San Diego Robotics 101
RoboSub 2014

Members: Ryan Ganton, Tushar Pankaj, Rahul Salvi, Rahul Keyal, Jason Ganton

ABSTRACT

Cubeception is an autonomous robot designed by San Diego Robotics 101 to compete in the 2014 RoboSub Competition held in San Diego. Cubeception is held together by a Polyvinyl Chloride (PVC) frame, and uses custom-made fixtures to attach pumps, cameras, torpedo launchers, etc. It is propelled by a set of twenty-four bilge pumps. The robot has also been given an Arduino, a Raspberry Pi, and a custom-made shield board, which serve as the robot's communications and computation systems.

INTRODUCTION

San Diego Robotics 101 is a team created to compete in the 2014 RoboSub Competition. While all individuals of our group have personal backgrounds in robotics competitions, this is our first competition together as a team.

MECHANICAL SECTION

PROPULSION

Cubeception is equipped with twenty-four bilge pumps, four on each of the six faces of the square. We decided on bilge pumps because of overall simplicity and convenience. Also due to limitations with our team budget, bilge pumps proved to be the best choice in terms of both efficiency and cost ratio. One other great benefit is that the whole pump is already waterproofed, including the wire. We tested them first with two motors facing the same direction on a Styrofoam boat float in a backyard pool and the experiment demonstrated that the bilge pumps would function smoothly. However, a problem arose in which the motors could only pump in one direction. To fix this, we required

that for every one thruster, we had another thruster facing the opposite direction so that the robot could tilt shift and transverse any direction in 3-D space. Attaching the bilge pumps to the PVC frame proved to be challenging, and as a result we had to engineer a custom 3-D printed part to solve the problem (Figure 1).

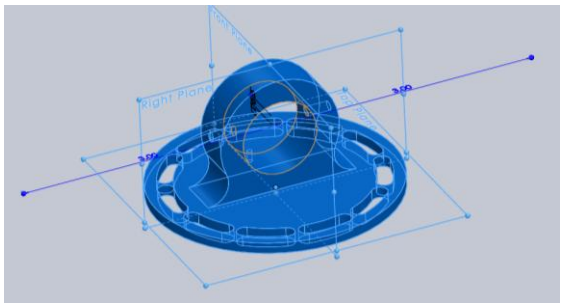


Figure 1

FRAME

The frame is 1/2-inch PVC pipe crafted as a cube so that we have equal efficiency moving in all directions (Figure 2). Instead of gluing the pipes, we decided to drill them together using nuts and bolts. This offers two advantages (as opposed to gluing them): first, it makes it easier to disassemble and reassemble; second, instead of adding weights to reduce the robot's buoyancy, we can simply let water flow through the robot's skeletal frame.

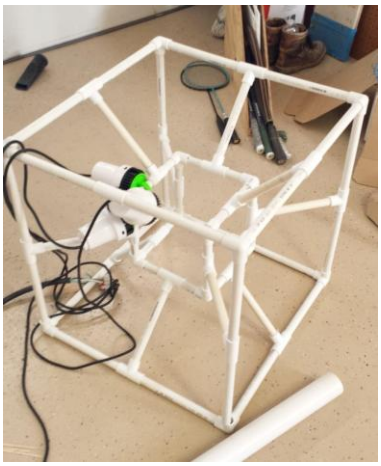


Figure 2

Since no electrical components are in the pipes, there is no need to glue and waterproof the insides of the PVC. We considered running wires through the pipes, but realized it would have created unnecessary complications. The outer frame holds most of the attachments for the game missions.

