Bruce: An ASV Solution for the 2016 Maritime RobotX Challenge

Leo Stanislas, Peter Smith, Brendan Tidd, Peter Kujala, Scott Nicholson, Akira Dawson, Riki Lamont, William Chamberlain, Remy Barton, Markus Eich, 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 2016 Maritime RobotX Challenge. Bruce is a *system-of-systems* comprising the ASV, a self-contained underwater robot, and an automated vison-enabled ball launcher. An upgraded sensor suite and new mission software architecture to deal with information discovery and task sequencing built on the Robotic Operating System (ROS) has been developed to allow completion of all challenge tasks. To facilitate software development and offline testing, a high-fidelity simulation model was developed and integrated with the software architecture. The ASV's control, mapping, and task-specific algorithms have been evaluated both in simulation and through field experiments. Results demonstrating capabilities as well as discussions on lessons learnt are also presented.**

I. Introduction

The Queensland University of Technology (QUT) presents our Autonomous Surface Vehicle (ASV) solution, named *Bruce*, shown in Fig 1 for competing in the 2016 Maritime RobotX Challenge. The objective of the 2016 Challenge compared to previous Challenge's has been to increase the level of autonomy on-board the ASV such that it must now react to new information gained as it executes that Challenge's tasks [1]. 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 [1]. In 2016, 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 Shape Identification" 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 stragety behind the ASV with Section III outlining the key vehicle hardware and software components. Section IV presents the novel Autonomous Marine Vehicle Simulator, Section V overviews the mission execution strategy. Section VI shows some experimental results, with Section VII providing concluding remarks.

II. Design Strategy

The student team's overall development strategy was built from experience gained at the 2014 Singapore Maritime RobotX Challenge and the task requirements of the 2016 Challenge. The team reflected on the design (both hardware and software) and made recommendations on what worked and what did not. These included improving the robustness of target detection and classification through adding confidence metrics. It was identified also that there was need for a high-fidelity simulator to allow off-line development which should also tie into the common software framework (ROS). In terms of hardware there was a need for rapid assembly and deployment during field trips and the wireless communication system needed upgrading to increase range.

Based on these reflections and task requirements, the bulk of the effort was particularly focused on 5 key areas: (1) simplifying and upgrading the hardware systems to improve in-field operations and overall system reliability, (2) the generation of a powerful vessel and course simulator to allow off-line algorithm development and performance evaluation, (3) extending the capability and functionality of Bruce by developing two additional automated systems for the "Underwater Shape Identification" and "Detect and Deliver" tasks, (4) establishing confidence metrics within each detection and classification software element to increase robustness, and (5) creating a mission execution approach to deal with course element prioritization and discovery of new information over time.

Fig 1. Burce – The Queensland University of Technology's Autonomous Surface Vehicle for the 2016 Maritime RobotX Challenge.

III. Vehicle Design

An overview of the Bruce ASV sensors, computing, and propulsion systems, as well as the software architecture is described in the following sections. Additionally, two independent systems that are attached to Bruce (autonomous racquetball launcher and underwater survey robot) are discussed.

A. Hardware Overview

Whilst a bulk of the technical effort went towards the software, simulation and autonomy solutions (Sections III, IV and V), Bruce underwent a complete hardware overhaul to improve robustness and operationalization when in the field. Figure 2 shows the primary hardware upgrades which include a sliding-tray frame for two electronics bays, and a removable sensor frame.

Fig 2. Top: The payload tray showing the sensor frame with Velodyne LiDAR, cameras, GPS and radar attached. Also visible is Gusto, the automated racquetball launcher and the Vaisala weather station. Lower: The sliding tray mechanism to allow shore-side access to the computing boxes.

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. On Bruce, all the electronics boxes are mounted on sliding trays which lock in place. The advantages of this approach are:

- trays to allow beach access without climbing over the vessel
- protection of the computers and batteries from rain and sun
- frees up the top for other tasks (e.g. platform for launching an UAV or supporting other science payloads).

The removable sensor frame provides a greater mounting height for the perception and navigation sensors described in the following sections.

1) Sensors

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 and a Dephi electronically scanning automotive radar provide the fundamental range information. These sensors are 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 Reson TC4032 hydrophones connected to a Roland Quad-Capture USB Audio Interface Device capable of sampling at 192 kHz.

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.

A Vaisala WTX-520 weather station on the ASV to provide 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.

2) Propulsion & Power Systems

In 2016, Bruce was upgraded to two 80lb 24V electric trolling motors to provide extra thrust and overall speed. In addition to the primary trolling motors which provide differential thrust for forward motion and turning, two Blue Robotics T200 thrusters were incorporated onto the tractable hydrophone brackets towards the front of the boat. These thrusters act like bow-thrusters to assist with turning particularly in high-wind conditions.

Bruce has two on-board batteries. The larger (Torqeedo 26-104) is used exclusively for powering the propulsion system. The smaller 12V 50Ahr LiPFe battery is used for powering the computers and sensors. This battery is connected to a 100W solar panel attached to the top of the payload tray for topping up the battery when under low-load conditions.

