Chapter 1

COOPROV: LOW-COST UNDERWATER REMOTELY OPERATED VEHICLE

1.1 Outline

The developed multi-image object recognition theory can be validated using an easily accessible open-source robot.

Section 0: Intro

Section 1: Commercially available underwater robots are expensive and provide limited flexibility for incorporation of new sensors and custom controllers.

Section 2: CoopROV based off of the open-source underwater platform openROV, marries the low cost hardware design with seamless integration to ROS for easy application development.

Section 3: An underwater experimental setup in a water tank was designed to collect underwater image data.

Section 4: IMU, depth sensor, stereo camera and AR Tags were used to localize CoopROV.

Section 5: The multi-image object recognition approach was validated using image data gathered from CoopROV.

1.2 Introduction

Field experiments related to underwater object recognition require a submersible robot with appropriate sensing and control payload to navigate through points of interest and capture necessary data through sensors. Depending on the level of manual

intervention involved to guide the vehicle, an underwater vehicle, can be classified as an Autonomous Underwater Vehicle (AUV), a Remotely Operated Vehicle (ROV) or a Human Operated Vehicle (HOV). AUVs navigate autonomously with minimal to no manual intervention. Both ROVs and HOVs are piloted by humans, with the difference being that the human operator is within the vehicle in an HOV, whereas in the ROV the human operator controls the vehicle from a remote location. Commercially available underwater vehicles come with wide range of capabilities in terms of sensing and operational depths. The sensing package or the type of vehicle required is dictated by the objective of the science experiment and its requirements. For validation and fine-tuning of the underwater object recognition algorithms like the multi-layered scallop recognition (Chapter **??**) and the multi-view object recognition (Chapter **??**), an underwater vehicle that can navigate to specific points and collect image data of underwater targets is needed. With these requirements in mind, a low-cost underwater research vehicle named CoopROV was developed. CoopROV is a ROV-class vehicle that is equipped with several sensors for data collection, and also has the capability to incorporate any customized controller.

1.3 Commercial Submersible Robots

Commercially available robots are expensive and often provide minimal access to their legacy controllers and software. Most commercial ROVs come with a graphical interface and joystick. AUVs come with an interface to allow the user to predefine the robot trajectory through a series of geo-tagged waypoints before deployment. Once deployed, the AUV tracks the specified trajectory. Such manufacturer interfaces do not provide considerable freedom to modify the core software running on the robot. For instance if someone wants to use their own controller, it becomes problematic to implement it without manufacturer supported software Application Program Interface (API)s. Cost is another factor that limits accessibility to commercial underwater robots. A relatively small ROV like the videoray [1] with a footprint of $14.75 \times 11.4 \times 8.75$ ($L \times W \times H$ in inches) and a sensor package comprising a camera,

gyros, accelerometers, compass, depth and temperature sensors can cost over 25 000 dollars. Bigger ROVs like the Outland ROV [2] can cost upwards of 75 000 dollars with the cost here being chiefly determined by the sophistication of the onboard sensing capabilities. Despite the high cost and minimal flexibility for customer modifications, the primary advantage of these commercial solutions is their well tested hardware that is resilient to harsh conditions that are characteristic to the marine environments.

The other league of underwater robotic solutions come in form of low-cost Do-It-Yourself (DYI) style robots like the OpenROV [3] and the Fathom One [4]. Such solutions offer a lot of potential for custom hardware and software modifications. However being small-scale development projects, the associated hardware in these cases is not well tested and can be prone to failures (e.g. water leaks). The capability of these systems to withstand the harsh conditions of a marine environment may be limited. In terms of software, the source code is often available and comes with an open-source license (as in the case of OpenROV). However such software often does not come with APIs to encourage users to extend or modify the capabilities of the robot. Additionally, the limited documentation available makes the task of directly modifying the source code cumbersome. Since the focus of most of these DYI projects are to offer low-cost options that primarily enable hobbyists to explore underwater, a teleoperation interface and access to a camera feed from the robot typically suffices. For researchers that may have very specialized interests, a professionally developed robotic solution should offer finer control of the hardware and software with options to easily extend the base capabilities of the system.

