Bruce: A system-of-systems solution to the 2018 Maritime RobotX Challenge

Leo Stanislas, Kyle Moyle, Emily Corser, Terrence Ha, Rhys Dyson, Riki Lamont and Matthew Dunbabin

***Abstract*—This paper provides an overview of the hardware and software systems developed for Bruce, the Queensland University of Technology's Autonomous Surface Vehicle (ASV) for entry in the 2018 Maritime RobotX Challenge. Bruce is considered a *system-of-systems* platform consisting of the ASV itself, a self-contained vision-based Autonomous Underwater Vehicle (AUV), and an automated vision-based racquet ball launcher. Bruce's software and hardware architecture builds on 4 years of development from the 2014 and 2016 RobotX Challenges. Key hardware upgrades include integrated bow thrusters and control system, a refined safety system, and an automated AUV deployment and retrieval mechanism. Key software upgrades include a fused LiDAR and vision object detection and classification system, a vision system for underwater ring detection, improved vision system for the ball launcher, and a new mission architecture with task optimizer. To facilitate software development and offline testing, a high-fidelity simulation model developed in 2016 was extensively used. The ASV's control, mapping, and task-specific algorithms were evaluated both in simulation and through field experiments. Results demonstrating capabilities as well as discussions on lessons learnt are also presented.**

Fig. 1: Bruce – The Queensland University of Technology's Autonomous Marine System (AMS) for the 2018 Maritime RobotX Challenge.

I. Introduction

The Queensland University of Technology (QUT) presents our *system-of-systems* Autonomous Marine System (AMS) for competing in the 2018 Maritime RobotX Challenge. The objective of the 2018 Challenge compared to previous Challenges has been to increase the level of autonomy on-board the ASV though linking information gained during missions to complete complex tasks [1] as well as integrate multiple robotic platforms for new tasks. Whilst this increases the overall complexity of completing the tasks, it has provided a unique opportunity to explore ideas for solving this problem.

The Maritime RobotX Challenge requires each team use a standardized base platform, the WAM-V developed by Marine Advanced Research Inc. The teams can then configure the platform as they please by adding sensors, computing, power and propulsion within the Challenge guidelines [2]. In 2018, the competition also introduced tasks that encourage a system-of-systems approach to extend the functionality of the ASV. This approach was taken by TeamQUT with the development of two systems for the "Underwater Ring Recovery" and "Detect and Deliver" tasks which were integrated into the higher level task execution software.

The remainder of the paper is structured as follows: Section II provides an overview of the design strategy behind the ASV with Section III outlining the multiple hardware systems, upgrades and software components. Section IV describes the obstacle detection and classification approach with Section VI presenting the strategy for the underwater ring recovery task. Section VII presents the mission planning strategy with Section X providing concluding remarks.

II. Design Strategy

The team's development strategy for the 2018 Maritime RobotX Challenge was guided by the experience and knowledge sharing from team alumni who competed in the 2014 and 2016 challenges, in addition to the new challenge task descriptions [1]. The team reflected on the design (both hardware and software) and made recommendations on what worked and what did not and exploited previous software development via the teams code repository and the 2016 Autonomous Marine Vehicle Simulator [3].

Based on these reflections and task requirements, the bulk of the effort was particularly focused on 3 key software and 2 key hardware areas. The three primary focus areas for software development were; (1) The need for a more integrated obstacle detection and classification system, (2) the need for a robust underwater vision system, and (3) the need for a revised dynamic task allocation mission framework that optimizes points, time and autonomous system capabilities. The two primary hardware focus areas were; (1) the need to integrate the autonomous underwater vehicle, and (2) the need for bow thrusters to aid in low speed manoeuvring and station keeping. The following sections outline the results of these design considerations in preparation for the 2018 Maritime RobotX Challenge.

Fig. 2: The complete Autonomous Marine System (AMS) showing the ASV (aka Bruce), the AUV (aka Thunder4) and the racquetball launcher (aka Gusto).

III. Vehicle Design

This section provides an overview of the hardware and software design review, considerations and lessons learnt in preparing the Bruce ASV and associated systems for the 2018 Maritime RobotX Challenge.

A. Hardware Overview

In 2018, Bruce underwent a significant design review based on the lessons learnt from previous challenges, as well as the need to integrate the Autonomous Underwater Vehicle and racquetball launcher. The resulting configuration is shown in Figure 2. The primary lessons learnt related to; (1) the need for bow thrusters to improve docking and station keeping during strong winds, (2) the need for robust wiring and sensor integration, (3) more robust power management systems to increase shoreline operational time, and (4) the need to for a modular mission/task management system.

1) Physical Structure: The primary electronics systems reside in a custom frame attached to the underside of the WAM-V payload tray. On Bruce, all the electronics boxes are mounted on sliding trays which lock in place. The sliding trays were inspired by years of field work and the difficulties of assembly, mounting and accessing hardware to the top of the payload tray. The advantages of this approach are the trays allow beach access without climbing over the vessel, protection of the computers and batteries from rain and sun, and frees up the top for other tasks (e.g. platform for launching an AUV or UAV, solar panels, science payloads). The removable sensor frame provides a greater mounting height for the perception and navigation sensors described in the following sections.

Two key structural upgrades were integrated onto Bruce in 2018. The first was a launch and recovery system for the AUV (see Section III-C). The second was a revised retractable bow thruster assembly which also duals as a mount for the hydrophones (see Section III-A3).

2) Sensors: The primary navigation sensor for Bruce is a Novatel FLEX6 GPS providing position updates at 20Hz. The GPS is capable of providing heading angle so a low-end magnetic flux compass was selected for providing a redundant compass/heading angle (9DOF Razor Inertial Measurement System). In addition to heading angle, this sensor provides roll, pitch and angular rates which are all used for navigation and mapping.