MECHANICAL SYSTEMS AND ACTUATORS

The robot's "Bottom side" is configured with two torpedo tubes powered by an electromagnetic/plasma tungsten point and CO_2 canister. The tungsten is present to hold an arc to punch a hole in the CO_2 canister. There is two grappler/tubes that hold the landing

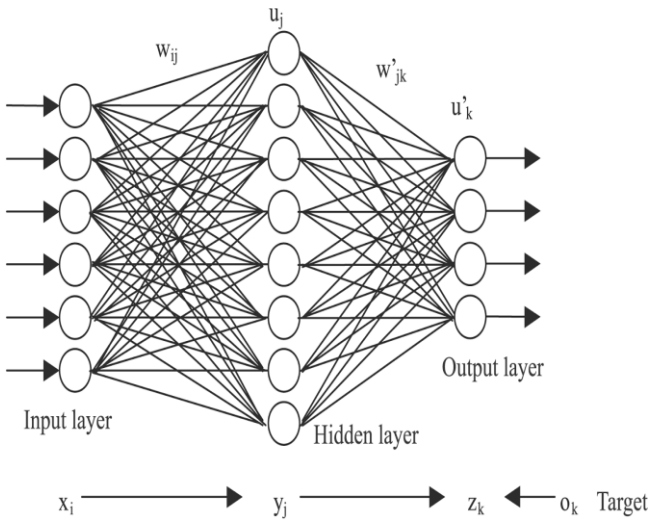
markers in place of which one simple servo switch will release the kraken. These grappler manipulators will also serve a dual purpose of grabbing and releasing samples from the moon sample bins.

SOFTWARE SECTION

The control system of our robot consists of two boards, an Arduino Due and a Raspberry Pi. The Arduino Due is used for controlling the bilge pumps and collecting all the inertial sensor data. The Raspberry Pi is dedicated to vision processing.

VISION PROCESSING

Two Raspberry Pi Cameras connect to the board using the native camera interface. Meanwhile we use the OpenCV library to perform computer vision tasks. Our algorithm thresholds the camera images based on color, resulting in a binary image. The binary image is then fed into OpenCV's SimpleBlobDetector to find objects of the thresholded color. We then use the results of the SimpleBlobDetector to crop the relevant part of the camera image and send it to a feed-forward neural network (multi-layered perceptron) implemented in OpenCV with a logistic activation function. The neural network was previously trained to distinguish between the various field elements in the RoboSub game, and it determined which of the field elements the detected object most closely matched. Once a field element is identified, the vision processing program computes a course of action and communicates it to the Arduino Due. The Arduino Due runs a separate program to navigate the robot to the right location. During this navigation, Raspberry Pi and Arduino Due continue to communicate to keep a loose feedback loop.



MOTOR CONTROL AND SENSOR INPUT COLLECTION AT ARDUINO DUE

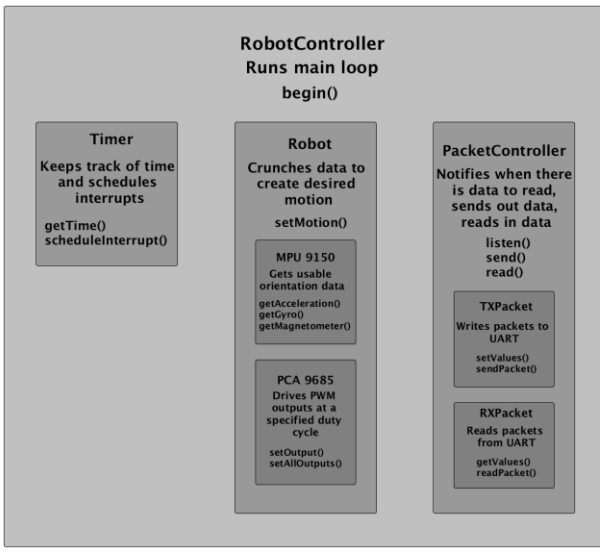


Figure 3

The program running on Arduino Due receives commands from the Raspberry Pi and uses inertial data to create the desired motion. The Due has a limited number of PWM outputs, necessary to drive our pumps, so we needed a way to produce multiple PWM signals with only a few pins. We settled on two NXP Semiconductors PCA9685 chips, which can produce up to sixteen PWM signals through a two-wire I2C interface. For orientation data, we used a MPU 9150, a chip consisting of a three-axis gyro, accelerometer, and magnetometer. We communicate with this chip over I2C as well. Because the Due has two I2C buses, we were able to split the communications between separate chips and speed up the system overall. The Due communicates with the Raspberry Pi over UART to chart the course for the robot. Given only simple commands from the Raspberry Pi, the Due figures out the values at which to run each bilge pump to create the desired motion. This frees up valuable computing power on the Raspberry Pi so that more resources can be used for vision processing. Additionally, it allows the packets exchanged between the boards to be much simpler and smaller.