B. Gusto: Automated Racquetball Launcher

The "Detect and Deliver" task entails the delivery of four payloads to within a target area, as well as detection of the target itself. The strategy taken by TeamQUT for this task is to propel each of the four balls individually via a novel visual-servoing pan-tilt electric robotic ball launcher, codenamed '*Gusto*'.

Figure 3 shows the automated ball launcher prior to attachment to the ASV. Delivery of the payload, a regulation racquetball, is achieved by pushing the ball through two horizontally opposed counter-rotating wheels separated by a gap fractionally narrower than the width of the ball. An automated ball feed mechanism transfers a single ball from the ball feed cartridge into between the wheels at a rate up to one ball per second. As the ball is squeezed between the wheels, they are propelled forwards by the rotational velocity of the wheels and friction between the contacting bodies (ball and wheels). The design is inspired by commercially available devices seen in sports such as tennis, cricket and baseball. It was also preferred over pneumatic/gas or pre-tensioned launch systems as being electrically driven, it can be completely isolated by an emergency stop mechanism without any stored energy. The manufacture exploited rapid prototyping techniques such as laser cutting and 3D printing.

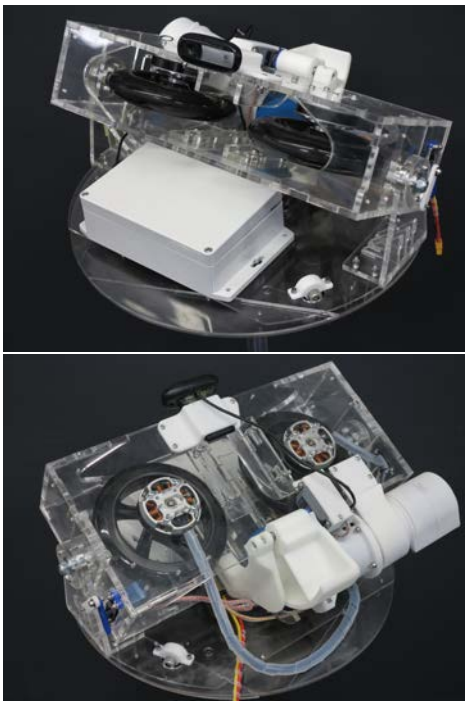

Fig 3. *Gusto* – the automated racquetball launcher for the "Detect and Deliver" task. (Top) Front view showing camera and electroics box, (Lower) rear view showing the automated ball feeder.

The main launcher device is mounted on a two degree of freedom actuated pan-tilt mechanism which enables the payload's launch direction to be controlled within the range of 0 to 35 degrees vertically, and 180 degrees horizontally. Mounted on the upper deck of the WAM-V surface vessel, this provides sufficient trajectory/angle to direct the projectile within the specified target range while the boat is docked perpendicularly against the target buoy.

The launcher is self-contained with its own electrics and camera systems. It communicates to the ASV via serial when required for targeting. Visual servoing using the launcher's own camera is used for tracking the target and directing the launcher accordingly. The target detection system is based on the symbol detection software described in Section V.

C. George: Autonomous Underwater Imaging Robot

The underwater tasks require collecting and processing images for symbol detection and wall tracking. In the preliminary rules, the ASV was not allowed within a prespecified zone of the underwater objects. This requirement inspired our strategy of using an Autonomous Underwater Vehicle (AUV), codenamed '*George*', to remotely (whilst tethered to the ASV) roam away and search an area. This exclusion requirement was later dropped by the Challenge organizers, however, we continued to pursue this approach due to initial feedback from the RobotX organizers that the water visibility may not be sufficient to see the bottom. Our approach provides a means to autonomously control its height above the seafloor which may be advantageous in low-visibility conditions.

Figure 4 shows George during testing in a pool. It is a self-contained AUV with all on-board power, motor controls and computing for image-processing and control. It is propelled by 4 x T100 Blue Robotics motors. Our custom AUV boasts 3 degrees of freedom allowing our vehicle to move just below the surface of the water to a target location, and dive down to an appropriate depth for image capture. A host of sensors keep track of position (GPS when at the surface) and orientation (IMU), as well as depth below the surface of the water (Blue Robotics pressure sensor), and height above the sea floor (underwater altimeter).

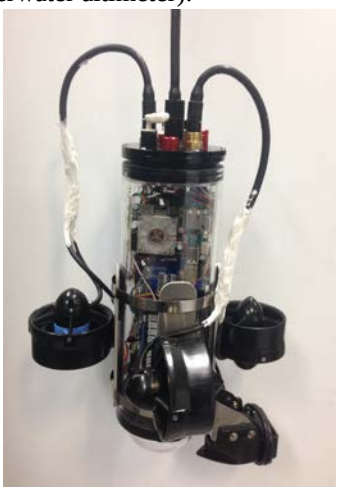

Fig 4. George – the autonomous underwater robot to provide imagery and symbol detection for the "Underwater Survey" task.

An Arduino MEGA accepts incoming data from each of the sensors, parsing the information into NMEA strings,

which are then sent via USB to the on-board CPU running a Linux system. The AUV communicates with the surface vehicle via a waterproof tether and employs a simple winch system with high strength fishing line for deployment and recovery.