The perception sensors provide most of the fundamental information for executing the Challenge tasks. They also provide the situational awareness for navigation and obstacle avoidance. On the sensor frame, a Velodyne HDL-32E LiDAR provides the fundamental range information. This sensor is positioned high to give the greatest situational awareness of the course. Many competition tasks require computer vision to interpret parts of the environment (e.g. symbols, buoy colors). Therefore, the sensor frame also has two Logitech HD 720p USB web cameras to allow real-time on-board processing. To measure the underwater acoustic environment, Bruce has two Aquarian Audio hydrophones connected to a Roland Quad-Capture USB Audio Interface Device capable of sampling at 192 kHz.

A Vaisala WTX-520 weather station on the ASV provides real-time wind-speed and direction data (at 1 Hz) to the on-board control system. Bruce has a range of current and voltage sensors for monitoring the motors and batteries which provides real-time feedback to the main software safety system. Monitoring these variables is critical for battery management and detecting any system fault.

3) Propulsion and Power systems: There are two propulsion systems for Bruce; The primary manoeuvring control is provided by two fixed 80lb 24V electric trolling motors that provide forward and differential steering motion. However, in 2016 it was found that this configuration alone was unable to successfully perform docking manoeuvres with strong side winds. Therefore, Bruce has been upgraded with the addition of two small bow thrusters on the fore section of each pontoon to aid in heading control and station keep.

These bow thrusters (Blue Robotics T200 thrusters) are fixed in position and only provide lateral control. As such a custom software controller was developed to determine the required control allocation to achieve static position hold as well as forward motion. Whilst many teams opt for steerable thrusters or mounted at 45 Degrees from forward to achieve holonomic control [4], [5], our configuration was chosen to maximise forward thrust and speed for normal operation and to reduce operational complexity which may be potential points of failure.

Bruce has two on-board batteries with the AUV carrying its own power system. A larger battery (Torqeedo 26-104) is used exclusively for powering the propulsion system (24V for the trolling motors and via a 24-12 DC converter for the bow thrusters) and are connected to the vessels safety management system (see Section III-A4). A smaller 12V 50Ahr LiPFe battery is used for powering the computers and sensors. A significant challenge in 2016 was managing power whilst on the shore for software debugging without being connected to mains power. As such, the computer battery is connected to

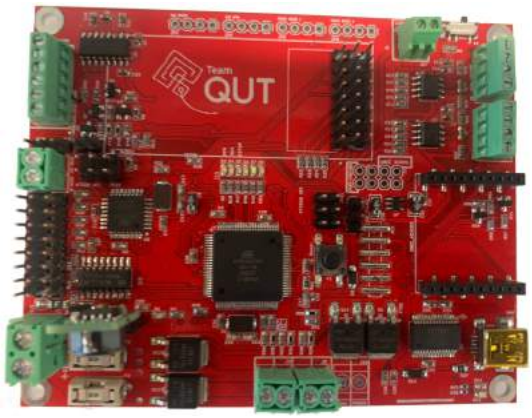


Fig. 3: The custom safety control board developed for Bruce. This board monitors all wired and wireless emergency stop buttons, manages RC and computer demands to the motor controller, and drives the status light system.

two 40W solar panels attached on the top of the payload tray to maintain battery capacity during a working day when under low-load (development) conditions.

4) Safety Management System: Operational safety of the ASV, AUV and ball launcher are paramount considerations in the overall system design. In 2018, the existing safety control system was redesigned and integrated onto a single board to improve overall operational robustness. Figure 3 shows the new safety system board. This board monitors all the E-Stops and manages all power to the motor relays and controls the mandatory status lighting system. A separately powered wireless E-Stop system is built onto the safety board. The board interfaces with a standard RC control handset to provide manual control capabilities and to trigger autonomous control. When in autonomous mode, the safety board directs motor control inputs to the motor controller from the computer.

B. Gusto: Automated Racquetball Launcher

The 'Detect and Deliver' task requires teams to first detect a target and then to deliver a payload (racquetball) within the target area. In 2016, TeamQUT developed Gusto a self-contained visual-servoing electric ball launcher. Gusto delivers racquetballs by feeding the ball through two horizontally opposed counter-rotating wheels. A feeding mechanism can launch the balls at a rate of one ball per second. Gusto is mounted towards the front of the upper deck of the ASV (see Figure III-B) and is connected to Bruces computer via USB.

During 2018, Gusto's entire low-level microcontroller software and ROS node were completely redesigned to improve vision-based window detection and tracking performance, safety and overall reliability. Additionally, a weather shied was created to allow operation in rainy conditions. The result is a robust system which demonstrates consistent delivery of balls into the smaller target area.

C. Thunder4: Autonomous Underwater Vehicle

A new task, the "Underwater Ring Recovery" task was introduced for the 2018 Maritime RobotX Challenge. This task

Fig. 4: Gusto – the automated racquetball launcher for the "Detect and Deliver" task.

Fig. 5: The Autonomous Underwater Vehicle (AUV), called *Thunder4*, used for the underwater ring challenge. This is a tethered hybrid (ROV/AUV) version of the RangerBot AUV.

requires the retrieval of yellow polypropylene rings suspended from an underwater PVC structure. TeamQUT's approach to completing this challenge is to use an Autonomous Underwater Vehicle (AUV) called *Thunder4* as shown in Figure 5. Thunder4 is a modified RangerBot AUV developed in collaboration between the Queensland University of Technology and the Great Barrier Reef Foundation. The AUVs sensors consist of two calibrated stereo camera pairs (forward and downward looking), an IMU, pressure sensor and a GPS. Two computing systems (Nvidia TX2 and a small embedded CPU) process all sensor streams and execute missions. It has six thrusters allowing full 6-DOF motion control and two on-board batteries giving an endurance of over 6 hours.

Whilst Thunder4 is capable of autonomous untethered operations, for the RobotX Challenge a custom tether modification has been made to the base platform to allow information sharing to the ASV using ROS. The tether also acts as an

Fig. 6: The Thunder4 AUV in the lowered position ready to drive out of the lifting cradle.

