Design and Implementation of Cubeception 3

Bryce Ballam, Cole Ballam, Ryan Ganton, Jason Ma, Patrick Paxson, Robert Quitt, and Rahul Salvi

***Abstract*—Cubeception's iterations over the years have traditionally boasted large, flat surface areas allowing for relative ease mounting external sensors and components compared to that of a curved surface. Cubeception 3.5 takes full advantage of this area now, with one face dedicated to an easy-access battery compartment, another to a sonar array mount, two faces for connectors and thruster arms, and two to portholes for cameras and ease of access to electronics. Six on-board Raspberry Pis allow for robust, dedicated processing of sensor data and mission parameters.This unique design allows for fast prototyping, ease of manufacturing, and low construction costs, all of which played an essential role in the completion of the finished core. As a realization of Cubeception 3's original design, this platform aims to achieve a wide variety of tasks at relatively low cost and manpower.**

I. Introduction

San Diego Robotics 101 is a highly interdisciplinary team consisting of members across multiple schools at the university and high school level with the purpose of building autonomous unmanned underwater vehicles. The focus of the team is on exploring the nature of cube-shaped underwater vehicles, particularly the high modularity and quick build time afforded by this design. San Diego Robotics 101 is also interested in the potential applications of its design to the autonomous underwater vehicle industry, where cost-effectiveness can be key to the design of an underwater vehicle. Traditionally, many components used in this industry are hard to machine or fabricate, significantly driving up costs of production. Cubeception 3.5's approach cuts down significantly on these costs through the use of metal sheets on the core, 3D printing, and the use of multiple Raspberry Pi boards as opposed to a single, powerful one. The process of construction involved CAD modeling, board prototyping, experimental analysis, software testing, simulations, and assembly as well as real-world trials, all of which were done over the course of 2 years by a team of 7 students. San Diego Robotics 101 participates in the annual AUVSI Foundation RoboSub Competition, which takes place in late July and is hosted at the SSC Pacific TRANSDEC in San Diego, CA. This competition puts AUVs like Cubeception 3.5 through a wide variety of missions which model tasks performed by professional AUVs. These tasks range from passing through a gate to touching buoys, locating pingers, surfacing within octagons, and handling objects, requiring precise maneuverability, mission prioritization software, and navigation capabilities. To achieve as many of these tasks as possible, San Diego Robotics 101 is divided into Mechanical, Electrical, Navigation, Sonar, and Computer Vision subteams.

II. Design Overview

A. Design Strategy

A high emphasis was placed on capability and maneuverability while keeping costs low in Cubeception 3.5's design. As such, Cubeception 3.5 features a low-cost set of sensors including 4 hydrophones for sonar and 1 camera for computer vision. This was determined to be the minimum set of sensors to enable long-range target detection and short-range precise movements. A minisub was also developed alongside Cubeception 3.5, with computer vision and actuating capabilities. Much of San Diego Robotics 101's time was spent researching and simulating sonar and computer vision to determine whether they were viable options on Cubeception 3.5. The analysis ultimately concluded that sonar was effective in bringing Cubeception 3.5 within close proximity of tasks, while computer vision would allow Cubeception 3.5 to precisely navigate through them, making a sonar and computer vision combination the most viable addition to Cubeception 3.5's design in terms of added capability and point scoring.

The additional complexity introduced with sonar and computer vision capabilities prompted San Diego Robotics 101 to implement a Raspberry Pi network in the hopes of increasing robustness and modularity against the possibility of a single sensor failure. Furthermore, significant changes in the sealing of the main core were introduced to reduce the possibility of human error. The batteries need to be removed relatively often compared to the electronics core, so having a separate battery enclosure with a clear lid for the operator to visually inspect the seal allows for faster and more reliable waterproofing.

San Diego Robotics 101 also has relatively limited manpower and funding compared to other AUV teams, so many design strategy choices were made with that in mind, balancing capability and reliability with resource availability.

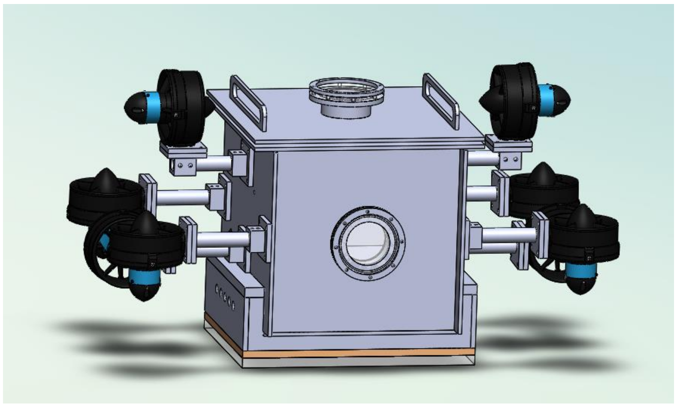
Fig 1. Full model of Cubeception 3.5.