Our AUV performs on-board image processing using an Odroid XU4. Images acquired from a Microsoft Lifecam Cinema USB camera are passed through a custom algorithm that improves image clarity (dehaze) and color corrects [2,3] prior to feature detection. Shape priors are then used to predict the presence of target shapes for the 'Underwater Survey' task, and combined with color thresholding for the "Find the Break" task. Example results are shown in Section VI.

D. Software Architecture

The software architecture developed for Bruce is shown in Fig 5. It is an event driven, cross-platform structure built on the Robotic Operating System (ROS) [4]. Other popular frameworks for marine vessels exist, such as MOOS [5], 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. 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 Matlab™).

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 Controller (see Section V).

The execution of the missions is performed by the Mission/Task Controller block. This block maintains a series of state machines for achieving high-level task 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 auxiliary 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. Finally, the Mission Controller communicates with the Messaging System block which provides the task-specific messaging interface to the remote Judges' display.

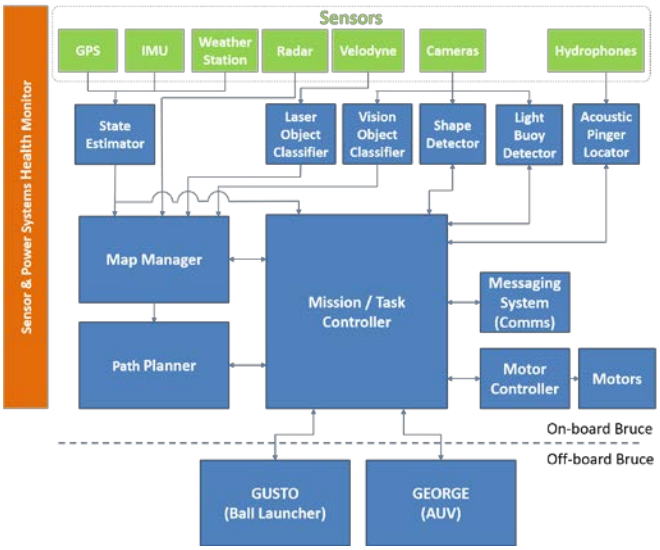

Fig 5. Overview of the Bruce ASV software architecture.

The Sensor & Power Systems Health Monitor provides a system wide assessment of the operational state of the ASV including monitoring motor and battery currents and voltages, and individual sensor performance with the ability to restart modules when necessary.

E. Mapping and Path Planning

1) Obstacle Map

The obstacle map is based on a 2D occupancy grid [6]. The area being mapped is discretized into fixed size cells with values reflecting the probability of occupancy given the Velodyne LiDAR data. Other approaches were considered such as octrees [cite: Armin Hornung OctoMap: an efficient probabilistic 3Dmapping framework based on octrees], an efficient 3D mapping algorithm, but was considered overkill for a 2D marine environment. Another approach, which was used in the previous RobotX competition, implements a hashmap to store each occupied cell of a 2D map only saving occupied cells in memory [7]. This approach has the advantage of being very memory efficient and scalable, but is less efficient in accessing cell values. Prior knowledge of the course size and abundant computer memory in the boat made a 2D occupancy grid the most viable option.

The link between sensor data and the map is managed by ROS coordinate frame transforms. Right handed coordinate frames are positioned on the main elements of the system such as the origin of the map, the base of the boat, and the center of the Velodyne. The GPS position of the boat (Easting (X_b), Northing (Y_b), Heading (H_b)) provides a transform between a world coordinate frame and the boat coordinate frame. The map is positioned in the world with a transform between the world coordinate frame and the map coordinate frame using pre-determined Easting, Northing and Heading values (X_m,Y_m,H_m). Finally the Velodyne coordinate frame is positioned with respect to the boat

coordinate frame using known hardware dimensions and Inertial Measurement Unit (IMU) rotation values to account for roll and pitch.

The occupancy probability of each map cell within a maximum range threshold is updated after every new Velodyne scan (10Hz). This maximum range threshold is set to only include the most reliable part of the Velodyne's scan data. The map cell values outside of this maximum range threshold are untouched and act as memory of the course obstacles. A minimum height threshold was set just lower than the smallest obstacle to limit the impact of spurious LiDAR returns from the water. Due to the movement of the boat, this height threshold could not eliminate completely spurious returns from the water, only limit them.

Multiple approaches to update map cell values from sensor data were tested during multiple field trials. The main difficulty in this process is eliminating spurious data points from the water in the LiDAR scans.

The first model was resetting the map cell values at each new Velodyne scan and increased cell values for each Velodyne point hitting them. This approach appeared to be too sensitive to spurious data from the water therefore complicating the path planning process.

The second model incorporated the fact that a spurious return from the water is rarely present in two consecutive LiDAR scans. This model compared two consecutive LiDAR scans and only considered the map cells that received points in both scans to be occupied. This method was robust to spurious returns from the water, however, only publishing a map every two LiDAR scans halves the update rate of the map to 5Hz which may be problematic when the boat is moving at 2.5ms^{-1} because it cannot map the area fast enough.

