Release the Kraken 3.0

San Diego City Robotics – Autonomous Underwater Vehicle 2015

Design Synopsis

Robert Pruitt (Faculty Advisor/Electrical Design Lead), **Patricia Shanahan** (Programming Advisor), **John Calderwood** (Electronics Advisor), **Jason Plojo** (Team Leader), **Shubhankar Agarwal** (Refuel Mission Lead), **Nick Cantrell** (SimBox Developer and OpenCV Filter Design), **Ed Eberle** (Webmaster), **Darik Houseknecht** (Audio Video Technician), **Luis Jarquin** (Electrical Integration Lead), **Matthew Pedler** (OpenCV Development Lead), **Carlos Mendoza** (Mechanical Lead), **Eric Mudd** (Technical Writer), **Nils Olsson** (Linux Applications Engineer)

The Kraken 3.0 is an AUV (Autonomous Underwater Vehicle) developed by San Diego City Robotics (SDCR) with extreme cost constraints in mind. The Kraken's SONAR subsystem also had improved programming and circuit changes. Since this was the second iteration of the Kraken 2.0 hull design, SDCR decided to use the same chassis from last year in order to reduce cost and increase time availability. The main strength of this year's vehicle entry is a new cross-platform (web browser-based) simulation environment, "SimBox," which allowed for the software team to begin vision testing in the beginning of 2015.

0 INTRODUCTION

SDCR (San Diego City Robotics) is a group composed of students, professionals, and faculty. SDCR currently is based in San Diego City College (SDCC), a junior college. The current SDCR team is divided into three groups; a mechanical build team, an electronics hardware team, and a software development team. For 2015, the decision was made to focus team efforts on process improvements which would enhance productivity in the years going forward. Infrastructure-building activities, which would reduce the learning curve for new team members, were made a top priority in order to maximize return on investment over time. By investing in people, this strategy should pay dividends as time goes on.

1 MECHANICAL BUILD TEAM

In order to justify the investments made to the 2014 Vehicle, Kraken 2.0, SDCR made the decision as a team to limit the mechanical changes made to the vehicle this year. Interested parties were invited to propose and develop alternative designs that would offer superior performance. Since no one was able to make improvements from the status-quo, the decision to reuse the 2014 vehicle was an easy one to make.

1.1 Component Development

The main components of the Kraken 3.0 were the same from the 2.0 model with the exception of the propulsion modules and waterproof connectors. In order to minimize risk, very few new components or mechanical systems were developed in 2015. A point-based analysis was conducted of the “difficulty-adjusted ROI” for all available options. Of those choices, the “Object Manipulation” and “Marker Bin” tasks appeared to be the most attractive candidates for the mechanical build team to develop new capabilities for the sub. Targeted changes allowed the team to focus more of its attention towards software and sonar development. These changes included custom “Object Manipulation Forks”, “Marker Bin Droppers”, and the purchase of industrial-grade waterproof connectors.

1.1.1 Object Manipulation Forks

To accomplish the tasks relating to object manipulation, vehicle “end-effectors” were designed and fabricated. To ease electrical integration, these so-called “Forks” were modelled after Forklift blades with simplicity in mind. The two aluminum blades were designed as a collaborative effort between Nicholas Cantrell and Shubhankar Agarwal. The fabrication was done by Shubhankar Agarwal.

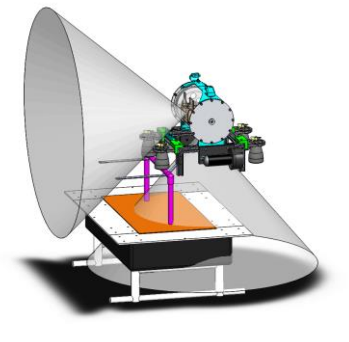

Fig. 1 Object Manipulation Forks

1.1.2 Marker Bin Droppers

The Marker Bin Droppers were designed as a collaboration between Shubhankar Agarwal, and John Calderwood. In order to conserve energy, the markers are passively held in position by a permanent magnet until activation. Once the solenoid is energized, the magnetic plunger is retracted allowing the Markers to fall from the vehicle. By minimizing the weight of the markers, the impact to vehicle buoyancy is negligible. Actual control of the solenoid is achieved through a two-channel Solid-State Relay (SSR) module with integrated OMRON® G3MB-202P optocoupler isolated relays.

