Nessie IV Autonomous Underwater Vehicle wins the SAUC-E Competition

Francesco Maurelli, Joel Cartwright, Nicholas Johnson, Yvan Petillot

***Abstract*— *Nessie IV* is the award winning autonomous underwater vehicle (AUV) from the Student Autonomous Underwater Challenge - Europe (SAUC-E) 2009. This paper aims to present the robot and the approach to the tasks carried out autonomously during the competition. With its robust design, equipped with sensors providing both vehicle state and environment information, and with a modular software architecture, *Nessie IV* has been proven to be not just a competition vehicle, but an excellent platform for research activities in the Ocean Systems Laboratory, at Heriot-Watt University.**

I. Introduction

The Student Autonomous Underwater Challenge - Europe (SAUC-E) is a European competition between Autonomous Underwater Vehicles (AUVs). It is a yearly event organized by Defence Science and Technology Laboratory (DSTL), the Heriot Watt University and the National Oceanographic Centre of Southampton. Inspired by the US competition from the Association for Unmanned Vehicle System International (AUVSI), the SAUC-E proposes a challenge in a European dimension for the competing teams. The robots need to perform autonomously realistic missions, with no human interaction, showing real time capabilities to successfully complete predefined tasks. This paper focuses on the *Nessie IV* AUV (Fig. 1), developed at the Ocean Systems Laboratory, at Heriot-Watt University, the winning entry of the competition.It was designed to compete in the 2009 edition held at the Qinetiq's Ocean Basin, Haslar, Gosport UK. Its capabilities and robustness go however behind the single tasks for the competition, as it is considered now a well established robotic platform for many different research projects in which the lab is involved. This paper will first describe the hardware designed for the vehicle and then proceed to the software architecture, with an emphasis on the mission of the competition. Innovations with respect to previous works are then highlighted, together with the conclusions.

II. Hardware Design

The vehicle is made up of two 22cm diameter cylindrical aluminium hulls surrounded by a Delrin polymer frame. This cage serves as a mounting point for sensors, protects the contained devices from impact, and keeps the thrusters safely out of the way of human divers. One hull, dubbed the motor hull, houses batteries and H-bridge controllers to drive the thrusters. The other, the PC hull, contains the embedded computers, interfacing and sensing electronics and

All the authors are with the Ocean Systems Laboratory, Heriot-Watt University, EH14 4AS Edinburgh, Scotland (UK). f.maurelli@ieee.org

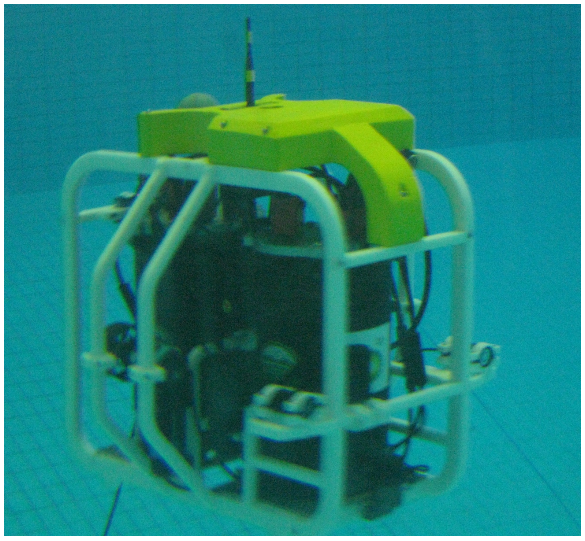


Fig. 1. Nessie IV AUV

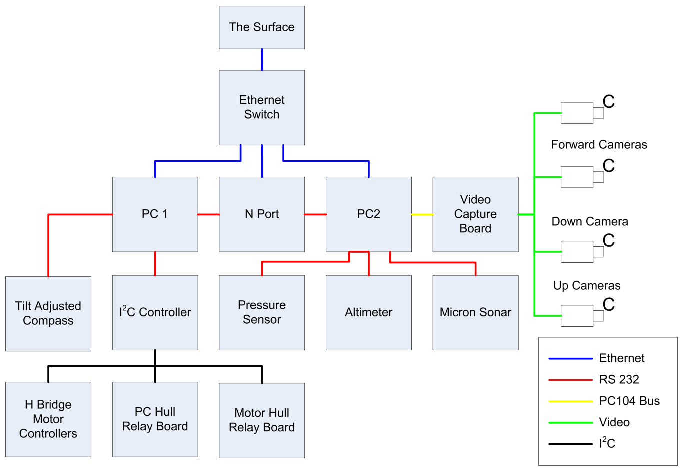


Fig. 2. Hardware Architecture

the batteries to power these. Separating the power supplies and electronics in this way provides a degree of isolation from noise and power fluctuation caused by the H-bridges. It also ensures that even if the thrusters drain their batteries to a low level, the computers remain operational and control is still possible. A schematic of the hardware architecture is given in Fig. 2. The various hardware components of the vehicle will now be described in more detail.