The third and best performing model is an improved version of the first one. Instead of resetting the map cell values at each iteration, the values are decreased by a step value "dec_step". For each cell hit by a LiDAR point, its respective value is increased by a step value "inc_step" that is greater than "dec_step" increasing the probability of occupancy of that cell over time as LiDAR points hit it. A minimum probability threshold is then used to classify a cell as occupied or free.

Once a cell is classified as occupied, a safety radius is applied around this cell increasing all the cell values around a detected obstacle. This safety radius is then used to compute a path that safely avoids any obstacle. Within this safety radius, the cells closer to the occupied cell have a greater value that the cells on the edge of the radius to mitigate risk of collision if the boat has to pass in between tight obstacles.

This obstacle map is then used to plan a path between the boat and a goal position while avoiding and detecting obstacle types to execute the different competition tasks.

2) Path planner

The path planning algorithm uses tree search in which each of the map cells is a node. The algorithm implemented is a version of the well-known A-star algorithm [8]. 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. This is an improvement from last competition where the search algorithm implemented was Breadth First Search [9] which provides a dramatically poorer performance. Other algorithms were considered, notably D-star [10] 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 5 to 10 seconds. With A-star being able to produce a path across the entire course in less than 5 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.

This cost function is defined as $F(n) = G(n) + H(n)$ where n is the last node on the path, $G(n)$ is the cost of the path from the start to node *n*, and $H(n)$ is a heuristic function that estimates the cost of the cheapest path to the goal. [Hart et al 1968] The algorithm explores the cells with the lowest F value first which influences the search speed and the path behavior.

The cost of the path function *G* is the sum of each map cell value from the start node to the current node, thus going through free cells will give a lower *G* value than going through occupied cells.

On Bruce, the heuristic function *H* is defined by the Euclidean distance from the current node *n* to the goal; consequently, the further from the goal a node is, the higher its heuristic value. A boat speed factor is also added to reflect that at higher speed the goal is reached faster.

The links between the different cells indicate which cells of the map are available from the current 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.

3) Obstacle Identification

To proceed to each competition task, it is important to regularly extract information from the elements around the boat using its different sensors. This information is organized in the form of a list of obstacles of different types (e.g. cylinder buoy, circular buoy, dock, light tower) and their position on the map. A list of obstacles is produced by each

perception sensor and then matched and fused in one global obstacle list wherein each obstacle is associated with a level of confidence.

The obstacle list from the LiDAR is computed by denoting a cluster of occupied cells of the obstacle map as a single obstacle with a centroid position on the map (x,y), a width (w), and a height (h). The obstacle list from the camera is created through buoy visual detection, returning a position of the buoy (x,y) and its color (c). The obstacle list from the radar is based on the position centroids in the map (x,y) directly given in the radar raw data.

IV. Autonomous Marine Surface Vessel Simulator

To assist in the development of the QUT ASV and maximize development time and reduce in-field software debugging, a novel, high-fidelity simulation to test all the autonomous subsystems of the vessel was developed. Experience from the 2014 RobotX Challenge highlighted that although testing systems in local dams provided an invaluable method of verifying the effectiveness of the autonomous systems a number of issues remained, including:

- restrictions on where autonomous marine vessels can operate, limiting testing locations to sites that often required long and time consuming journeys
- a lack of course elements meant that only small sections of the competition course could be constructed at a given time, making complete mission planning impossible
- multiple people are required for field testing along with strict health and safety regulations limiting access, resulting in scheduling conflicts.

The key functions required for a simulator to be an effective model of the QUT ASV included a high-fidelity camera, LiDAR (Velodyne HDL-32E), IMU and GPS sensor simulation, as well as buoyancy and physics simulation to integrate the motors. A search for an existing simulator to efficiently and accurately meet the requirements included investigation of a number of simulators including UWSim [11], Gazebo, and V-Rep [12]; unfortunately, all lacked support in various key components required for the project.

The *Autonomous Marine Surface Vessel Simulator* was created to fill this missing functionality gap in existing simulators. In particular, the software is designed to simulate key sensors including camera, LiDAR, IMU and GPS by implementing the influence of a marine environment on the readings. The focus of this simulator was to realistically and efficiently implement a simulation environment that models, with high fidelity, LiDAR point clouds, buoyancy of surface marine vessels, and the effect that water has on image reflection and refraction. 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. To efficiently simulate LiDAR, a method known as depth buffer sampling was implemented allowing for hardware acceleration through OpenGL whilst also allowing custom shaders to simulate absorption of LiDAR beams below the surface of the water. To implement a buoyancy model, the JBullet physics library was extended to add functionality that applied buoyancy forces to the vessel. 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 from the convenience of the QUT labs, or the comfort of the home. Figure 6 shows example images of the ASV during simulating the docking and obstacle fields.

