal that an object is "found" by beeping. A single beep can be used to indicate that any type of object has been found. Three beeps will indicate that an unmapped object has been found. All beeps must be generated when the robot is within 30cm of the object. The judges will determine whether beeps are meaningful, and random beeping will be penalized.
A component is considered "transported" if it resides within 60cm of the construction area when time expires.

Construction

The construction site can either be provided as input on the map or computed by your robot given constraints on the size and the orientation of the desired structure (which are again up to you to define). Your robot will have to signal the identification of the construction site by stopping within 30cm of the site and beeping 5 times at least once during the final run. The robot will have to bring the construction objects to the construction site. You should aim away from randomly placing the objects at the site and towards carefully placing them in a pile or a wall.

Nudging

If your robot gets off course, you can manually (and physically) put it back on course at most two times during the final run.

Part 1: Solution Design and Team organization

The course challenge is the most complex task your robot will face during this term. Your robot will have to scale up its performance (both the complexity and the duration). So far, your robots have performed in half-minute chunks. For the challenge your robot will have to be reliable for 10–20 minutes, an order of magnitude longer! This raises many issues at the level of algorithmic design, software implementation and integration, and solution reliability.

Many of the modules you will need have already been implemented as part of individual labs: e.g. wall following, motion planning, grasping, object transport, and visual servoing. For each of these modules you may use your solution, our solution, or you may build on your lab experience and implement something completely new.

The first step in thinking about how to put it all together is deciding your "philosophical" approach to control. Do you want to choose a reactive control structure, a deliberative control structure, or a hybrid? What are the trade-offs? A reactive approach to control may rely on wall following and complex sensing to locate objects. A deliberative approach relies on a map to get access to a model of the world and therefore may rely on simpler sensing. Think about your robot's performance and the trade-offs in complexity between the various approaches and then choose an approach. Is your robot's hardware sufficient for the approach you have chosen? If not, what modifications will be needed? You may modify the robot's hardware, but keep in mind the deadline for freezing the hardware on your robot and the budget limit, both described above.

One important consideration to keep in mind is that your robot will have to travel to many specified points. How do you choose to implement precision navigation and error detection and recovery? One option is to rely on a reactive strategy that uses odometry augmented by wall following and the topology of the environment. Whenever the robot detects a topological feature such as a corner it could update its idea of where it is located. This strategy is simple and robust if all the goal points for the robot are reachable by wall following. However, your challenge map will have significant open spaces where there will be no walls to follow so you will need to develop a solution for robust navigation within open spaces as well. Another option is to use visual landmarks to ensure the robot has an accurate estimation of its location. An appendix below explains how you can solve the robust navigation problem using vision.

Before you begin coding, define a high-level algorithm for how you will solve the challenge. Include the details of how your solution will be expressed in terms of the basic robot capabilities provided by ROS, implemented in previous labs, or that are self-contained behavior modules for which you have a concrete implementation plan.

The next step is to define the APIs between all the software modules. After that, consider what sequence of module development, testing and integration makes sense. (Remember that a spiral approach will allow your robot to acquire a gradation of skills within each task while it fulfills the requirement of demonstrating each sub-task.) Then, match the sequence to a schedule. When developing the schedule be sure to include time for module development and testing, time for software integration, and time for project testing and evaluation. At this point, make appropriate programming assignments so that you can parallelize your efforts.

Part 2: Programming and Development

We have provided a GUI, `Challenge.ConstructionGUI`, for you to use to display information from your robot and to help you debug your code. We have provided code: `Challenge.GrandChallengeMap`, which can parse map files including `Challenge.Fiducial` markers and `Challenge.ConstructionObjects`. We have also provided two launch files to start the GUI and draw the map; you may want to adjust these launch files to use the `roscore` on the netbook/workstation.

At this point, also check the course Wiki for other errata, which may include instructions to update other files in your group repository to new versions we have now provided.

We have provided you with a map file (`challenge_2012.txt`). As the new environment is finalized, we may provide you with updated versions.

Part 3: Go For It!

We will approach this as a two-stage process. There will be a dry run on 5/7/12 which will give you an opportunity to do "dress-rehearsal" for your robot. This will also give you a chance to learn about the weaknesses of your solution so that you can improve your robot's performance for the final run.

Each group will be required to participate in a 20 minute review meeting during each Monday lab session from now until the due date for this lab. This meeting will be conducted in parallel with the other lab activities on that day.

A word of advice: don't just hack! Think carefully about your solutions and make sure they have some degree of generality.

Part 4: Wrap Up and Part II of Lab Report

To Hand In: Part II of your lab report on the Wiki should detail your solution to the challenge. Please justify your choices, and provide any testing data you collected during development. Include time spent on each part of the lab in person-hours and indicate what elements were done individually, in pairs, as triples (and by whom) or as a full group. This is due on 5/14/12.

Remember: Each student must also turn in an individually authored essay entitled "Building Brains for Bodies," about the lessons you learned by doing all the labs and the course challenge.

Appendix: Visual Navigation with Fiducials

This section describes the use of known fiducial objects (visual landmarks) to infer robot pose. You should also refer to your handouts and notes from the Localization and Mapping lectures.

Visual Navigation

This section guides you through the modification and extension of your VisualServo module (or equivalent), which you have developed in prior labs, to enable your robot to infer its approximate position and heading (i.e., absolute orientation) in the global coordinate frame of the maze.

Recall that, in prior labs, you implemented object range and bearing sensing using the camera. For visual navigation, extend your code so that it returns the *calibrated* range in meters and bearing in radians of an observed landmark.

Using two such landmark observations, your robot can determine its position and (absolute) heading in the global frame. When three or more landmark observations are available, your robot can use the additional constraints to combat noise, for cross-validation, or for both purposes.

Camera Calibration

Position the camera as close as possible to $(x, y) = (0, 0)$ in robot frame, facing straight ahead (along the $+x$ direction). Think about how you can use external markings, e.g., ruled lines on the floor, to help you position and orient the camera precisely. Observe a gradated meter stick or measuring tape. Make sure that the tape is aligned with the middle horizontal raster of your image. Make sure that the 1m mark is at the center pixel, and that the number of pixels subtended by the left segment is the same as that by the right segment. This ensures that your calibration object lies in a plane very nearly parallel to the image plane (do you see why?). Now measure the perpendicular standoff distance from the camera to your calibration object, and work out the relationship between pixel x-coordinate and the body-relative bearing angle in radians to the corresponding point on the calibration object.

This gives you the calibrated bearing to an object. To get the calibrated range (distance), pick two points symmetrically spaced about the 1m mark on the calibration object and work out the relationship between their actual physical separation, the observed separation in x-coordinate of the corresponding pixels, and the actual distance to the 1m mark on the calibration object.

Solving for Robot Pose

Combining your existing blob tracker, and the additional bearing calibration information from the previous section, yields a metric range-and-bearing sensor for any detected fiducial, provided you know the actual position and size of the fiducial in the scene. Refer to your course notes to compute the possible location(s) of the robot given two range-and-bearing observations. Remember to take into account the height (z coordinate) of the fiducial, and the height of your camera's focal point above the ground plane ($z = 0$).