A. PC104 embedded computer

The computers used in the vehicle are industrial MSM800 PC104 embedded PCs. This model was chosen both for its relatively small size and its low power consumption which results in longer battery life and less heat. It has a AMD Geode LX800 500 MHz Processor, 512 mb of RAM, 4 USB 2.0 ports, 2 serial ports and one 100 Mbit Ethernet port. Two of these embedded PCs are used in the vehicle; one for low level sensing and control, and another for video/sonar capture and processing. These are connected via Ethernet. This split ensures that that primary control PC is never starved of resources by the image processing algorithms. To make the computer more robust, flash based solid-state hard disks are used instead of standard magnetic disks. These further reduce the power requirements of the system and make it more robust to bumps and jerks as are expected in a mobile vehicle. In case of boot faults, an Ethernet accessible dual serial port (N-Port) is connected to the serial console of each PC104, which allows the boot progress to be monitored seconds after the vehicle is powered on,without opening the PC hull.

B. *Power supply*

Each of the vehicle's hulls contains two 10 cell Nickel-Metal-Hydride (NiMh) battery packs connected in series to provide a nominal 24 volts. The capacity of the cells is 9 Amp hours, which allows the vehicle to run continuously for several hours. Each hull is fitted with a charging connector that enables the batteries to be charged in place using external battery chargers. The packs are fitted with appropriate fuses to prevent dangerous current levels during charge and discharge. A 25 amp rated 100 metre power tether may be employed for constant running. The tether is driven at 30 volts at the top end, using an earth-protected 20 amp bench supply.

C. *I^2C Interface*

The primary PC104 employs a USB I^2C interface (Total Phase Aardvark) to control the thruster H-bridges and the bespoke relay boards.

D. *Thrusters and H-Bridges*

The vehicle has been equipped with five SeaBotix H.P. thrusters. These thrusters run at a maximum of 4.25 A and 19V. They weigh 700g in air and 350g in water. The five thrusters are arranged thus: port, starboard, lateral, and two vertical. This configuration provides 4 degrees of freedom in control: surge, sway, heave and yaw. Two vertical thrusters are used for additional power in descent/ascent. The thrusters are driven by individual SeaBotix H-Bridges controlled using the I2C interface.

E. *Submersible switches and visual feedback device*

A bespoke submersible module combining reed switches and LEDs was constructed for basic system control and visual feedback. One reed switch is activated by a magnetic key placed in the socket (Fig. 3), serving as the PC on switch. Another can serve as a generic start/stop signal, e.g. for mission start. Three LEDs (red, yellow and green) can display visual feedback to the operator. The red LED shows power state, and yellow and green provide feedback on the mission and targets detected by blinking a binary code. Fast action relays are used within the hulls, which allow the vehicle to be switched from tether to battery power on the fly, with no PC downtime. The end plate of the motor hull also has a reed switch behind it, which activates power to the motor hull via a relay. Removing the clearly visible key from this face plate will cut power to the thrusters.

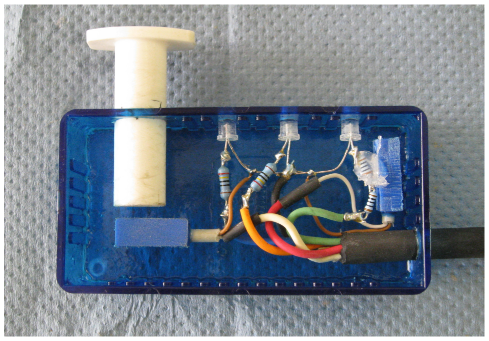


Fig. 3. Switch module

F. *Altimeter*

The Tritech PA500 is used to measure the distance between the vehicle and the floor of the tank. The maximum operating range is 50 metres, well over the depth of the competition tank. It uses an RS232 connection to the PC104.

G. *Depth Sensor*

A Keller Series 33X depth sensor is used to measure the distance between the vehicle and the water surface. For a flat environment, this sensor gives an information very similar to the altimeter, but its precision close to the bottom is higher. Additionally, considering the SAUC-E 2009 event, the assumption of a completely flat environment does not stand (eg. docking box, hovering over a target with a buoy in the middle) It is connected to the PC through a RS485 serial interface and a guaranteed range from 0 to 10 bar (0-90 m in water). Absolute accuracy is 0.005 bar, equal to 0.05 m, with a precision of 0.0002 bar, equal to 0.002 m.