1.1.3 Waterproof Connectors

For 2015, the San Diego City College District generously purchased the SDCR team a complete set of Teledyne Impulse “XSJ” Glass Reinforced Epoxy Series Dry-Mate Connectors. These connectors are rated for 20,000 PSI (approximately 1350 bar) in their mated condition, and can withstand 5,000 PSI (approximately 340 bar) of open-faced pressure. Consistent with this year’s goal of making investments in the future, these components will be of great-value to future teams.

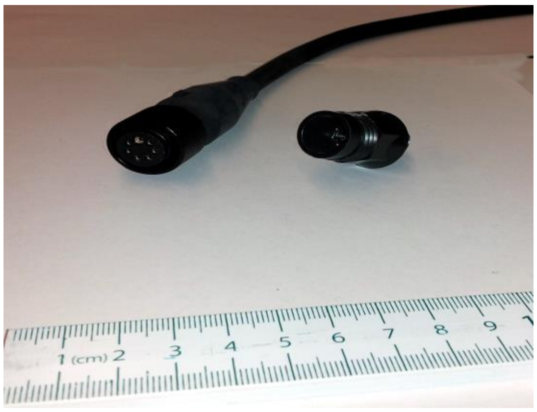

Fig. 2 Teledyne waterproof connector

1.1.4 Pressure Housing

The pressure housing is composed of a .25”(6.35mm) wall thickness 6.47”(164.34mm) ID Cast Acrylic tube (McMaster #: 8486K591). Two aluminum end-caps (6061 Aluminum) were machined in 2014 by SDCR team members Nicholas Cantrell and Oscar Aguilar. The rigidity of the aluminum faces resisted buoyancy changes under pressure and provided a flat smooth surface for the underwater connectors to seal against.

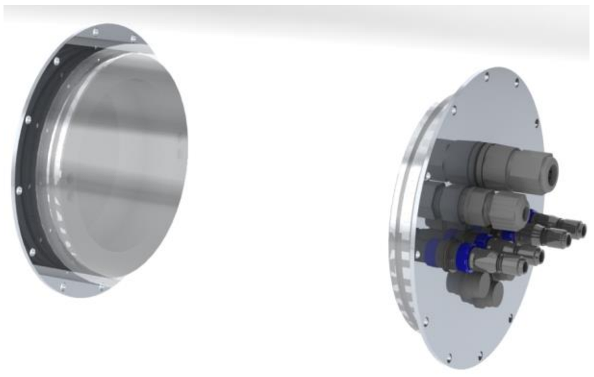

Fig. 3 Pressure Housing Model

1.1.5 Propulsion

The four dive motors are Marine Rule 27D 1100 bilge pumps. These pumps have a displacement rating of 1100 gallons per hour. As packaged, the motors had a hydro-dynamically, unfavorable shape that imposed drag on the vehicle; adversely affecting its performance. The motors were modified into a conical shape by applying Bondo. Once done, a mount was attached with epoxy to the top of the motor which could be easily installed into a machined bracket. These surface/dive thrusters are also used to control pitch and roll for the Kraken.

The drive motors are Seabotix BTD-150 thrusters. These are used to control yaw and propel the Kraken forward or backward. These were a natural choice for SDCR as they are inherently waterproof and are made for AUV applications.

1.2 Mechanically Integrated Vehicle

In 2014, a meticulous and deliberate design process was followed to ensure that the finished vehicle would be up to the competition challenges faced by SDCR. A high level requirements spreadsheet helped the team generate a parts list, estimate buoyancy, and size batteries. Multiple Finite Element Analysis (FEA) studies conducted by the design team supported the belief that the pressure housing would be sufficient to withstand the pressures met by operating at the depth requirements decided in advance of component purchases.

This approach was highly successful and resulted in an extremely stable vehicle in pitch and roll which required very few buoyancy modifications when placed in the water for the first time. Yaw stability was identified to be a problem without a vertical stabilizer fin, so a “tail” was added at the 2014 Robosub competition to supplement the MPU6050 AHRS and BTD-150 Seabotix thrusters in navigating the vehicle through the qualification gate.