There are two primary requirements for a professionally developed robotic solution that is intended for research. The first is the need for a tough exterior shell that can hold up against high values of water pressure which is common with deep sea operation. This exterior shell should also be resilient to water leaks and at the same time offer the ability to retrofit sensors. The second requirement is for the robotic system to offer a well built software interface with APIs that allows researchers to extend the capabilities of the system. It will be beneficial if the software APIs adhere to popular

robotics research architectures like the Robot Operating System (ROS) [5, 6] or Mission Oriented Operating Suite (MOOS) [7, 8]. A robot with ROS-support will greatly enhance the versatility of the robot by allowing access to a plethora of existing software tools. Furthermore, software development based on open-source tools like ROS can be supported by an active base of researchers who can provide feedback, tips and advice. Currently the underwater research community lacks a variety of low-cost submersible robotic platforms with resilient hardware design and well-engineered software.

1.4 CoopROV

The chief requirements that led to the design of CoopROV was the need for a low-cost submersible robotic platform that can serve as a test-bed for computer vision and control algorithms. Before the effort to build CoopROV was initiated, existing solutions were evaluated. Commercial robotic solutions like the Videoray were clearly outside the budget of this research project. OpenROV 2.5 [3], the first version of OpenROV, was a good candidate primarily due to it sub-1000 dollar price tag. The original OpenROV design came with a beaglebone and browser based graphical user interface that used node.js [9] in the backend to drive the robot. However, multiple tests revealed that the water proofing design and electronics package had several glitches making it unsuitable for deployment in its original configuration. Furthermore, most of the algorithm development till that point was in ROS, hence ROS support was one of the preferred requirements for the submersible robot. The initial experimental evaluation of OpenROV provided useful insights into the core components and associated problems that need to be tackled to design a submersible robot. The design of CoopROV was initiated to address the shortcomings of OpenROV's hardware in terms of waterproofing and electronics. Additional features of the target design was to allow easy interfacing of new sensors along with full ROS software support for sensors and actuators.

In the current stage of development, CoopROV shows some resemblance to

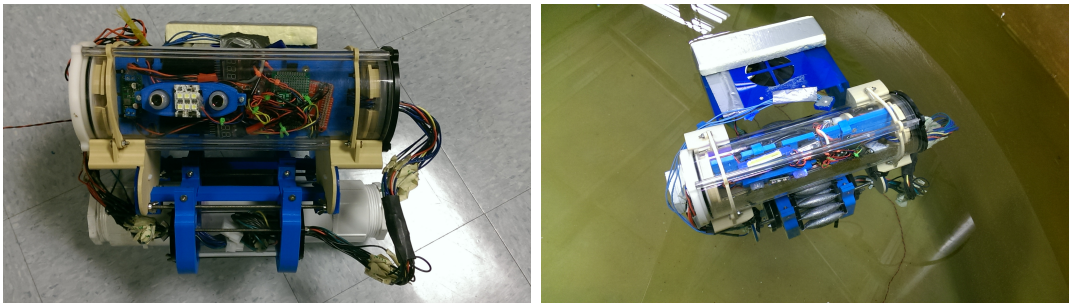


Figure 1.1: CoopROV shown in two different views

OpenROV as both of them share the same frame and actuators. However, all electronics, power-supply and sensors in CoopROV diverge completely from OpenROV 2.5. Two views of CoopROV a frontal view and its appearance when deployed in a test tank are shown in Figure 1.1.

1.4.1 System Overview

CoopROV is a ROV-class submersible that is connected to a surface station through a tether. The surface station is a laptop computer. CoopROV and its surface station are connected by an ethernet interface and run a distributed ROS system. In teleoperation mode, a joystick controller connected to the surface station can drive the robot. The joystick controller can actuate the 3 thrusters (two at the back and one on top) offering the robot a 6-Degree of Freedom (DOF) motion capability. This data flow between the different system components is graphically illustrated in Figure 1.2. The computing on the robot is handled by a RaspberryPi [10] and Arduino Mega [11]. The sensor package includes a stereo camera, a 9-DOF Inertial Measurement Unit (IMU), depth, temperature and voltage sensors. The robot is powered by two sets of Lithium Polymer (LiPo) batteries which can provide up to an hour of run time. For a brief list of CoopROV specifications see Table 1.1. A more detailed account of different sub-systems comprising hardware, electronics, software, sensors and power supply is given in the following sections.