H. *Compass*

A TCM 3 compass measures the vehicle's heading with a precision of 0.5 degrees. It is able to compensate for the vehicle's tilt up to 80 degrees. Firmware calibration routine is provided for hard and soft iron compensation. Connection to the PC is through RS232 serial communication. During the competition, it performed quite well, apart from a magnetic deviation given by a big metallic structure in the middle of the tank.

I. *Cameras*

The vehicle is fitted with four 50 metre rated underwater colour cameras, with optics specifically designed for minimal distortion underwater. They interface to the second PC104 with a PC104 form-factor video capture board. This can capture up to 25 frames per second, at 24 bit colour depth. Two cameras are arranged in pairs, looking forward, for stereoscopic vision. They have been used during the competition for mid-water target detection and tracking and for docking box entrance detection. The other two cameras are one looking down and one looking up. The downward facing camera is used for ground target detection and docking box top detection. while the upward facing camera is not used in the SAUC-E context, but in other missions carried on by the lab.

J. Sonar

An extremely compact Micron DST Sonar is mounted on the vehicle for obstacle avoidance, mapping and localisation. With a maximum range of 75 m, a big portion of the competition tank can be mapped and used for navigation. Vertical beamwidth is $35\,\mathrm{deg}$ and horizontal beamwidth is $3\,\mathrm{deg}$. Communication with the PC is through a RS232 interface.

K. Wireless/wired Communication

A D-Link 802.11G Wireless Access Point is used for wireless communication with the vehicle when on the surface. This was stripped of its case and unnecessary connectors and moulded with a high gain aerial in rubber compound to make it waterproof. The access point is connected to an Ethernet connection and 5V power source in the PC hull via a wet-mateable connector. This same hull connector may also be used to attach an industrial 100m Ethernet tether instead of the wireless access point, if communication is desired for submerged testing.

III. Software Design

A. Architecture

The software architecture for Nessie IV is illustrated in figure Fig. 4. All modules are implemented as separate processes, and communicate using the OceanSHELL system developed in the Ocean Systems Laboratory. This is a lightweight UDP based communications protocol for distributed, modular systems. Each module process is monitored by a *watcher* process, which will restart the module if it fails. The software modules used in the vehicle will be described in more detail in the following sections.

B. Autopilot

The motion of an underwater vehicle is described by a combination of six velocities (surge, sway, heave, yaw, pitch and roll). These constitute the vehicle's six degrees of freedom. In most hover capable underwater vehicles, only surge, sway, heave and yaw are controlled, maintaining pitch and roll approximately zero. This happens also with Nessie IV. The autopilot control system is comprised of a bank of cascading Proportional, Integral, Derivative (PID) controllers, and an axis force to thrust controller. In each of the controlled axes, the current position and set point are fed into the *position* PID controller which outputs the velocity set point. This combines with the current velocity to feed the *velocity* PID, which outputs the desired thrust to meet that velocity. The control system functions as the core of the Autopilot software system. The role of the autopilot is to accept navigation waypoint commands from the mission controller, and send appropriate thruster commands in to manouvre the vehicle to the waypoint.

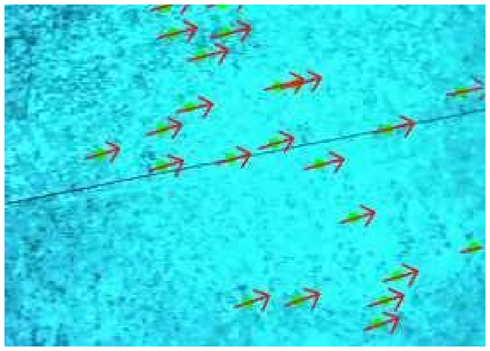


Fig. 5. Optical flow estimation with Lucas-Kandade method

C. Navigation

The navigation system designed and implemented for Nessie IV is composed by two main parts: motion estimation and global localisation. Motion estimation is performed with a so-called *Video Velocity Log*, using the down-looking camera. A hydrodynamic model of the vehicle has been designed and used for motion estimation, when there are not enough features on the ground to estimate the optical fow. Global localisation is necessary to correct the incresing drift of the motion estimation. For this section, the sonar has been used and a particle filter based localisation algorithm has been successfully integrated in the vehicle.