COMMUNICATION BETWEEN RASPBERRY PIE AND ARDUINO DUE

The Raspberry Pi was also connected via UART to an Arduino Due through our custom shield board. The Arduino was connected to the PWM controllers for our bilge pumps and also to I2C sensors like an accelerometer, gyroscope, and magnetometer. The Arduino sent the sensor data to the Raspberry Pi over the UART interface and the Raspberry Pi responded with directional movement commands computed based on the

neural network output and the data collected from the I2C sensors.

RASP => ARDU

	7	0
0	HEADER	
1		
2	VEL_X	
3	VEL_Y	
4	VEL_Z	
5	ROT_X	
6	ROT_Y	
7	ROT_Z	
8	POS_Z	
9		
10	TORPEDO_CTL	
11	SERVO_CTL[0]	
12	SERVO_CTL[1]	
13	SERVO_CTL[2]	
14	SERVO_CTL[3]	
15	SERVO_CTL[4]	
16	SERVO_CTL[5]	
17	SPARE	
18	CHECKSUM	
19		

ARDU => RASP

	7	0
0	HEADER	
1		
2	ACC_X	
3	ACC_Y	
4	ACC_Z	
5	MAG_X	
6	MAG_Y	
7	MAG_Z	
8	PRESSURE	
9	SPARE	
10	CHECKSUM	
11		

ELECTRICAL SECTION

POWER DISTRIBUTION BATTERY

Our robot is powered by two Tenergy 12V 7 AH batteries that will work in tandem to supply power to the entire system. We compared the energy, cost, weight, and size of many different lead acid and lithium ion batteries and found that this battery was one of the most efficient batteries.

POWER MANAGEMENT

All power cables and electrical components are held in the one aluminum box located at the very heart of our robot (the inside center cube). There is one port hole in the box for the camera to look forward, a separate hole for another camera to look down. The robots core housing will contain an integrated cable clamp on the back for the bilge pump wires, and other motors, in order to reduce the probability of leakage.

SHIELD BOARD

The shield board was designed to solve the problem that the Arduino did not have enough ports to satisfy the amount of bilge pumps and servos that the robot required. Cubeception has twenty-four bilge pumps and thus required more control ports than the Arduino contained. A shield board was designed to solve this problem. Due to the size of this board it was determined a slightly larger board could hold connectors for the Arduino and Rasp Pi and thus reduce cabling. This allows the two boards to communicate through a UART interface.

OTHER SECTION

GAME STRATEGY

We calculated the total game points to determine which missions we should accomplish by ordering them according to point values and relative complexity. That being said, we do not plan on competing in the hardest one, the reroute mission, due to its difficulty and isolation from the rest of the competition. The highest priority however is the sample recovery mission in which we grab underwater samples and place them in a basket. For this particular mission we're not aiming to sort the objects but instead to place the objects in both bins and grab as many objects as we can. For the first several missions we intend to complete them routinely. For the 'Invitation to Brunch' mission we plan on firing into the larger holes from medium distance or firing the invitation (missile) into the small hole. For the landing marker mission (Figure 4) our plan is to do a quick survey for the primary and secondary targets, but if the camera does not find either of the targets on the first try we will drop the markers into two random bins.

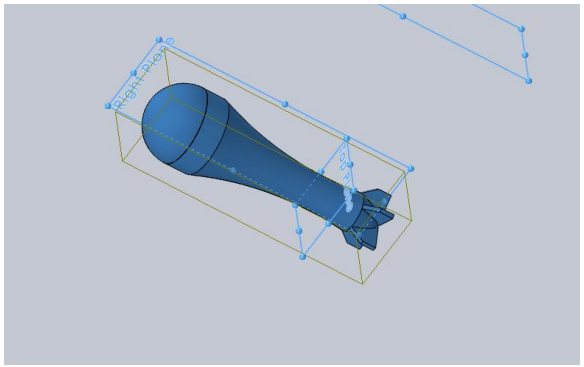


Figure 4