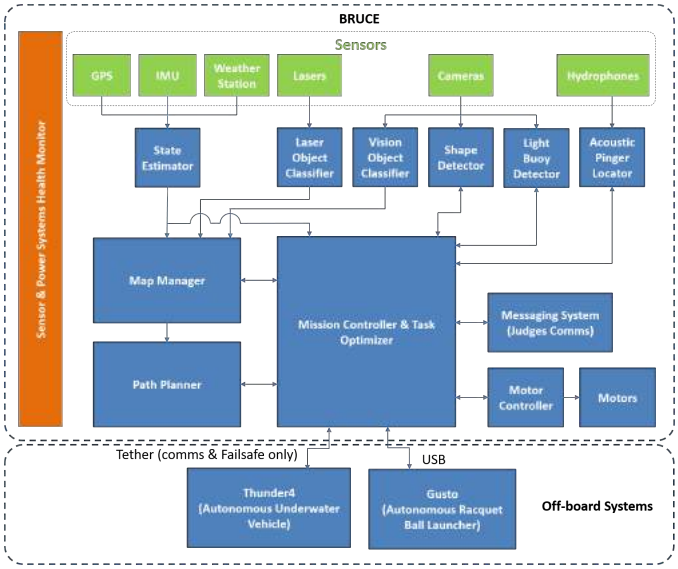


Fig. 7: Overview of the 2018 software architecture for Bruce.

emergency stop system which when disconnected triggers motor power shutdown. To facilitate launch and recovery of the AUV including tether management, a custom scissor lift mechanism was installed on Bruce (see Figure 6). In order to collect the rings from the subsurface structure, a passive latching mechanism is attached to the AUV which catches the ring when driven towards it. Once latched the AUV then reverses to release the ring from the structure.

D. Software Architecture

The software architecture developed for Bruce is shown in Figure 7. It is an event driven, cross-platform structure built on the Robotic Operating System (ROS) [6]. Other popular frameworks for marine vessels exist, such as MOOS [7], however, cost-benefit-risk analysis led to ROS being chosen as it provided seamless integration and extension of the individual software modules representing sensing, localization, planning, and control as well as the different robotic platforms (e.g. Thunder4). It also facilitates more rapid software development within the project team capabilities as it caters to the programming languages preferred by the individual team members (C++, Python, Java, and MatlabTM).

The software platform executes actions in parallel and asynchronously. The individual sensor modules provide data asynchronously to the state estimator and image, laser and acoustic classifiers. The State Estimator maintains the pose of the ASV for localization and control. The State Estimator, laser, and vision classifiers (for buoys) all feed the Map Manager which maintains an up-to-date, globally referenced map of real and virtual obstacles as the ASV traverses the courses. The Path Planner produces viable (efficient and obstacle free) trajectories for the ASV based on the obstacle map and the desired waypoints provided by the Mission/Task Optimizer (see Section VII).

The execution of the missions is performed by the Mission/Task Optimizer block. This was completely rewritten in 2018 to facilitate a new mission structure allowing dynamic execution of sub-tasks within main tasks for achieving high-level objectives and overall system function. The decision support structure within the state machines is parameterized allowing the controller to retry, abort or skip specific tasks, or components therein. The Mission Controller also determines the required motor control inputs sent to the Motor Controller for driving both the primary and bow propulsion systems.

To facilitate load management on the primary CPU (Intel Core i7-4790K, 4.00GHz), particularly for computationally expensive computer vision tasks, the Mission Controller can put each classifier block into an idle state until required. The Thunder4 AUV has two independent computing systems (see Section III-C) and communicates via ROS to Bruce using the multi-master library [8]. Finally, the Mission Controller communicates with the Messaging System block which provides the task-specific messaging interface to the remote Judges display and the technical directors network. The Sensor & Power Systems Health Monitor provides a system wide assessment of the operational state of the ASV and AUV including monitoring motor and battery currents and voltages, and individual sensor performance with the ability to restart modules when necessary.

IV. Mapping, Path Planning and Obstacle Classification

This section describes the 2018 approach to obstacle detection and classification to increase the robustness of task execution.

A. Obstacle Detection

In 2016 we based our obstacle detection and mapping approach on a probabilistic Occupancy Map [9]. This method is fast and adapted to the computing hardware available on the ASV. However, we encountered issues with false obstacles generated by ripples on the water. Our previous attempt to reduce false obstacles was based on height and intensity

thresholds but was only partially successful. In 2018 we decided to keep the Occupancy Map implementation, but we overhauled our obstacle detection approach.

Our new approach is based on work by Suger et al. [10], where we analyze the raw Lidar points to identify which points are hitting a solid obstacle (e.g. competition structures) and which points are from false obstacles (e.g. water ripples). For each Lidar point, we compute the *stepHeight* and *incline* angle. The *stepHeight* of a particular LIDAR point is the vertical distance between the two closest neighboring points of the same azimuth angle, while the *incline* angle is calculated between the line that connect those two points and the horizontal plane of the LIDAR. A LIDAR point is considered as hitting an obstacle if both *stepHeight* and *incline* values are greater than respective thresholds. This obstacle detection step is key to reject spurious outliers from ripples in the water but reliably detect competition structures at various distances. Following this process, each LIDAR point hitting an obstacle is integrated into the probabilistic Obstacle map using Bayes update rule. This new mapping process is described in depth in our latest journal paper [11].

Finally, a traversability estimation step is performed to translate the obstacle map into a cost map of same size and resolution. Any cell of the obstacle map with a probability greater than a defined threshold is considered non-traversable (high cost) in the traversability map so that a path cannot be planned through this cell. A safety disk is added around non-traversable map cells to promote the planning of safer paths.

