Design Philosophy-Driven Development of Autonomous Underwater Vehicle *Alfie*

Seth Isaacson, Daniel Yang, Ginger Schmidt, Kyle Rong, Diana Lin, Omari Matthews
Harvey Mudd College
Claremont, CA, USA

***Abstract*—This paper discusses the development of MuddSub's autonomous underwater vehicle, *Alfie*. We will compete at the 2019 International Robosub Competition. As MuddSub is a new organization, *Alfie* was not designed to be a robot with state-of-the-art, groundbreaking functionality. Instead, *Alfie's* main innovation is in its unique design philosophy: Simplicity, Stability, and Scalability. The principles of our design philosophy are intended to reflect the limitations that new teams face when competing at RoboSub for the first time. The design philosophy is a top-level, governing principle which dictates our competition strategy, and in turn influences our vehicle design.**

By introducing this philosophy in our first year of competition, we hope to contribute to the underwater robotics community in two ways. Firstly, we wish to influence pre-existing teams by demonstrating how adhering to a strict set of goals allows for a effective competition strategy and creative systems engineering, while minimizing over-engineering. Secondly, as Robosub is a ever-increasing community, we believe *Alfie* will be a model of an efficient and effective way to compete in a team's first year. Finally, we hope to encourage growth in the RoboSub community, and pave the way for novel, impactful research in autonomous underwater vehicles.

I. Design Philosophy

The conception, design, and manufacture of *Alfie* was guided by three interconnected principles. These three principles, in decreasing order of priority, are as follows:

A. Simplicity

We define simplicity as avoiding superfluous complexity. In practice, simplicity embodies the restrictions inherent to being a new team.

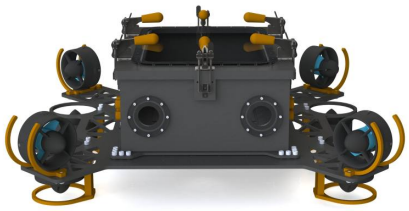

Fig. 1. From competition strategy to implementation, *Alfie* is developed on our design philosophy.

These restrictions include development time, human resources, and monetary resources. In terms of development time, all members of MuddSub are full-time students at a rigorous institution, and over the summer we are all balancing this project with full-time research positions. In total, *Alfie* was developed by six people, including three who worked over the summer. In terms of monetary resources, the team had a total of $ 10,000 dollars for the project, of which $6,000 were spent on components on the final iteration of the robot.

Our sensors are thus limited to an Inertial Measurement Unit (IMU), a depth sensor, and cameras. This means *Alfie* will not have DVL values for accurate localization, and rely only on cameras for obstacle localization. However, many tasks of Robosub can be completed with

only these sensors, and the DVL is sometimes excessive to achieve reliable results in vision-based tasks. Through simplicity, we maximize our underwater autonomous capability despite the limitations we faced.

B. Stability

We define stability as designing for minimal change to *Alfie's* core systems. Stability further serves as our measure of reliability. To exercise our desire for stability, we are prioritizing concepts in a greedy strategy: rather than focusing our efforts on accomplishing many of the complex tasks, we only work on tasks we believe are achievable for the next competition. By doing so, our team can maximize our potential for success in this year, while also laying the foundation for success in future years. In order to achieve stability, we need to prioritize design decisions which would work consistently this year and would require minimal modifications to the robot in the future. Stability is connected to simplicity as robots with high simplicity tend to be stable.

C. Scalability

We define scalability as the possibility of future expansion. Scalability is inherently linked to stability. In further years, we would like to incrementally modify *Alfie* without complete redesign. There is an inherent relationship between scalability and stability because reducing the changes made to existing systems will allocate/divert more time and resources to new systems.

II. Competition Strategy

Our competition strategy follows directly from the design philosophy.

A. Task Selection

In order to adhere the tenets of our design philosophy, *Alfie* will only attempt vision-based tasks. While it is tempting to incorporate a sonar system, which has a significant point value, we decided this would fundamentally undermine the simplicity of the robot. A sonar system introduces a level of complexity that compromises the success of the vision-based tasks given our new-team restrictions. For a rookie team, distributing resources across a variety of tasks decreases the likelihood of success on each individual task.