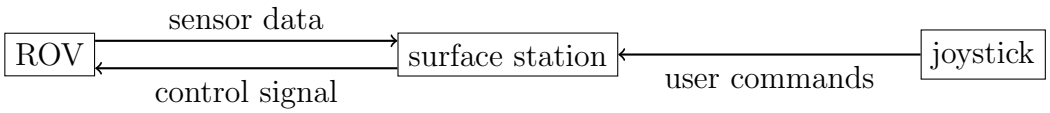


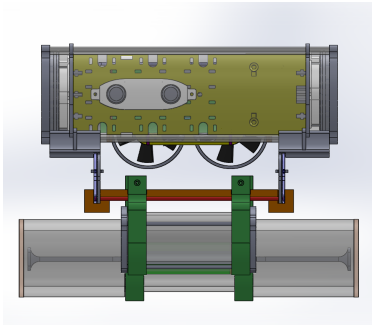
Figure 1.2: Block diagram of high-level data flow between different components in CoopROV

Weight	7 Kg
Dimensions (L x W x H)	0.30 x 0.28 x 0.39 m
Run time	1 hour
Sensors	Minoru stereo 3D webcam
	LIS3MDL 3-axis magnetometer
	LSM6DS33 3-axis accelerometer and gyro
	MS5803 pressure and temperature sensor
	LiPo tester voltage sensor

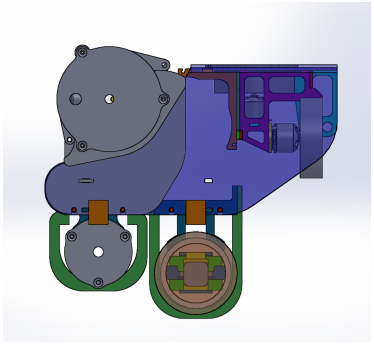
Table 1.1: CoopROV Specifications

1.4.2 Hardware

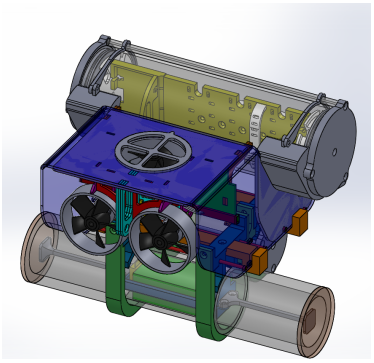
The exterior shell of CoopROV is composed of an acrylic frame that holds all housing tubes together. There are 3 housing tubes in total: an electronics tube, a connector tube and a battery tube. A Computer Aided Design (CAD) model of CoopROV in Figure 1.3 illustrates the position of the housing tubes on the CoopROV frame. The electronics tube is a long transparent acrylic tube that holds all sensors and most electronics of the robot. The connector housing tube is a relatively short transparent acrylic tube with a molex connector plug inside it. This connector tube serves as a water proof connector plug that allows to change the sensor package by swapping out the electronics tube. The battery tube is a PVC pipe that holds 11.1 V Li Ion batteries. The clips and clamps that hold the tubes in place were made out of 3D-printed plastic parts. One big design improvement over OpenROV is the water proof seals. The waterproof seals comprise a delrin end cap with an O-ring seal along with 3 threaded rods to maintain the tension between the end caps and the acrylic tube.



