Carnegie Mellon University Africa

18-799 Introduction to Cognitive Robotics

Assignment 3
Robot Manipulation

Deadline: 18:00 Tuesday 16th March 2021

Problem Definition
Write a robot control program for the LynxMotion AL5D robot arm simulator to pick up three small coloured bricks and stack them on top of each other.

The (x, y, z) position of each brick and its orientation ϕ, where $0 \leq \phi < 180°$, is specified in an input file `assignment3Input.txt` located in the data subdirectory in the assignment3 package. You can assume that the x axis of the brick is aligned with its major axis; see Fig. 1. You can also assume that the dimensions of the brick in the x, y, and z directions are 31.8 mm, 15.8 mm, 11.4 mm respectively.

The (x, y, z) position and orientation ϕ of the first brick in the stack is specified on the fourth line of the input file.

All poses – T5, gripper, and bricks – should be specified in a Cartesian frame of reference using frames (i.e. homogenous transformations).

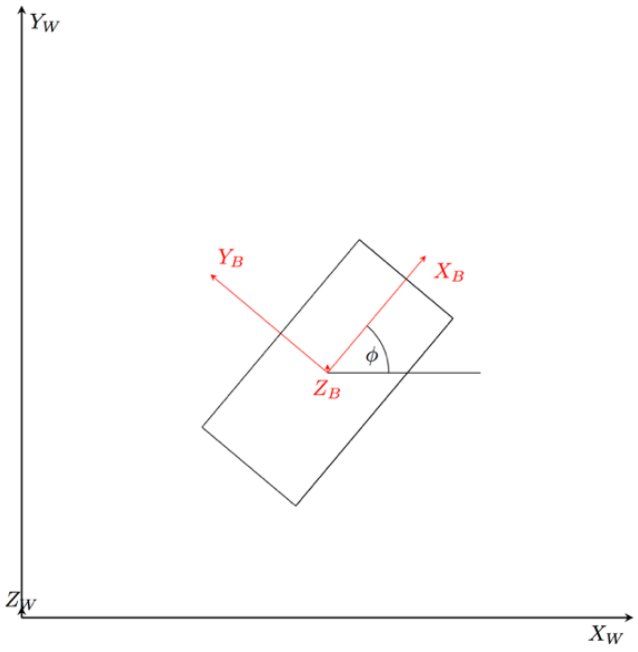

Figure 1. Pose of a brick.

Input
The first three lines each contain four numbers corresponding to the *x, y,* and *z* coordinates of a brick and its orientation ϕ. The fourth line contains the *x, y,* and *z* coordinates of the stack location and its orientation ϕ.

Sample Input

```
-40 150 0 90
 0  150 0 90
 40 150 0 90
 0  220 0 0
```

Instructions

Update the simulator

Before starting this assignment, you must get the current version of the Lynxmotion AL4D robot simulator. To get this, do the following.

```
$ roscd lynxmotion_al5d_description
$ git remote set-url origin https://github.com/cognitive-robotics-course/lynxmotion_al5d_description.git
$ git  pull origin master
$ roscd; cd ..
$ catkin_make
```

Build the Sample Program

To get you started, a pick-and-place program is provided in a ROS package `module4`.

This program reads **two** sets of four numbers. The first set corresponds to the *x, y,* and *z* coordinates and orientation ϕ of the brick to be picked up. The second set specifies where it should be placed. To get the program, do the following.

```
$ roscd; cd ../src
$ git clone https://github.com/cognitive-robotics-course/coro_examples.git
$ cd ..
$ catkin_make
```

Run the sample program

Run the program, as follows.

Open three terminals.

In the first terminal, run the ROS master:

```
$ roscore
```

In the second terminal, launch the simulator:

```
$ roslaunch lynxmotion_al5d_description al5d_gazebo_control.launch
```

In the third terminal, spawn a brick. Use the coordinates and orientation given in the input file `~/workspace/ros/src/coro_examples/module4/data/pickAndPlaceInput.txt`

```
$ rosrun lynxmotion_al5d_description spawn_brick -n brick1 -c red -x -0.04
-y 0.15 -z 0 -Y 0.785
```

Run the pick-and-place program:

```
$ rosrun module4 pickAndPlace
```

Verify that it works by inspecting the behaviour of the robot in the simulator.

When the pick-and-place action is complete, kill the brick:

```
$ rosrun lynxmotion_al5d_description kill_brick brick1
```

While you at it, try running the example robot programming application.

```
$ rosrun module4 robotProgramming
```

Close the Gazebo simulator when you're finished.