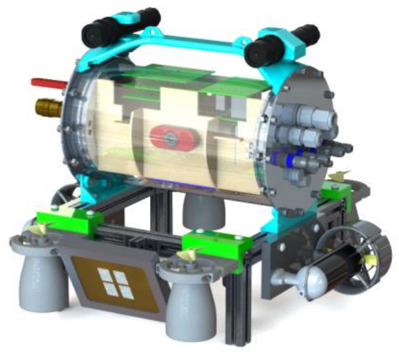

Fig. 4 Solidworks Model of chassis

2 ELECTRONICS HARDWARE TEAM

The Electronics Hardware Team was divided into three subgroups: (1) an electrical design team that handled specific circuitry and power for the different accessories for the sub; (2) a computing team that handled the motherboard and its accessories; and (3) the sensors team.

2.1 Electrical Design

During the process of creating the “requirements tracking matrix”, electronics components were assigned electrical properties such as voltage, current, and power requirement in watts. This was a valuable asset during the process of designing the electrical and mechanical systems as it allowed for the batteries to be appropriately matched in capacity, voltage, and max discharge power.

Components powered from a 12VDC or 5VDC input were preferred because of the local availability of supplies such as fuses, relays, and DCDC converters in these voltages.

2.2 Computers

The 2013 Kraken design used a micro-ATX motherboard with an Intel processor. When the vehicle took on water, the motherboard and CPU were both lost. Although this equipment damage was unfortunate, the tragedy became an opportunity to select a new mainboard. This freedom allowed for the vehicle's displacement to be greatly reduced by selecting a smaller footprint component. Ultimately the board selected was an IFC6410 Pico-ITX Single-Board Computer (SBC) from the Inforce Computing ® Corporation.

2.2.1 Main Computer – Inforce Computing ® - IFC6410

This Snapdragon™ based board was generously donated to SDCR. It features an ARM® architecture Qualcomm® Snapdragon™ S4 Pro APQ8064 quad-core processor clocked at 1.7GHz, as well as an Adreno 320 GPU. The smaller Pico-ITX form factor allowed for a reduction in pressure housing displacement. It contains 2GB of on-board DDR3 RAM and a 4GB eMMC (embedded MultiMedia Card) drive for storage. It also includes a MicroSD card connector that team uses for the storing of the Kraken's software logs.

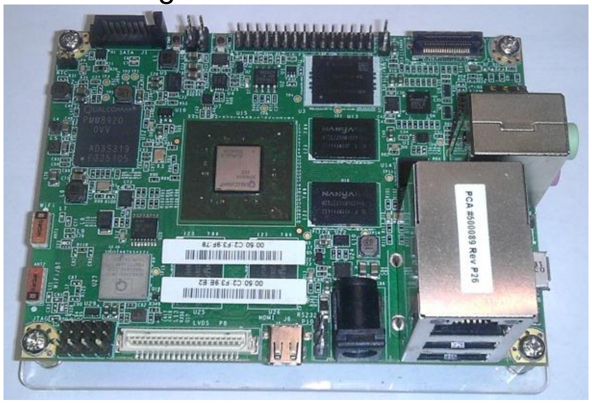

Fig. 5 IFC6410 - DragonBoard

2.2.2 Arduino Mega 2560

The standard Arduino Mega used in 2011 was upgraded to an Arduino Mega 2560 microcontroller. It is based on the Atmel® ATMega2560 microcontroller. The Arduino Mega doubles the Kraken's flash memory. The Arduino reads the current heading and depth information provided by the IMU and pressure sensor. It also receives data from the Kraken's sonar subsystem via the Two-Wire Serial Interface (TWSI, the Atmel version of the I^2C bus). From that data, the Arduino (via NavBox) then determines the appropriate speed, heading, and depth of the Kraken; then sends PWM signals to the motor controllers.

2.3 Sensors

The Kraken achieves a "high level of autonomy" through accurate sensor data. All the Kraken's sensors were selected with both quality and ease of integration in mind. As a result an Invensense® MPU-6050™ based Attitude and Heading Reference System (AHRS) was selected as a low-drift sensor for tracking vehicle attitude. For ease of integration, a differential analog pressure transducer (PX181B-030G) from OMEGA Engineering Inc. was selected to measure vehicle depth.

2.3.1 IMU (Inertial Measurement Unit)

The process of maintaining an accurate position in a given space is a prerequisite for much of the sub's activities during the course of the competition. This year the team decided to use the MPU-6050™ based "Triple Axis Accelerometer and Gyro Breakout" from SparkFun Electronics®. It features MotionFusion™ technology which uses an on-board processor to provide sensor fusion calculations from the integrated three-axis gyroscope, and three-axis accelerometer. This sensor provides the heading estimate to NavBox.

