Melbourne School of Engineering
ELEN90066 Embedded System Design

Kobuki Obstacle Course Project - REMOTE

Overview

The first SIX workshops (Workshops 1–6) for the subject comprised a structured sequence of exercises designed to give you experience and develop competency using the tools required to simulate and control a real-life embedded system, a Kobuki robot platform.

The next THREE workshop sessions (Workshops 7–9) are devoted to optimising the design of the Kobuki in order to navigate an obstacle course in the final workshop in Week 12 of the semester. There are no pre-lab or in-class tasks to be performed so it is up to your team to determine how to best spend your workshop time.

You will be writing a team report that documents your design and testing procedure so it is important that you document your design, implementation and testing procedure as you go.

Your Kobuki obstacle course algorithm MUST be coded as a Finite State Machine (FSM) using Stateflow and embedded in a Simulink model. It must be compatible with the standard ESD VM provided on the LMS (i.e. must not rely on any modifications to the models or software contained in the VM).

Obstacle Course Design Requirements

The physical Kobuki has several buttons, one of which can be used to trigger the execution of a program used to control it. You will need to provide a switch at the top level of your Simulink model that represents this button and gives the user control over executing the obstacle avoidance algorithm on the virtual Kobuki (see the next section for details). This switch will be referred to as the ‘Go’ switch in this section.

The following are the design requirements for the obstacle course navigation algorithm of the robot.

1. **Startup**: When the robot is loaded into the Gazebo world, it shall not move until the 'Go' switch is activated by switching it to 1 in Simulink.

2. **Run**: The robot shall begin movement the first time the 'Go' switch is switched to 1 and continue running until paused or stopped.

   (a) **Ground Orientation**: The *ground orientation* of your robot is the direction the front of the robot is pointing when it first runs. The ground orientation does not change after subsequent pausing/resuming; only after a power cycle, reprogram or restart of the robot or its FSM.

   (b) **Drive**: Your robot shall maintain ground orientation and drive forward while on level ground and clear of obstacles.

   (c) **Obstacle Avoidance**: Your robot shall avoid obstacles.

      i. **Cliff Avoidance**: Your robot shall not fall off of cliffs or edges. losing contact with the ground. If a wheel loses contact with the ground, your robot shall attempt to recover and move around the incident hazard.

      ii. **Object Avoidance**: Your robot shall avoid objects in its path. It is acceptable for your robot to touch objects as long as it immediately changes course in an attempt to avoid.

      iii. **Avoidance Robustness**: Obstacle avoidance must always be satisfied, even if multiple obstacles are encountered simultaneously or in short succession.

      iv. **Reorientaton**: After avoiding an obstacle, your robot shall eventually reorient to ground orientation.

   (d) **Hill Climb**: Your robot shall have a Hill Climb ability.

      i. **Hill Climb**: When an incline is encountered, your robot shall drive uphill towards the top of the incline. It must not go over any edge of the incline.

      ii. **Hill Plateau**: When the top of an incline is reached your robot shall remain driving forwards along the plateau (top flat section) of the hill.

      iii. **Hill Descend**: When an downward slope is encountered, your robot shall drive downhill towards the bottom of the slope. It must not go over any edge of the slope.

      iv. **Ground Stop**: After climbing and descending, when the bottom of the slope is reached, your robot shall stop and terminate execution within a set distance of the bottom (40cm) as its final goal has been achieved.

3. **Pause/Resume**: The 'Go' switch in the Simulink model shall start/resume or pause movement of the robot. At any point the robot is moving, switching the 'Go' switch to 0 shall cause it to immediately and completely stop. Subsequently switching the 'Go' switch to 1 shall cause the robot to resume operation.

4. **Performance**: When moving, your robot must satisfy the following performance characteristics:
   (a) **Turnabout**: Your robot shall never rotate in place more than 180°.
   (b) **Chattering**: Your robot shall not move erratically or exhibit chattering.
   (c) **Abnormal Termination**: Your robot shall not abnormally terminate execution, with the exception of software or simulation failure.
   (d) **Obstacle Hugging**: Your robot shall not repeatedly encounter ("hug") an obstacle for the purpose of navigation.
   (e) **Timeliness**: Your robot shall achieve its goals in a timely manner.

Hint: Your robot does not need to follow a specific path or trajectory, or return to a specific path or trajectory following obstacle avoidance – it need only eventually return to and maintain its original ground orientation.

Simulink Obstacle Course Navigation Model Design Requirements

Your obstacle course navigation algorithm must adhere to the following rules

- It must be implemented as a Stateflow chart embedded in a Simulink model.
- It must be saved as a single `.slx` Simulink model file compatible with MATLAB R2020b and submitted through LMS **before 9am** on the day of your workshop in Week 12. You must use the file naming convention `WS10_XX_YY.slx` where `XX` represents your class number and `YY` represents your group number.
- It must have a Stop/Go switch at the top level of the Simulink model exactly as shown in Figure 2 to facilitate control by the assessor

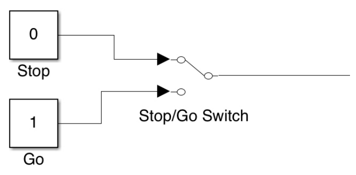

Figure 1: Stop/Go switch in Simulink to control the robot

- It must only rely on the provided standard ESD VM (i.e. no modifications to the VM)

- It can only make use of sensors (or derived sensor data) that are available to the real-world Kobuki and available on the ROS topics:
  - `/mobile_base/events/bumper`
  - `/mobile_base/events/cliff`
  - `/mobile_base/sensors/imu_data`
  - `/odom`

  You may however process the data from these sensors in Simulink in any way you wish (e.g. filtering) using standard Simulink operations / blocks.
- It can only control the robot through the following ROS topics:
  - `/mobile_base/commands/moto_power`
  - `/mobile_base/commands/reset_odometry`
  - `/mobile_base/commands/velocity`
- It must NOT use any MATLAB or external toolboxes, other than the ROS toolbox
- It must NOT have any hard-coded IP addresses in the model (these should be set in the MATLAB Command Window via MATLAB environment variables when testing).