Create a new package

First, create a package `assignment3` in the ROS workspace:

```
$ cd ~/workspace/ros/src
$ catkin_create_pkg assignment3 roscpp
```

Copy `~/workspace/ros/src/coro_examples/module4/CMakeLists.txt`
to `~/workspace/ros/src/assignment3/CMakeLists.txt`

Copy `~/workspace/ros/src/coro_examples/module4/package.xml`
to `~/workspace/ros/src/assignment3/package.xml`

Create a subdirectory `~/workspace/ros/src/assignment3/data`

Copy `/workspace/ros/src/coro_examples/module4/data/pickAndPlaceInput.txt`
to `~/workspace/ros/src/assignment3/data/assignment3Input.txt`

Copy `~/workspace/ros/src/coro_examples/module4/data/robot_3_config.txt`
to `~/workspace/ros/src/assignment3/data/robot_3_config.txt`

Copy `~/workspace/ros/src/coro_examples/module4/include/module4/pickAndPlace.h`
to `~/workspace/ros/src/assignment3/include/assignment3/studentid.h`
(Be careful to replace `studentid` with your own ID everywhere in the following.)

Edit the `studentid.h` as follows.

> Replace `#define ROS_PACKAGE_NAME "module4"`
> with `#define ROS_PACKAGE_NAME "assignment3"`

Copy `~/workspace/ros/src/coro_examples/module4/src/pickAndPlaceApplication.cpp` to `~/workspace/ros/src/assignment3/src/studentidApplication.cpp`

Edit the `studentidApplication.cpp` as follows.

Replace `#include <module4/pickAndPlace.h>`
with `#include <assignment3/studentid.h>`

Replace `ros::init(argc, argv, "pickAndPlace");`
with `ros::init(argc, argv, "studentid");`

Replace `strcat(filename, "pickAndPlaceInput.txt");`
with `strcat(filename, "assignment3Input.txt");`

Replace `printf("Error can't open input pickAndPlaceInput.txt\n")`
with `printf("Error can't open input assignment3Input.txt\n")`

Copy `~/workspace/ros/src/coro_examples/module4/src/pickAndPlaceImplementation.cpp` to `~/workspace/ros/src/assignment3/src/studentidImplementation.cpp`

Edit the `studentidImplementation.cpp` as follows.

Replace `#include <module4/pickAndPlace.h>`
with `#include <assignment3/studentid.h>`

Replace `ros::init(argc, argv, "pickAndPlace");`
with `ros::init(argc, argv, "studentid");`

Edit the `CmakeLists.txt` file as follows.

Change `project(module4)` to `project(assignment3)`.

Replace `pickAndPlace` with `studentid`.

Remove the part that refers to `robotProgramming` application.

Edit the `package.xml` file as follows.

Change `<name>module4</name>` to `<name>assignment3</name>`.

Change `<description>Cognitive robotics module4 package</description>`
to `<description>Cognitive robotics assignment3 package</description>`

Build this package as follows.

```
$ roscd
$ cd ..
$ catkin_make
```

Run the Copied Program

If necessary, open three terminals.

In the first terminal, run the ROS master (if necessary):

```
$roscore
```

In the second terminal, launch the simulator (if necessary):

```
$roslaunch lynxmotion_al5d_description al5d_gazebo_control.launch
```

In the third terminal, spawn a brick. Use the coordinates and orientation given in the input file:

```
$ rosrun lynxmotion_al5d_description spawn_brick -n brick1 -c red -x -0.04
-y 0.15 -z 0 -Y 0.785
```

In the third terminal, run the robot program:

```
$ rosrun assignment3 <your_own_studentid>
```

Verify that it works by inspecting the behaviour of the robot in the simulator.

In the third terminal, when the pick-and-place action is complete, kill the brick:

```
$ rosrun lynxmotion_al5d_description kill_brick brick1
```

Adapt the Copied Program

Now, adapt this code to satisfy the requirements of this assignment, i.e. pick three bricks in turn, placing them one on top of the other at the specified destination. You might consider encapsulating the pick-and-place functionality of the example application in a function, in which case put the target function to do this in the implementation file, the declaration in the interface file, and the function calls in the application file.

Submit your Assignment

Submit the three source code files named with your student ID, i.e.. `mystudentidApplication.cpp`, `mystudentidImplementation.cpp`, `mystudentid.h`, and your `assignment3Input.txt` file in a zip file named with your student ID by the deadline shown above.

Do not forget to include your name in the internal documentation.

Marking Scheme

Marks will be awarded based on blind tests using data similar to those in the sample input above. Marks will be awarded for the effectiveness of the robot path and the time taken to achieve the goal pose, given the algorithm that is being used to effect the locomotion.

Equal marks will be awarded for each test case.