B. Competition Route

Leveraging our vision-based system, we will prioritize passing through the gate and 'vampire slaying' (the buoy task) in our competition route. Time permitting, we may also attempt to compete garlic drop (the bins task) to the best of our ability. First, we will determine the departure direction of *Alfie* based on a random coin toss. From the dock, we will attempt to search for the gate through vision code. When Alfie is confident enough that it found the gate, it will move closer towards it. Once close to the gate, *Alfie* will identify the smaller section of the gate, and pass through the gate while attempting revolutions for style. Next, *Alfie* will attempt to head toward one of the targets at slay vampires, completing our competition route. From there, attempting the bins task is considered a stretch goal.

C. Task Development

To maximize stability, we focused on thoroughly addressing a limited number of tasks rather than attempting every task we are theoretically capable of. Thus, we spent sufficient time developing each task before beginning work on future, more challenging tasks. For example, while many teams ded-reckon through the gate, we spent several months developing a reliable vision system for the gate before moving on to the path markers and buoys.

In the current state of the robot, we are confident in our ability to accomplish the gate task. We are now incrementally developing the buoys task and, only once confident in our ability to complete buoys, we may begin work on the bins.

III. Vehicle Design

Alfie's design was determined by our competition strategy, and thus, adheres to the three cornerstones of our design philosophy. We decided to split *Alfie's* design into the three traditional subcategories: mechanical, electrical, and software.

A. Mechanical Systems

The mechanical systems comprise an enclosure which houses all electronics, and a base plate which mounts the enclosure and all other hardware. In accordance with our design philosophy, the main priorities in design of the mechanical systems were simplicity and accessibility of the internal components. Thus, all components are mounted in a single rectangular enclosure, sealed by one large o-ring seal.

Our design made the robot simple to manufacture and assemble. Symmetry in design also decreased overall unique custom parts count, which made the build process much more efficient. All machining operations were completed in-house by team members, with the exception of the initial waterjet cut (performed by the shop manager on campus) and the welding (outsourced to a professional). The total cost of the frame, enclosure, and out-sourced welding was under $1500, a fraction of what we know comparable robots to cost.

B. Electrical Systems

The electrical systems are built entirely of off-the-shelf components. The hardware is able to support vision-based tasks. The electronics are centered around an Nvidia Jetson TX2 comptuter. A Vectornav VN-100 IMU provides inertial data, while a pair of Blackfly cameras with Theia wide-angle lenses supply images.

1) Power System: Alfie's power system presents a novel solution to a common problem in compact systems which handle relatively large amounts of power. Because of the presence of 8 thrusters - which are large, inductive loads - any power lines are likely to become extremely noisy. This can damage the sensitive

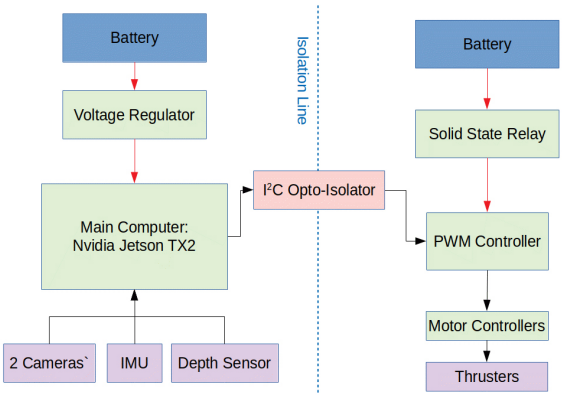

Fig. 2. Alfie's electronics infrastructure includes two opto-isolated circuits to prevent damage of sensitive electronics

electronics in the enclosure. To address this, *Alfie* has two isolated circuits. One is dedicated to the thrusters, while the other manages the computer and auxiliary components. The computer communicates with the thrusters' speed controllers through an opto-isolated i^2c line.

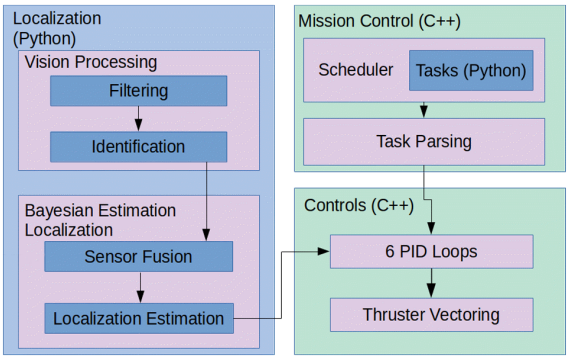

Fig. 3. Alfie's software stack is split to localization and a ROS-based backend which handles communication and task execution.