2.3.2 Camera

The USB Cameras were selected in 2014 to maximize compatibility with Linux powered ARM SBCs such as the IFC6410 and Raspberry Pi. One of the few USB webcams with Linux drivers available known to be compatible with the original Raspberry Pi was the Logitech C210. The Logitech C210 has a published Video Resolution of 640 x 480 and a reported current draw of 170mA. This low image resolution helps with faster vision processing and the minimal current draw allows the camera to be powered from a wide range of ARM SBCs. All of these attributes made it our preferred choice for ease of development.

2.3.3 Sonar

In conjunction with Professor Pruitt, SDCR was able develop a viable sonar option. The Hydrophone Array consists of 4 custom fabricated piezoelectric elements that are potted and mounted into an anodized aluminum adjustable scan angle frame.

The design challenges we faced included signal level variation with location, unwanted frequencies, and the mathematical complexity of converting arrival times to pinger heading angles. One problem was the variation in received signal level depending on the Kraken's location in the pool. This was solved by using a high gain operational amplifier circuit which outputs a constant voltage square wave.

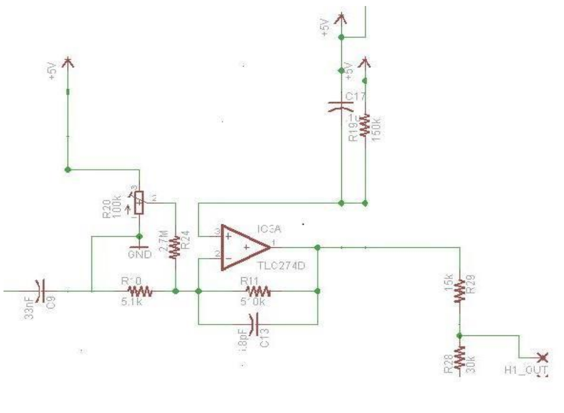

Fig. 1 High Gain Op Amp

There was a need to discriminate against frequencies outside of the desired frequency range. The fixed-level square wave signals are sent to a programmable 4th order (MAX7490) switched capacitor filter, leaving only a sine wave of constant level. The center frequency of each MAX7490 chip is controlled by a Direct Digital Synthesizer (DDS) generated clock, which allows on-the-fly frequency selection in the 22 KHz to 40 KHz range. Because distortion was observed at the filter output, a voltage divider was then added between the amplifier and the filter, resulting in a cleaner output from the filter circuit. The final analog processing step was to delay one side of each signal pair using an all-pass phase shifter circuit.

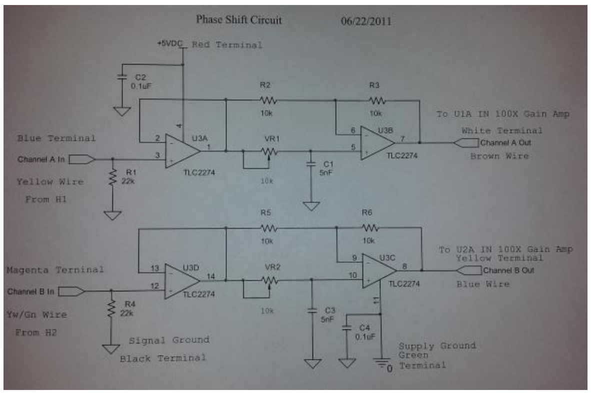

Fig. 2 Phase Shifter circuit

The sonar microcontroller, an ATMega16, counts the number of crystal clock pulses between the two arriving data streams and converts the difference into distance, and then pinger angle. A counting timer in the code is set to only store a measurement if there has been silence for more than 1 second to keep the sub from chasing echoes coming from the wrong direction. This microcontroller is on a custom PCB that was designed by Robert Pruitt, Keith Dwyer, and Frank Alfaro.

Several problems arose in the mathematical calculations necessary to convert the timer counts to information that was useful to the commander module. The original microcontroller clock was chosen to be 4 MHz for stability, but the time it took the code to measure the clock counts was too close to the actual time difference being measured for small degree values to the far left. By quadrupling the clock frequency to 16 MHz, usable horizontal and vertical resolution was obtained.

In the original plan, a fixed lookup table at the median pinger frequency of 26 kHz was used to speed up calculations. This year the electronics team reviewed how the fixed frequency lookup table performed. They found out that it produced a margin of error of 6°. The group decided that was too much and changed the way the lookup table was produced. They allowed the microprocessor to do the mathematical calculations for the specific pinger frequency and placed generic arcsine values in the lookup table. The TWSI bus on the microcontroller accepts the daily pinger value from the commander module. This then produced a margin of error of only 0.5°. The sonar is much more accurate now.

