Intelligent Robotics Exercise 1: ‘Let’s get started’ and ‘Going for a walk’

Abstract

The Pioneer 3DX (P3DX) from Mobile Robots is an advanced research robot that has an on-board PC, a range of sensors (including an optional laser range finder), and communicates via Wireless Ethernet (WiFi) [1]. It is the world's most popular research mobile robot [2]. This report, submitted in requirements of the Intelligent Robotics module of the University of Birmingham, presents the research and subsequent solutions implemented by our team to achieve fully autonomous exploration of the robotics lab by a single Pioneer robot. This exercise highlights the necessity for trial-and-error, research driven approaches to Robot programming.

Introduction

As a team, (comprised of Matthew Flint, Laurence Stokes, Robert Minford and Alex Bor), we have been tasked with setting up a pioneer robot from being able to respond to human control (via a joystick) to fully autonomous exploration of the robotics lab in the University of Birmingham Computer Science department. Programming the robot was done using Python 2.7 and facilitated through the Robot Operating System (ROS) framework, an open-source, flexible framework for writing robot software [3].

This report presents the research approached and subsequent solutions implemented by our team to the end of achieving fully autonomous exploration.

Tasks

This section of the reports details the tasks our team solved. Readers primarily interested in the results and discussion may wish to skip this section.

Prior to engaging in the exercises our team covered some basic administrative duties. Principally, we discussed what times were suitable to meet and work beyond the established lab times, if and where necessary, in order to complete the exercise (as well as

future exercises which will likely demand more time). We also set up a Facebook group to have an expedient means of contact outside of the assigned lab hours.

The first section of the exercise, 'Getting started', simply required following the setup intstructions as per the module webpage (available: http://www.cs.bham.ac.uk/internal/courses/int-robot/2014/exercises/0getstarted.php) and we demonstrably achieved joystick control of the Robot.

The second part of the 'Getting started' exercise involved setting up the ROS packages *Stage* and *Rviz*. Stage creates a simulated robot in a predefined world and Rviz provides a 'robot's eye view' of the world by displaying the data being published by the laser.

The final part of the 'Getting started' exercise was to write a small python script to drive the Robot forward until it reached an obstacle, whereupon the Robot would stop. Our script is documented below:

```
#! /usr/bin/env python
# -*- coding: utf-8 -*- #

#---- imports ----#
import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan

"""
Intelligent Robotics 2014/15
Exercise 1, Part 1 (c)
Alex Bor, Laurence Stokes, Robert Minford, Matthew Flint
"""

#------Main Method definitions------#

def callback( sensor_data ):
	'Callback method that executes on retrieving sensor data from the
Pioneer'
	base_data = Twist()
	#print len(list(sensor_data.ranges))

	#boolean switch to stop motion
	stop = False

	#iterate over sensor values
	for value in list(sensor_data.ranges):
		if value < 0.5:
			stop = True
```

```
        if stop:
                base_data.linear.x = 0;
        else:
                base_data.linear.x = 0.3

        #publish the data back to the Robot to execute
        pub.publish(base_data)


#-------------- Main Method-------------#
def main():
    try:
                rospy.init_node('simple_mover_node')
                rospy.Subscriber('base_scan',LaserScan, callback)

                #make pub a global variable
                global pub
                pub = rospy.Publisher('cmd_vel',Twist)

                rospy.spin()
    except KeyboardInterrupt:
        raise KeyboardInterruptError()
    except SystemExit:
        sys.exit(1)


if __name__ == "__main__":
    main()
```

Discussion regarding this script is presented in the results and discussion section.

