# Applications for Robotics

Chris Brown
Duke University Class of 2013
Computer Science

Mentor: Chad Jenkins, Ph.D.
Robotics, Learning and Autonomy Lab
Department of Computer Science
Brown University
Providence, RI/Moss Beach, CA

# 1. Introduction

The overall goal of this internship was to discover ways to make it easier to control robots for users. I spent my summer doing research through the DREU program with Dr. Chad Jenkins and the Brown University Robotics group. Most people do not know how to code in a high-level programming language, so it would be difficult for them to operate a robot as it is now. To help find solutions to this problem, we used rosbridge, a program that allowed users to control robots through different programming interfaces that are easier to use, created by Brown University researchers[1]. Rosbridge allows the ROS code to be translated into other languages such as Processing, an art-based language; Scratch, an educational programming language for children; XNA, the programming language used for XBOX to control a robot with an XBOX controller; HTML/JavaScript to control a robot through a webpage; and even more. We were able to work with the iRobot Create robot and the AR Drone Parrot Quadricopter, both which use the same ROS programming for the most part.

# 2. Learning the Background Information

At the beginning of this research experience, our mentor asked us to download all of the software and a virtual machine necessary to run Robot Operating System (ROS) to control the robots and create programs. Then we were asked to complete the first two assignments that he gives his class "Building Intelligent Robots" (CS 148), to help us get familiar with the software. After downloading all of the software, I had to go through tutorials and learn how to use the ROS instructions, Linux commands which ROS runs from, and Python which was compatible with ROS. There is a version of ROS that reads Java, which I am very experienced in, called "rosjava", but the VMware image that we downloaded with all of the Brown University Robotics files saved on it only used "rospy" for Python. After becoming more familiar with these things, I began to work on the two assignments that he gave us. The first assignment was entitled "Enclosure Escape"[2]. We had to program an iRobot Create to turn in a different direction after it bumps into an obstacle. Most of the code was provided for this project to help us learn about importing packages from ROS and the robot itself such as the bump sensor. The second assignment, "Object-Seeking", I completed from scratch. The goal of this assignment was to have an iRobot Create seek out a specific Augmented Reality tag, move toward it, then spin and seek the next tag, move toward it, etc. until all of the tags have been found. This assignment was much more difficult and took longer to finish than the first one. A PlayStation 3 camera was used for the iRobot Create to see where it was going. After completing the second assignment, I adapted my program so that it would work with the Drones. The code was basically the same but some of the numbers had to be changed around.

# 3. Project

## 3.1 First Project Idea

After completing the first two assignments, we were pretty much done with our time in Providence, RI and I was able to go with my mentor and his family to Moss Beach, CA along

with the other summer interns and a postdoc student for the remainder of the internship. By this time, I decided that I wanted to work on the ScratchROS project to build a better interface for using ROS with Scratch. Scratch is an educational programming tool for children where the user could just drag and drop an instruction into the editor and an animated cat called "Sprite" would act out the code. I chose this because this would greatly simplify robot control by dragging and dropping instructions and essentially anybody would be able to control a robot. I also chose it because I am familiar with Alice, another educational programming tool for kids that uses the "drag-and-drop" interface, but its animations use 3D objects and Scratch is in 2D. There was a version of Scratch that worked with ROS, but it was the actual Scratch program. There were many parts of the interface that were not used because there was no ROS command for them. I was going to create a new interface that only contained the instructions that could be translated into ROS, to get rid of all of the unused commands, and to possibly add some other ones that were not present in Scratch. I became familiar with Scratch then I had to download a different virtual machine image to use ScratchROS. It took a while to get it to work, but I was finally able to control a Create with the ScratchROS program and I created a program to drive the robot with the arrow keys and recreated the "Enclosure Escape" program, which was much faster and easier that creating the code in Python. Below is a copy of the "Enclosure Escape" code in Python and a copy of the same code in Scratch to show how much simpler creating code for robots would be with a Scratch Interface.

