

Nessie II Autonomous Underwater Vehicle

Hassan Assalih, Joel Cartwright, Vincent Chalencon, Ben Davis, Andrew Durrant, Emeric Faraud, Flora Figiel, Dan Harber, Mara Jiminéz, Nick Johnson, Andrew Lees, Peter Long, Javier Mayor Pérez, Zhizhuang Qiang, Jamil Sawas, Chris C. Sotzing and Yvan Petillot

Abstract—An AUV has been designed and built in the Ocean Systems Laboratory at Heriot-Watt University to compete in the 2007 SAUC-E competition. Using a robust hardware and software design the vehicle is able to successfully complete the tasks set out in the competition and will be an excellent platform for further development.

I. INTRODUCTION

THIS paper will describe the Nessie II autonomous underwater vehicle developed at the Ocean Systems Laboratory at Heriot-Watt University. The vehicle was designed to compete in the 2007 Student Autonomous Underwater Challenge Europe (SAUC-E) competition held at the Haslar Ocean Basin Tank in Gosport, UK. This paper will first describe the hardware designed for the vehicle and then proceed to the software architecture.

II. HARDWARE DESIGN

The Nessie II vehicle is made up of two 22cm diameter cylindrical aluminium hulls surrounded by a protective metal cage. On hull contains the batteries, the other is dedicated to the electronics. The vehicle is controlled by 4 thrusters giving it a full 6 degrees of movement. An image of the vehicle can be seen in Figure 1.

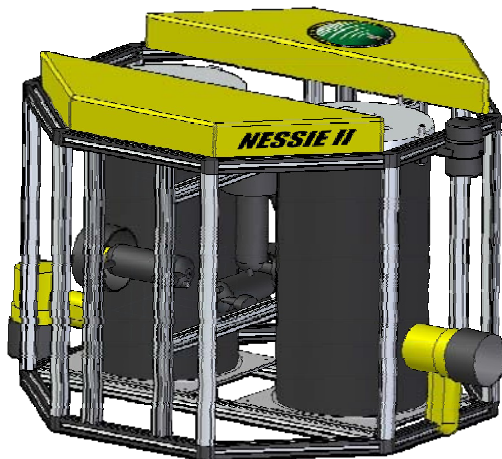


Figure 1: The Nessie II autonomous underwater vehicle.

In this section the hardware used in the vehicle will be described in detail including the batteries, motors, sonar, altimeter, INS, cameras, communication, drop marker system, and computers. The schematic of the hardware architecture can be seen in Figure 2.

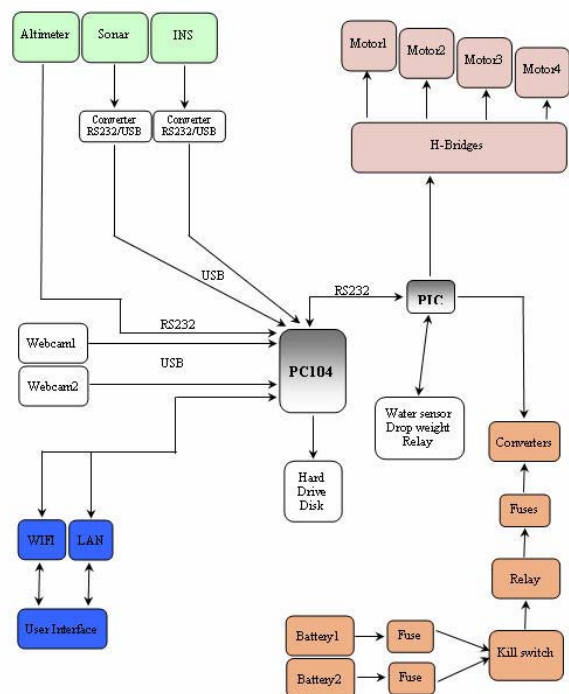


Figure 2: Schematic of the Nessie II hardware.

In this section the hardware used in the vehicle will be described in detail including the batteries, motors, sonar, altimeter, INS, cameras, communication, drop marker system, and computers.

