Cubeception: A Low-cost Alternative to Underwater Exploration

Jason Ma, Ryan Ganton, Rahul Salvi, Robert Quitt, Apoorv Agarwhal, Cole Ballum, Bryce Ballum

***Abstract*— In this report, we present a low-cost, labor-efficient autonomous underwater vehicle (AUV) design which displays a high degree of modularity and compartmentalization. Our strategy focuses on maximizing the amount of points gained across multiple years given constraints on budgeting and more importantly, testing time. This goal is achieved by combining the effects of a cube-shaped hull design and cluster of six Raspberry Pi computers, which has enabled quick modifications and additions such as a sonar array, arm, and battery compartment throughout the vehicle's four year lifespan. The adaptability of the vehicle to additional requirements over the years has led to many improvements at minimal cost while remaining easy to change for future iterations of the design.**

I. Competition Strategy

San Diego Robotics 101 (SDR101) is a very uniquely distributed team during the school year, with members all attending different colleges internationally. This had significant impacts on the way the team strategizes and assigns tasks. A compartmentalized approach was favored to accommodate the different timelines of individual members, and this is reflected in our vehicle designs.

In recent years, SDR101 has focused on improving the functionality of its two vehicles, Cubeception and Minisub, while keeping costs and maintenance effort low. Cubeception, the main vehicle, takes full advantage of its flat surfaces by allocating two faces to portholes for cameras and sonar mounts, two faces for connectors, thrusters, and arms, one face for a quick-access battery compartment, and one face for miscellaneous components such as a pressure sensor, manipulator, and kill-switch. The Raspberry Pi cluster logically splits the processing of incoming sensor data, compartmentalizing both the data and programming for simplified development. The addition of a secondary vehicle (Minisub) with a camera and communication unit onboard further augments the capabilities of the main vehicle with minimal cost and effort.

With the available hardware and testing time, SDR101 determined that focusing on maximizing starting gate, buoy, and marker drop points would be an ideal outcome. SDR101 has relatively limited manpower and funding compared to other AUV teams, so many design and competition strategy choices were made with that in mind, balancing capability and reliability with resource availability. Vision is unique from other software tasks in that it requires much less interaction with the assembled vehicle to test algorithms, so a high priority was placed on tasks relying on vision and movement. This enabled the software team to work in parallel using competition and go-pro footage without needing to be in San Diego with the actual robot up until the in-water tests. Although Cubeception has a SONAR array, it is more difficult to set up the appropriate test environment and allocate the time to test both accurate localization and surfacing, which would be necessary to achieve the surfacing task. On another note, in the competition layout, focusing more on tasks closer to the starting dock increases the amount of attempts a vehicle has to achieve those tasks due to a much lower turnaround time. A failed attempt on the Expose to Sunlight task would have much higher consequences on run time than a failed attempt at 8x gate style points or Slay Vampires. As shown in Fig. 1, the competition strategy is currently the following:

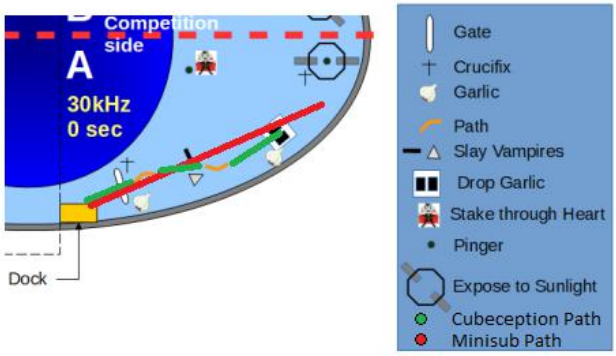

Fig. 1. Vehicle Strategy

- Cubeception is deployed with Minisub attached
- Cubeception carries Minisub through the starting gate while performing all gate tasks
- Cubeception communicates the status of the mission to Minisub resulting in a detaching of the two vehicles
- Cubeception will then perform the Slay Vampires and Drop Garlic tasks while Minisub will gather footage.

It is also important to note that SDR101 has always favored a learning approach, so many mechanical and electrical components are designed and manufactured by hand. Likewise, much of the software for the vehicle is written from scratch.

II. Vehicle Design

A. Mechanical Design

Cubeception's core was originally a singular welded aluminum cube with pylons mounted for attaching thrusters and a transportation rack, but several modifications have been made in recent years to accelerate in-water preparation and testing procedures.