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('enclosure_escape')
import rospy
from geometry_msgs.msg import Twist
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True

def hello_create():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('hello_create')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 2
`           bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0; twist.linear.y = 0; twist.linear.z = 0
```

```
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.1
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        hello_create()
    except rospy.ROSInterruptException: pass
```

**Fig. 1-** This is the "Enclosure Escape" program code in 40 lines of Python code.

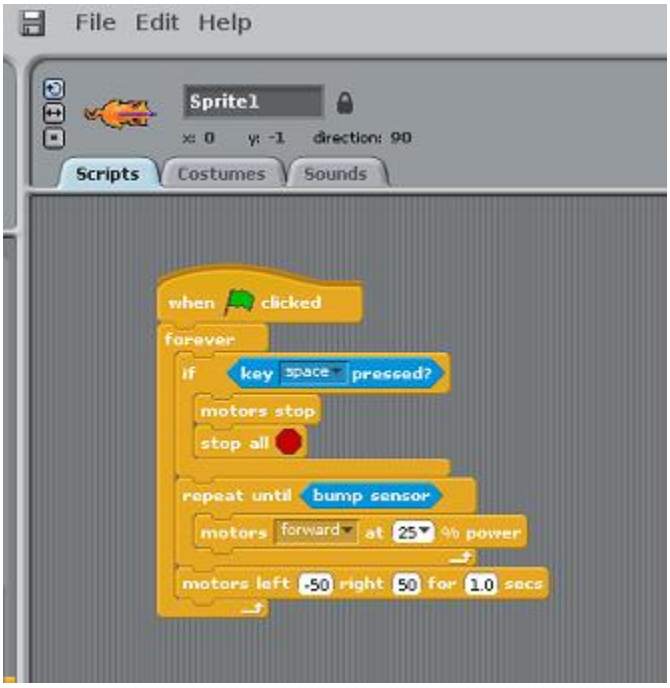


**Fig. 2-** This is the "Enclosure Escape" program code using Scratch in only 8 lines.

As seen above, Scratch makes programming for robots much easier to code and understand for people who are not familiar with coding. The differences in the code for the "teleop_twist_keyboard" program to drive a robot with the user's keyboard were even more drastic. This program was created by the Brown University Robotics Group. In Python, the code was 106 lines long, but my version with the Scratch code was just 15 lines long and can be seen in Figure 3. The Scratch version, however, could not do all of the tasks that the Python version could such as increasing/decreasing the linear speed and the turning speed. The Scratch version can only move forward, backward, turn left and right, and move forward at an angle to the left and right by pressing the designated keys on the keyboard.

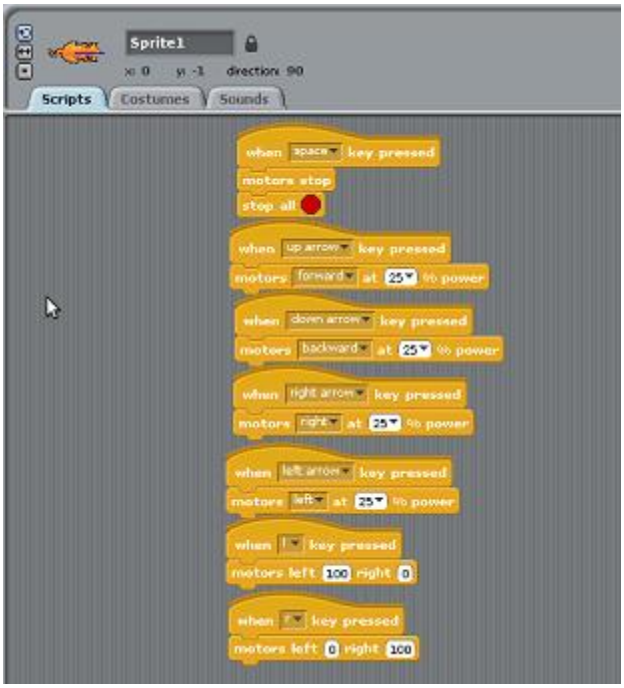

**Fig. 3-** My version of the "teleop_twist_keyboard" program in Scratch code.

While the Python version of "Enclosure Escape" took me about a week to complete after becoming familiar with Python and getting the code to work, learning how to code in Scratch and then creating the Scratch versions of the "teleop_twist_keyboard" and "Enclosure Escape" programs took only one day. An interface that used ScratchROS instead of rospy would be much better for non-ROS users. The interface that I would have created would have most likely used the Internet and Rosbridge to have the user click and drag an image or click a button to use a specific instruction and control the robot. I was not, however, able to work much on the new interface, but it is possible that I will continue to work on it throughout the semester or try to return to Brown next summer to complete an interface that will use Scratch and Rosbridge to program robots efficiently.

## 3.2 Final Group Project

Before I was able to begin working on the new interface for Scratch and ROS, Dr. Jenkins gave all of the summer interns a "Hack-a-Thon", or a project to work on using telepresence and it had to be completed within one week. Telepresence is the concept of being in one place, while being able to observe and move around in a different environment simultaneously. For example, if businessmen from around the world would need to have a conference meeting, they could simply all use telepresence robots in one place to meet and look around at each other. We were thinking about applications with rosbridge and we came up with the idea to create a Virtual Waiting Service. The premise was that at a party or any other event, someone can log on to our website and tell a teleoperator what he wants to drink and where he is located. Then, the teleoperator tells the chef in the kitchen what drink to get. After the drink is ready, the teleoperator drives the robot to where the customer is, delivers the drink, then he drives the robot back to the kitchen to prepare for the next order. We separated the tasks, and my job was to create the website interface and use rosbridge to control the robot through my