B. Path Planner

The Path Planner is used by the Mission Controller to provide a safe path to any position on the map from the current position of the boat. We use a tree search approach in which each of the map cell is a node. The algorithm implemented is a version of the well-known A-star algorithm [12]. This algorithm is optimal (will return the best path) and complete (guaranteed to find a path if it exists) while maintaining a very high performance. Other algorithms were considered, notably D-star [13] which has the same search properties as A-star and is faster to execute but is more computationally expensive. As the boat dynamics are relatively slow, a new path is only needed every second on average. With our A-star implementation being able to produce a path across the entire course in less than one seconds, it was chosen to conserve computational power.

The two main features to define for the A-star algorithm are the cost function, and the links between the tree nodes (map cells). A-star uses a cost function to guide its search while exploring the map cells, the definition of this cost function is crucial and defines the behavior of the search. The cost of a cell reflects the difficulty for the boat to reach that cell. The algorithm explores the cells with the lowest cost first which influences the search speed and the path behavior. The links between the different tree nodes (map cells) indicate which cells of the map are available from any particular cell. These links play a very important role in reflecting the boat dynamics to produce a realistic path for the boat to follow (e.g. a boat cannot do a sharp turn at high speed). Therefore, a dynamic model of the boat was built and improved over time using different GPS trajectory data from field trials. This model accounts for the current boat speed and proposes different turning behaviors. It also incorporates a model of the boat acceleration and deceleration capabilities. This path planning implementation allows Bruce to safely navigate on the course and through challenging obstacle fields.

C. Obstacle Classification

Our obstacle classification framework is based on both structural and appearance features extracted from LIDAR data. LIDAR measurements are accurate and robust to ambient lighting conditions and so are the features derived from them, making them especially useful in outdoor scenarios.

From the Obstacle Map we extract a list of obstacles by grouping adjacent occupied cells. For each obstacle we compute the following features based on the information in the group of cells that it represents:

- LIDAR intensity mean
- Maximum Height
- Maximum Width

LIDAR intensity mean is an appearance feature computed statistically from the cells belonging to each obstacle. This feature is used to differentiate objects of different materials (e.g. wooden structure and metal structure will have different LIDAR intensity returns). Structural features (Maximum height, and maximum width) are also computed from Velodyne data to obtain a sense of proportion of the elements and used to make the distinction between obstacles of different physical shapes and sizes.

We perform classification on the LIDAR features using the Random Forest algorithm introduced by Breiman [14]. This classifier is fast to execute and was used successfully in multiple laser-based classification cases [15], [10]. The classifier is trained using hand-labeled LIDAR data recorded during previous RobotX competitions. The different objects of the competitions are split into four classes: *Round Buoy, Straight Buoy, Light Tower structure, Detect and Deliver/Dock structure*. These classes were chosen to be informative to the mission planner to complete the competition tasks while being recognizable solely from LIDAR data. Fig 8 depicts an example of a scene from the obstacle field task with buoys and totems correctly identified.

V. Vision

A robust vision system is required to accurately detect the properties of the coarse elements such as the color of all the buoys, color and shape of the symbols/placards in the detect and deliver as well as the docking tasks, and the color sequence of the light tower in the scan the code task. The following sections provide a brief overview of the vision systems developed for these tasks.

A. Buoy Color Detection

Whilst the buoys are able to be detected by the LIDAR, their color is determined by the vision system. Furthermore, due to

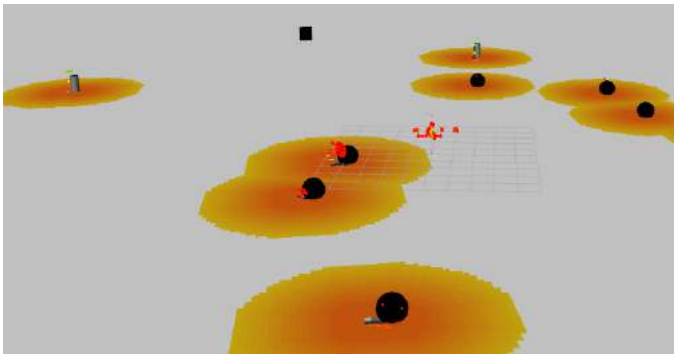


Fig. 8: Predicted obstacle types superimposed on the obstacle map and raw LIDAR points. Cylinders are totems, spheres are obstacle field buoys, and squares are unknown.

the black and shiny nature of the obstacle buoys, the LIDAR has difficulties detecting them and so the vision system is also used to identify these buoys.

Due to unpredictable changes in lighting conditions throughout the day over the course of the challenge (see top row of Fig. 9), a more traditional approach based on manually selected image thresholds is not viable. Instead, a water detection system is proposed whereby the water was identified and removed with and the remaining objects tested for color and shape. The water is isolated by apply a Gaussian filter to the image, followed by Canny edge detection. The obstacles and buoys produced edges, allowing identification of areas of water as shown in the second row of Fig. 9. The water can then be removed by applying a color threshold. The resulting blobs are then filtered using size and ratio. The buoys were separated from the obstacles because they were taller and then tested for color. The HSV color space is used for determining the color of the buoys. For each blob, their mean hue was compared to hue values for red, green, blue, and yellow and the color of the buoy was selected as the closest match, with the difference used to calculate a confidence value. The resulting bounding box, color, and confidence was provided to the mapping system as shown in the example in Fig. 10.