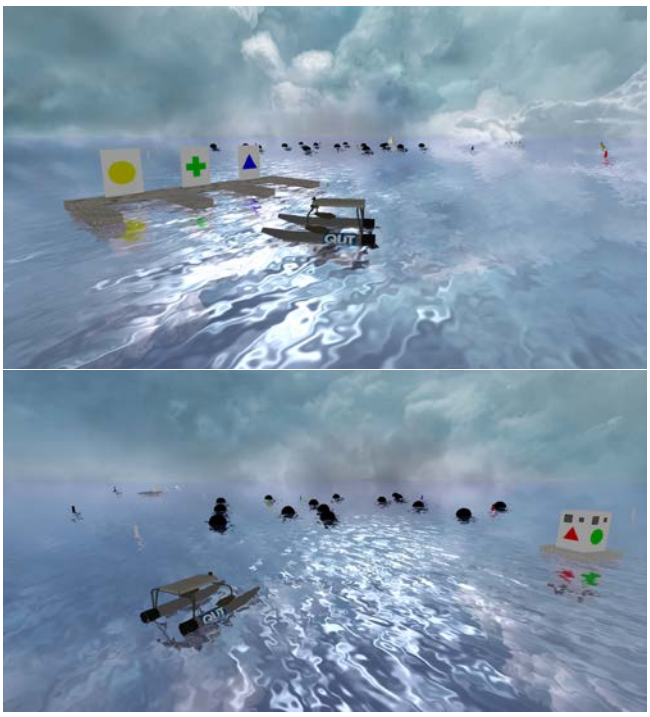

Fig 6. Screen images from the Autonomous Marine Surface Vessel Simulator whilst the ASV is exectuting a docking task (top), and attempting the obstacle field (lower).

V. Autonomous Mission Execution

Missions are performed by the combination of a set of behaviors (control primitives) and task specific detetctors/identifiers. In the 2014 Challenge, the required tasks were specified in order and all required information (e.g. symbol representing docking bay, bounding box of each task location) was specified. However, in the 2016 Challenge, despite being more task elements on the course, only partial information is given by the organizors with potential links between each task to obtain the necessary information to complete the other tasks. This reduces the ability to “guess” the task sequence or elements to detect

within each task meaning that the ASV must discover new information during task execution. The following sections provide an overview of the *Mission Controller* developed for the 2016 Challenge.

A. Behavior-based Architecture of Bruce

Behavior based robotics is a robot control paradigm which is represented by internal states [13]. 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 articectures exist within ROS (e.g. FlexBE [14] and SMACH [15]), for the RobotX Challenge we partially modellied 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) standoff target, and (4) circle buoy. In 2016, a new behavior "*explorer*" was created. 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 Scheduler & Decision Making

The RobotX Challenge emphasizes autonomy and decision-making to complete the course. The vessel must therefore perform autonomous task scheduling given the currently known information, the possible informational pay-offs of executing a particular task, and the course layout. Additionally, due to limitations on sensor ranges and accuracy, the complete course layout cannot be fully determined at the start of a mission. Therefore, the geographic layout of the course has to be built while navigating and completing tasks. This is the role of the *Map Manager* and *Interpreter*.

A core component of the *Mission Controller* is the *Task Scheduler*. 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, 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 scheduler/decision maker is responsible for determining the task sequence, and is actioned after the 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 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 the task score *s*, task information dependency *d* of task *j* on task *i*, task required information *r*, and the estimated cost *c*, the task *t* to schedule next is determined by the vote *v* for a task *i*, calculated from:

$$v_i = \frac{si}{ci} * \sum f(d_{ji}, r_j) * g(r_i) \quad (1)$$

$$f(d_{ji}, r_j) = \begin{cases} 1 \text{ if } r_j \text{ is complete} \\ 0 \text{ otherwise} \end{cases} \quad (2)$$

$$g(r_i) = \begin{cases} 1 \text{ if } r_i \text{ is complete} \\ 0 \text{ otherwise} \end{cases} \quad (3)$$

$$t = \max(v_i) \quad (4)$$

If $t > 0$, then execute task with $\max(v_i)$, otherwise execute the *explorer* task.

The task information dependencies are read from a configuration file at mission initialisation, with a flag which allows the tuple to be retained but disabled for rapid configuration on the day. The information discovery component is linked to the *Intepreter* and *Map Manager* and runs concurrently with task scheduling and completion. In general, the *Interpreter* tries to estimate confidences levels within each of the interpreted elements (e.g. actual buoy color, symbols). Only once the confidence level exceeds a threshold is that piece of information considered known for that particular element. If no task can be excuted due to missing information, the *explorer* task can be executed in an attempt to contribute more information.

C. Task Specific Detectors

The *Mission Controller* executes a set of task specific detectors/identifiers to facilitate information discovery and task execution. On Bruce, these are collectively managed by a construct known as the *Interpreter*. The role of the *Interpreter* is obtain the higher-level information from the course elements for use by the *Map Manager* and *Task Scheduler*. The following sections provide an overview of the key detectors implemented on the ASV.

1) Scan the Code

The "Scan the Code" task of the 2016 Challenge is similar to that of the 2014 challenge. However, a fourth color, yellow, has been added to the existing red, green and blue. 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.

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.

