Introduction

In this final project, you will develop software packages for intelligent human-centered robots. This project is an opportunity for you to obtain an understanding of the concepts you have learned in the class or to explore new robot capabilities that are not fully covered in the lecture but provided by the Robot Operating System (ROS). Students are required to implement the software package in ROS that can work on a real physical robot. This project is typically done as a team effort (students who want to do an individual project must discuss the opportunity with the instructor), and each team is required to implement one and only one topic that can be freely selected by each team (although the instructor will provide suggestions on the difficulties of the selected topic). You will notice that Project 1 and Project 2 can be helpful for you to implement these topics.

Robotic Platform

The projects must be implemented in ROS that work on the TurtleBot2 robots provided in the class. A number of 14 TurtleBot2 robots will be made available for students in this course. TurtleBot2 is an excellent robotic platform that is still one of the most widely used robots in robotics education (and research). The following are some information about the robot:

  Or [http://wiki.ros.org/Robots/TurtleBot](http://wiki.ros.org/Robots/TurtleBot)
- A very interesting tutorial of learning TurtleBot and ROS: [http://learn.turtlebot.com/](http://learn.turtlebot.com/)

Turning in your project

The Final Project has multiple due dates. **Each team must submit a single, integrated report by the team leader via Canvas before the deadline.** The contents required to be included in each submission are listed as follows:
Deliverable 1 (Proposal Report, NOT necessary to use LaTex):
(1) Select one and only one topic for your team final project;
(2) Specify a team leader (or point of contact) who is responsible for contact;
(3) Provide a description of the objectives, methods, evaluation of success, etc.
(4) Clearly state whether Turtlebots are required and how many.
(5) Include a timeline and workload breakdown to show your plan of finishing the project.

Deliverable 2 (Progress Report, NOT necessary to use LaTex):
(1) Describe the progress of your final project (working hardware, design, method, etc.);
(2) Include a workload report (agreed upon by all team members) that states what each team member did on this project, along with a percentage breakdown (totaling 100%).

Deliverable 3 (Code and Demo):
(1) Submit a video demo of your final project (required);
(2) Submit the ROS packages/code you develop as a team
(3) Include a detailed instruction of how to compile, use and test the code.
(4) In general, your code must compile, be well-documented, and run without crashing.

Note: If the file size of your code and demo is too big (usually the case), please upload it to Dropbox or Google Drive and then submit the download link to the Canvas.

Deliverable 4 (Final Report and Poster):
Please follow the guidelines to prepare final report and poster in the next sections.

In-class presentation (live demo is highly welcome) will be scheduled on April 24 and May 3 (across 4 classes). During that period, you will have the chance to show off your awesome implementation as well as share the technical details of how you make it happen. Graduate teams will present first, then students working on individual projects, and lastly undergraduate teams. Each presentation will be 10-12 minutes (We have 16 projects this semester!) plus an additional 5 minutes for Q/A. Peer review will be used.

Final Report Guidelines

As part of your completed final project, each team must prepare a paper describing your project (Minimum 3 pages for undergraduate teams, and minimum 5 pages for graduate teams, but not exceeding 6 pages). Your paper must be formatted using LaTex and the standard 2-column IEEE conference paper format. A Latex template was provided in Project 2: inside.mines.edu/~hzhang/Courses/CSCI473-573/Projects/Project-2/LatexTemplate.zip

The final report you turn in must be in pdf format and to Canvas. Your paper must at least include the following sections:
- Abstract: An abstract of 200 to 300 words summarizing your final project and findings.
- Introduction: An introduction describing your final project, including the task and formulation of research problems, a brief description of your methods, and structure of your final paper.
• **Approach**: A detailed description of your approaches to solve the problem, with enough information to understand and enable someone to recreate your system.

• **Experiments**: Experimental results plus an explanation and discussion of the results, such as in what situations your system can obtain the best performance, when it fails, the efficiency of your program, etc.

• **Conclusion**: A conclusion and future work section that summaries your final project, point out future work you believe would improve your implementation, and any other insightful observations you’d like to make.

• **Appendix**: If it’s a team work, a *workload report* in Appendix (agreed upon by all team members, which states what each team member did on this project in detail, along with a percentage breakdown (totaling 100%).

**Poster Guidelines**

Posters can be portrait format (vertically tall) or landscape format with a size by the A0 standard (33.1x46.8 in or 841x1189 mm).