B. Hardware Overview

Cubeception 3.5’s core is based off of Cubeception 3’s core, with the same screw-fastened lid and portholes for cameras to see out of. The addition of a more maintainable battery enclosure makes Cubeception 3.5 a more robust and maintainable platform than ever before.

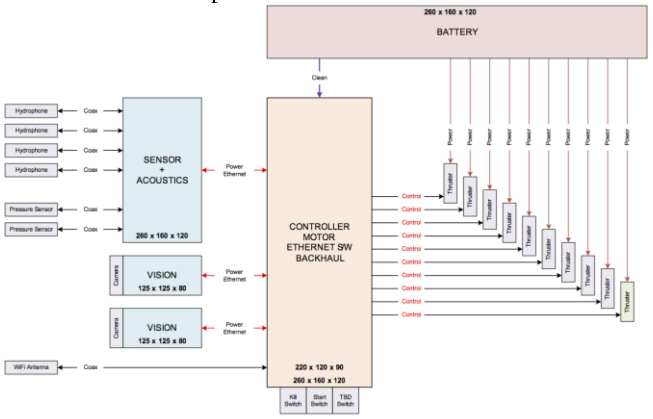

Fig 2. High level overview of Cubeception 3.5’s hardware.

C. Electronics Overview

Cubeception 3.5’s electrical hardware has been completely redesigned for this year. Each of the many custom circuit boards was designed in the EAGLE PCB design suite, then sent out for fabrication to a local PCBA manufacturing business. Many design considerations were made to effectively meet the requirements that were placed on Cubeception 3.5’s design. The final result is an electrical subsystem that efficiently delivers power to and carries signals between the different parts of the robot. Having custom FPGA boards also allows for more sensor data throughput, further increasing the effectiveness of intensive software subsystems.

D. Software Overview

Cubeception 3.5's new distributed computation model improves the performance of each individual component and promotes high parallelism in its software functionality. The cluster of Raspberry Pi 2 computers onboard naturally allows many processes to be run at the same time, increasing the overall throughput and decreasing the latency of the system. Individual computers are given specialized tasks, creating clear divisions between responsibilities. Instead of a “tall” software stack with many processes built upon each other, Cubeception 3.5’s stack is “wide”, with many processes sharing information, but able to proceed with the loss of any individual machine.

III. Mechanical Design

A. Thrusters

Cubeception 3.5 features a major upgrade in thrusters over previous designs. One of the major issues in previous designs was the complexity and unreliability of maintaining 24 bilge pumps, which broke often and did not provide much thrust. When deciding on a new system to replace the previous 24 bilge pump system, VideoRay thrusters and Blue Robotics T-100 thrusters were the primary candidates. Both types of thrusters required a significantly greater amount of power, although the VideoRay thrusters would require too much. Therefore, the best option was 8 T-100’s with future plans of implementing the VideoRay thrusters as a main source of forward thrust. This system requires only a third of the number of motors from previous designs, and provides twice the amount of thrust, reducing complexity and increasing performance significantly.

B. External Mounting

The PVC rectangular prismatic frame was removed from Cubeception 3.5, as it did not provide enough benefits to justify its continued use. Although this exposes the core and outer components to potential impacts, it also simplifies cable management for the robot, allows for easier maintenance of the thrusters, decreases drag during movement, and allows the motors to run with less interference against the sensors located on the core. Pylons will extend from the sides of the robot to house these Blue Robotics thrusters, away from the robot and each other.

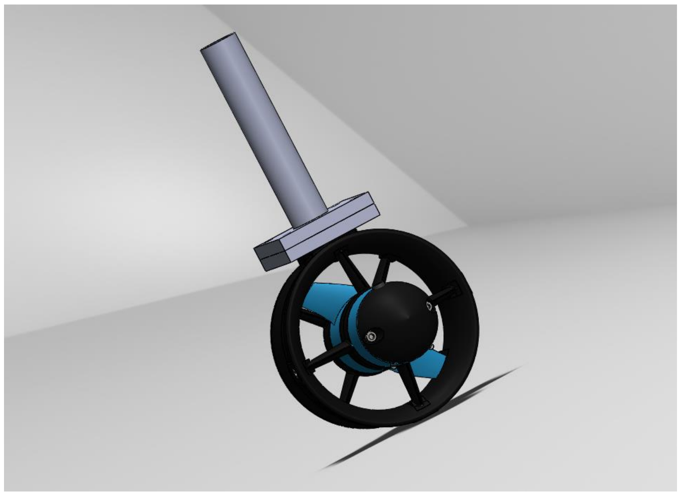
Fig 3. Pylon mount for Blue Robotics T100 Thruster.