The addition of a separate battery compartment reduces the likelihood of a leak damaging the electronics core by

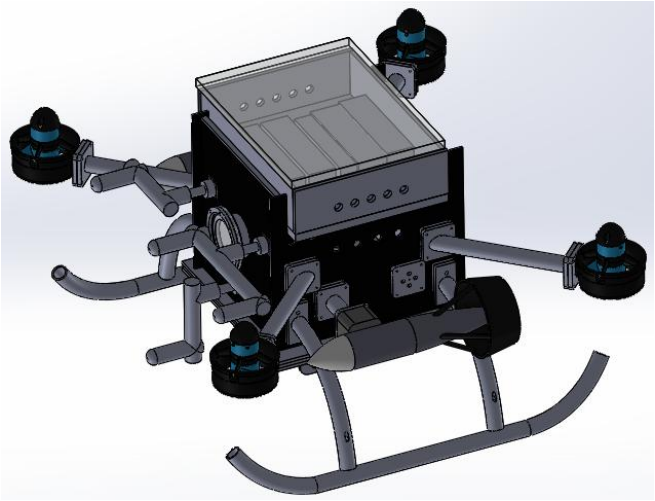

Fig. 2. Cubeception model

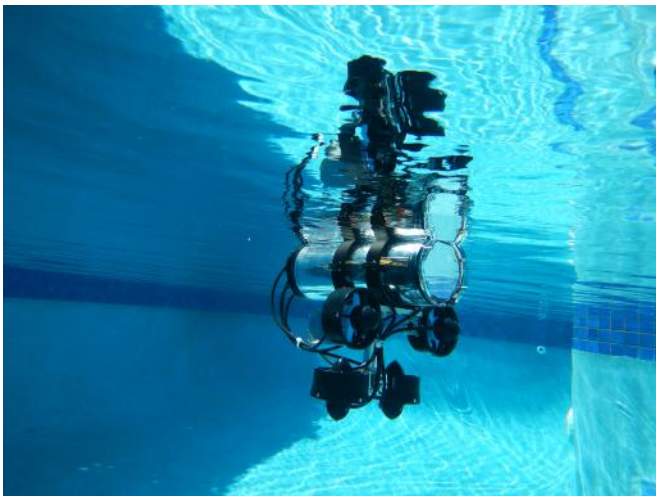

Fig. 3. Minisub in Action

lowering the rate at which the electronics compartment seal is opened and resealed. At the same time, this offers fast access to the batteries via a clamped lid while maintaining a robust seal on the electronics compartment via screws.

Due to difficulties with BlueRobotics ESCs blowing out, custom external housings were created for faster ESC changes.

Most recently, an arm was created to hold pre-loaded markers and release them at the appropriate moment into bins. The arm operates by producing an electro-magnetic field to push a steel pole between the locked and open states, the latter of which will result in the weighted marker detaching and dropping.

Minisub is designed with robustness in mind, with the electronics in a tube at the top of the sub and thrusters at the sides and bottom. Minisub's tube is made of acrylic with an aluminium back cap, while most of the assembly is made of aluminium.

B. Electronics Design

Cubeception's electrical hardware features many custom circuit boards including the motor driver, power distribution board, and FPGA shields for high-throughput sensors like the SONAR array.

To control each motor individually on the vehicle, a custom motor driver board with a Pololu Maestro 12 servo controller generates the appropriate PWM signals for all thrusters, with indicators to show which motors are active. These signals are also opto-isolated, which helps prevent sudden power spikes from affecting other electronics subsystems.

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.

To address the Raspberry Pis 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.

Cubeception 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.

To produce sonar pings and listen for their return, Cubeception 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 systems. 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

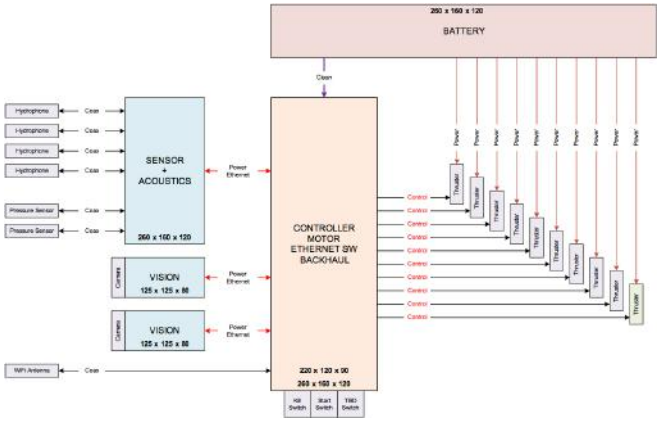

Fig. 4. Cubeception Hardware Connections

listen for responses.

Minisub's electronics feature two custom PCBs to control power distribution and hold digital components such as ESCs and sensors. Powered with a 3 cell 4000mAh LiPo battery, Minisub has enough power to run for 2-4 hours.