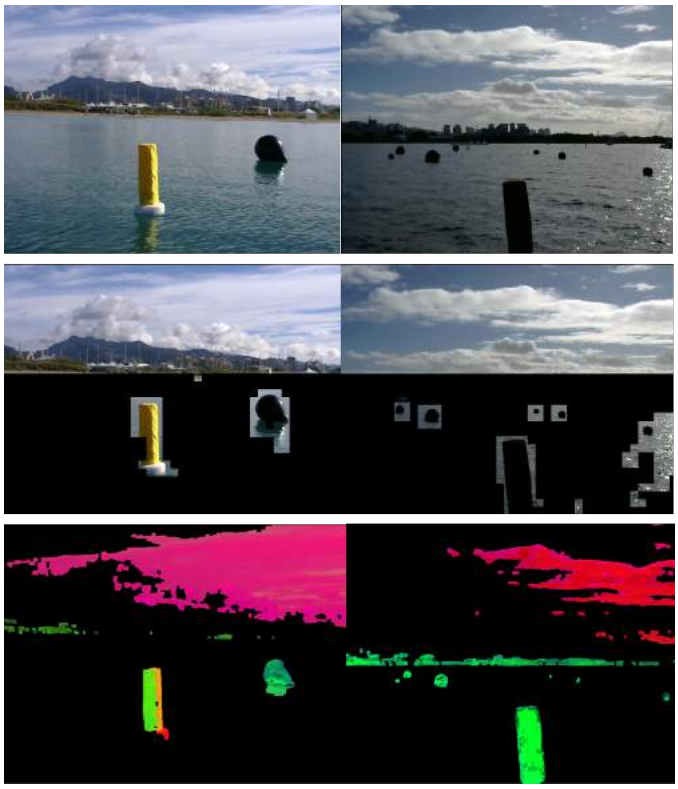


Fig. 9: An example showing work flow for vision-based buoy detection in both sun (left column) and shade conditions (right column). Top row: original images; Middle Row: results after Gaussian filter and Canny Edge detection, Lower Row: the resulting threshold image colored in HSV format.

B. Symbol Detection and Identification

The symbol detection and identification algorithm was built on the robust code-based system developed for the 2016 Challenge [16]. The OpenCV and ROS-based module is capable of detecting three colors (red, green, blue) and three shapes (circle, triangle, cruciform). It was updated for improved scale invariance (allows detection at greater distance) as well as improved robustness to symbol orientation (e.g. if the triangle was sideways facing or upside down). In addition, it reports back the confidence score for each detected for use in mission execution. The algorithm performs well in practice and is capable of detecting all nine symbol combinations in a single image.

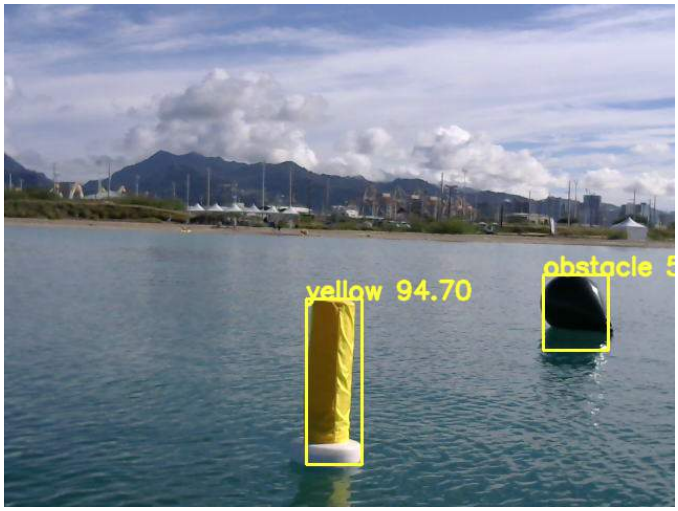


Fig. 10: An example image showing the output from the vision-classification scheme highlighting the region of interest, its predicted color and the associated confidence.

C. Light Tower Sequence

Our approach for Scan the Code task is based on work from both the 2014 and 2016 challenges. The computer vision algorithm was used for detecting the sequence consists of two stages; (1) LED panel segmentation and color identification, and (2) a confidence based sequence algorithm. To segment the panel color from the rest of the image, a custom process built on OpenCV was developed for precise filtering of features based on size, aspect ratio, angle and circularity. An adaptive thresholding approach using parameters selected from experimental data allows robust detection in a wide range of lighting conditions. In 2018, the yellow color was removed by the judges, therefore, the system was reverted to only detect red, green and blue LEDs.

Following LED color detection a custom sequence algorithm is used to build temporal confidence in the color sequence. Once the confidence exceeds a threshold, the sequence is reported to the judges display.

D. Pinger Localisation

The method for pinger detection implemented in 2018 is identical to that of 2016. Here the signals from two Aquarian Audio hydrophones at fixed spacing (across the fore hull section) are recorded and a custom processing algorithm used to calculate the Time-Difference-of-Arrival (TDOA). Using the TDOA, the angle to a pinger can be determined. However, as the detected angle has two possible solutions (positive and negative about an axis through the two hydrophones), information from the Obstacle Identifier is to compare the TDOA calculated angle and ray trace to the buoys to check the validity of the solution. The temporal confidence in the solution is calculated and used by the mission planner.

Figure 11 illustrates the signal processing from one hydrophone on data collected at the 2016 RobotX Challenge showing it can robustly identify the acousitc pinger signal from within the raw input.

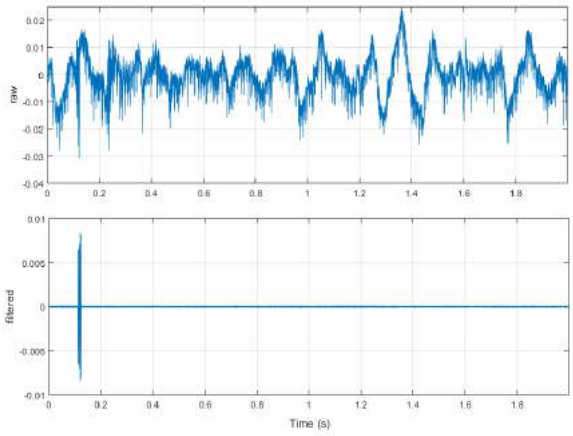
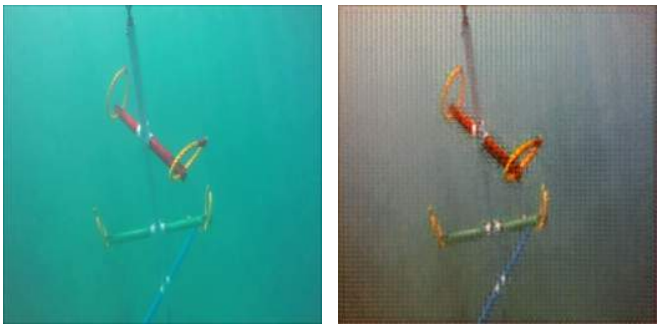


Fig. 11: Example of the raw and filtered recordings from the a single hydrophone of a pinger at 32 kHz taken during RobotX 2016 in Hawaii.



