Next Year
Embry-Riddle Aeronautical University
Autonomous Underwater Vehicle

http://www.roboticsassociation.org/RoboSub.html

Team Leader:

Donald Patrick Bennett Jr.

Team Members:

Robert Goring, Evan Williams, Martin Lehmeier

Faculty Advisors:

Dr. Charles Reinholtz
Professor and Chair - Mechanical Engineering Department

Dr. Brian Butka
Professor - Electrical, Computer, Software, and Systems Engineering Department
&
Dr. Timothy A. Wilson
Professor and Chair - Electrical, Computer, Software, and Systems Engineering Department

Embry Riddle Aeronautical University
600 S. Clyde Morris Blvd.
Daytona Beach, FL 32114

Abstract
The Robotics Association at Embry-Riddle (RAER) is proud to present its entry, Next Year, for the eighteenth Annual International RoboSub Competition. The year's RoboSub team consists of two veteran, and two new members with mentorship being provided by students and faculty members who are well versed in the unmanned and autonomous systems regime. The team working on Next Year consists of undergraduate students representing different majors from the College of Engineering at Embry-Riddle. The RoboSub competition consists of several tasks, which require navigation, image processing, and object manipulation. These tasks model the applications of current commercially available AUVs. The Embry-Riddle RoboSub team has continued developing on the platform designed for last year's competition.

1. Introduction
The Association of Unmanned Vehicle Systems International (AUVSI) and Office of Naval Research (ONR) are hosting the eighteenth annual international RoboSub Competition to be held in San Diego, California at the SPAWAR Transdec facility. The goal of this competition is to create a vehicle that can autonomously complete a set of tasks, which are mimic the abilities of current autonomous vehicles in industry. To create a vehicle and system capable of accomplishing all of these tasks requires years of development and testing. Recognizing the concept of complexity through iteration, the RAER RoboSub team has devised the goal of creating a simple system that can perform the required tasks, with room for future modifications.

The Next Year System is based on the BlackFin system developed for last year's completion. New deployment to the system includes a new camera system, new main computer, updated software, and an improved power supply subsystem.

2. Hardware and Mechanics
The electronics, computer, and the majority of our sensors are housed in a Pelican case mounted to the top of the vehicle. This packaging allows for easy transport and maneuverability of the vehicle.

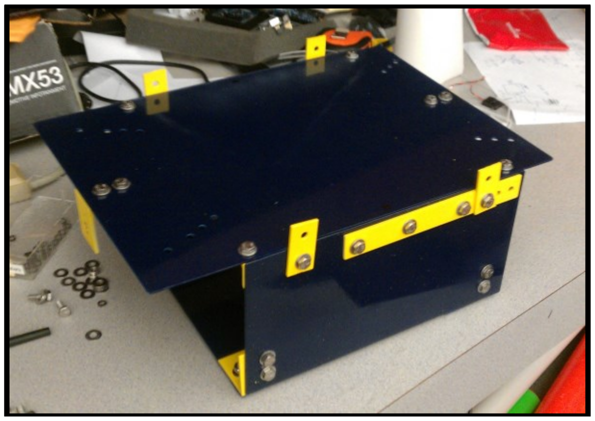


Figure 1 - Frame without the main hull

2.1 Frame / Structure
With the exception of the addition of external camera housings, the structure developed last year will continue to be used. The structure was developed to be lightweight, compact and durable. This frame is composed of sixteenth-inch plate aluminum and one-eighth-inch right angle extruded structure for support and motor mounts. A Pelican Storm IM2200 case is used as the waterproof hull. This case was selected due to its durability, optimum size, and low cost. The hull has in interior volume of 15,000cm^3 (950 in$^{3)}$ equating to a buoyancy of 15 kg in fresh water.

