Actuation and Sensing, Familiarization with ROS
Evan House
Alaina Kafkes, Wenbo Yu
10/1/17

ABSTRACT:

In the field of robotics, a robot is a system capable of interacting with the environment on its own. Effectors, actuators and sensors make up some of the most important parts in building a robot. In this lab, we built a humanoid robot, and tested out the robot's demo behavior. Our lab group then collected data on the robot's demo behaviors. We then used the Robot Operating System to write a few behaviors that were to be carried out by the robot. Each of our behaviors depended on use of the robot's infrared (IR) sensor. In the process of the lab, we found that there is often uncertainty involved in our robotic system specifically, and in robotics more generally.

I. INTRODUCTION:

A robot is defined as an autonomous system that is capable of both sensing and acting upon the world around it. In order to be a robot, the object must be autonomous, meaning that it is able to make its own decisions and act accordingly.

Robots use sensors to understand the surrounding environment. There are two types of sensors that are commonly used in robots. They are the proprioceptive and exteroceptive sensors. Proprioceptive sensors are those which sense properties about the robots own internal state, such as the positioning of its leg; exteroceptive sensors are those which sense properties of the external world such as distance to an object. These sensors form what is called the robot's perceptual system. There also exists a distinction between passive and active sensors. Passive sensors use a detector take a measurement of some physical property in the external world; active sensors use an emitter to emit its own signal and make measurements according to the interaction of the environment with the signal. A common sensor type is an infrared (IR) sensor.

There are also different levels of sensor processing. In order for a robot to process what it senses about the environment, it requires a number of different parts: sensors, computation and connectors. Whenever a task is assigned to a robot, it is important to consider a solution that takes into account each of these aspects of processing. Different sensors are also categorized in different levels. Vision is a high level of sensory processing, speech and sonar are medium levels, and bump sensors and odometery are low levels of sensory processing.

In order to act upon the surrounding environment, robots use what are called effectors and actuators. Effectors are the mechanisms by which the robot has an impact upon its environment. Effectors include parts such as arms and legs. Actuators are the mechanisms that allow the effectors to act upon the environment. There are various types of actuators in robots such as electric motors and hydraulic cylinders. The most common type of actuator in robots are electric motors because they are both simple and cheap.

In robotics, Robot Operating System (ROS) is one method of writing robot behaviors. ROS is an open-source language which is commonly used for developing robotics. ROS is widely used in both academia and in the engineering industry. ROS code is contained in one node, and different nodes communicate with one another. There is a ROS master node which runs the rest of the nodes.

II. METHODS:

The first step of our assignment was to build our robot. Our lab group used the Battle Droid platform to build our robot. Our robot included eight different actuators. Two corresponding to each of the robot's effectors. The robot had four effectors in total: two arms and two legs. All of the actuators we used were electric motors. We also added one IR sensor to the robot. The IR sensor is both an active sensor, as it emits an IR signal, and a exteroceptive sensor, as it perceives something about the external world. A microcontroller was attached to the top of the robot,

and all of its sensors were attached to this. A bluetooth sensor was also attached to the microcontroller, and this is what allowed our code to interact with the robot's actuators and IR sensor.

After building the robot, we ran the pre-existing demo behavior and made three measurements related to the demo behavior. The three variables we measured were the time it took for the robot to assume the "battle pose", the distance the robot walked in 10 seconds, and the distance at which the IR sensor first detected an object in its path.

Once we finished running the demo behavior, we then "flashed" the robot, removing the demo behavior from the robot microcontroller memory, adding ROS to the memory of the microcontroller. Once this was complete, we set up our computer files to write and run code that controls the robot's actions. We then built our template ROS package, and built our ROS workspace. We used ROS in this assignment to make the robot perform a few simple behaviors, so it was important for us to understand the basics of writing code that affects the robot's behavior. Once our environment was set up, we tested whether our code was communication properly with the robot.

We tested this in two ways. The first was by testing if the sensor was working properly. There was a pre-existing function that would get the value measured by the robot's IR sensor. When nothing is measured by the sensor, the function returns a value of zero, and as an object comes into its range, the function returns increasing positive numbers. We next tested sending commands to the robot's actuators. We simply used another pre-existing function that took a motor ID and a position value as inputs; the output was a corresponding action by the specified actuator. The position value ranged from 0 to 1024. In this stage we tested out all of the different motors at various positions.

Our final task was to write a few simple behaviors for the robot to carry out using rospy, a Python client library for ROS (Robot Operating System). The first behavior was a function that would cause the robot to "disco" continuously, when it did *not* sense any object in its path. One of the difficulties in this case involved writing code that would ensure the behavior was continuous. This is because when we tell the same actuator to move to two different positions, the actuator would not reach either position, but just move back in forth quickly between the two position inputs. The robot would react to the second command before the first command had been completed. Because of this problem we had to come up with a simple solution to ensure that the robot's actuator would reach the first input position before moving to the second position and vice versa. We wrote two different pieces of code that each corresponded to boolean variables. In this way, we could write code that caused the robot to move in one direction, then once it reached its position, the second section of code would be called, and this would continue in succession so that the robot would be dancing continuously.