webpage. I was already familiar with HTML, but I had to read more about "rosjs", a program that connected JavaScript and Rosbridge to learn how to use them. Adding the appropriate JavaScript code to my website that communicated with Rosbridge would allow a connection to be made, and a user could drive our iRobot Create using the webpage as well as receive a feed from the PlayStation 3 camera that was connected to it through rosbridge signals. Other tasks for the project were to create the hardware of the robot to hold a laptop and drinks, receive sensor messages from the iRobot Create (made in XNA in a different interface), find a video chat software that would work well, and documentation of our project. We were able to put all of the parts together and run a successful demo of our project in one week's time. After we finished, there was so little time left for our internship that we just made improvements to our project. I enhanced the website to make it look more professional by adding a template background for all of the pages. I also added a page to contact all of the summer interns and another page for pictures and videos. We also wanted to make a better video of our demo at a restaurant, but the free Wi-Fi at all of the places we tried was not able to support the connection with rosbridge between a laptop to a netbook connected to the robot. We determined that these networks were port-blocking, and the only solution would have been to go to the restaurants' ISP. Our group ended up using our robot to check out a book at a local public library, where the Internet connection allowed the connection that we needed. We used our telepresence robot to check out a book in the library, as if we were unable to make it to the library. This also provides another application for robotics and telepresence robots in our society.

# 4. Conclusion

I really enjoyed being in the DREU program this summer. I was able to learn a lot about robotics and I learned about programming in ROS, Linux, Python, and JavaScript, all of which I had never used before this program. I was able to successfully create programs in Python and implement JavaScript and Rosbridge in a website. Unfortunately, I was not able to create the Scratch interface to Rosbridge to help simplify the coding for ROS and greatly improve the human interaction for non-computer scientists. The skills that I learned will help later on in my future with Computer Science. Our mentor and his grad students provided many different helpful websites with tutorials, cheat sheets, and practice for different programming languages, and also improved my ability to learn new aspects of computing in a short period of time. This REU also helped me to be able to work with others better. For almost all of the summer, I worked with 4 other undergraduate summer interns on the first two assignments and our final project. The trips that we took also helped me to realize what it takes to succeed in Computer Science field. Most of the companies we went to maintained a very cool and an informal environment with game rooms, gyms, etc., but they were very serious about their work. Whether it was finishing a product before a deadline like a movie or video game, or just researching the next big idea. Both the tour guides at ILM and EA Games said that work gets really intense before the release date of the game or movie coming out. The one at Google said that if they haven't come up with or started working on anything new in about 3 months, they were behind, which he said applied to the work ethic of the entire Silicon Valley. If I want to work in the Computer Science field, I will have to keep working hard and keep learning more and evolving as new technologies arise to compete with my co-workers. This internship has deeply helped me prepare for a future in Computer Science in graduate school and in the work force.

# References


Brown University Google Code Repository:

- Robotics Learning and Autonomy at Brown. “brown-ros-pkg: Brown University Repository for ROS Packages”. *Google Code*. Brown University. < http://code.google.com/p/brown-ros-pkg/>.

Brown University Summer Interns’ Google Code Repository:

- Robotics Learning and Autonomy at Brown. “brown-reu-interns: Development space for REU student interns at Brown Robotics”. *Google Code*. Brown University. < http://code.google.com/p/brown-reu-interns/>.

1. Christopher Crick, Graylin Jay, Sarah Osentosiki, Benjamin Pitzer, and Chad Jenkins. “Rosbridge: ROS for Non-ROS Users”. Brown University. <http://aduni.org/~chris/Crick11b.pdf>

2. Jenkins, Chad. “CS 148 Assignment Enclosure Escape”. *MediaWiki*. Brown University Computer Science. 5 October 2010. < http://brown-robotics.org/index.php?title =CS148_Assignment_Enclosure_Escape>

3. Jenkins, Chad. “CS 148 Assignment Object Seeking”. *MediaWiki*. Brown University Computer Science. 25 September 2010. < http://brown-robotics.org/index.php?title= CS 148_Assignment_Object_Seeking>