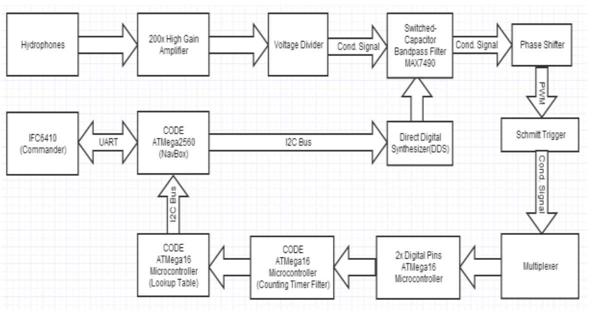
Hydrophones
200x High Gain Amplifier
Voltage Divider
Cond. Signal
Switched-Capacitor Bandpass Filter MAX7490
Cond. Signal
Phase Shifter
PWM
IFC6410 (Commander)
UART
CODE ATMega2560 (NavBox)
I2C Bus
Direct Digital Synthesizer(DDS)
Schmitt Trigger
Cond. Signal
I2C Bus
CODE ATMega16 Microcontroller (Lookup Table)
CODE ATMega16 Microcontroller (Counting Timer Filter)
2x Digital Pins ATMega16 Microcontroller
Multiplexer

Fig. 6 Sonar Signal Flow

The arcsine is calculated and converted to signed pinger angles via the lookup table and then the data is sent physically via the TWSI to the Arduino Mega 2560. NavBox then instructs the motors on how to steer.

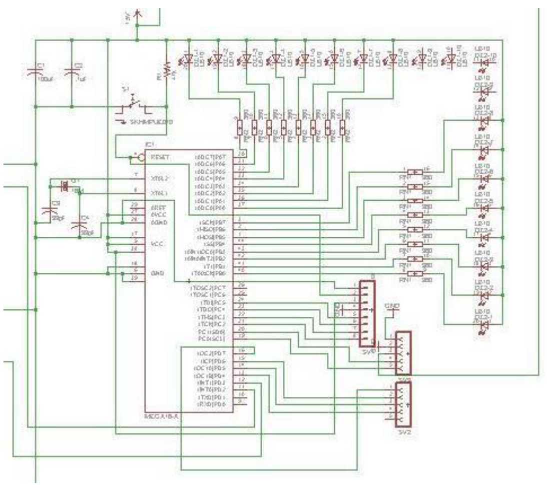

Fig. 7 Microprocessor/LED circuit

3 SOFTWARE DEVELOPMENT TEAM

The goal of the computing system was to be small, require little power and eventually allow the Kraken the capability to attempt every competition task. Based on the aforementioned requirements, the new system consisted of an IFC6410 computer, an Arduino Mega 2560, and two cameras. All software is built upon a Linux operating system and written in a mixture of C (which is used for the microcontroller), C++ (used for the main application), and Assembly (used for the Custom Sonar Board). This year's software is based upon the success of previous systems. The structure features vision processing, control systems, and a testing and debugging scheme.

The application coded by the software development team consists of 4 major parts: (1) the Dashboard, a graphical user interface (GUI); (2) the Commander, the overseer of the Krakens subsystems;(3) the Navigation Box (NavBox), movement control code for the Kraken that gets sent to the Arduino; and (4) the TRANSDEC Simulator (SimBox).

3.1 Dashboard

Dashboard is a GUI developed solely for testing, debugging, and maintenance.

It features a manual override that assists the team in the debugging process. It enables real-time data viewing and remote control of the sub while submerged via an ethernet cable tether. A server running on the Commander module transmits data (such as heading, depth, and thruster duty-cycle) to the client machine running Dashboard. Upon receiving the information, Dashboard displays and logs the data, with the ability to replay the session at a later time. Dashboard can be disabled for complete vehicle autonomy, independent of operator control.

3.2 Commander

Commander consists of 3 major subsystems: (1) Mission control, which determines what task/mission is current; (2) Communication (CommLink), which interfaces with the Arduino Mega 2560 and (3) Vision, which is used to process the vision input from the cameras.