We also discovered that when setting the robot's actuator to a certain position value, the actuator never moved to the exact position. Upon recognizing this, we were able to implement another fix in our code to account for it. In order to be aware of the robot's position at all times, we found it helpful to write a simple function that would move the robot in to what we called the "ready position" before the disco function was called.

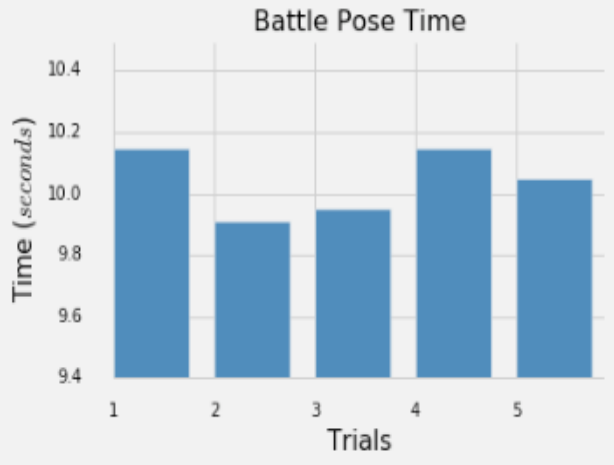
Our second function was written to move the robot's actuators in both its arms and legs so that the robot is in a position similar to a ballet pointe position. However, the robot would only go into this position when it detected that an object was in its path. Once the robot's IR sensor no longer detected an object in its path it would return to the rest position. This function was simpler to write as the robot was not moving continuously. We wrote two sections of code, one section corresponding to when the IR sensor detected an object, and another section corresponding to when there was nothing detected by the IR sensor.

III. RESULTS:

Once the robot was put together, we began by running the demo behavior. In the demo behavior, the robot would begin by assuming the battle pose. After assuming battle pose, the robot would begin by walking forward in a straight line. If any object is detected in front of the robot, the robot swings its right arm forward, then its left arm forward. If an object is no longer detected, the robot would begin walking again. In this lab, we collected data regarding three aspects of the robot demo behavior: the time it took to assume the battle pose, the distance the robot walked in 10 seconds, and the distance at which the robot's IR sensor detected an object. We repeated the measurement for each individual test a total of five times.

Battle Pose Time Data:

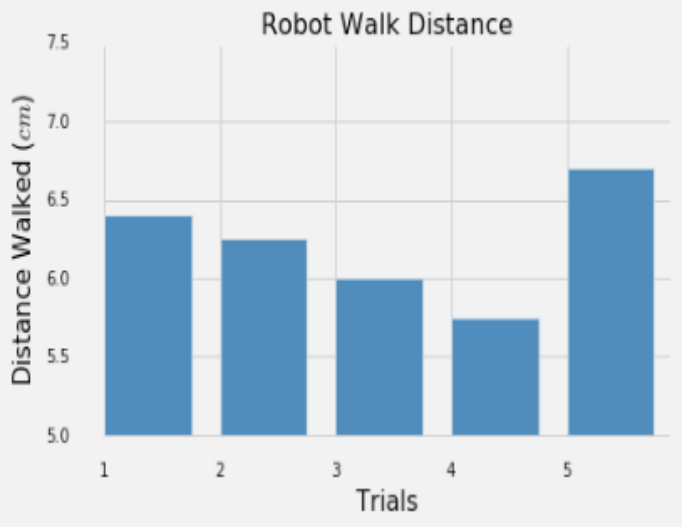
Trials	Battle Pose Time (seconds)		Battle Pose Time (seconds)
1	10.15	Mean	10.042
2	9.91	Standard Deviation	0.099
3	9.95		
4	10.15		
5	10.05		



We first measured the time it took to assume the battle pose. It is important to go through the initial robot behavior before the battle pose. Once the behavior is started, the robot would quickly swing its right arm forward, then its left arm. After this, the robot would simultaneously lift both its right and left arms slowly above its head. At the high point the robot would pause, before lowering its arms and beginning to walk forward. Our group defined the battle pose as the position where the robot is holding both of its arms above its head. One member of the lap group would make the measurement by simultaneously beginning the robot's demo behavior and a timer. Once the robot had finished entering the battle pose, we stopped the timer.