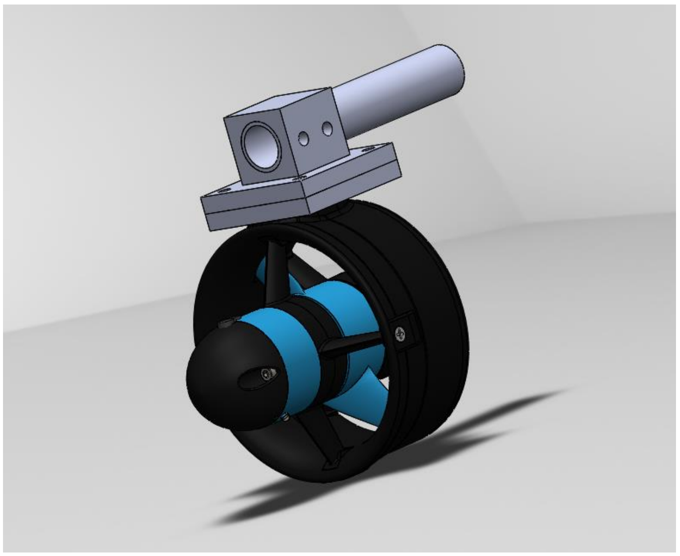
Fig 4. Alternate pylon mount for Blue Robotics T100 Thruster.

C. Cube Core

As an improvement over Cubeception 3’s core design, Cubeception 3.5’s core features greater ease of access to the batteries while improving the robustness of the seal by replacing threaded holes with latches. With the introduction of a separate battery box outfitted with these latches, the lifespan and reliability of the seal has increased greatly.

The two most appealing options for such an enclosure were two smaller enclosures on the side of the core or a large one integrated into the core. Although the two smaller enclosures offered a more even distribution of weight, the single integrated enclosure was easier to build and required fewer changes from Cubeception 2's core. More space was also available on the sides of the core for thruster mounting in the single enclosure design. This decision resulted in a trade-off involving slightly increased complexity in software for greatly reduced complexity in hardware.

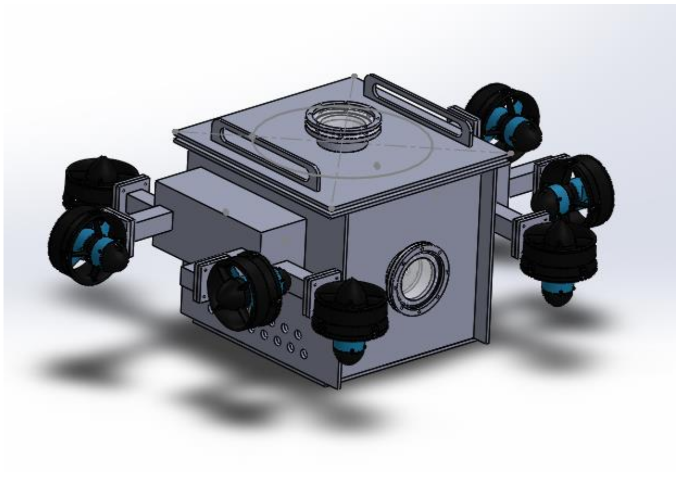
Fig 5. Alternate design with two battery enclosures.

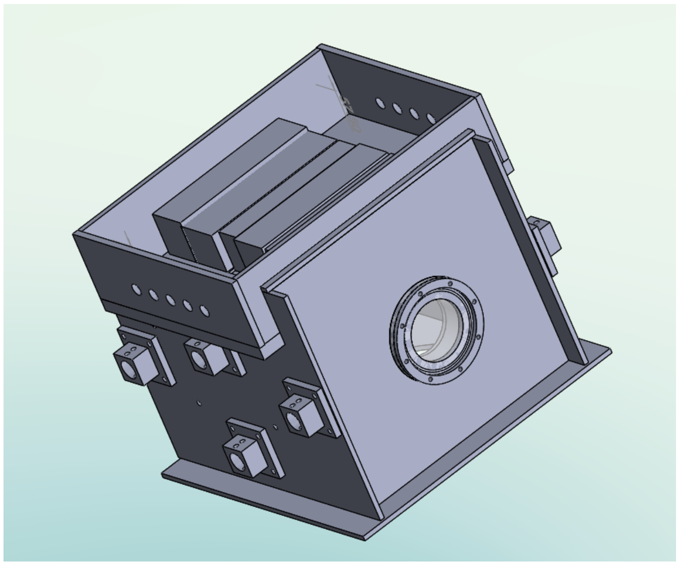
Fig 6. Cube core with integrated battery box on top.