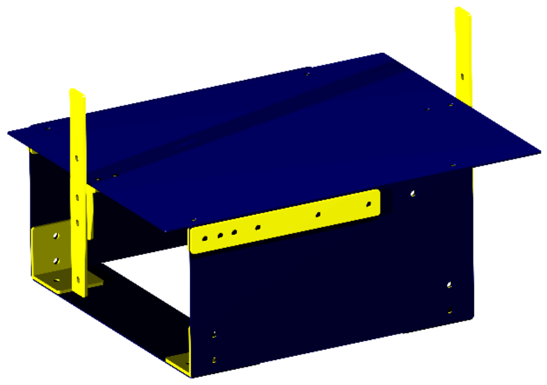


Figure 2 - CAD Rendering of Frame

The vehicle is designed to be statically stable in order to simplify maneuvers in the water. The center of buoyancy was placed above the center of mass. This gives the vehicle the capability of being self-righting from various orientations. Due to the lightweight nature of aluminum, and a large buoyant hull, the vehicle is intrinsically positively buoyant. In order to increase the efficiency of the system, weights were added to the vehicle to bring the buoyancy to one half of a percent of the vehicle's mass.

2.2 Internal Layout

All components for the system with the exception of the thrusters, and camera are housed within the main hull.

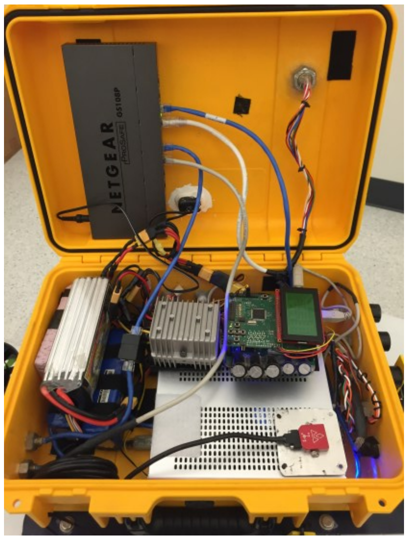


Figure 3 - Vehicle Internal Layout

The Ethernet Cameras are housed in a separate metal housing.

2.3 Thrusters

Four SeaBotix BTD150 thrusters are used to maneuver the platform. Each thruster provides 22N of continual thrust. Two thrusters provide vertical thrust, while two are used for horizontal movement. This configuration provides four degrees of freedom: translation (horizontal and vertical), yaw, and pitch. This setup was determined to have the optimal level of maneuvering freedom to perform most tasks. Plans are in place to upgrade the system to add two more thrusters. By having additional thrusters in the lateral direction the vehicle could holonomically move with five degrees of freedom. This capability would assist with the completion of several tasks.

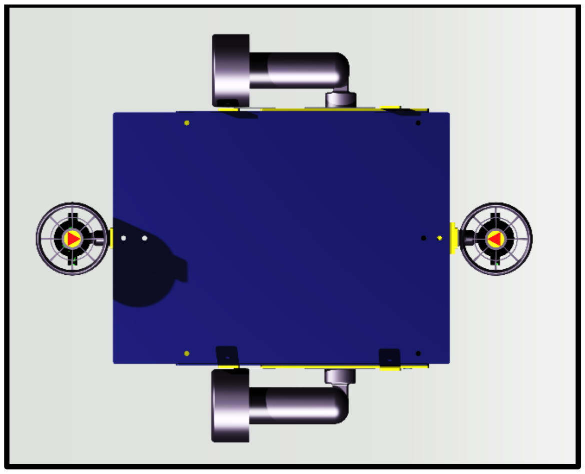


Figure 4 - Thruster Positions

Current motor configuration allows for simplified control algorithms. In this setup, motors are placed symmetrical to each other to prevent unexpected movements. To achieve reliable motion in the horizontal plane, only the side thrusters need to be balanced.