C. Software Design

Cubeception's distributed computation model improves the performance of each individual component and promotes high parallelism in its software functionality. Having a cluster of Raspberry Pi 2Bs naturally allows many intensive 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 very logical divisions between responsibilities. Less resources need to be devoted to scheduling processes and ensuring timing accuracy at the cost of increased interprocess communication. Instead of a tall software stack with many processes built upon each other, Cubeceptions stack is wide, with many processes sharing information, but able to proceed with the loss of any individual machine.

Cubeception 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 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

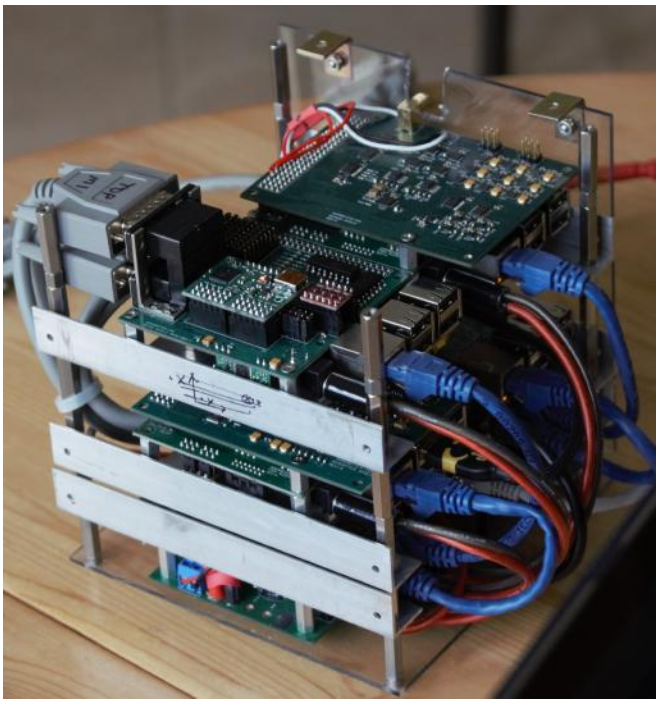

Fig. 5. Electronics Core

of the DSM software by moving the most frequently accessed data into contiguous structures.

Cubeceptions 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.

Cubeceptions logging backend 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 after runs to test new algorithms and find potential issues. This processed data can also be plotted in real time for visualization and error identification.

Cubeception 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

Fig. 6. Minisub Electronics Board

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. To integrate the new physical motor layout on Cubeception, a new set of kinematic equations defining the robots 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 Cubeceptions geometry and motor outputs. This precise physical model, combined with sensor inputs, allows Cubeception to maintain a controllable state-feedback system.

Because the competition environment is translucent, Cubeception and Minisub use vision as a short range localization tool. Both vehicles use a forward-facing camera to detect mission elements in the main axis of movement, while Cubeception features an additional downward-facing camera to detect path markers to follow. Due to the limited processing power of Raspberry Pis, the detection algorithm changes the camera resolution depending on the situation. For navigation, processing higher resolution frames provides sufficient throughput. However, for buoy face recognition, it is necessary to lower the resolution to speed the pipeline up.

Utilizing the sonar board and FPGA shield board, Cubeception is capable of performing both active and passive sonar by profiling the received samples from all 4 hydrophones, filtering out large surfaces, and detecting anomalies which can be matched to objectives by depth. The anomalies in the time domain can be 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 find a pinger or supplement navigation between mission elements.

Communication between Cubeception and Minisub is handled using light signals, which can relay messages including when a certain depth is reached and when to detach. These signals are picked up by Minisub's forward-facing camera and logged for post-run analysis.

On Minisub, a Teensy 3.2 board acts as a motor controller and sensor monitor. It takes short commands as an input through UART, and is capable of moving 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 Raspberry Pi 3B sends commands to the Teensy based on images it takes through the camera unit. Along with controlling Minisub through commands sent to the Teensy, the Raspberry Pi keeps a log of all sensor input sent from the Teensy, along with all the pictures taken. The pictures taken are interpreted using Pythons OpenCV module, which allows for basic color recognition and blob detection.

III. Experimental Results

Cubeception and Minisub have been modified multiple times to improve performance, maintainability, and deployment times based on in-water testing outcomes. The oldest parts on Cubeception have been in service for four years now, but components such as the battery compartment, electronics rack, and software have undergone significant experimentation to determine what worked the best over the years. SDR101 spent the Fall using a newly gathered image dataset from competition runs to test gate and buoy detection algorithms. During Winter and Spring, hardware changes were made to the ESC housings and an arm was added. Since then, the vehicles have been prepared for in-water testing and are currently being tested on the qualifying maneuver, buoy recognition, communication, and marker drop tasks. Some additional simulations were conducted for vision, SONAR, and sensor fusion to determine reasonable resolutions, precision, and sensor drift.