3.2.1 Mission Control

Mission Control is the name given to the subcomponent of Commander that runs on the IFC6410 and is the mission/task logic software. Decisions on which task to attempt is based on time out slots. It makes use of the information generated by the vision code and sonar to make decisions about specific target headings. Once the Mission Control has designated what mission/task to run, and how to run it, the Mission Control relays the mission specifications to the Navigation Box. From that, NavBox will turn on certain motors to move the Kraken to appropriate positions.

In previous models of the Mission Control software, the logic relied heavily on several “if”, “then”, “else” statements; so a more organized format was needed. The fix was a more efficient version written through the use of the idea of a “Finite State Machine.” This programming model allows each decision situation to have its own logic and will then transition to another situation when a certain condition is met.

3.2.2 Communication (CommLink)

The Communication interface, or “CommLink,” is a second subsystem of the Commander software. This leg of Commander uses the MaiaXmlRpcServer library and allows any accessory attached to the Arduino to communicate with the rest of the system. Data can both be received and sent to the Arduino via CommLink and a USB port on the IFC6410. The Kraken’s sonar subsystem uses CommLink for communication to and from the hydrophones.

3.2.3 Vision

The “Vision” processing system is based on OpenCV; making extensive use of its existing libraries, but also includes custom code written by the software team. The “Vision” module algorithm begins by pre-processing image frames using a histogram equalization in YCrCb space, a Gaussian blur filter and basic HSV thresholding. The resulting binary image is fed to a canny edge detector producing the necessary contours allowing for contour approximation, convexity checks, structural analysis and shape detectors.

3.3 Navigation Box (NavBox)

NavBox is the motor driver and sensor controller which runs directly on the Arduino. When given a target depth and heading by the Commander, it reads in the data from the IMU and pressure sensor. NavBox then makes the necessary adjustments to the Kraken’s six motors via a Pulse Width Modulation (PWM) signal to reach the desired depth and heading. Commander also relays speed constraints to NavBox for backwards and forward movement.

3.4 Transdec Simulator (SimBox)

A recurring problem faced by SDCR has been the balance of distributing programming workload throughout the course of a year. In order to enable software testing without the prerequisite of a mechanically complete vehicle, a simulation environment was envisioned which would allow for continuous computer vision and vehicle controls development. This workflow would enable the software team to validate their work while the mechanical build was still underway.

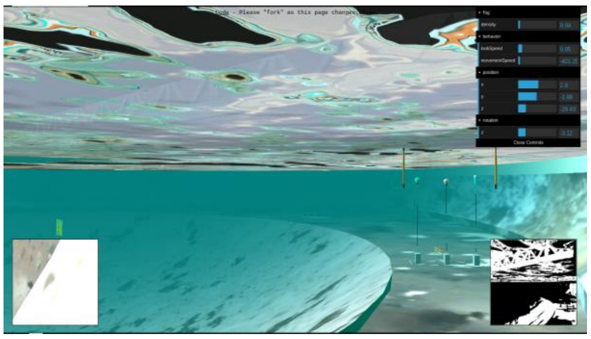
Fig. 8 SimBox screen capture

3.4.1 SimBox Description

SimBox is a tool that allows a navigable 3D environment to be embedded in the user's web browser. In this environment: lighting conditions, physics, and textures can all be controlled via a well-documented JavaScript API (threejs.org/docs/) to achieve a more accurate simulation. Through the "looking glass" in the bottom right-side corner of the screen, SimBox allows an experience where the developers can view the environment from the perspective of the vehicle in order to enhance their understanding of the problems faced during autonomous decision making.

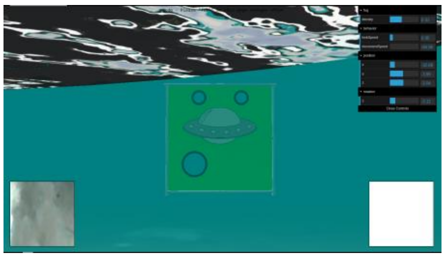
Fig. 9 SimBox screen capture – "Torpedo" task

In order to better support a diverse variety of possible end-user hardware, a web browser was considered to be the most universal means of making this resource available to engineering students. This desire to minimize barriers to entry suggested that WebGL should become the core technology "under the hood". To ease development, a JavaScript library called, "three.js" (threejs.org) was selected. This decision was motivated by the large supply of published examples in addition to the existence of a thriving development community.