(a) Raw image. (b) Enhanced image.

Fig. 12: Figure 12a shows an example raw image of the underwater rings and frame highlighting the reduced visibility (extracted from video content published by RoboNation [18]). Figure 12b shows the image after processing through the GAN-based image enhancement module.

VI. Underwater Ring Recovery

A significant vision system development for 2018 was for the detection of the rings in the underwater task. Our approach is to use the Thunder4 AUV which has a forward looking color stereo pair to locate the PVC frame and identify rings to allow visual-servoing. The following sections describe the image processing pipeline and general state-machine for task execution.

A. Underwater Ring Identification

The underwater vision system has been designed with a combination of deep learning and classical visual image processing techniques. There are a number of challenges surrounding the use of robotic vision methods in underwater scenes; many of which arise from degraded image quality. Pre-processing image enhancement methods offer improved performance to vision systems, typically by reducing haze, enhancing contrast, or normalising colours. A deep learning-based image enhancement method was found to exhibit competitive performance when applied to challenging underwater scenes. Thunder4's vision system uses recent research by Fabbri et. al. [17] surrounding the use of a generative adversarial network (GAN) to enhance the colour and features present in underwater images. An example of this pre-processing stage of the vision pipeline are demonstrated in Figure 12, where raw image data such as Figure 12a is passed into the pre-trained GAN, producing an enhanced image as shown in Figure 12b.

The enhanced images are then converted to the HSV colourspace, where binary thresholding methods are applied to segment the image into regions of interest. An example output of this stage of the process is shown in Figure 13. To reduce the noise indicated in Figure 13, Gaussian blurring and morphological filters were applied to the thresholded image. The final analysis isolated the rings by using known prior information, (e.g. that the rings are typically found at the horizontal extremities of the detected structure). This method allowed estimates of the location of a given ring to

Fig. 13: The underwater ring segmentation process required multiple iterations. The red pole was not able to be directly segmented out without losing the rings, so is removed using binary subtraction.

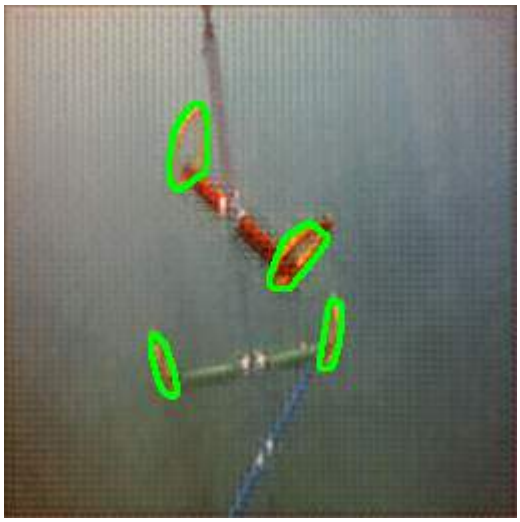


Fig. 14: An example output from the Thunder4 vision system. Rings are segmented (green lines) using binary object masks created by classic thresholding methods. Note the estimated ring shapes are not exact due to the morphological operations during the segmentation removing some fidelity.

be estimated either directly, or through inference based on the detection of a corresponding horizontal pole. Figure 13 demonstrates both methods, where the rings on the green pole are segmented directly, while the rings on the red pole had to be segmented indirectly by examining and subsequently removing the red pole. The resultant ring detection from the Thunder4 vision system is demonstrated in Figure 14.

B. Underwater Ring Collection

The other major design component for the underwater ring recovery task is the AUV's state machine. Evaluation of the low-level movement requirements for Thunder4 provided

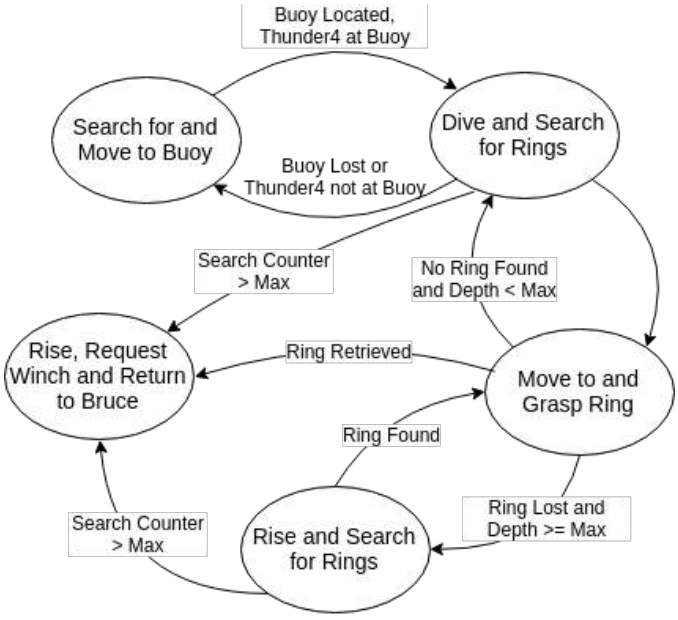


Fig. 15: A high-level representation of the Thunder4 AUV state machine system. Error states and non-critical states have been omitted, and linked states combined for presentation purposes.

insight into the potential states which the robot could be in at any given time. Due to the requirement for the AUV to remain tethered to the WAM-V surface vehicle at all times, Thunder4 required an additional set of autonomy states relating to winching requirements. Figure 15 presents an overview of our state machine system diagram, simplified for presentation purposes, with error states omitted.