C. Software

1) Backend: Alfie's software stack is built around ROS. The backend is designed to be as user-friendly as possible, providing a range of QT-based utilities for interfacing with the code. In the spirit of simplicity and stability, the back-end controls were left extremely standard. We run 6-axis PID control; however, due to the physical stability of the robot we do not

anticipate needing the pitch or roll control loops. State queuing and execution is handled by the ROS SMACH library, a powerful and well-documented tool for creating hirearchical state machines.

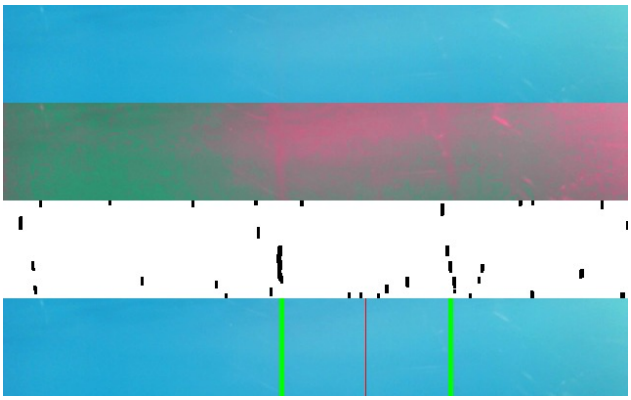

Fig. 4. These are the image pre-processing steps that demonstrate how traditional computer vision tools can accentuate a gate that even the human eye has difficulty recognizing.

2) Image Pre-processing: After the advancement of machine learning object detection such as the popular YOLO, the prevalent vision strategy has been to simply pass a given image into a network, and receive back a bounding box of the feature [1]. Although object detection networks are powerful, some of the functionality it provides is overkill for RoboSub, and sometimes, better results can still be achieved with more traditional methods. For instance, consider the detection of a gate: given that the depth of the gate is constant, identification of the gate relies only on detection of the x-coordinates of each leg.

For this reason, we decided to attempt an OpenCV-based gate detection. Our algorithm increases the contrast of wanted features, filters out noise, then runs a binarization to detect the features. To achieve these tasks, we first convert the image into the YUV color space, where Y is the luma (brightness) component, and U and V are the cool color and warm color components respectively. Since the background of an underwater image is generally blue and the gate is generally yellow, we enhance the contrast in the warm color space by linearly scaling the pixel values, and remove the cool color space. After this, we apply additional noise treatment and binarization, and localize the gate by finding the location for the peaks of sums of pixel values columns. To correct for gate legs positioned at an angle, our code experiments with the binarization under different rotations and reports the localization results of the most probable rotation.

The traditional computer vision-based approach provides several benefits over neural networks. Firstly, the computer-vision based code runs in less than 0.1 seconds; even if the detection is not perfect, statistical methods can then figure out the true location of the gate. Secondly, and more importantly, object detection networks need a substantial training data set. These images are not readily available in bulk for new teams, despite the best intentions of participating teams. Furthermore, the lighting conditions and environment of TRANSDEC is considerable different than that of a recreational swimming pool. Apart from on-site training, the neural network will likely perform poorly compared to OpenCV-based code. To demonstrate the significance of the lack of training images, a benchmark between YOLO and OpenCV is included in the experimental results section. The last benefit is the reuseability of the code. Since most of the OpenCV code is designed to clarify features of images, much of the preprocessing can be reused for detection of other obstacles, such as the bins or buoy tasks. Finally if in the future we determine a machine learning application to be more applicable to certain tasks, the preprocessing will still increase the success rate of the object detection.

IV. Experimental Results

A. Watertight Testing

To verify our mechanical systems, we completed a series of water tightness tests with *Alfie*. We first used our vacuum pump to determine if the enclosure was airtight. After fixing a few minor issues, we proceeded to complete

underwater tests. For each trial, we placed the water in the robot, weighed it down such that it sank to the bottom of the pool, and retrieved it after one hour. In general, we found three sources of leaks. The first was leaks caused by our Blue Robotics hull perpetrators, which we discovered needed to be extremely tight to function properly. The second was a faulty weld which had a thin pore through which water dripped. We repaired this issue with marine epoxy. Thirdly, our primary o-ring seal was slightly unreliable. We fixed that by using a slightly larger diameter o-ring. Since making those three adjustments, we've experienced no leaks in over twenty hours of testing.