3.4.2 SimBox Benefits

Unlike recorded video inputs or image "stills," SimBox allows the vehicle to interact with its simulated environment as a form of "Software In Loop" (SIL) controls development. This phase of testing borrows from a "Model-Based Design" (MBD) methodology and is intended to make computer vision unit-tests as meaningful as possible (prior to actual testing of a fully integrated vehicle). This enables vision code testing which goes beyond passive object tracking or identification and actively adjusts the vehicle orientation to keep the region of interest centered in the field of view.

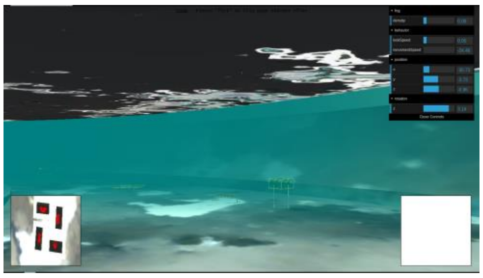
Fig. 10 SimBox screen capture – "Marker Bins" task

This testing can be conducted under a variety of conditions because SimBox allows for a high degree of control over environmental variables such as: lighting, water clarity, currents, and surface behavior. To accommodate the large number of possible permutations for these input parameters, toggle buttons and keyboard shortcuts have been included which allow for “input fuzzing” with random numbers. This ability to “get lost” on demand, combined with the optional variation of signal-to-noise-ratio (SNR), enhances the robustness of the controls code by improving tolerance against bad state estimation (while measuring the acceptable SNR range). The benefits of this workflow extend back into the earlier stages of development; to include procedural generation of large data sets for machine learning (caffe.berkeleyvision.org/) while providing “ground truth” known-values for quantitative analysis.

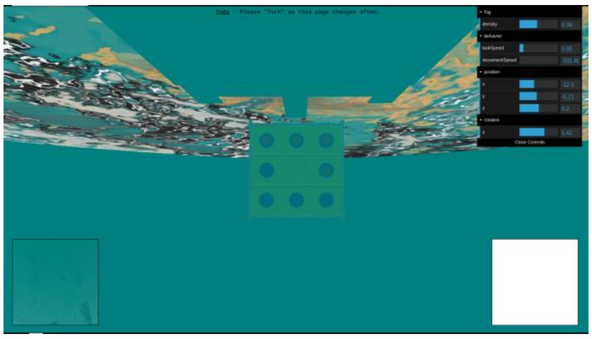
Fig. 11 SimBox screen capture – "Manipulation" task

3.4.3 SimBox Model Generation

All 3D modelling, and texture mapping, was done in SolidWorks with only minor adjustments required in Blender to the resulting .Obj and .Mtl files. Modifications to the environment can be achieved through a simple workflow which leverages industry standard software. For simple behavior or environmental changes, slider bars have been provided. If more complex modification or extensibility is desired, only a basic knowledge of JavaScript is required to modify the “index.html” file which serves as the configuration interface for the supporting libraries.

4 CONCLUSION

This year SDCR set out to further integrate OpenCV with our vehicle autonomy code. For the sake of limited resources, avoiding unnecessary rework was a priority. Multiple long-term infrastructure investments were made to help ensure that 2015-2016 is a more productive year. This paper covered how some of those changes defined the "Kraken 3.0" in contrast from our previous vehicle, as well as what changes we have made to our own processes as a team to make us more successful in the future.

5 ACKNOWLEDGMENT

SDCR is made possible thanks to the generous support of our sponsors: San Diego City College and the San Diego City College Associated Students Government (ASG). SDCR would like to specifically extend their gratitude to the Dean of the Engineering Technologies, Mathematics, Sciences, and Nursing departments.

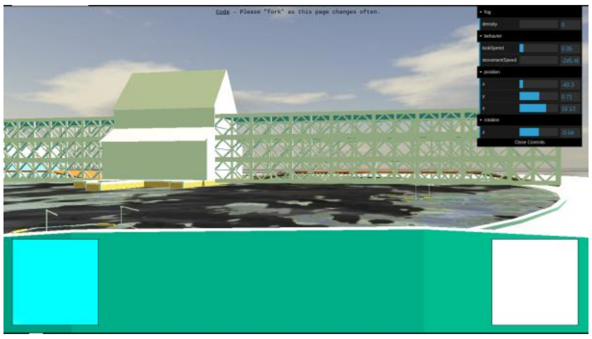
Fig. 12 SimBox screen capture "wear sunscreen" task