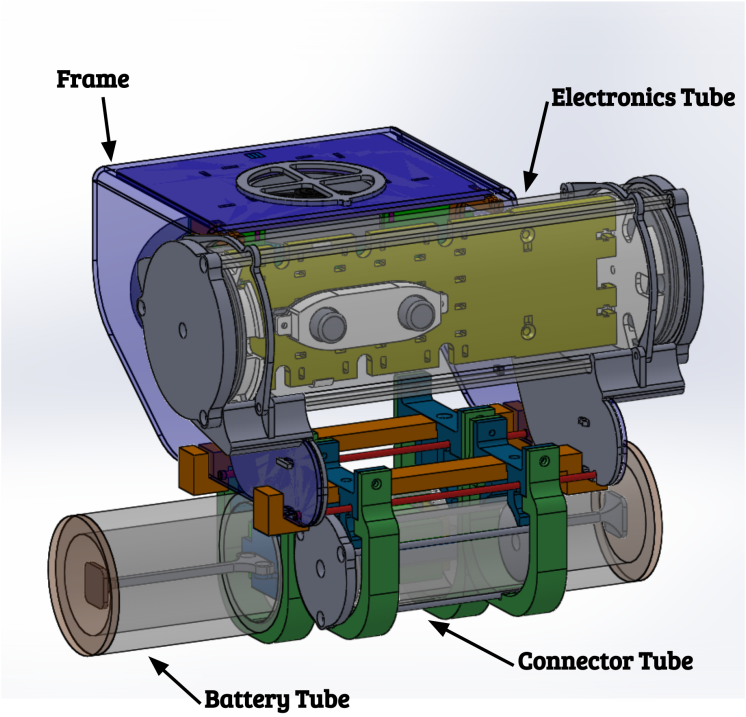
(a) Front view



(b) Side view



(c) Isometric view (back)



(d) Isometric view (front)

Figure 1.3: Different views of the CoopROV CAD model with some labeled parts

1.4.3 Electronics

The electronic components inside the electronics tube are responsible for controlling the thrusters and relaying the sensor data back to the surface station. The RaspberryPi and the surface station together constitute a distributed ROS system. For communication purposes, a Transmission Control Protocol (TCP) link is established between the RaspberryPi and the surface station via the tether cable. The Arduino is connected to the RaspberryPi via the serial bridge and behaves like a node on the ROS system. The Arduino logs the data from the depth sensor and the IMU. Additionally, the arduino is also responsible for translating the motor commands into signals for the motor driver. Image data acquisition from the stereo camera is directly handled through a usb interface on the RaspberryPi.

There are two parallel data streams. One enables the flow of control commands throughout the system and other handles the data being logged from the various sensors. The control commands from the joystick flow through the surface station and RaspberryPi to reach the Arduino. The Arduino translates these motor velocity commands into Pulse Width Modulation (PWM) signals, that enable the motor driver to run the propellers. The sensor data logged by the Arduino and RaspberryPi are served to other nodes on the ROS system through a *publish-subscribe* communication paradigm [12]. The block diagram in Figure 1.4 provides a graphical overview of the connections between the major electronic components on CoopROV. It is important to note that only one motor and motor driver are shown in Figure 1.4, but in the actual system, there are 3 motors with dedicated motor drivers. Figure 1.4 also shows the two power supplies in the system. More details on the power supply components can be found in next section.

1.4.4 Power Supply

Power supply is a critical part of any robotic system. Almost all mobile robotic systems require batteries onboard. However tethered systems like the ROVs can have an alternative resource in form of power supply through the tether. Running power

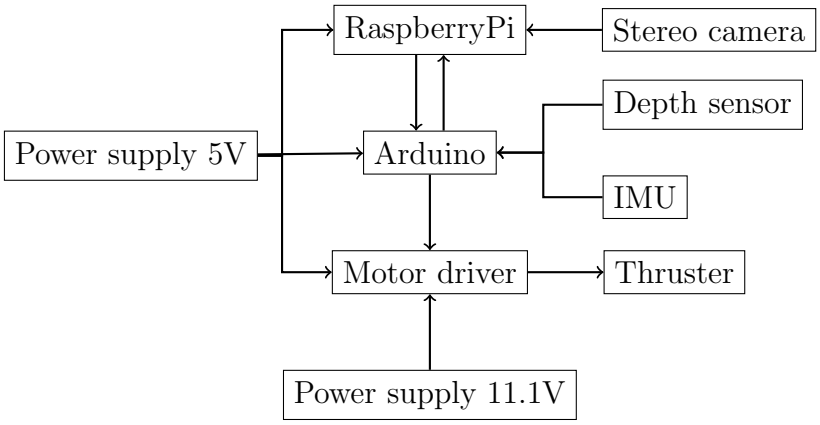


Figure 1.4: Block diagram depicting the connectivity between major components in the electronics schematics

through a tether comes with its own set of challenges. Direct Current (DC) transmitted at low voltages result in significant voltage drop due to the impedance of the long tether cables. Conversely using AC power at high voltages circumvents this problem but then runs into the need for onboard transformer/rectifier to transform the input Alternating Current (AC) power into usable DC form.