2) Buoy Identification

Although the LiDAR is capable of detecting the majority of buoy metadata (e.g. position, size, aspect ratio) it cannot detect color. The same thresholding and feature detection processes are used to distinguish many different colored buoys in the image. A spatial filter is then applied to obtain an estimated range and bearing of each detected buoy. This information is combined with laser information and combined to add color to the existing metadata. A temporal confidence metric of a detected buoy color is also recorded for use in the mission tasks. The approach has provided robust performance on sunlit and shaded buoys as shown in Fig 7.

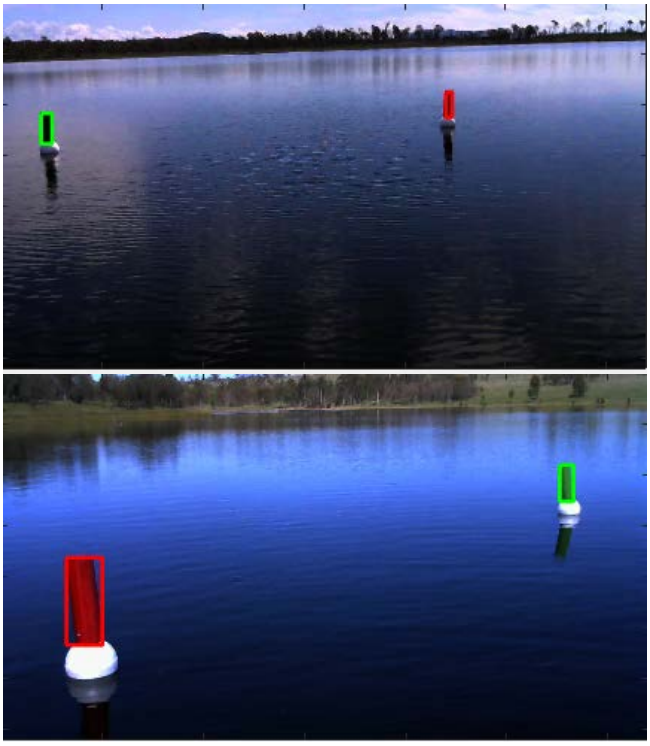

Fig 7. Example images of buoy color detection when approaching buoys from both the sunlit and shaded directions.

3) Symbol Detection and Identification

A robust symbol detection system is required for use in the docking and detect and deliver tasks. The symbol detection and identification algorithm was built on the robust code-based system developed for the 2014 Challenge. In 2016, the OpenCV and ROS-based module was extended to include 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. The basis of this approach was also used for the "Underwater Survey" task as described in Section III. Figure 8 shows example images of correct detection in different scenes and distances.

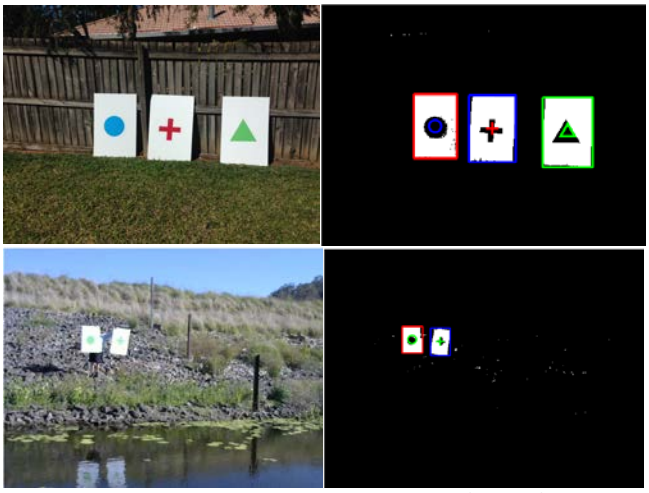

Fig 8. Examples showing symbol detection and classification from the raw images (left) and processed images (right) at different viewing distances and scenes. The squares indicate the shape (red=circle, blue=cruciform, green=triangle) with the marker indicating the detected symbol color.

D. Pinger Localisation

The method for pinger detection implemented in 2016 is identical to that of 2014. Here the signals from two hydrophones at fixed spacing 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 determine. 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.

VI. Experimental Results

In addition to the evaluation results presented in individual detector systems described in Sections V, Bruce had over 30 hours of in-water testing to evaluate the performance and refine various subsystems and task elements. An overview of some experimental results are presented below.

A. Vehicle performance

An evaluation of Bruce's new motor configuration showed an achievable top speed of approximately 2.6 ms^{-1} (compared to 1.4 ms^{-1} achieved by the 2014 competition boat). In differential mode, Bruce can turn on the spot at a rate of approximately 30 Degrees per second. Whilst not a significantly high-speed, it is deemed sufficient for executing the tasks and overcoming the trade-winds expected in Hawaii. It is estimated that the on-board propulsion battery could sustain the top speed for approximately 1.5 hours.

B. Launcher performance

A key requirement for the Detect and Deliver task is to accurately propel a racquetball into a hole in the target. An evaluation into the targeting accuracy of *Gusto* was undertaken to assess repeatability and maximum firing range. This was deemed necessary as Gusto is a relatively low-speed ball launcher (measured to be approximately 12.5 ms^{-1})and the trajectory could be affected by wind or changes in friction between the ball and wheels. Figure 9 shows measured results of anticipated trajectory range with launcher angle which is used by the launcher's controller.