Workshop 10 (Week 12) - Robot Obstacle Course

In your final scheduled workshop for the subject in Week 12, your team will be running the virtual Kobuki robot on an obstacle course in order to determine if your navigation and hill climbing algorithm is successful. The following subsections will provide details on the obstacle course set up and logistics for the final workshop.

The course

The course will consist of an elevated area approximately 5m $\times$ 3m that will contain the following:

- Barriers (walls) that mark the edges of the course. These may be bumped into by the Kobuki but not "hugged";
- Several shaped obstacles (rectangular, triangular, cylindrical) of different sizes that will be in random positions on the course;
- Pits of several shapes and sizes that the robot must not fall into;

- A hill section that comprises of a downward sloping ramp. The Kobuki is not to drive off the side of the hill.

An example course layout is shown in Figure 2. Note that this course does NOT correspond

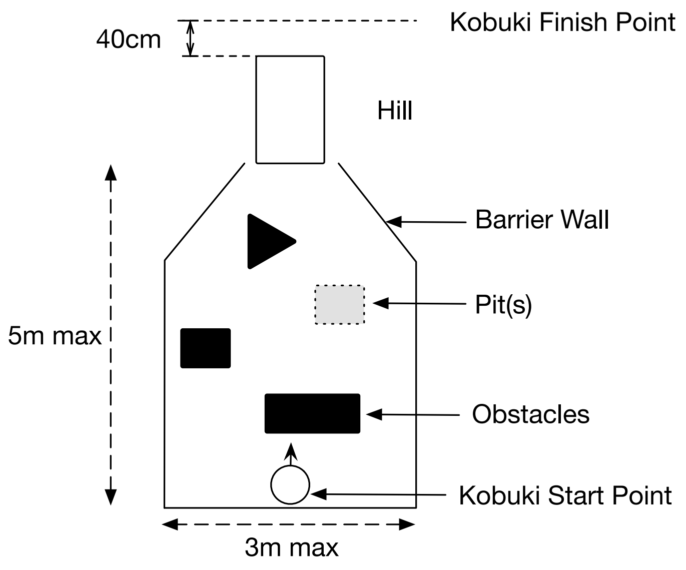

Figure 2: Obstacle course layout (note that items are not to scale)

to the precise layout that your robot will be tested on – you will not know the locations of the objects on the course before you run your robot on it. The only information known about the course is that it will narrow towards the hill, which will be at the end of the course as shown in Figure 2. Your robot MUST stop **within 40cm** of reaching the bottom of the far side of the hill.

As you will NOT know the precise configuration of the obstacle course for the final workshop your team must make sure that you have simulated and tested the robot enough so that you are confident it can handle any configuration of the course. You are responsible for designing your own test worlds in Gazebo.

Pre-workshop

Make sure that your team has submitted a single `.slx` Simulink model (R2020b) file through LMS **before 9am** on the day of your workshop. You must use the file naming convention `WS10_xx_Gyy.slx` where `xx` represents your class number and `yy` represents your group number (from LMS).

Workshop Logistics

Once the class begins, each team's algorithm will be tested with the following procedure:

- A team will be selected at random;
- A Kobuki robot will be spawned into a world representing the test obstacle course by the assessor and screen shared to everyone in the class. The starting direction is the **ground orientation**;
- The team will have their Simulink model file loaded by the assessor;
- The assessor will toggle the "Go" switch in Simulink and start a timer;
- The attempt is considered **complete** if any of the following occur:
    1. the robot completes the course by successfully avoiding all obstacles, ascending and descending the hill and then stopping within 40cm of the bottom of the downwards sloping section;
    2. the robot breaks any of the rules in the 'Obstacle Course Design Requirements' section;
    3. the elapsed time is greater than 180 seconds;
    4. the robot is interfered with in any way.

Assessment (20%)

The assessment for the Kobuki Project is worth 20% of the final mark for the subject. Of this, the report itself is worth 18% and the project outcome worth 2%.

Project Outcome (2%)

Your team will be graded on how successful the robot was in achieving the objective of navigating the obstacle course. The marking rubric for this component is integrated into the marking rubric for the report on LMS.

Project Final Report (18%)

The report must contain (but is not limited to):

- a complete FSM diagram (including pause states) of the robot algorithm using the notation covered in the lectures. If you use an Extended state machine, or state refinements, you must show all states, variables and transitions
- a description of the Kobuki sensor data available in ROS and how it was used in your algorithm (including any processing done)

- your design procedure (this could include work from earlier workshops)
- your testing procedure (this could include simulation data or screenshots)
- your validation procedure (this could include reliability or reachability analyses)
- the outcome of the run in the final workshop
- a discussion section (how you applied knowledge from the lectures, what you would improve in the future)

The report is to be no more than **20 pages in length**. A marking rubric for the report will be available on LMS.

Submission Details

Requirements for submitting the report :

- Submission is via the LMS.
- Submissions must be in PDF format.
- ONE submission per team.
- The report is to be no more than **20 pages in length**, not including any appendices.
- Late submissions will be penalised at the rate of 10% per day.
- **Submission date :** 11:59pm Monday November 1, 2021.