Robot Walk Distance Data:

Trials	Distance walked (cm)		Distance Walked (cm)
1	16.256	Mean	15.799
2	15.875	Standard Deviation	0.829
3	15.24		
4	14.605		
5	17.018		

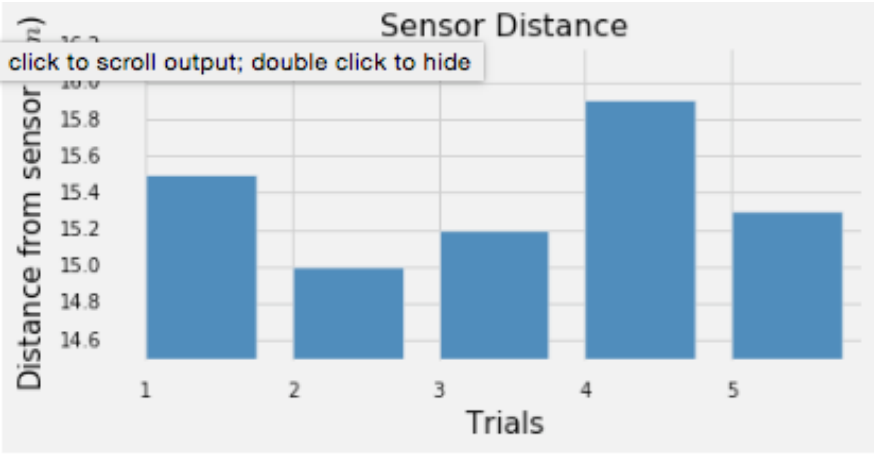


We then measured how far the robot walked in 10 seconds. We began by lining up the front of the robot's feet at the zero-centimeter position of a ruler. One member of the group would turn on the robot and start the demo behavior. Because there is a significant amount of time in the demo behavior before the robot begins walking, we decided to begin the timer immediately *after* the robot struck the battle pose. The member of the lab group recording would the time, would signal the other member to stop the robot once the timer had reached 10 seconds. We then measured the distance the robot walked, making our measurements from the robot's front most foot.

Sensor Distance Data:

Trials	Distance from Sensor (cm)		Distance from Sensor (cm)
1	15.5	Mean	15.38
2	15	Standard Deviation	0.306
3	15.2		
4	15.9		
5	15.3		

The last demo behavior we measured was the distance at which the robot's IR sensor detected an object. In order to improve the ease and accuracy of taking this measurement, our lab group decided to detach the IR sensor from the robot body. We were able to do this, because for this measurement, it was not important that the robot be walking; it was only important to note when the robot reacted to an object blocking its path. We fixed the IR sensor on the table at the zero-centimeter position of a ruler. We then set a small, flat cardboard box standing upright on the table, at the opposite end of the ruler. We slowly moved the cardboard box closer and closer to the ruler until the robot reacted to the presence of the surface. The robot reacted by stopping its previous walking motion and "punching" both arms forward in succession. If the object was continually in front of the robot, it would continue this motion; if an object was no longer detected, the robot would resume walking.



IV. DISCUSSION:

The hardware building phase was a relatively straightforward process. We did have to redo some of our work to fix some wiring that would make working with the robot later on more clean, however there were no significant problems as we successfully assembled the robot's effectors, actuators and IR sensor.

We were able to properly run the demo behavior and make the corresponding measurements for the assignment. There was room for improvement in the data collection process. In the case of the robot walking distance, some of the variables that certainly changed slightly from trial to trial were the time at which the timer was started, and more significantly, the time at which the robot was stopped. Because the robot was moving, it was difficult to stop the robot from walking at precisely the moment when the timer reached 10 seconds. In the case of the sensor distance, it is possible that the object being sensed was moved slightly past the IR sensor distance before it was stopped and the measurement was taken. This is one possible cause of increased variation in the results. In measuring the time that it took to assume the battle pose, the one uncertainty was stopping the timer right when the robot had reached the battle pose. All of these uncertainties definitely made the experimental testing phase more difficult than the hardware building phase.

The fact that we were able to write two simple behaviors to control the robot's actuators was a positive result. At the same time, there was room for improvement in our function that caused the robot to do a disco like dance. While we were able to write code that successfully made the robot disco, our implementation was more complicated than necessary. There was likely a simpler solution that could have been used in order to implement the same behavior.

Through the results of the demo behavior and the writing of two simple behaviors, we became more aware of some of the difficulties in the robotics field. For example, in the process of writing our first behavior, we found that the position of the robot's actuator was never in the exact position the function called for. The variation in the measurements of both sensor distance and walking distance also seem to suggest that the robot behavior is not always exact. For much larger functions than the ones we wrote, these small inaccuracies can amount to a big problem. Moving forward, it is clear that one of the challenges in future experimentation will be writing algorithms that can help account for these inaccuracies.

V. CONCLUSION:

In this lab assignment, we first put together the robot with its various effectors and actuators and its IR sensor. Once put together, we ran the demo behavior to see some of what the robot was capable of and to test its functionality. In the process, we took measurements on three different parts of the demo behavior. After running the demo behavior, we set up our python and ROS environment. We then wrote two functions, creating simple behaviors of our own. A common difficulty in both the demo behavior and our created behaviors was the uncertainty that was involved in the actuators and IR sensor.