VII. Autonomous Mission Execution

In 2018, the entire mission execution stack was rewritten to allow sub-tasks to be dynamically executed within main mission tasks. This has the benefit of simplifying the task primitives (building blocks) allowing easier debugging and complex task creation. Learning's and code from 2016 was used to create the required behaviours, with new functions added to each class including individual tune-able cost functions for the allocation optimizer.

A. Behavior-based Architecture of Bruce

Behavior based robotics is a robot control paradigm which is represented by internal states [19]. Each state is independent from other states while information can be passed between the states. Each state can contain sub-states and the outcome of a single state decides which state is triggered next. States can be encapsulated in sub-states and thereby allowing a hierarchical implementation of complex behaviors. Whilst a number of common architectures exist within ROS (e.g. FlexBE [20] and SMACH [21]), for the RobotX Challenge we partially modelled our architecture on FlexBE in which each state has an event loop which allows to modify the execution of each state dynamically. Each state has several hooks which are

called depending on the event. These events are: start, enter, stop, exit, pause, resume. This is mandatory for the RobotX Challenge because we may wish take over control of Bruce any time, including the option of starting, stopping or cancelling the current task.

The current behaviours (control primitives) on Bruce include: (1) waypoint, (2) station-keep, (3) launch and recover, (4) dock, (5) pinger locate, (6) standoff target, and (7) circle buoy. In 2016, a new behavior explorer was created and repurposed for 2018. The purpose of the explorer behavior is to search the course for any missing information that may be needed to compete a specific task/s and is called by the Task Scheduler when required.

B. Mission Task Optimizer and Decision Making

A core component of the Mission Controller is the Task Optimizer. The Task Scheduler is composed of a mission data initialiser, a mission data recorder, a task information dependency map, and the task scheduler/decision maker. At the start of a mission, the mission data initialiser reads a configuration file that lists all known task-related information such as task scoring and the on-the-day details (e.g. the Acoustic Pinger frequencies) in addition to flags for all of the information which remains unknown at the start of the mission. As information is discovered during the execution of a mission, it is both updated in-memory and written to a parallel date-stamped mission data file by the mission data recorder. The recorded files can be used for restarting missions if required.

The task optimizer is responsible for determining the task sequence and is actioned at the start of each run and during the run after completion of each task. The task sequence is based on the information required by each specific task and the information currently known, the pay-off (maximum anticipated points that can be scored) and risk associated with executing a particular task, the distance costs between tasks, and the remaining time available on the course. As this information is initially incomplete, the task scheduler depends on updates from the information discovery component as the mission progresses. The next task to complete is determined by a voting in which each currently incomplete task votes for the task(s) which can provide it with information. Given a set of n tasks $t_{i...n}$, we define the task score s_i, task information dependency d_{ij} of task t_i on task t_j, task required information q_i, and the estimated risk factor r_i. The task t_{best} to schedule next is determined by minimizing the predicted overall run cost (maximum score). If this cost is above a threshold, the explorer task is executed. The task information dependencies are read from a configuration file at mission initialization, with a flag a flag that allows the tuple to be retained but disabled for rapid configuration on the day.

VIII. Autonomous Marine Surface Vessel Simulator

In 2016 we developed a high-fidelity simulator to assist with the development of the different software components. This was upgraded during 2018 and made open-source [3]. This was a crucial advantage with minimal in-water testing opportunities and no early access to the competition structures. Since the last competition, our simulator was improved with better physics models to obtain more realistic movements of the boat, see journal paper [22]. The new competition structures were also added such as the single detect and deliver/dock structure.

Fig. 16: Screen image from the Autonomous Marine Surface Vessel Simulator with the 2018 Dock and Detect and Deliver task element.

The simulator includes a high-fidelity camera, a LIDAR sensor(Velodyne HDL-32E), IMU and GPS sensors, as well as a buoyancy and physics simulation to integrate the motors. In order to realistically model the effects of reflection and refraction for image simulation, custom rendering techniques and graphics shaders were implemented using the programmable pipeline of OpenGL 3.0. The fidelity and performance of the simulator was benchmarked against the existing best available alternative simulator, V-Rep, to show that the Autonomous Marine Surface Vessel Simulator has better performance and fidelity when simulating marine environments.

This simulator has been used to provide a method of testing the autonomous subsystems on an entire simulated course. Fig. 16 shows an example image of the ASV in simulation going from the obstacle field to the 2018 docking task.

IX. Operator Control System

In order to convey mission and status information of Bruce and the associated systems, a custom operator control system was developed. This system allows rapid assessment of behaviours and debugging. Figure 17 shows example shots of the two screens. The top image shows the task summary page, with the lower image showing the ASV situational awareness page.

X. Conclusion

This paper has presented Queensland University of Technology's autonomous system solution for the 2018 Maritime RobotX Challenge. The complete solution is a system-of-system platform comprising of Bruce (ASV), Thunder4 (AUV) and Gusto (autonomous racquetball launcher). For 2018, key hardware upgrades included an integrated bow thruster system, a refined safety system, and an automated AUV deployment and retrieval mechanism. Key software upgrades include a fused LiDAR and vision object detection and classification