IV. Electronics Design

A. *Motor Driver Design*

To control each of the many outputs that must be driven on the vehicle, a motor driver shield board is used. The main IC in use to create the pulse-width modulation (PWM) outputs is a MAX6696ATE. The board attaches to a Raspberry Pi and communicates with it over SPI. Software can command any of the 20 channels to produce a specific PWM signal. These signals are either available as singled-ended drive, as in the Raspberry Pi, or opto-isolated drive for use with the motors. This helps eliminate potential ground loop problems that can occur in large systems.

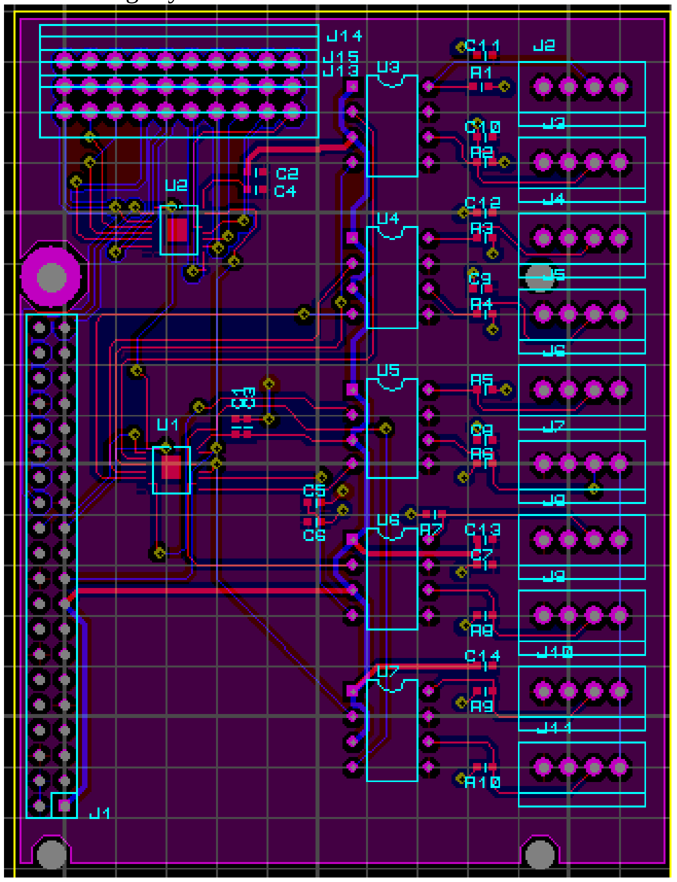

Fig 7. Layout of motor driver shield.

B. *Power Distribution Board Design*

Efficiently supplying each subsystem with power is done by using a custom power distribution board. The design features several DC converters to take the 11.1V input from the batteries and create 5V, 3.3V, and 1.8V outputs. As the battery voltage is subject to dropping largely when multiple motors are in use, creating large fluctuations, the board splits power between "dirty" and "clean" sides. Dirty power may fluctuate greatly, but clean power is regulated to produce a steady voltage, ideal for sensitive electronics.

C. *FPGA Shield Board Design*

To address the Raspberry Pi's lack of high throughput I/O, a custom FPGA shield board was designed. An FPGA was found to be the most suitable system because of its high configurability, making the shield a modular attachment to any Raspberry Pi. The card is capable of both input and output, so it is critical to both the sensor and sonar subsystems. The onboard Xilinx Artix-7 FPGA is programmed in VHDL using Xilinx tools and is more than powerful enough for each task to which it is assigned.

D. Sensor Board Design

Cubeception 3.5 features a custom circuit board for polling sensor data from its IMU and pressure sensors. The board has a total of nine STMicroelectronics LIS3DSH accelerometers, nine STMicroelectronics LIS3MDL magnetometers, and four InvenSense MPU3300 gyros, exploiting the advantages of multiple-sensor systems to great effect. Additionally, there are analog to digital converters for the Measurement Specialities UltraStable 300 pressure transducers on the vehicle. A Raspberry Pi is incapable of the I/O throughput necessary to read all the sensors in reasonable time, so the FPGA shield card is incorporated with the sensor board. The FPGA is able to read each sensor while the Raspberry Pi processes the data for other subsystems to consume, resulting in a fast and effective sensing system.

E. Sonar Board Design

To produce sonar pings and listen for their return, Cubeception 3.5 employs 4 Aquarian Audio H1C hydrophones and a specialized circuit card. The PCB turns a digital ping signal into analog output, specifically a gated 15Vp-p 200 kHz sine wave to drive the hydrophones. In addition, the card listens to the analog input from the hydrophone and converts it to a digital signal for further processing, using a quadrature detection method popular in radio system. Both pathways are bandpass filtered to eliminate undesirable noise introduced by the mechanical systems. Switching TX to RX is designed to happen extremely rapidly, to allow each hydrophone to act as both a sender and a receiver. This means that each hydrophone can send out a ping and immediately listen for responses.