2.4 Connectors

In-line bulkhead connectors from SubConn are used as the primary connector system. Each thruster requires two pins, power and ground, so the motors are paired together and use four pins of a five-pin connector. This allows for thrusters to easily be swapped out if need for replacement or repair. Each camera requires a Gigabit PoE connection. Two eight-pin connectors are used to provide a waterproof as well as a detachable solution. A third eight-pin connector is used to provide an Ethernet tether to the vehicle. All of the connectors used are rated for 150 meters.

Amphenol cable glands are used to provide a waterproof solution to pass the Ethernet cable into the camera housing. These IP 68 rated glands were selected due to their durability, as well as their ability to reduce interior housing space.

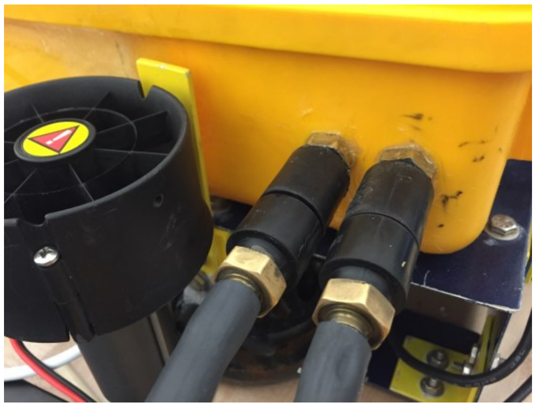


Figure 5 - External SubConn Connectors

2.5 Camera Housing

Two matching custom housings are used to encase the camera systems. These housings were manufactured by ¼" thick aluminum tubing, as well as ¼" acrylic. A custom 3D printed mount has been created to hold the camera in place. The cameras are mounted so that one faces forward, while the other faces straight down. This allows for the system to visually inspect objects in front of the vehicle as well as those below.

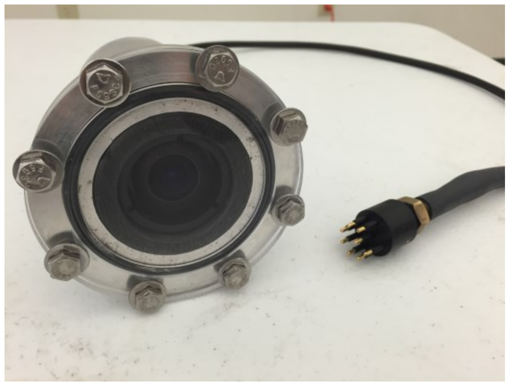


Figure 6 - Camera Housing

2.6 Current Development

At the time of the Journal Paper submission, Next Year does not yet have systems for completing all of the tasks. Currently, the droppers and torpedo launchers are being developed. These systems are anticipated to be developed in time for this year's competition.

The dropper system in development consists of small 3D-printed torpedoes that are released by a servo. These 3D-printed torpedoes are hollow so that they can be weighted to fall straight.

The torpedo system consists of Toypedos that are spring launched and released by a servo. Having them spring launched ensures that they are released at a safe speed and won't cause injury to the pool or diver.

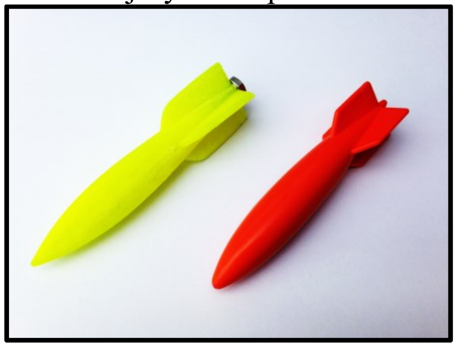


Figure 7 - Torpedoes for Dropping (left) and Firing (right)

3. Electronics

Next Year's electronic system consists of purchased, donated, and developed in-house components. The electronics package was designed to be compact and modular. For added protection to the computer, a compact Mini ITX case was installed. This ensures all components are safe as well as providing strong airflow. This case also functions as an easily accessible area to mount the majority of the electronics. These electronics are mounted to the top of the computer case seen in Figure 8 using Velcro.