A robot's power supply components ought to be chosen, such that they can handle the expected operational load (also transients). For CoopROV, several power supply systems were tested. Four different battery types were evaluated: alkaline C, lithium ion, lithium phosphate and Lithium Polymer (LiPo). Tests revealed that the high transient current requirements (>20A) of CoopROV were only met by the LiPo composition. Thus, two sets of LiPo batteries were used to independently power the motors and electronics. This physical isolation of the power sources improves stability of the robot's power supply system and prevents electronic components from being affected by occasional erratic power draws of motors. More information on the two power sources is given in Table 1.2.

The two independent power sources (Table 1.2) cater to different sub-systems of CoopROV. The 5V power source drives all the onboard electronics. Additionally, the 5V power supply is stepped-down and regulated to protect the electronic circuits against voltage variations. The 11.1V (unregulated) power source meets the power

Table 1.2: CoopROV Battery Specifications

Power Source	Battery Specification
11.1V	LiPo 3-cell 11.1V 5400mAh
5V	LiPo 2-cell 7.4V 800mAh

demands of the motors. Figure 1.4 shows the different power supplies along with the different components powered by them. An approximate running time of the ROV is 1 hour can be obtained by doubling the number of LiPo batteries on each power source.

1.4.5 Sensors

CoopROV comes with 4 sensors; namely, stereo cameras, IMU, depth sensor and voltage sensors. A low-cost off-the-shelf Minoru 3D webcam [13] was used to capture stereo images at 5 Frames Per Second (FPS) with a resolution of 320×240. The 9-DOF IMU on the robot provides instantaneous acceleration in the x,y and z directions along with roll, pitch and yaw. The depth sensor measures the water pressure and temperature to compute the robot's operating depth. The voltage sensors are connected on to the LiPo batteries to measure and report the instantaneous battery voltage. They also double as a protection to the LiPo batteries; they signal an alarm if the battery voltage drops below a critical value.

1.4.6 Software

The software of CoopROV is built on a ROS architecture. The different core functions of the robot are sub-divided into independent modules or *ROS-nodes*. Each ROS-node has a specific task and can communicate with the other ROS-nodes. Additionally, the ROS architecture allows ROS-nodes running on different systems to communicate seamlessly: agnostic to where they are running. The interaction between different ROS-nodes is handled through a *publish-subscribe* communication paradigm [12]. These characteristics of the distributed ROS system was exploited to build CoopROV's software stack. The CoopROV setup comprises a distributed ROS architecture, where

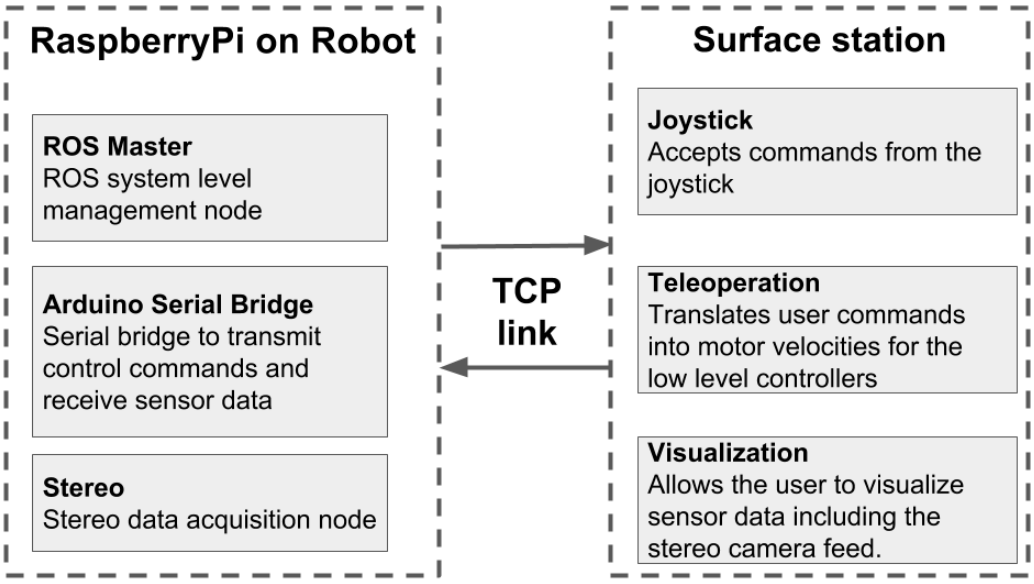


Figure 1.5: CoopROV software architecture showing all the ROS-nodes along with a brief overview on their functionality