V. Software Design

A. Sensor Data

Cubeception 3.5 has several on-board pressure and inertial sensors. The data from these sensors is filtered and processed to create an accurate representation of the vehicle's position and orientation. To reduce noise in IMU readings, several identical sensors are placed in close proximity. The individual signals are then averaged together to produce a final signal with less overall noise.

The data from the 9-axis IMU is fused together using an unscented Kalman filter to produce an estimate of the vehicle's orientation and velocity. Internally, quaternions are used for the orientation representation to prevent gimbal lock; however, a simple conversion in to yaw-pitch-roll format is used in monitoring software because it is more human-readable.

The depth of the vehicle is calculated using two pressure sensors. The sensors are mounted opposite each other on the vehicle such that when the vehicle rotates without changing depth, one sensor reads an increasing pressure while the other sees a decrease in pressure. This setup allows for a stable depth regardless of orientation.

B. Sensor Calibration

Sensor calibration consists of measuring an offset in the output of each sensor then correcting against this. A special case is magnetometer calibration where nearby metal objects can cause an output that is scaled and shifted. To handle this, the magnetometer output is found as the vehicle rotates, then scaled back to a normal level.

C. Networking

Cubeception 3.5 uses a distributed shared memory(DSM) architecture to facilitate communication between the various subsystems on board. A server process is created at startup for each processing node in the system. Other processes running on the same node can request a shared memory buffer and then access it normally. In the background, the server process will synchronize the buffer with other nodes. Additionally, the server will update local copies of remote buffers with new information as it is received. From the perspective of the client, complex networking has been replaced with simple operations on local data. This separation of complexity allows for faster development and easier debugging.

The implementation of the DSM software has been improved in several key areas. A new triple-buffered lock-free approach to updates allows clients present new information to the server without blocking. Reading new information has also been made into a non-blocking operation. These new methods allow for better real-time scheduling because the time taken per operation is much more consistent. A performance boost has also been attained by improving the cache performance of the DSM software by moving the most frequently accessed data into contiguous structures.

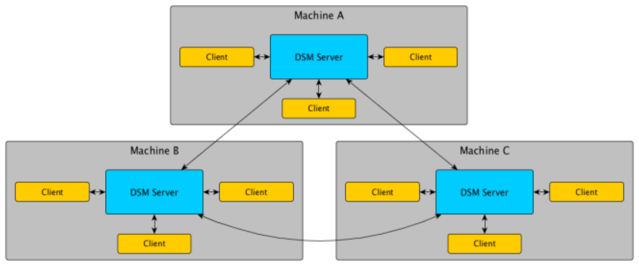

Fig 8. Cubeception 3 network layout.

D. Raspberry Pi Imaging

Cubeception 3.5's distributed computing introduced a new challenge for maintainability. Ideally, replacing an individual board in the system does not disturb the other boards. Additionally, replacements should be fast to set up and not require specialized instructions. To meet these needs, a solution was devised using both hardware and software methods. Each board is initially loaded with the same disk image on a microSD card. Specialized GPIO inputs are given to the Raspberry Pi at startup. A setup script then runs on the Raspberry Pi, recognizing these inputs and initializing the appropriate services and setting a static IP address and hostname. Setting the IP address of the board based on its role means that minimal changes are necessary on the other boards to incorporate a replacement. In addition, SSH can be used over Wi-Fi to remotely debug and interface with each Raspberry Pi.

E. Logging

A focus on logging was placed on this year's software design.

In the past, only processed data was logged, limiting the potential uses of the data. Cubeception 3.5's logging backend this year heavily leverages the distributed shared memory system to log all data, raw and processed, to local files and over the network. Raw data can be fed back through the processing pipeline to test new algorithms and find potential issues. This processed data can be plotted in real time for visualization and error identification.

F. Navigation

Cubeception 3.5 has control over all six degrees of freedom, and the software controlling it attempts to make optimal use of this fact. Compensation for buoyancy combined with the input from the sensor and sonar subsystems allows the vehicle to maneuver nearly identically in any orientation. Utilizing this, higher level programs can give a simple representation of complex paths to the controller process and have it calculate the appropriate propulsion system outputs. This level of abstraction makes it easier to develop the software for individual tasks in the pool. To further assist this goal, the vehicle can use dead reckoning to attempt to find various waypoints within the pool. From these waypoints, specialized software for a task can take over, giving more precise movement commands.