A. Batteries

The vehicle is powered by four packs of ten alkaline GP476A batteries each amounting to 12V. The batteries are contained in their own separate vessel. An image of one of the battery packs can be seen in Figure 3.



Figure 3: Nessie II battery pack.

Two packs of batteries are connected in parallel to supply 12V to the electronics. The other two packs are used to supply 12V to the motors.

B. Motors

The vehicle has been equipped with four SeaBotix H.P. thrusters (Figure 4). These thrusters can produce 80W of power by using 19V at approximately 4.5A. They weigh 700g in air and 350g in water.



Figure 4: SeaBotix H.P. thruster.

The thrusters are driven by four H-Bridges that are controlled using pulse width modulation (PWM). The PWM is created by the PIC18F6525 and has the capability of controlling the speed and direction of the thrusters.

C. Sonar

The Nessie II AUV uses the Trittech Micron Sonar (Figure 5) that weighs 324g in air and 180g in water. It has an operating range of 2-75m and an operating frequency of 650 kHz to 750 kHz. It is supplied by 12V and uses an RS232/USB interface.



Figure 5: Trittech Micron Sonar.

The sonar is used to detect the corner of the tank in order to find the vehicle's position. It is also used to detect targets in the vicinity of the vehicle.

D. Altimeter

The Trittech PA500 (Figure 6) is used to measure the distance between the vehicle and the floor of the tank. The operating range of 0.1 to 10 meters has been employed as the maximum depth of the water tank is 5 meters. It uses an RS232 connection to the PC104 and is powered by a 12V supply.



Figure 6: Trittech PA500 altimeter.

E. INS

The Xsens MTi (Figure 7) is a miniature, gyro-enhanced Attitude and Heading Reference System (AHRS). It is used to measure the vehicle's 3D orientation and its angular velocity by using a combination of gyroscopes, magnetic field sensors and accelerometers. The INS weighs 50g and is supplied by 5V.



Figure 7: Xsens MTi attitude heading reference system.

To determine the vehicle's orientation the INS uses three gyroscopes. Each gyroscope is aligned to a different axis. This allows the device to determine the vehicle's angular position about the X, Y and Z axis (Roll, Pitch and Yaw). The device uses the magnetic field sensors to detect magnetic north. Both of these techniques are used for navigation purposes.

F. Cameras

The vehicle uses two Logitech Pro 4000 Web Cams (Figure 8) to view the two underwater targets. One is situated at the front of the vehicle to detect the mid-water target. The other is designed to detect the target situated on the bottom of the tank. The Web Cams have a digital video capture speed of 30 frames per second and a colour depth of 24 bits. Both web cams are connected to the PC via USB connections.



Figure 8: Logitech Pro 4000 webcam.

G. Wireless

The ProSafe 802.11G Wireless Access Point (Figure 9) is used to communicate between the user interface and the vehicle without the use of an Ethernet cable. This is achieved by floating and umbilical cable to the surface.



Figure 9: Netgear ProSafe 802.11G Wireless Access Point.

The 802.11G uses a 12V supply and weighs 574g. It is connected to the PC104 via a RS-232 connection. An Ethernet connection can also be used to connect to the user interface if necessary.

H. Drop Marker System

A custom drop marker system was developed to allow for precise target marking as well as recognition of marking success. There are three main parts to the project: the dropper mechanism, the markers and the flash detector which won't be used in the competition because although a semi-working model was developed it was not ready for use due to time restraints.

The dropper design was approached considering two designs, one a linear and one a rotary. The linear design was chosen because it appeared to be superior. The control electronics were much simpler and more reliable than that of the rotary design because it involved only a few components to connect it directly to the Programmable Integrated Circuit (PIC). The use of a solenoid meant that it was not necessary to waterproof the electro-mechanical part of the device, it being a low resistance coil meant it could be used 'naked' in fresh water, which can be a difficult process. The shape of the linear design lent itself more to being integrated into the robot than the rotary one. The final advantage of this design was that all the components were easily manufactured with the resources available. An image of the drop target system can be seen in Figure 10.