1) Video Velocity Log (VVL): Digital video cameras capture motion by taking a series of frames in particular time steps (frame rate). From these sequences the motion of the objects in the scene can be determined by estimating the differences from one frame to the next. The spatial differences between pixels in the current image and corresponding pixels in the previous images are expressed through motion vectors. The algorithm for estimating the optical flow between two frames of the video stream used for the VVL is the Lucas-Kandade iterative image registration technique [1]. This method uses the spatial intensity gradient and a Newton-Raphson iteration to find matches between feature points. The function is computationally cheap and can deal with rotation and other distortions of the images. The main advantage of the method is that it assumes that the shift between two images is small. This assumption is certainly true for the available frame rate and the speed of the AUV. We are using the pyramidal implementation of the Lucas-Kanade feature tracker from the OpenCV-library. Fig. 5 shows the motion vectors of the optical flow between two frames estimated with Lucas-Kandade-method. Details about this technique can be found in [2].

2) Global localisation: AUV localisation is a subject widely explored, due to the difficulties of the underwater environment. GPS signal does not propagate underwater, and it is thus very difficult to have a reliable global position. If

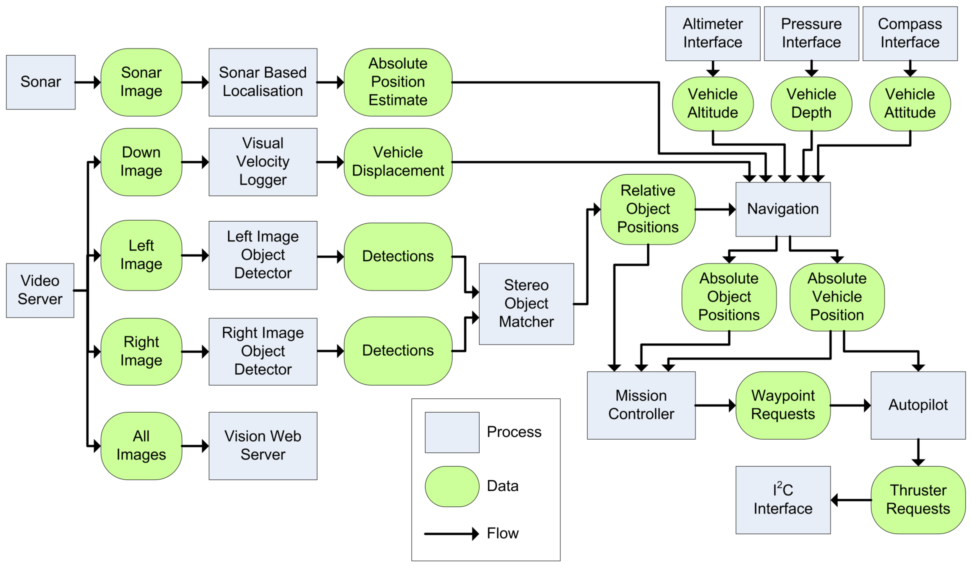


Fig. 4. Software Architecture

the vehicle is moving in a known environment, however, it is possible to analyse the sensor data and estimate the position and orientation with respect to the environment. The chosen approach is an improved particle filter algorithm, using sonar sensor measures, already tested succesfully in different scenarios [3]. A Particle Filter samples the probability distribution function in a discrete number of samples (particles). The more dense the particles are in a specific portion of the state space, more high is the probability for the vehicle to be in that state.

Two features are very helpful in our approach: the capability to recover from wrong convergence, due to the instantiation of a portion of random particles and the capability to deal with incomplete and non perfect map.

The first feature is very helpful, because a wrong convergence could be potentially very damaging, while the second one fits very well the SAUC-E context. Dimensions of the pool are given, but not the position and orientation of the objects inside the pool, which are therefore not in the a-priori map.

D. *Visual Object Detection*

One of the goals during the SAUC-E competition is to be able to localize different objects in the tank, and once they are recognized and localized, interact with them. The Nessie IV vehicle uses video cameras for this purpose, as visibility in the tank is very good, and cameras offer faster frame rates than our sonar system. We found 5 frames per second for the video processing was sufficiently fast to achieve the tasks. As the algorithms were hand-coded for speed, the video system could have run at a faster frame rate, but we chose to save the CPU cycles for other processes. Some investigations have been performed with the sonar as well, with very promising results. The sonar target detection has been considered as a backup detection algorithm, in case of unexpected failure of the video one. The video capture board used in the vehicle provides images with a resolution of 348 x 276 pixels. For image processing, the Intel Open Computer Vision Library 1.0 is used. This implements many common image operations, allowing robust detection algorithms to be created with the minimum of programming overhead. Several different methods are used to detect the objects, described below.

1) Ground Target: The competition includes a black circular target on the bottom of the tank, with an acoustically bright ball mounted on the centre. This target must be identified, and the vehicle must hover above it at a distance less than 30 cm. At first, Canny algorithm for edge detection has been applied. Then circularity and dimension constraints are applied, in order to correctly identify the target. An example detection is shown in Fig. 7.