For vision, Cubeception had a tough time keeping the processing pipeline up, sometimes only getting through 1 image every 5 seconds when performing an intensive recognition task. This proved to be problematic when current drift shifted the position of the vehicle off from where vision localized it, or even moved the mission element out of frame. Lowering the resolution of the camera solved a few of the speed issues, and simplifying the algorithms used in detection also cut down on processing times.

Several SONAR simulations have been conducted to determine the precision of our subsystem given a sampling rate and SONAR array separations. By using time of arrival to 4 hydrophones and ellipse triangulation, it is possible to use active sonar to determine a 3D vector representing the position of an object to the vehicle. Passive sonar was simulated on heading detection, where time of arrival was used to determine the heading of the source signal.

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

TABLE I

ANGULAR DRIFT OF STATIONARY INVENSENSE MPU-9150 GYROS OVER 15 MINUTE PERIOD

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. As seen in Table 1, 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 Cubeceptions sensor performance was found.

ACKNOWLEDGEMENTS

SDR101 would like to thank all of its sponsors and supporters for making Cubeception and Minisub 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., Apollo Manufacturing Services, and xQC Engineering Inc. were also integral in the creation of Cubeceptions electronics and core.

APPENDIX

Subjective Measures	Maximum Points	Expected Points	Points Scored
Utility of team website	50	50	
Technical Merit	150	150	
Written Style	50	50	
Capability for Autonomous Behavior	100	100	
Creativity in System Design	100	100	
Team Uniform	10	10	
Team Video	50	50	
Pre-Qualifying Video	100		
Discretionary Points	40	40	
Total	650	550	
Performance Measures	Maximum Points	Expected Points	Points Scored
Weight	128.5 / vehicle	123.5 + 68	
Overweight/Size Penalty	-500 / marker		
Gate: Pass Through	100	100	
Gate: Maintain Fixed Heading	150	150	
Gate: Coin Flip	300	300	
Gate: Pass Through 60% Section	200		
Gate: Pass Through 40% Section	400	400	
Gate: Style	100 (8x max)	800	
Collect Pickup: Crucifix, Garlic	400 / object		
Follow the ”Path”	100 / segment	200	
Slay Vampires: Any,Called	300, 600	300	
Drop Garlic: Open, Closed	700, 1000 / marker	700	
Drop Garlic: Move Arm	400		
Stake Through Heart: Open Oval, Cover Oval, Sm Heart	800, 1000, 1200 / torpedo		
Stake Through Heart: Move Lever	400		
Stake Through Heart: Bonus - Cover Oval, Sm Heart	500		
Expose to Sunlight: Surface in Area	1000		
Expose to Sunlight: Surface with Object	400 / object		
Expose to Sunlight: Open Coffin	400		
Expose to Sunlight: Drop Pickup	200 / object		
Random Pinger First Task	500		
Random Pinger Second Task	1500		
Inter-vehicle Communication	1000	1000	
Finish mission with T minutes	Tx100		
Total		4141.5	

TABLE II

POINT EXPECTATIONS

Component	Vendor	Model/Type	Specs	Cost
Buoyancy Control				
Frame	SDR101	Custom		
Waterproof Housing	SDR101	Custom		
Waterproof Connectors	SDR101	Custom		
Thrusters	BlueRobotics, VideoRay	T-100, Pro 4 Brushless	5lbs, 20lbs thrust	$600
Motor Control	RISE	ESC	0.5A/5V Linear BEC	
High Level Control	SDR101	Custom		
Actuators	SDR101	Custom		
Propellers	BlueRobotics, VideoRay			
Battery	Floureon	Li-po RC Battery	11.1V, 5500mAh, 35C	$200
Converter				
Regulator				
CPU	Raspberry Pi Foundation	6x Raspberry Pi 2B	CPU: 4x Cortex-A53 1.2GHz	$210
Internal Comm Network		Ethernet		
External Comm Interface		Wi-Fi		
Programming Language 1		Python 3		
Programming Language 2		C++		
Compass	STMicroelectronics	LIS3MDL		
Inertial Measurement Unit (IMU)	SDR101	Custom		
Doppler Velocity Log (DVL)				
Camera(s)	Raspberry Pi Foundation	Raspberry Pi Camera Module v2	8MP, 1080p30, 720p60	$30
Hydrophones	Aquarian Audio	H2A		
Manipulator	SDR101	Custom		
Algorithms: Vision	OpenCV	OpenCV 3		
Algorithms: Acoustics	SDR101	Time of Arrival		
Algorithms: Localization and Mapping	SDR101	IMU-based, Vision-based		
Algorithms: Autonomy	SDR101	State machine		
Open Source Software				
Team Size	8			
HW/SW Expertise Ratio	1:3			
Testing Time: Simulation	50 hours			
Testing Time: In-water	10 hours			

TABLE III
COMPONENT SPECIFICATIONS