3.1 Computer

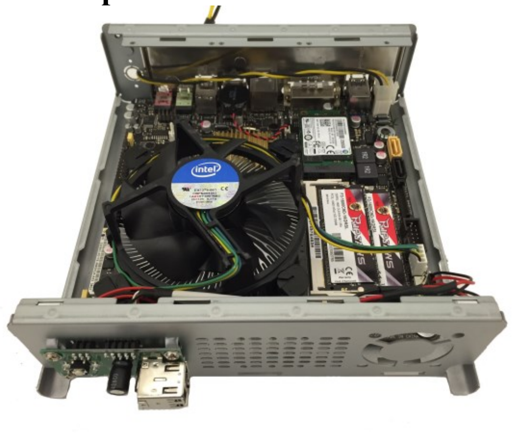


Figure 8 – Computer case with Cover Removed

The computer consists of entirely off-the-shelf components. The computer specifications are seen below in Table 1.

Table 1 - Computer Specifications

Motherboard	Asus H81T/CSM
Processor	Intel Core i7-4790S @ 3.2GHz
Memory	16GB DDR3
Storage	250 GB Solid State
Power Supply	Integrated 19V DC Input

3.2 System Voltage Converters

The power supply is given by two separate systems: a 6 cell Lithium-Polymer battery which outputs 25 volts, and a 5 cell Lithium-Polymer battery which outputs 21V. The first of these batteries supplies power to various subsystems of differing voltages, for which the voltage converters are listed below.

The latter of the two batteries is used exclusively to power the thrusters, which carries the benefit of isolating the innately noisy motors and their controller board.

Table 2 – Voltage Converters

24VDC	19V DC (Motherboard input)
24VDC	48V DC (PoE capable Switch)

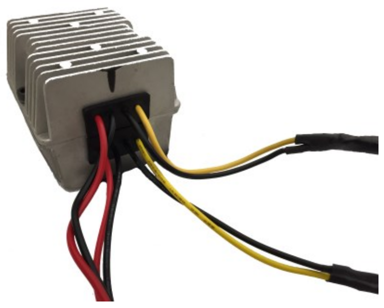


Figure 9 - Voltage Converter

3.3 VectorNav Compass

Next Year uses a VN-100 Rugged Inertial Measurement Unit (IMU) to navigate in consistent directions. This IMU was generously donated to RAER by VectorNav.

Figure 10 - VectorNav VN-100

The data used from this sensor are the Euler Angles of the sub, yaw, pitch, and roll. The main features of this IMU are its .02º RMS precision measurements, simple 2D magnetic field calibration, and ability to automatically filter out soft and hard iron disturbances. These filtering capabilities are vital when operating in close proximity to large switching currents, such as the motor controller.

3.4 Main Interface Board

On the top of our board stack (Figure 11) is our interface board, custom designed and built by a student. The board connects over USB and serves the following functions:

- Outputs vehicle status on the display (battery voltage, temperature, etc.)
- Read analog values from sensors.
- Monitors battery voltage and current draw.
- Drives servos.

This custom made board features:

- 10 PWM outputs
- 10 analog inputs
- I2C
- Micro SD slot
- 5 buttons
- LCD Screen

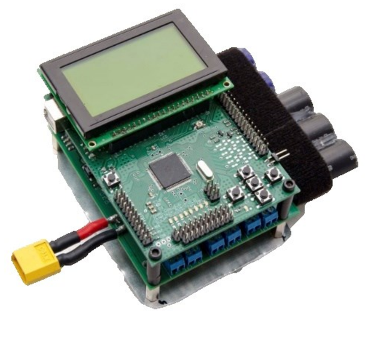


Figure 11 - Main Interface Board

3.5 Motor Controller Board

The motor controller board supplies power to the microcontroller for any servos and controls up to six thrusters. This custom-built board is capable of supplying -18 to +18 Volts at up to 15A to each of the six bi-directional channels.