2) Docking target: AUV docking is an area with increasing interest in the research community. Autonomous docking is very beneficial for offshore applications. These kinds of operations are now performed by ROVs - Remotely Operated Vehicles, with consistent limitations, due to the tether, management and cost associated (support vessel, specialised pilots...). Some solutions for the navigation to the docking station have been proposed in [4], [5], [6]. The general idea proposed by the authors is to use sonar for mid-range navigation and vision for short-range navigation. The sonar approach is particle filter based for vehicle state estimation with respect to the docking box, while the vision position algorithm calculates the distortion of a known pattern on the image, inferring the relative position and orientation of the vehicle. As the competition took place in clear water, a vision-only approach was enough.

The detection of the docking box top works with Canny edge detection, plus a thresholded segmentation, in order to extract the filled top of the box. Hough Line detection and corner extraction help to determine the orientation of the vehicle with respect to the docking box. Then the vehicle

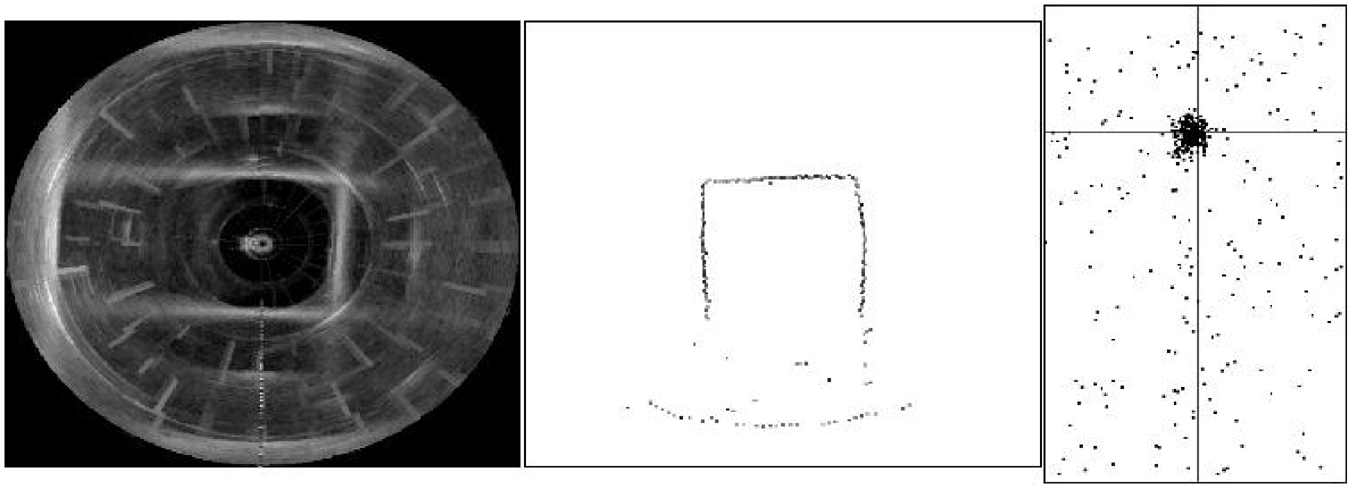
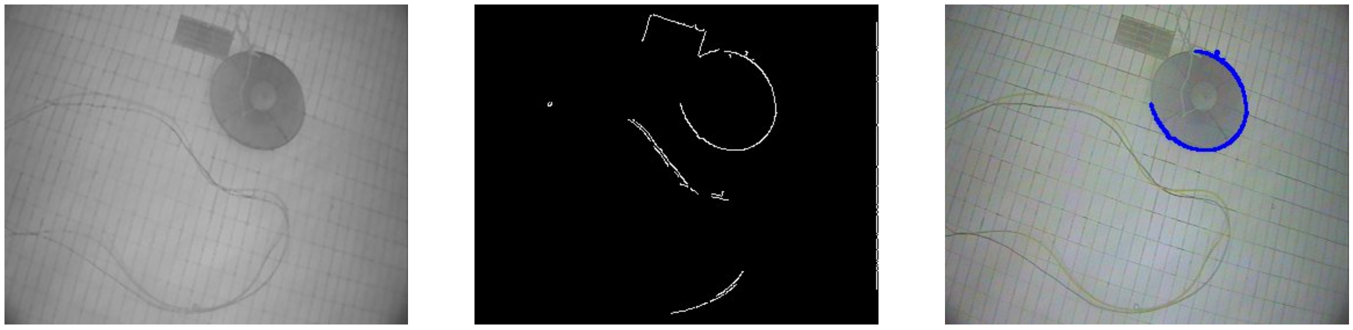


Fig. 6. (left) Original sonar image in the sonar reference system; (centre) Sonar segmentation in the robot reference system (the sonar is mounted upside-down on the bottom of the vehicle and rotated of 90 deg to protect the wiring; (right) localisation. The rectangle represents the pool and the point a discretization of the probability density function of the robot pose in the environment.