**Grading**

Your grade (totally 10 points) will be based on:

- 0.5 points: The proposal report
- 0.5 points: The progress report
- 4 points: Code and demo (3 points for code, 1 point for video demo)
- 1.5 points: In-class presentation and demonstration
- 1.5 point: Poster
- 2 points: Final report

The following are some clarifications of the grading procedures:

**How grading will be done for teams**: The grading for this assignment, when working in teams, will be as follows. All members of the team will receive the same grade as a starting point, based on the submission. Then, the Workload Report information (submitted with Progress Report and Final Report) either results in the scores staying the same, or it can result in one student’s score moving down by some amount (maximum of 1 point).

**Graduate vs undergraduate teams**: Graduate teams will be graded more strictly on quality of the research and paper presentation. The instructor expects a more sophisticated implementation and a thorough analysis of the results for graduate students.
Potential Topics of Final Projects

Students are strongly encouraged to consider their own topics of interest for the final project. The following is a couple of ideas (some are learned in the class, others are new but enabled by ROS):

A. Robot Playing of “Simon Says” Games
Simon Says is a child's game for multiple players where 1 player takes the role of "Simon" and issues instructions (usually physical actions such as "jump in the air" or "stick out your tongue") to the other players, which should only be followed if prefaced with the phrase "Simon says", for example, "Simon says, jump in the air". Players are eliminated from the game by either following instructions that are not immediately preceded by the phrase, or by failing to follow an instruction which does include the phrase "Simon says". More information: https://en.wikipedia.org/wiki/Simon_Says.

Notes:
(1) The robot will take the role of “Simon”, who issues instructions and recognizes activities of two human players. The robot Simon can issue the instructions using voice. You can directly use ROS packages to generate synthetic voice for the robot.
(2) The robot must recognize the activities of two players in real-time, and take actions after the recognition to indicate whether a human is failed (not able to follow the instructions), through for example, moving forward and touching the failed human, or simple facing to that individual and using voice for indication.

B. 3D Simultaneous Localization and Mapping (SLAM)
The students may explore and improve an open-source SLAM method, and use a robot equipped with a structured-light sensor (Kinect or Asus Xtion Pro Live) to construct a 3D map of a room and a loop in a campus building (e.g., Brown Hall or library). The package could be RTAB-Map (Real-Time Appearance-Based Mapping), which is a RGB-D Graph SALM approach based on a global Bayesian loop closure detector. At the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) in 2014, a team using the RTAB-Map for SLAM won the Kinect navigation contest.
The RTAB-Map is available: http://introlab.github.io/rtabmap/.

Notes:
(1) The robot can be manually controlled or semi-autonomous (can be more challenging).
(2) A 3D map of (i) a room environment and (ii) an indoor loop of a campus building need to be generated.

C. Other Example Topics
C.1 Package Delivery: Create an application that will allow the Turtlebot to deliver certain packages to certain people and places. The basic version of this project might involve building a map of the world

---

1 Many example topics are collected from the Internet, including:
https://courses.cs.washington.edu/courses/cse571/12wi/project-ideas.html
http://web.engr.oregonstate.edu/~smartw/me539/labs/final/final.html
(e.g., using Gmapping in ROS) and making sure the robot is localized in it. People and places are associated with particular places on this map. The robot is then given a stack of packages to deliver, and must plan a path (using the build-in ROS path planners) around the world to deliver the packages to the correct people and places. You can probably assume that people will be cooperative, and the Turtlebots do not have to directly manipulate the packages themselves. More advanced versions of this project might involve optimal planning for the entire route, reading the name labels on the packages (rather than having them available in a file), and the ability to add packages during a delivery round.

C.2 Sentry Robots: The robot patrols a specified area, looking for intruders. If an intruder is spotted, it takes appropriate action. More advanced versions of this project might involve a GUI to specify the patrol area, guarantees of coverage (this is similar to versions of the art gallery problem in theoretical computer science), and identifying subtle changes in the environment (and not just people walking through it).

C.3 Human or Object Tracking and Following: Track a rolling or bouncing ball (or another object) or humans. The robot should have the capability that when the target is lost (e.g., when turning a corner), after the object or human comes back into the scene, the robot should continue tracking the object or human. The robot should also be able to adjust its speed/distance according to the speed of the target.

C.4 Robot Drawing: Develop an interaction software that when a user draws a shape on the computer screen using mouse, the robot can physically reproduce the same drawing on the ground.

C.5 Robot-assisted Rescue by Wall Following: An efficient way to perform indoor rescue (e.g., during a firefighting) is to follow walls. This topic will focus on developing an algorithm (based on heuristics, reinforcement learning, or other methods) that allows a robot to follow the wall (for a long distance).