The board accepts a battery voltage of up to 40 volts, ensuring that a variety of LiPos are safe to use with it. The board is controlled serially through a USB cable to the main computer.

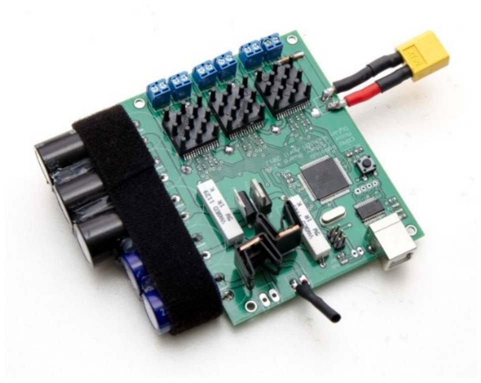


Figure 12 - Motor Controller Board

3.6 Power over Ethernet

A system providing a high-bandwidth, easily accessible protocol was needed for the computer vision cameras (see section 3.7). In addition, it is desirable to keep wiring external to the main compartment minimal. As a result, an implementation of Gigabit Ethernet and Power over Ethernet was utilized for the vision systems. The hardware facilitating the implementation is composed of a single component – A Netgear GS108P Gigabit PoE switch. This switch provides up to 50W of power, divided between the four PoE ports.

Figure 13 - NETGEAR GS108P

3.7 Computer Vision Cameras

While standard webcams and other low-cost cameras are useful for computer vision, a higher quality implementation is needed for the sub-optimal lighting conditions natural to an underwater environment. As such, a camera which utilizes a high fidelity CMOS sensor and operates via PoE gigabit Ethernet was selected and integrated – a Point Grey BlackFly camera with 2.3MP resolution at 27 FPS maximum (BFLY-PGE-23S2C-CS). Contained within is a Sony IMX136 CMOS sensor, which provides for high color accuracy. Fujifilm 2.2-6mm lenses are used. These F1.3 lenses were selected due to their variable focus and large aperture for increased light. The camera with attached lenses and custom 3D printed mounting bracket is displayed in Figure 14.

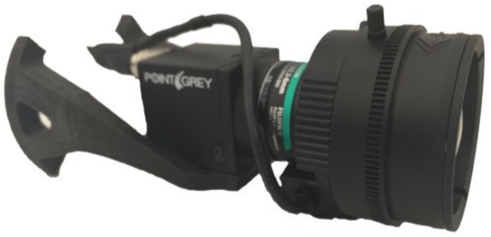


Figure 14 - Point Grey BlackFly

4. Software

Almost all of our software is written in the programming language Python. This language was used for its simplicity and ease of development. Python is so simple and intuitive in its implementation, that even freshman with very little programming experience can develop software. In addition, the ability to avoid compiling means the process of coding, testing, and re-coding is much quicker. The pySerial and openCV packages are used to interface with sensors and for computer vision, respectively. With pySerial it is easy to set up serial connections to multiple COM ports with custom bit rates, allowing for communication with Arduino boards and a high bit-rate compass. The openCV package is a powerful open source computer vision library that gives python powerful tools for interpreting images.