(a) Smoothed Image (transformed to Gray)

(b) Edge Detection with Canny

(c) Target detection

Fig. 7. Steps for ground target detection

needs to position itself aligned with the box, and perform a kind of U-manouvre (go backwards, go downwards, go forwards (docking), go backwards (exit from the docking box). In the ”go forward” step, the internal light in the docking box is tracked and the vehicle centers itself with respect to it.

Fig. 8 shows the steps for line and corner detection.

During the practice tests in the competition area, the vehicle successfully performed all the steps of autonomous docking, while during the competition itself, the vehicle was able to enter into the docking box, but had some problems to exit autonomously.

3) Mid-water Targets: The mid-water orange ball target is detected with the forward camera. Three detection methods have been implemented: colour-based detection, detection of circles, and a Mean-Shift Algorithm. According to the conditions and the performances evaluated on the site, we chosed the colour detection method (the only one described in this paper, for brevity), with a circularity check, for extra safety.

The image colour space is mapped from BGR (Blue, Green, Red) into YCrCb (Luma, Red Chroma, Blue Chroma). For the red ball, the red chroma is selected and a threshold applied to highlight the orange colour of the ball. An open morphological transformation using a disk-shape structuring element is then applied (an erosion followed by a dilation), in order to remove noise. A more robust method consists in subtructing the green channel from the red one, instead of considering only the red one. Finally, the circularity of each remaining element is calculated, and those above a given threshold are taken to be orange balls. This method is very simple, robust and efficient.

Unlike the previous years’ competitions, the mid-water target was moving during SAUC-E 2009. A control strategy has been developed in order to track the moving target. The goal is to face the target, trying to maintain a constant distance. In the case that the target is further away than the predefined ideal distance, the vehicle will attempt to close the distance, whilst turning to face the object. Alternatively, if the target is too close, the vehicle will move backwards and laterally while turning to face the target. In this way it will essentially ”side step” the target when its movement brings it closer to the vehicle, and then continue following in its wake once it has passed. We consider this to be an ideal strategy for pursuing a moving target, as the vehicle will avoid being ”in front” of its target, therefore staying outside