different ROS-nodes run either on the RaspberryPi or the surface station. Both these platforms are connected via a TCP link. The different nodes running on the robot and the surface station along with their functionality is explained in the software architecture diagram in Figure 1.5.

1.5 Test Infrastructure

Initial waterproofing tests were performed in the University of Delaware indoor swimming and diving pools. Later integration tests for software and electronics along with some experimental runs were performed in a circular tank at Robotic Discovery Laboratories (RDL) in University of Delaware.

1.6 Localization Experiments

Preliminary localization experiments were performed by placing an Augmented Reality (AR)-tag on the bottom of the water tank. Images from the stereo camera were used to detect and track the AR-tag. Further experiments were performed by fusing the the 6-DOF position obtained from the AR-tag localization and IMU values.

Since there was no reliable reference measure to compare the reported position of the ROV, the quality of the results were not verified.

1.7 Conclusion

CoopROV was built as a research platform to conduct underwater object recognition and control experiments. It offers a low-cost solution with a easily reconfigurable hardware and extensible ROS-software interface. ROS-software support allows the robot to use open source perception and control algorithms available as a part of ROS software infrastructure. The sensor package comprising a stereo camera, IMU and depth sensor can be leveraged for object recognition and motion control experiments. The capabilities of the CoopROV system can be easily expanded to suit any research experiment like the multi-layered scallop recognition (Chapter **??**) and the multi-view object recognition (Chapter **??**) discussed in this dissertation.

Acronyms

AC Alternating Current. 9

API Application Program Interface. 2, 3

AR Augmented Reality. 11

AUV Autonomous Underwater Vehicle. 2

CAD Computer Aided Design. 6, 7

DC Direct Current. 9

DOF Degree of Freedom. 5, 10, 11

DYI Do-It-Yourself. 3

FPS Frames Per Second. 10

HOV Human Operated Vehicle. 2

IMU Inertial Measurement Unit. 5, 8, 10–12

LiPo Lithium Polymer. 5, 6, 9, 10

MOOS Mission Oriented Operating Suite. 4

PWM Pulse Width Modulation. 8

RDL Robotic Discovery Laboratories. 11

ROS Robot Operating System. 4, 5, 8, 10–12

ROV Remotely Operated Vehicle. 2, 3, 5, 8, 10, 12

TCP Transmission Control Protocol. 8, 11

BIBLIOGRAPHY

[1] VideoRay Pro 4 standard base datasheet. http://www.videoray.com/images/specsheets/2016/2016_P4STANDARDBASE_FINAL.pdf. Accessed: 2016-09-17.

[2] Outland ROV. http://www.outlandtech.com/product_info.php. Accessed: 2016-09-17.

[3] OpenROV. http://www.openrov.com. Accessed: 2016-09-17.

[4] Fathom One ROV. http://fathomdrone.com. Accessed: 2016-09-17.

[5] Robot Operating System (ROS). http://www.ros.org. Accessed: 2016-09-17.

[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and AY Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[7] Mission Oriented Operating Suite (MOOS). http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php/Main/HomePage. Accessed: 2016-10-03.

[8] PM Newman. MOOS-mission orientated operating suite. Technical Report 08, Massachusetts Institute of Technology, 2008.

[9] node.js. https://nodejs.org/. Accessed: 2016-09-17.

[10] RaspberryPi. https://www.raspberrypi.org/. Accessed: 2016-09-20.

[11] Arduino Mega. https://www.arduino.cc/en/Main/arduinoBoardMega. Accessed: 2016-09-20.

[12] PT Eugster, PA Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

[13] Minoru 3d webcam. http://www.minoru3d.com/. Accessed: 2016-09-20.