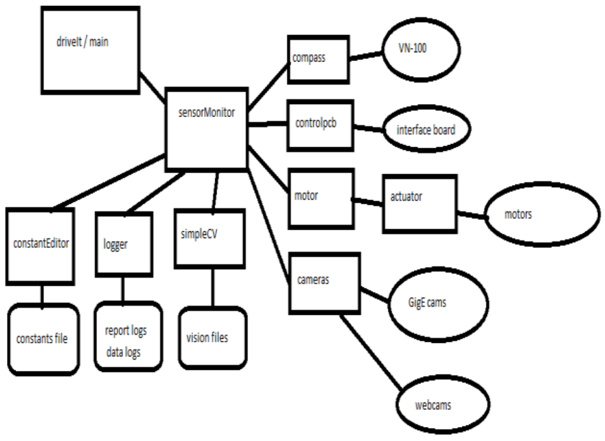


Figure 15 - Software Diagram

4.1 Motor control

The motor controller board's firmware is written in Arduino code. The code was written such that the state of a motor can be changed by sending a serial command with the direction (forward or reverse), motor ID number, and the power level. Also, for safety, a single command can be sent to stop all of the motors at once. The actuator class handles all serial communication with the motor control board. The motor class handles translating movement commands such as "move straight" and "turn at half power" into what the commands for the individual motors should be. The separation of the actuator and motor classes is to allow for a simpler transition to a different motor configuration. If the sub were to transition to a holonomic configuration with angled motors, only the motor class would need to be changed. The serial communication and any motion command from higher in the software would remain unchanged. This structure will hopefully allow for future development of the submarine to focus on higher-level aspects of the software, even if there are major hardware changes.

4.2 Sensors

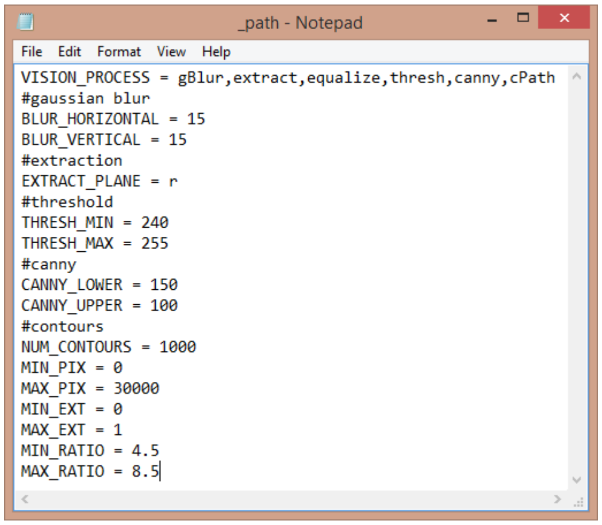
The main interface board is very similar to the motor controller board, as it is also based off of the ATmega1280. The interface board's firmware is written in Arduino code, and is also interacted with by the use of serial commands. To read the sensor values, a serial command is sent to the board, and the board responds with the data delimited by commas. To control servos, commands almost identical to those sent to the motors are sent. All commands to the board are handled by the control controlpcb class. Much like how the actuator class handles all communication with the motors, the controlpcb class handles all communication with the interface board. Unlike the interface board, the compass outputs its data automatically at 40Hz. The compass class organizes the data by looking for the start and end characters < and >. Like the motors, the software was structured in this way to allow for sensors to be easily replaced if there is a failure. If the compass needed to be replaced during competition with one that had an output with a different format, the only software that would need to be swapped out is the compass class.

4.3 Cameras

With development currently in using the PointGrey GigE cameras, and adding a down-facing camera, it is very important to have software flexible enough that the number and type of cameras can be easily changed. The cameras class handles the setup and connection to both webcams and GigE cameras. The connections are stored in a dynamic list, and frames can be retrieved from the cameras by either the index they were added to the list, or by a name such as "front" or "bottom". This allows for the higher-level code to easily adjust to varying developmental needs.