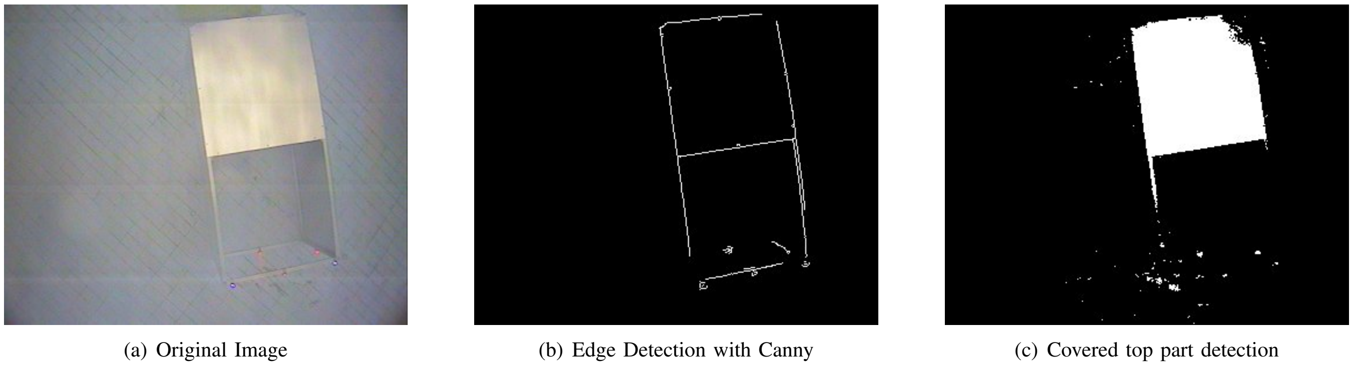


Fig. 8. Steps for docking detection

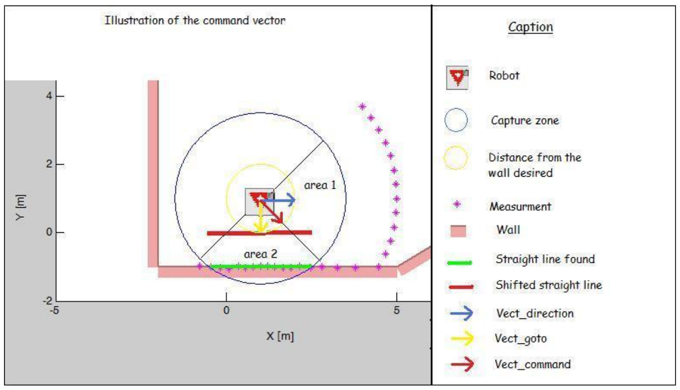


Fig. 9. Wall Following Strategy: calculation of the command vector

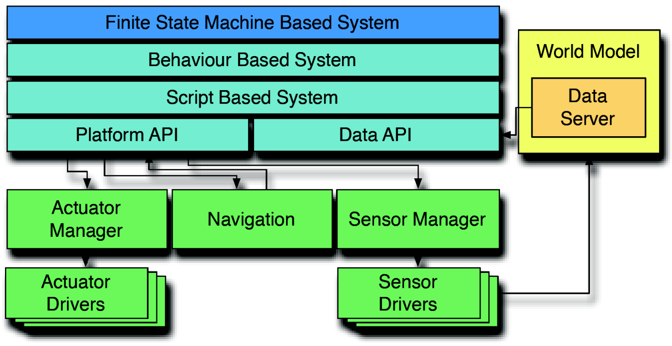


Fig. 10. Mission Control System

its most likely active sensing zone. The relative location of the target is judged in three dimensions using stereo vision, but could easily be adapted to work with sonar.

4) Stereoscopic Vision: By combining object detections from two images taken from differing viewpoints, it is possible to obtain estimates of relative 3D position. Firstly, an object is tracked in both images using the detection algorithms mentioned above, which give the centre of the object. Epipoloar geometry is used to reject outliers, and application of the intrinsic matrices for each camera converts from pixels to metric values. Finally, the triangulation method is used to calculate the estimated 3D positions. In practice, the 3D position estimation is good, with an error of between 0.01 and 0.05 metres at a distance of 2.5 metres from the cameras. The process is described in more detail in [2].

E. Wall following

Wall inspection and following are receiving growing interest both for industrial and military applications. The chosen solution uses the Least Square Method, to find the best line fitting a set of points. From the raw sonar image, points are extracted for each beam, converting the image generated by the sonar in range values. A simple threshold for each beam is applied in order to segment the image and to extract the wall points. Two approaches have been evaluated. In the first one, the vehicle is always facing the wall and has a limited field of view. This means that the vehicle will change its orientation according to the shape and direction of the wall. The second approach consists of always havig the same orientation, with the vehicle orientation parallel to the wall to be surveyed. The field of view is set to $180\,\deg : [-\frac{3}{4}\pi; \frac{\pi}{4}]$ or $[-\frac{\pi}{4}; \frac{3}{4}\pi] radians$, according to the position of the wall with respect to the vehicle, being $0\,\deg$ at forward direction. This second approach only allows complete surveys of a tank if the field of view is changed dynamically, but has the great advantage that the vehicle does not need to rotate, which results in better and more reliable sonar images. As for the competition there is the need to survey only one wall, even if not always staight, this second approach has been chosen. Fig. 9 shows the different vector components of the command given to the vehicle. An early attempt was using the DVZ technique, Deformable Virtual Zones, a technique used for obstacle avoidance and obstacle tracking. We decided to change approach as this algorithm was not reliable enough with the delays typical of the sonar sensors.

F. Mission Control

The mission control system (see Fig. 10) consists of several layers of abstraction, working up from a vehicle systems interface to a parameterised behaviour based finite state machine system [7].

At the highest level, a Finite State Machine (or FSM) based controller oversees the mission. Each state in this FSM represents a particular behaviour with a particular set of parameters. Exactly one behaviour has control of the vehicle at any one time, and the controller monitors its condition, switching to different states as required by the mission specification.

Generic behaviours are defined either in Java or a dynamic scripting language called *Groovy* [8]. These are able to pass instructions to the low level vehicle systems through a *Platform* API. These instructions could be setting the mode of a sensor or actuator, or requesting movement to a relative or absolute position. This API also provides the script with basic information about the vehicle, such as its current position and attitude.

Information received from the sensor processing systems is entered into a *data server*, access to which is provided by the *Data API*. This allows the script to both read and write to the data server, allowing access to information about the current position and state of objects in the world and also for different behaviours to pass information to each other. Example tasks which a behaviour may carry out are: passing through the validation gate, searching the mission area, and hovering over a ground target.