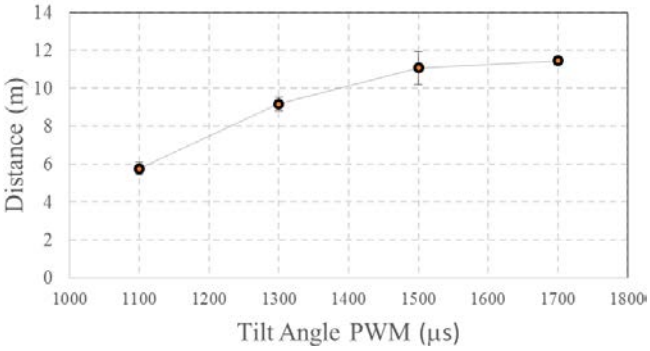

Fig 9. Measured racquetball launch distance against tilt angle (in PWM inputs to servo). The error bars show variability in range for 16 measurements at each angle.

C. Obstacle detection and classification performance

A set of evaluations were conducted to assess the ability of the obstacle detection, to link with the classification systems as well as the path planner. Figure 10 shows an example of the boat correctly approaching a set of navigation buoys. The photo on the left shows the real-time image used by the ASV with the image on the right showing the obstacles with safety zones around them and the planned trajectory through the centre of the buoys.

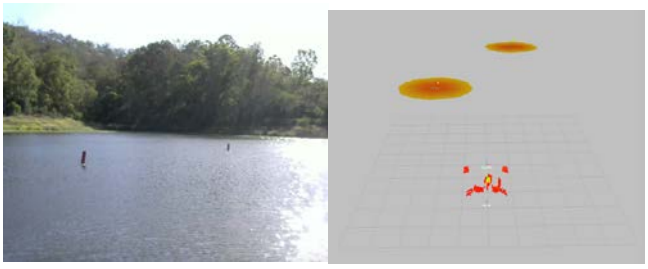

Fig 10. Example image taken from the AUS during field testing (left) and the correctly detected and classified objects and safety regions within the Map Manager.

D. Acoustics

An experimental evaluation was conducted with multiple pingers (Vemco fish tags with individual frequencies in the range 54-60 kHz) attached to buoys in the reservoir. Figure 11 shows raw and filtered time-histories of a detected ping. The top traces shows the results when the boat is moving and motors on, with the lower trace showing a ping when stationary. As can be seen there is significant noise introduced by the motors even though they are at least 3m away from the hydrophones. These results have driven the mission execution software to now turn off the motors briefly to improve signal to noise ratio when looking for the pingers.

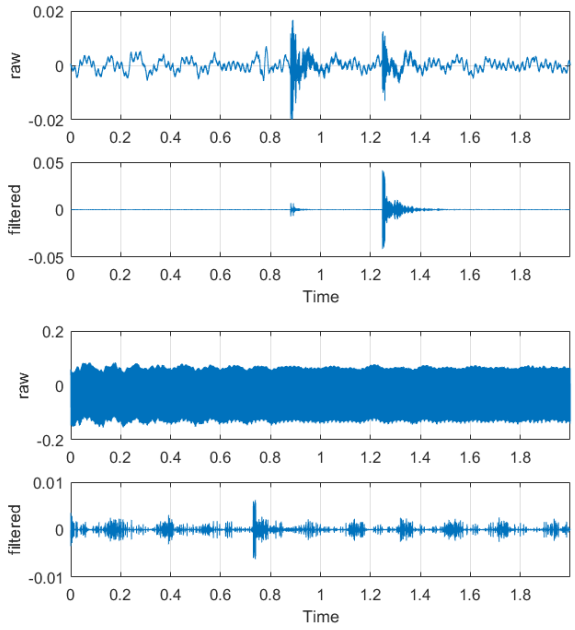

Fig 11. Examples of raw and filtered recordings from the hydrophones of two pingers (57 and 60 kHz) when the motors are turned off (top two traces), and turned on (lower two traces). The noise from the motors can be clearly seen in the lower images. The filtering is a band-pass filter centered at 60kHz.

E. Underwater Vision results

A set of images of an underwater symbol target were collected in Moreton Bay, Queensland, Australia. They were obtained by setting a target down on the seafloor and lowering a camera over it. The visibility was approximately 5m based on a Secchi Disc. Figure 12 shows an example image taken from approximately 2 meters above the target along with the visibility enhanced and symbol detection algorithm. Although the symbol could be detected without enhancement, the overall quality of the image has been improved and allows detection at even greater altitudes.

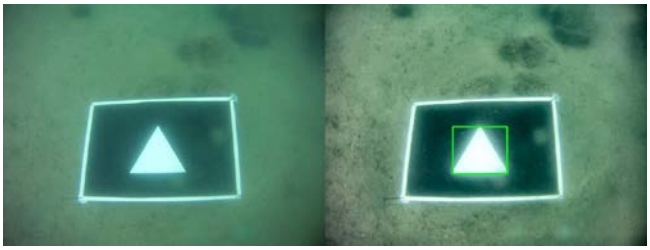

Fig 12. Example image from Moreton Bay, Queensland, showing an underwater target (left). The visibility enhanced and classified image (right).