The final part of the Exercise, titled 'Let's go for a walk' was the most technically challenging. Building on the 'Getting started' exercise, the task for this exercise was to make the Pioneer robot turn, rather than stop, on encountering an obstacle, enabling continuous unhindered movement.

```
#! /usr/bin/env python
# -*- coding: utf-8 -*- #

#---- imports ----#
import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan

"""
Intelligent Robotics 2014/15
```

```
        Exercise 1, Part 2, 'Let's go for a walk'
        Alex Bor, Laurence Stokes, Robert Minford, Matthew Flint
        """

        #------Main Method definitions------#

        def median(lst):
              'Method to find the arithmetic median of a list passed in as the explicit
parameter'
              sortedList = sorted(lst)
              lstLen = len(lst)

              index = (lstLen-1) // 2

              if (lstLen % 2):
                    return sortedList[index]
              else:
                    return (sortedList[index] + sortedList[index + 1])/2.0


        def callback( sensor_data ):
              'Callback method that executes on retrieving sensor data from the
Pioneer'

              base_data = Twist()
              #print len(list(sensor_data.ranges))

              stop = False
              index = 0

              #Segments to break the peripheral environment into
              farRight = []
              right = []
              middle = []
              middleLeft = []
              middleRight = []
              left = []
              farLeft =[]

              #iterating over all the values in the sensor data
              for value in list(sensor_data.ranges):

                    if value < 0.3:
                          stop = True

                    if index < 100:
                          farRight.append(value)
                    elif index < 200:
```

```
                right.append(value)


        elif index < 233:

                middleLeft.append(value)
        elif index < 267:

                middle.append(value)
        elif index < 300:

                middleRight.append(value)

        elif index < 400:
                left.append(value)
        else:
                farLeft.append(value)


        index = index+1

#print median(list1), median(list2), median(list3)

#if(median(list1) < 1 and median(list2) < 1 and median(list3) < 1):
        #base_data.andular.z = 0.5

if median(middle) < 1:
        base_data.angular.z = -0.5
elif median(farRight) < 1:
        base_data.angular.z = 0.5

elif median(right) < 1:
        base_data.angular.z = 0.5
elif median(middleRight) < 1:
        base_data.angular.z = 0.5



elif median(middleLeft) <1 :
        base_data.angular.z = - 0.5

elif median(left) < 1:
        base_data.angular.z = -0.5
elif median(farLeft) < 1:
        base_data.angular.z = -0.5
if stop:
        base_data.linear.x = 0;
else:
```

```
                base_data.linear.x = 0.3


        #publish the data back to the Robot to execute
        pub.publish(base_data)


#-------------- Main Method-------------#
def main():
    try:
                rospy.init_node('simple_mover_node')
                rospy.Subscriber('base_scan',LaserScan, callback)

                #make pub a global variable
                global pub
                pub = rospy.Publisher('cmd_vel',Twist)

                rospy.spin()
    except KeyboardInterrupt:
        raise KeyboardInterruptError()
    except SystemExit:
        sys.exit(1)


if __name__ == "__main__":
    main()
```

Discussion regarding this script is presented in the results and discussion section.

Results and Discussion

This section details the the research and design choices we implemented in order to complete the exercises.

Due to the nature of the first two tasks of the 'Getting Started' exercise we feel it is unnecessary to discuss them in any detail; instead we will begin discussion on the third and final part of the 'Getting Started' exercise, which was to to write a small python script to drive the Robot forward until it reached an obstacle, whereupon it would stop.

The most important step in accomplishing this task was to first read and interpret the laser sensor data. To this end, we loaded the laser sensor data into a List . We then iterated

through the list for any values lower than a set predefined value, 0.5 - or 50cm[1] - and if a value less than this distance was present we instructed the Robot to assume that it was close to an obstacle and thus to stop movement. As a team, we decided not to deliberate on this task, so as to focus on the next task (essentially an extension of this one).

The final task of this exercise, 'Let's go for a walk' required making the Pioneer turn when encountering an obstacle, rather than stopping. Similarly to the previous task, stellar to accomplishing this task was successfully reading and interpreting the sensor data. Initially, we investigated how the Robot was scanning its peripheral environment and we quickly discovered it was doing a 180 degree laser scan from right to left.

Our first, crude solution to the task was to split the sensor data into 3 arrays of 60 degrees, essentially a 'right', 'middle' and 'left' environment. From there, we took the median value of the sensor data of each environment in order to gauge whether there was an obstacle in any of the right, middle or left vicinities ; if the median was below a threshold, initially 0.5m, we assumed there was an obstacle in that direction. Through trial and error we concluded that 1m was the best value to set the threshold. The median was chosen (over the mean) as the median is less victim to skewing from outlying data.

In order to improve the solution we further split the array into 5 parts, (farRight, right, middle, left, farLeft) so that each segment referred to a smaller, more precise, angle of the sensor data. This allowed us to more accurately determine in which direction the obstacle was detected and thus react more suitably. We then split the middle section further still, into three parts, (middleLeft, middle, middleRight) so that we could better see what obstacles were in the frontal proximity. We also altered the distance at which the robot would react to an obstacle for this section (so that it would react earlier), which helped when navigating out of 'dead ends'.

The overall success of the task notwithstanding, there were a few challenges that we faced. The sensor on the robot, by design, is simply too high to pick up low standing objects. Additionally, certain obstacles, such as the chairs in the computer science lab, were difficult to accurately detect as the legs were thin and narrow .

Finally, laser scanners cannot pick up glass and so in this instance other obstacle detection tools, such as a sonar sensor, would be required.

References

[1] We tried several values for the stopping distance but settled at 0.5. Smaller values led to the Robot not having enough distance to stop.

1. *'MobileRobots Pioneer P3DX',* Microsoft Developer Network, Available: http://msdn.microsoft.com/en-us/library/bb483031.aspx [Online] (Accessed 10/10/2014)

2. *'Pioneer P3-DX'*, Adept Mobile Robots, Available: http://mobilerobots.com/ResearchRobots/PioneerP3DX.aspx [Online] (Accessed 10/10/2014)

3. *'About ROS'*, ros.org, Available: http://www.ros.org/about-ros/ [Online] (Accessed 10/10/2014)