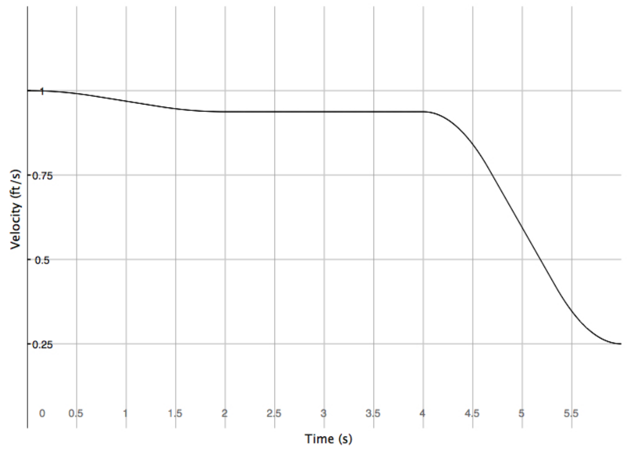

Fig 9. Sample velocity over time curves for a motion profile. This demonstrates smooth acceleration and deceleration.

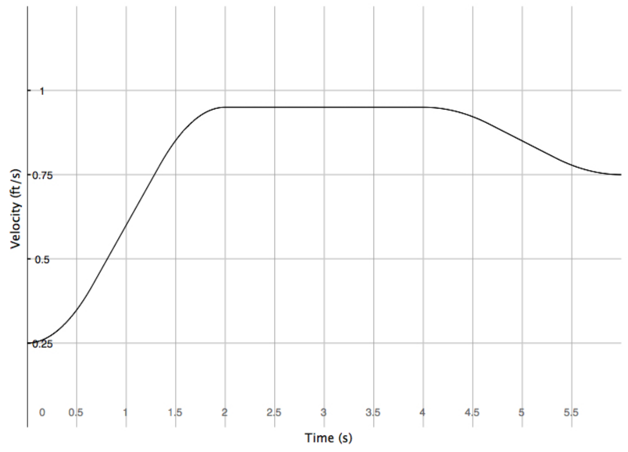

Fig 10. Another velocity over time curve for a motion profile.

To integrate the new physical motor layout on Cubeception 3.5, a new set of kinematic equations defining the robot's motion were derived. The focus of this work was to enhance the predictability of the system without environmental sensor intervention. Forces and torques that act on the robot can be estimated using knowledge of Cubeception 3.5's geometry and motor outputs. This precise physical model, combined with sensor inputs, allows Cubeception 3.5 to maintain a controllable state-feedback system.

G. Computer Vision

Cubeception 3.5's vision software utilizes a 5 megapixel Raspberry Pi camera connected to a Raspberry Pi 2 running OpenCV. Cubeception 3.5's modular design allows for a distinct separation of tasks. Since processing the images in real time is resource intensive, having the computer vision systems separate from the rest of the robot yields resources elsewhere for smooth operation. Due to the fault-tolerant nature of using the distributed shared memory architecture, Cubeception 3.5's other processes can continue to function without the vision system in case of an emergency.

Because the water is translucent, Cubeception 3.5 only uses computer vision as a short range maneuvering method. It is not used for long-range navigation, as many visual cues are not consistently discernible at range.

A forward-facing camera allows for detection of objects directly in front of Cubeception 3.5. The computer vision algorithm on this camera begins by averaging all the pixels to determine the hue of the background water in the image. This dynamic color-determining algorithm allows for objects in the frame to be isolated and categorized. They are put through a shape recognition algorithm to determine whether they are buoys, gates, or not of interest. From there, a contour-detection algorithm and trigonometric calculations yield how far and in which direction the objects are so Cubeception 3.5 can maneuver in an appropriate manner.

H. Sonar

Utilizing the sonar board and FPGA shield board, Cubeception 3.5 is capable of performing both active and passive sonar by profiling receiving signals on all 4 hydrophones, extracting peaks, and calculating time differences.

From there, the received times are used to create ellipses representing possible object locations. Target headings and distances can be determined from the intersections of these ellipses, and this data can be used to supplement navigation between mission elements or find pingers.

VI. Secondary Vehicle

A. Mechanical Design

The minisub is designed to fit the required components in a small space and place the motors in optimal positions. The electronics are placed in a tube at the top of the sub. Motors are placed beside and beneath the tube. The minisub tube is made of acrylic with an aluminium back cap, while most of the assembly is made of aluminium. The mounting hardware is mostly from Actobotics, with a custom adapter plate to mount the tube and position the motors in the correct place. The vertical tube used for mounting the bottom motors is filled with lead fishing weights to act as a ballast and to reach the desired buoyancy.