B. Vision Benchmark

To verify our theory that OpenCV outperforms neural network such as YOLO given limited training images in objection localization, we trained and tested OpenCV and YOLO codes with 83 images from past competitions. We use only competition images, because we want to have object detection codes usable during competitions. In particular, including non-competition images will not promote the functionality of the neural net, because neural networks typically require more than 2000 images to be fully flexible and functional. Using the 83 competition images we obtained, we randomly choose 54 of them as training data and 29 of them as testing data. Our test focuses the accuracy of the code in detecting the x-coordinate of the leg of the gate. For a 640 by 640 images, we consider an error of 30 pixels acceptable. Using this standard, we find that the f-measure (f-score) is 0.88 for OpenCV localization and 0.58 for YOLO trained on 100 epochs. Moreover, while YOLO on average takes 0.46 seconds, OpenCV-based detection only needs 0.2 seconds on average to generate a prediction. The reason for the less ideal performance of YOLO code is that YOLO inherently requires more training images than we currently possess. We believe with more images, YOLO can improve its performance substantially and this will be our goal next year.

Fig. 5. MuddSub Team Members, 2019

ACKNOWLEDGEMENTS

MuddSub would like to thank all the individuals who helped make our work on *Alfie* within a single year possible. We would like to thank Harvey Mudd College's Shannahan Fund, which served as our primary source of money. We would also like to thank the Harvey Mudd College Department of Computer Science for providing a workspace and access to the underwater robotics testing laboratory. This work would not be possible without our advisor Zachary Dodds. Furthermore, machine shop manager Paul Stovall provided extremely valuable advice and assistance relating all machining tasks. We are grateful for our corporate sponsor Connect Tech Inc. for providing us with a Jetson TX2 carrier board which drastically reduced the footprint of our computer. We would also like to acknowledge Theia Technologies for providing us with high quality camera lenses. Finally, we wish to thank VectorNav for providing our team with an IMU.

REFERENCES

[1] Redmon, Joseph, and Ali Farhadi. *Yolov3: An incremental improvement.* arXiv preprint arXiv:1804.02767 (2018).

APPENDIX A: EXPECTATIONS

Subjective Measures			
	Maximum Points	Expected Points	Points Scored
Utility of team website	50	40	
Technical Merit (from journal paper)	150	150	
Written Style (from journal paper)	50	50	
Capability for Autonomous Behavior (static judging)	100	100	
Creativity in System Design (static judging)	100	100	
Team Uniform (static judging)	10	10	
Team Video	50	40	
Pre-Qualifying Video	100	100	
Discretionary points (static judging)	40	40	
Total	650	630	
Performance Mmeasures			
	Maximum Points		
Weight	See Table 1/Vehicle	80	
Marker/Torpedo over weight or size by <10%	minus 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" (2 total)	100/segment		
Slay Vampires: Any, Called	300, 600	600	
Drop Garlic: Open, Closed	700, 1000 / marker (2+ pickup)		
Drop Garlic: Move Arm	400		
Stake through Heart: Open Oval, Cover Oval, Sm Heart	800, 1000, 1200 / torpedo (max 2)		
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 (Crucifix only)		
Random Pinger first task	500		
Random Pinger second task	1500		
Inter-vehicle Communication	1000		
Finish the mission with T minutes (whole+factional)	Tx100		

APPENDIX B: COMPONENT SPECIFICATIONS

Component	Vendor	Model/Type	Specs	Cost(if new)
Buoyancy Control				
Frame				
Waterproof Housing				
Waterproof Connectors				
Thrusters				
Motor Control				
High Level Control				
Actuators				
Propeller				
Battery	HobbyKing	Turnigy	Li-Po battery	
Converter				
Regulator				
CPU	NVIDIA	Jetson		
Internal Comm Network				
External Comm Interface				
Programming Language 1	Standard C++ Foundation	C++14		Free
Programming Language 2	Python Software Foundation	Python 3		Free
Compass				
Inertial Measurement Unit	VectorNav	VN-100T		Donated
Doppler Velocity Log	n/a			
Camera(s)	FLIR	FlyCapture		
Hydrophones	n/a			
Manipulator	n/a			
Algorithm: vision	OpenCV	Python CV2		Free
Algorithm: acoustics	n/a			
Algorithm: localization and mapping	Gaussian-based Localization			
Algorithm: autonomy				
Open source software	Canonical Ltd.	Ubuntu 16.04		Free
Team size	6 team members			
HW/SW expertise ratio	1:1			
Testing time: simulation	0			
Testing time: in-water	15 hrs			