4.4 Vision

The goal when making the computer vision software was to create something that was easily customizable and simple to use. The problem with writing, testing, and rewriting vision code is that it takes too long, and the parameters to the openCV methods aren't always intuitive. Team Next Year's solution is simpleCV. The simpleCV class takes a core set of openCV methods, simplifies their parameters, and allows them to be implemented with a simple text file. The following example is how the path markers are located.



```
VISION_PROCESS = gBlur,extract,equalize,thresh,canny,cPath
#gaussian blur
BLUR_HORIZONTAL = 15
BLUR_VERTICAL = 15
#extraction
EXTRACT_PLANE = r
#threshold
THRESH_MIN = 240
THRESH_MAX = 255
#canny
CANNY_LOWER = 150
CANNY_UPPER = 100
#contours
NUM_CONTOURS = 1000
MIN_PIX = 0
MAX_PIX = 30000
MIN_EXT = 0
MAX_EXT = 1
MIN_RATIO = 4.5
MAX_RATIO = 8.5
```

Figure 16 - Example Configuration File

The first line contains the list of processes applied to the original frame in the order they occur. The subsequent lines change the parameters for the methods used. When the values in the text file are changed, the computer vision process instantly updates while it is running. As seen in Figure 17, the process successfully finds the path, and its angle relative to the vertical axis.

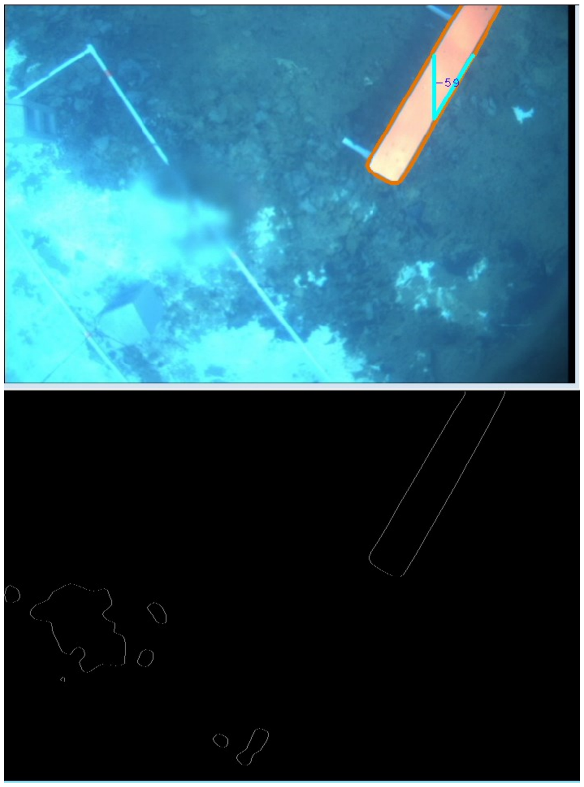


Figure 17 - openCV recognition of Path

An exciting new development is the addition of optical flow. Up until now, the only information available to the submarine about its state was if it was trying to move, if it was spinning, and if a task was in front of it. With optical flow, the hope is to be able to track position and velocity. Both methods tested so far have shown promising results, with accuracies within five percent when test at constant depth. The first method tested was the Farneback[1] method, which divides the images into neighborhoods and attempts to track their position from frame to frame. This method was found to have issues if the sub was moving too quickly, and uses a significant amount of processing power. The second method tested used the Lucas-Kanade[2] method on sets of good features. This second method requires much less processing and handles quick motion better, but both are still being tested to try to obtain the highest accuracy.

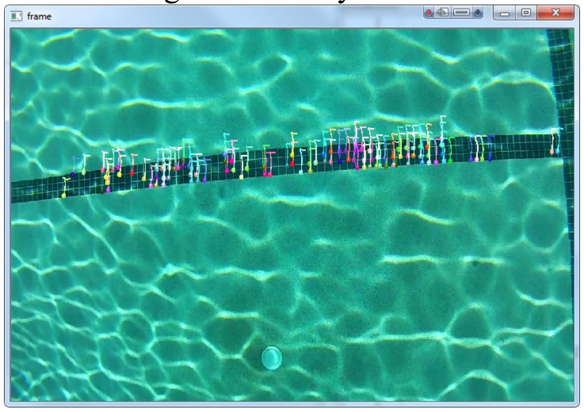


Figure 18 - Optical flow displaying motion vectors

4.5 Data Hub

The heart of the sub's data and processing is in the sensorMonitor class. Last year, all of the data processing was run in a single thread. Though this appeared to be sufficient, it resulted in a general instability to the software, and slower read-rates for all of the sensors. It all started with the discovery that controlling the motors and reading the compass and the interface board in the same loop could only update at five times a second. In the new system, the motor control, compass, interface board, cameras, optical flow, data logger, and health monitor each have their own thread. This greatly increases the speed of the software, and has the added bonus of being able to start, stop, and reset the different processes individually.