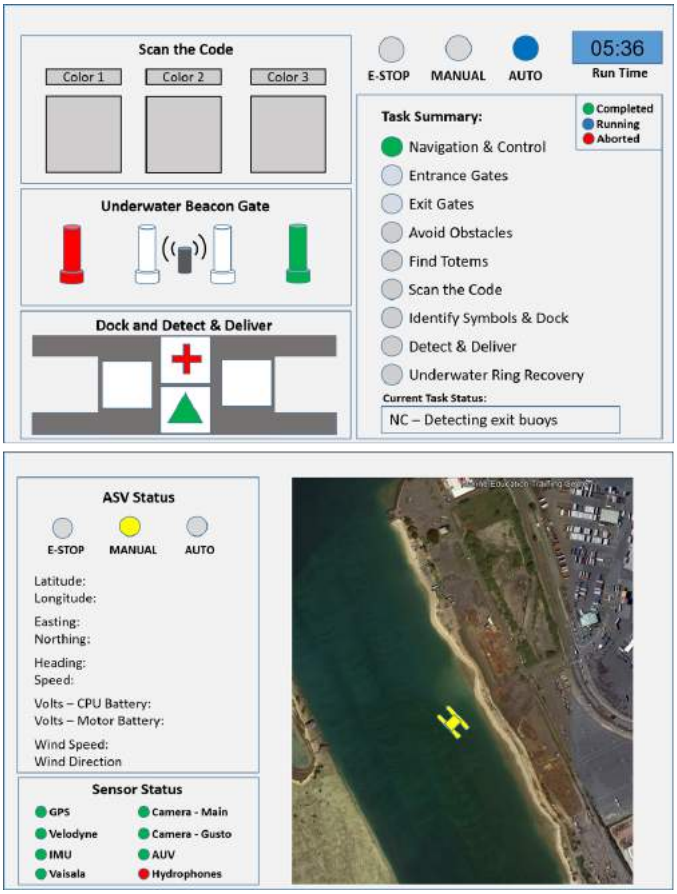


Fig. 17: An example image of the Task Summary page (top) and ASV Situational Awareness page of the Operator Control System.

system, a vision system for underwater ring detection, improved vision system for the ball launcher, and a new mission architecture with task optimizer. The strategy and associated innovations have been successfully evaluated through simulation and on-water testing giving confidence for a strong performance at the 2018 Maritime RobotX Challenge.

Acknowledgements

The team gratefully acknowledges the generosity of our sponsors and supporters in this project. These are; The Australian Government through the Defence Science and Technology Group and the Australian Institute of Marine Science (AIMS), the Australian Centre for Robotic Vision, GFB Robotics Pty Ltd, Cirrus Robotics and BotBitz. Those listed above have provided financial support, equipment loans and discounted products. We would also like to acknowledge the technical support from Paul Rigby, from our industry partner at AIMS, Steven Bulmer from QUT, and the mentoring from previous RobotX alumni Peter Smith and Scott Nicholson. Their guidance on component selection and design and manufacturing techniques has been invaluable to the team.

References

[1] AUVSI, "Maritime robotx challenge preliminary task descriptions," *AUVSI*, 2018. [Online]. Available: https://www.robotx.org/images/RobotX-2018-Tasks_v2.0.pdf

[2] ——, "Maritime robotx challenge task descriptions," *AUVSI*, 2016. [Online]. Available: https://www.robotx.org/images/files/2016-MRC-Tasks-2016-11-28.pdf

[3] P. Smith, "Autonomous-surface-vehicle-simulator," *Github*, 2018. [Online]. Available: https://github.com/P3TE/Autonomous-Surface-Vehicle-Simulator

[4] A. Webb and et al., "Development and testing of the topcat autonomous surface vessel for the maritime robotx challenge 2016," *RobotX*, 2016. [Online]. Available: https://www.robotx.org/images/files/FlindersUni_2016_RobotX_Journal.pdf

[5] D. Frank and et al., "University of florida: Team navigator ams," *RobotX*, 2016. [Online]. Available: https://www.robotx.org/images/files/UniFlorida_2016_RobotX_Journal.pdf

[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[7] P. M. Newman, "Moos-mission orientated operating suite," ., 2008.

[8] A. Tiderko, F. Hoeller, and T. Röhling, "The ros multimaster extension for simplified deployment of multi-robot systems," in *Robot Operating System (ROS)*. Springer, 2016, pp. 629–650.

[9] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *IEEE Computer*, 1989.

[10] B. Suger, B. Steder, and W. Burgard, "Terrain-adaptive obstacle detection," *IEEE International Conference on Intelligent Robots and Systems*, 2016.

[11] L. Stanislas and M. Dunbabin, "Multimodal sensor fusion for robust obstacle detection and classification in the maritime robotx challenge," *IEEE Journal of Oceanic Engineering*, 2018.

[12] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[13] A. Stentz, "The d* algorithm for real-time planning of optimal traverses." CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Tech. Rep., 1994.

[14] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[15] S. Laible, Y. N. Khan, K. Bohlmann, and A. Zell, "3D LIDAR- and Camera-Based Terrain Classification Under Different Lighting Conditions," *Autonomous Mobile Systems*, 2012.

[16] L. Stanislas and et al., "Bruce: An asv solution for the 2016 maritime robotx challenge," *RobotX*, 2016. [Online]. Available: https://www.robotx.org/images/files/QueenslandUni_2016_RobotX_Journal.pdf

[17] C. Fabbri, M. J. Islam, and J. Sattar, "Enhancing underwater imagery using generative adversarial networks," *IEEE International Conference on Robotics and Automation*, 2018.

[18] RoboNation, "Robotx 2018 ring recovery capture 1," 2018. [Online]. Available: https://www.youtube.com/watch?v=oLq_mMXRiS4&feature=youtu.be

[19] R. C. Arkin, R. C. Arkin *et al.*, *Behavior-based robotics*. MIT press, 1998.

[20] S. Maniatopoulos, P. Schillinger, V. Pong, D. C. Conner, and H. Kress-Gazit, "Reactive high-level behavior synthesis for an atlas humanoid robot," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4192–4199.

[21] J. Bohren and S. Cousins, "The smach high-level executive [ros news]," *IEEE Robotics & Automation Magazine*, vol. 17, no. 4, pp. 18–20, 2010.

[22] P. Smith and M. Dunbabin, "High-fidelity autonomous surface vehicle simulator for the maritime robotx challenge," *IEEE Journal of Oceanic Engineering (in press)*, 2018.