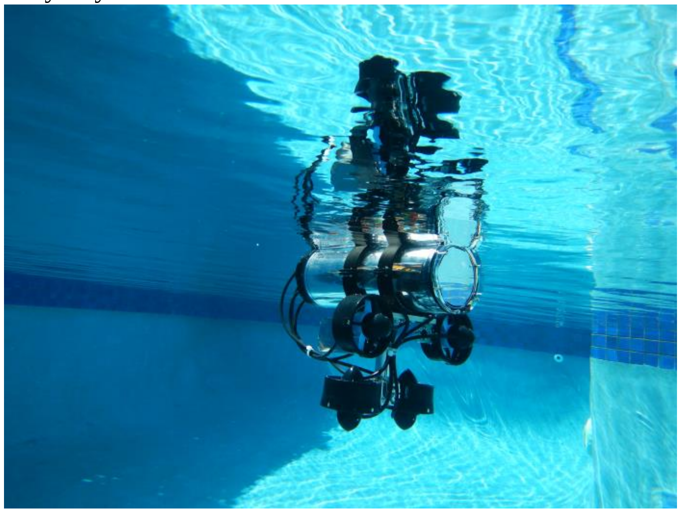

Fig 11. Minisub mechanical layout.

B. Propulsion

The minisub is powered by four T100 thrusters from BlueRobotics. Two of these thrusters are located on the sides of the tube and are used for forward propulsion. The other two thrusters are mounted on the bottom and are used to maintain depth. The pairs of thrusters are counter-rotating to keep the sub pointing straight.

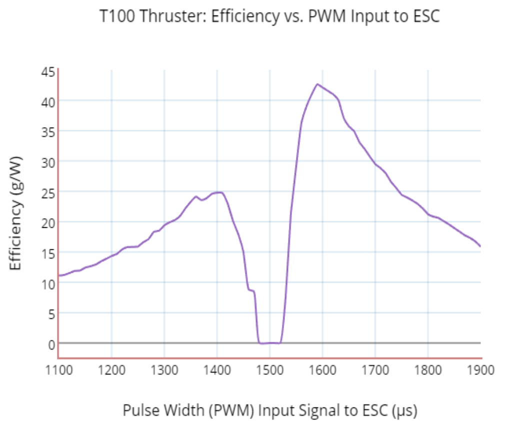

Fig 12. Thruster efficiency of T100.

C. Electronics

Two custom PCBs are held in the minisub's waterproof acrylic tube. One controls the minisub's power, and the other holds the digital components. The power board contains the ESCs that are used to power the motors. The logic board converts the voltage to 5v so our components can run. These components are a Teensy 3.2, a Raspberry Pi Zero, a Bluetooth radio, servo mounted camera, and an IMU. All components are socketed and thus are removable. There are also labelled status LEDs to indicate the status of the sub, even when the radio is unreachable. Our minisub is powered by a 3 cell LiPo battery with a 4000mAh capacity. This gives the minisub enough power to run for approximately 2-4 hours.

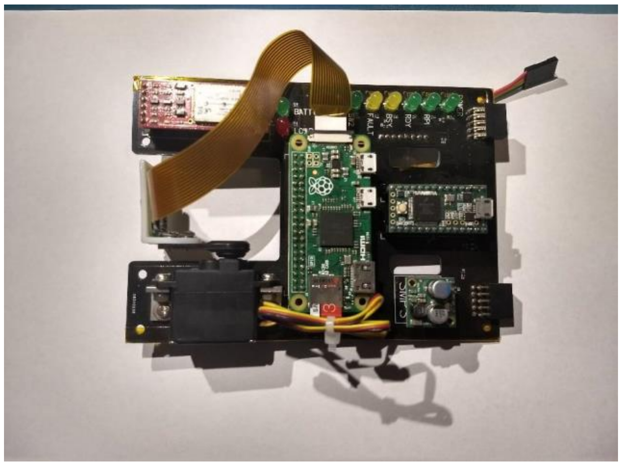

Fig 13. Custom PCB for minisub.

D. Software

The Teensy acts as a motor controller and sensor monitor. It takes short commands as an input through UART, and is capable of moving the minisub in all directions, setting a delay between actions, adjusting the vertical motor speed used for hovering, changing the position of the camera, or stopping all movement and surfacing. The software controls the depth by varying the level of thrust to achieve a set target depth. The Teensy also monitors the battery voltage and will surface and ignore all commands if the battery output voltage drops below 9.6V.

The Pi Zero sends commands to the Teensy based on images it takes through the camera unit. Along with controlling the minisub through commands sent to the Teensy, the Pi Zero keeps a log of all sensor input sent from the Teensy, along with all the pictures taken. The pictures taken are interpreted using Python's OpenCV module, which allows for basic color recognition and blob detection.

VII. Experimental Results

A. Sonar Simulations

After experimenting with various hydrophone array setups in Matlab and Python simulations, the sonar subteam found that arranging three hydrophones in a linear fashion and one hydrophone offset at a right angle from the middle hydrophone allowed for reasonably accurate resolution of target headings and distances in 3D space. The obtained data suggested that sonar is best used as a way to move close enough to mission elements so that Cubeception 3.5's onboard cameras could detect colors and features more accurately. The current model is accurate at determining one dimension of the target's location better than others, so strategic placing of the hydrophone array on the vehicle allows for exceptional depth detection. After target location resolution, the type of mission involved can then be guessed based on the depth it is located at.