For complex tasks in dynamic and unknown environments, the finite state machine based layer can easily be replaced with a full mission replanner. This system is fully implemented but not deployed on the vehicle for the competition as it is not required. During the final, the vehicle integrated five tasks in a single mission: passing through the validation gate, finding and hovering over a ground target, finding and following a mid-water moving target, following a non-linear wall, and docking. Due to a communication problem between different modules, it stopped after succesfully completing the first four tasks. The docking task and the gate avoidance tasks were demonstrated successfully separately in single task missions, as there was not enough time to run the complete mission again. Team Nessie was the only one managing to successfully put together in a single mission more than three tasks.

IV. Video

The attached video presents the evolution of the robot, since its construction to the most difficult tasks. Unfortunately the DSTL has not yet released the video recordings, captured by the underwater cameras during the final of the competition. At present, only a few short clips are available and we have put those related to *Nessie Team* in the presented video (ground target hovering, docking). The parts of the video related to other tasks show the robot performing in our testing facilities (wall following, ball tracking).

V. Innovation

The main innovations concerning Nessie IV are related to both hardware and software. She has more sensors and more accurate sensors than Nessie III, both for state estimation (new compass, new depth sensor) and for sensing the environment (new micron sonar).

The global localisation algorithm is very helpful, as relying only on relative motion estimation can be very dangerous. The navigation system that we have used this year has no precedence in SAUC-E, both in the method (able to recover from wrong convergence) and in accuracy (up to 10 cm).

Other relevant software innovations concern the new targets to be detected. Ball tracking is quite a solved problem for land robotics, but not yet for the underwater world. Our solution not only tackles the problem posed in the rules, but has a control strategy designed to also address possible real scenarios.

Wall surveying is a task with growing interest (eg. ship hull inspection) and our solution is reliable and efficient.

Last, but not least, autonomous underwater docking is far from being a solved problem. Although some ideas have been proposed in the literature, a reliable and widely used algorithm is still under research. With our solution we attempted to provide one more piece to the *autonomous docking toolbox*, which will eventually provide a robust solution. It should be mentioned that *Team Nessie* was the only team which not only attempted the docking task, but that successfully completed it.

VI. Conclusion

The *Nessie IV* AUV is a fully autonomous robotic platform, equipped with reliable sensors and comprehensive software. Its excellent performance at the SAUC-E 2009 has demonstrated once again its capabilities in successfully carrying out underwater missions. The expertise gathered in designing and building this vehicle will be channeled into a completely new vehicle, which the Ocean Systems Lab is building to compete for the SAUC-E 2010 (summer 2010, NATO Undersea Research Centre - NURC).

Acknowledgment

The authors would like to thank Pierre Regis, Guillaume Bossant and Pierre-Louis Garmier for their invaluable help before and during the competition.

References

[1] J.-Y. Bouguet, “Pyramidal implementation of the lucas kanade feature tracker - description of the algorithm.” Intel Corporation, Microprocessor Research Labs., Tech. Rep.

[2] J. Cartwright, N. Johnson, B. Davis, Z. Qiang, T. L. Bravo, A. Enoch, G. Lemaitre, H. Roth, and Y. Petillot, “Nessie iii autonomous underwater vehicle for sauc-e 2008,” in *Proceedings of UUVS 2008, Southampton, UK*, 2008.

[3] F. Maurelli, S. Krupiński, A. Mallios, Y. Petillot, and P. Ridao, “Sonar-based auv localization using an improved particle filter algorithm,” in *Proceedings of IEEE Oceans ’09, Bremen, Germany*, 2009.

[4] S. Krupiński and F. Maurelli, “Localisation and guidance in the auv docking problem,” in *Workshop in visual guidance systems for small autonomous aerial vehicles, IEEE/RSJ IROS 2008; Nice, France; September 2008*, 2008.

[5] A. Brighenti, L. Zugno, F. Mattiuzzo, and A. Sperandio, “Eurodocker-a universal docking-downloading recharging system for auvs: conceptual design results,” *OCEANS ’98 Conference Proceedings*, vol. 3, pp. 1463–1467, Sep-1 Oct 1998.

[6] L. Brignone, M. Perrier, and C. Viala, “A fully autonomous docking strategy for intervention auvs,” *OCEANS 2007 - Europe*, pp. 1 – 6, 2007.

[7] N. A. Johnson, “Abstracting the planner down down: An architecture for planner based control of autonomous vehicles,” in *Proceedings of the 27th Workshop of the UK Planning and Scheduling Special Interest Group*, 2008.

[8] G2One, Inc., “Groovy dynamic scripting language for the java virtual machine,” Web Page : http://groovy.codehaus.org/. Last Checked: 18/11/2008, 2008.