4.6 Loading parameters

Much like the vision software, there are many variables that can only be effectively tuned while the system is active. To facilitate the editing of these variables, the constantEditor class was created. Just like in the vision software, but with a broader scope, certain constants can be loaded from a text file. The effect of changes to the PID controller are much more visible when they are changed during operation of the sub.

[1] Farneback (2003)
[2] Bruhn and Weickert (2004)

4.7 Data Logging
Keeping good logs is the key to fully understanding the errors that pop-up. Next Year keeps two types of logs, status reports, and sensor data. The status reports track the status of all of the processes of the sub. Whenever one of the threads is started, restarted, loads a new set of parameters, or has an error, a time stamped message is directly written to the session's status report. These reports are crucial for checking that everything is starting in the correct order, and knowing what errors are occurring. The errors tracked range from dropped camera frames to not being able to connect to sensors. The sensor data logs record at a rate of five times a second, the position, velocity, acceleration, rotation, and target states. The camera is recorded from at the same rate.

4.8 Health monitoring
A new feature of Next Year is a much-needed addition of robustness to the software. As mentioned previously, the multithreading of the system allows for individual systems to be easily restarted, and the logs keep track of whenever sensors lose connection or stop functioning properly. The product of these two features is automatic health monitoring and action. An issue that occurred at the last competition was that during semifinals, the sub repeatedly received bad data from sensors and even lost connection to the motors. This error hadn't been seen before in previous testing or qualifying runs, crippling the sub in an otherwise successful competition. The hope is that the ability for the vehicle to self-diagnose and correct may prepare it for even the unexpected.

4.9 Mission Process
The mission plan currently focuses on navigational tasks, and is made up of two different types of commands, vision based movement, and waypoint movement. Vision based movement is the most obvious requirement of the competition, as most tasks are composed of "find bright object X and do Y." The vision mission parameters are the names of the files, which contain the vision settings, the maximum amount of time to spend attempting the mission, the speed at which to move, and a name for identification. The type of interaction with the object detected is handled set within the vision settings files. Waypoints are used when vision tasks aren't visible from the current location, or it's just easier to use, such as the gate. The waypoint missions differ from the vision ones in that they are given a point or a direction instead of a vision task. All of this allows for the mission plan script to focus on what movements need to be made to complete a task instead of how to implement them. Below is a simple example of how the first two tasks would be implemented.

5. Acknowledgments
We would like to thank Embry-Riddle Aeronautical University's College of Engineering specially the Department of Electrical, Computer, Software, and Systems Engineering and the Mechanical Engineering Department for their continued and substantial support of Team Next Year. We would especially like to thank our faculty advisors for all their support, encouragement, and advice. We would also like to thank all of our sponsors for the outstanding support and donations that have made this project possible.

5. References

BRUHN, A., & WEICKERT, J. (2004, April 22). *Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods.* Retrieved from ee.oulu.fi: http://www.ee.oulu.fi/research/imag/courses/Kokkinos/bruhn-ijcv05c.pdf

Farneback, G. (n.d.). *Two-Frame Motion Estimation Based on Polynomial Expansion.* Retrieved from diva-portal: http://www.diva-portal.org/smash/get/diva2:273847/FULLTEXT01.pdf

VectorNav. (2014, June 26). *VN-100 Rugged IMU / AHRS.* Retrieved from VectorNav: http://www.vectornav.com/products/vn100-rugged