Initially, the sonar subteam created simulations where the simulation would calculate a grid of times for objects placed in the field of view, comparing each of those times to program input to see if an object could possibly be in that location.

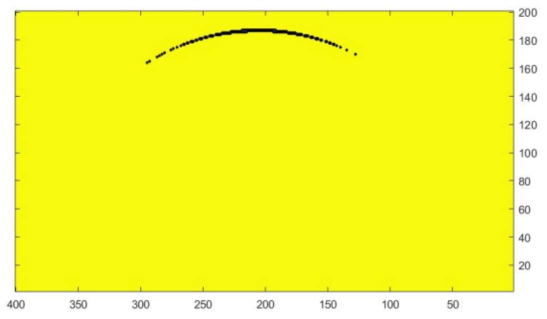

Fig 14. Two hydrophone array comparing received times to a time table generated based on expected times. The black curve represents possible locations.

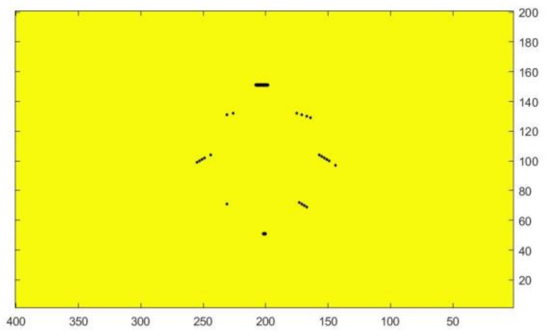

Fig 15. Three hydrophone array detecting eight objects placed in an approximately circular fashion using time tables.

The sonar subteam quickly found that using ellipse triangulation with a linear three hydrophone array was more effective, as it was possible to pinpoint the heading and distance of the object on a plane. However, this results in a circle of possible locations, so a 4th offset hydrophone was needed to resolve the location in 3D space down to a point. Used in conjunction with previous data and navigation data, this would allow for robust detection of targets over time with increasing confidence when they are detected continuously.

B. *Computer Vision Testing*

The computer vision subteam's first attempt at an algorithm was to detect circular edges. This failed since it was difficult to pick up edges in the water based on only light values. The algorithm was refined to use hue to distinguish objects, since the background water was a relatively uniform color compared to buoys and other objects. Many test images and videos from last year were tested to make the software perform well under various lighting conditions.

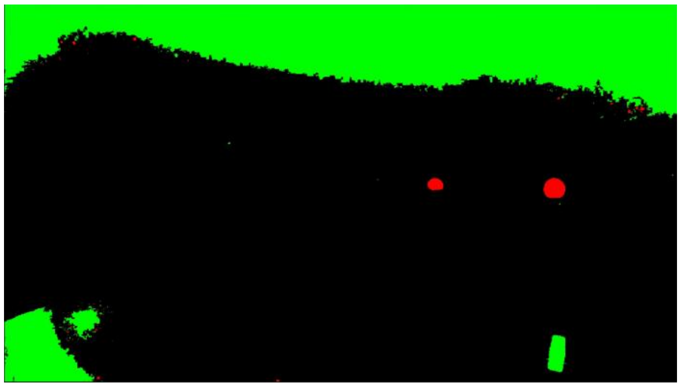

Fig 16. Computer vision on buoys.

C. *Multisensor Calculations*

Many of the same type of sensor on a single platform has several distinct benefits. From a logistical standpoint, having several sensors increases the redundancy of the system, as a single sensor failing does not bring down the entire processing pipeline. Additionally, repairs are not immediately necessary, improving the overall uptime. Considering the quality of data produced, several sensors used together can produce a better signal with some processing. Simple empirical testing saw improvements of up to 30% in angular random walk (ARW) for a gyro. With this in mind, a cost-effective solution to enhancing Cubeception 3.5's sensor performance was found.

Sensor	Gyro 1	Gyro 2	Average Data
Drift (degrees)	0.241	0.273	0.197

Fig 17. Angular drift of stationary InvenSense MPU-9150 gyros over 15 minute period.

D. *Component Testing*

Before Cubeception 3.5 is assembled, it goes through a leak test, all of the electronics are tested outside of the vehicle to ensure compatibility, and the software is unit tested to reduce runtime errors. Integrated testing for any potential problems will continue until the RoboSub competition begins.

VIII. ACKNOWLEDGEMENTS

San Diego Robotics 101 would like to thank all of its sponsors for making Cubeception 3.5 a reality. Solidworks provided invaluable software which made prototyping, simulating, and animating models a very streamlined process. MathWorks provided useful computational software which was utilized heavily by the sonar development team for simulations. Vinatech Engineering, Inc. and Apollo Manufacturing Services were also integral in the creation of Cubeception 3.5's electronics and core.