F. Lessons Learnt

The field campaigns were invaluable to gain new knowledge on how to design for robustness and check the overall fidelity and accuracy of the simulator. Due to the remoteness of the field site and 4WD access requirements, the ASV was dismantled and packed onto a 6 x 4 ft trailer each trip. This allowed us to refine assembly and minimise hardware setup time to less than 25 minutes.

A particular challenge was the lack of course elements available to the team for practice. Whilst the simulator provides a powerful evaluation framework, it does not replace final in-field evaluation. Again due to the remote nature of the test site, it was not possible to install course elements. In the future, it is proposed to construct lightweight "fold-away" replicas of the courses that can be easily deployed and transported with minimum setup effort. It was also identified that for tasks that required significant setup (e.g. obstacle field), the use of virtual obstacles in the obstacle map which can also be used concurrently with live obstacle detection would be beneficial. This is not available in the current version of the software but will be added in the future.

VII. Conclusions

This paper has presented the strategy behind the hardware and software systems that provide a complete autonomy solution for Bruce, the Autonomous Surface Vehicle developed by the Queensland University of Technology. Core to this innovative solution is; (1) An upgraded hardware layout and component integration to improve reliability and facilitate management in the field, (2) the generation of a powerful physics driven simulator to allow off-line algorithm development and performance evaluation, (3) two additional automated systems for the "Underwater Survey" and "Detect and Deliver" tasks, and (4) creation of a mission execution framework to deal with course element prioritisation and discovery of new information. The strategy and associated innovations have been successfully evaluated through simulation and on-water testing giving confidence for a strong performance at the 2016 Maritime RobotX Challenge.

Acknowledgments

The team gratefully acknowledges the generosity of our sponsors and supporters in the development of the Bruce ASV. These are; The Australian Government through the Australian Institute of Marine Science (AIMS) and the Defence Science and Technology Group, the Queensland University of Technology, the Commonwealth Scientific and Industrial Research Organisation (CSIRO) GFB Robotics Pty Ltd, RoPro Design Inc., Blue Robotics Inc., and BotBitz.

Those listed above have provided financial support, equipment loans and discounted products. We would also like to acknowledge Seqwater, the operators of the on-water test site by the team, in addition to the technical support of individuals from our industry partners; Melanie Olsen from AIMS and Ray Russell from RoPro Design. Their guidance on component selection and design and manufacturing techniques has been invaluable to the team.

References

[1] Maritime RobotX Challenge Preliminary Task Descriptions (Version 0.91, Updated 03 November 2016), accessed 10 November 2016 < http://robotx.org/images/files/2016-MRC-Tasks-2016-11-03.pdf>.

[2] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior." in *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '09). IEEE, 2009, pp. 1956–1963.

[3] M. Roser, M. Dunbabin and A. Geiger, "Simultaneous underwater visibility assessment, enhancement and improved stereo", *2014 IEEE International Conference on Robotics and Automation* (ICRA), Hong Kong, 2014, pp. 3840-3847.

[4] Quigley, M., Conley, K. and Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A.Y. "ROS: an open-source Robot Operating System", In Proc. *ICRA Workshop on Open Source Software*, 2009.

[5] P.M. Newman. "MOOS-mission orientated operating suite." Massachusetts Institute of Technology, Tech. Rep 2299.08 (2008).

[6] A. Elfes. "Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation", PhD thesis, Carnegie mellon Univ., 1989.

[7] R. Lamont, S. Nicholson, Z. Renando, C. Dirkis, P. Smith, D. Jakes, J. Vanmali, S. Veitch, A. Chong Bang. "The Endeavour ASV: Hardware, sensor & software overview", [online] http://robotx.org/files/TeamQUT_RobotX_Journal_Paper_2014-10-05.pdf.

[8] P. E. Hart, N. J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in IEEE Transactions on Systems Science and Cybernetics, IEEE 1968, pp 100-107.

[9] S.E. Dreyfus. "An appraisal of some shortest-path algorithms." Operations research 17.3 (1969): 395-412.

[10] A. Stentz. "The D* Algorithm for Real-Time Planning of Optimal Traverses". Technical report, DTIC Document, 1994.

[11] M. Prats, J. Perez, J.J. Fernandez, P.J. Sanz. "An open source tool for simulation and supervision of underwater intervention missions", *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS), pp. 2577-2582, 7-12 Oct. 2012.

[12] E. Rohmer, S. P. N. Signgh, M. Freese, "V-REP: a Versatile and Scalable Robot Simulation Framework,'" IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2013.

[13] R.C. Arkin, An Behavior-based robotics. Cambridge, MA: MIT Press, 1998.

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

[15] J. Bohren and S. Cousins. "The SMACH High-Level Executive [ROS News]", *IEEE Robotics and Automation Magazine*, *17*(4), pp.18-20, 2010.