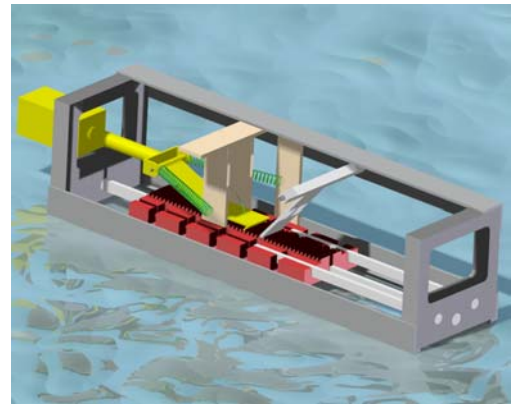


Figure 10: Nessie II drop marker system.

Testing the mechanism resulted in modification, the first one being that it was seem idly not possible to acquire reasonably an appropriate push solenoid. After testing one that seemed to be appropriate it was realised that the figures given were not for the whole stroke length but for its maximum force on the stroke. A much larger one would be needed but was not available however a suitable one was available in the pull format so the design was altered. During the final tests it was noticed that the solenoid arm had a tendency to drift off centre so a guide had to be added.

The principal concept of the markers was that they were to detect if they had hit the target using Hall Effect detectors coupled with an inbuilt magnet to detect the targets ferrous property and would then produce a flash. A simple flasher Printed Circuit Board (PCB) was then designed which strobed a Light Emitting Diode (LED). The case that was used was a 10ml syringe because of its hydrodynamic and sealing properties. A camera battery was used as a source of power because it best fit the requirements.

I. PC104

The computer used for this project is a MSM800 PC104. This model was chosen both for its relatively small size and its low power consumption which results in less heat and therefore no need for a fan. It has a AMD Geode LX800 500 MHz Processor, 512 mb of RAM, 4 USB 2.0 ports and 2 serial ports.

To make the computer even more robust an 8 gb solid-state hard drive was used instead of a conventional one. The combination of a low power, fanless computer and a solid state hard drive results in a computation system that is low power, low heat and very robust to the bumps and jerks that are expected in a mobile vehicle. An image of the PC104 is shown in Figure 11.

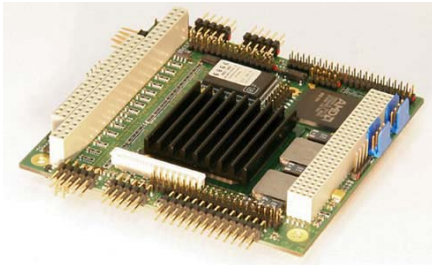


Figure 11: MSM800 PC104 embedded computer.

J. PIC

The PC104 communicates with the PIC in order to control the motors and receive sensor data. The PIC controls the motors via the H-bridges. It also controls the power in the vehicle by switching the relay on or off. In addition, it receives data from the water sensors which is then sent on to the mission planner. In this way, leaks can be easily detected and dealt with. The PIC used in the Nessie II vehicle is a Microcontroller PIC18F6525.

III. SOFTWARE DESIGN

A. Architecture

The software architecture for the Nessie is based around a standard three level control architecture. These systems are divided into three distinct layers: the deliberative high level layer for planning, the control execution layer for vehicle instruction and sensor processing, and the functional reactive layer for data input and motor control [14]. The software architecture used in the Nessie II AUV is shown in Figure 12.

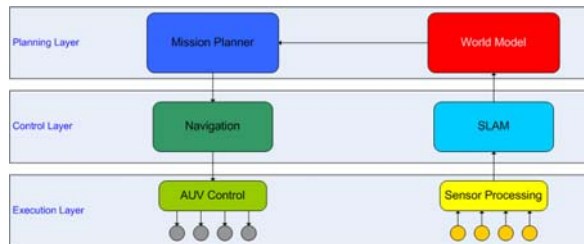


Figure 12: Software architecture for the Nessie II AUV.

In this architecture, data is taken from the sensors (2 cameras, sonar, altimeter, INS) and passed into the SLAM. The SLAM collates this information to discover the vehicle's position as well as that of all the targets in the world. This information is output to the world model. The mission planner uses the information in the world model to choose the best possible task and once this task is selected, instructs the navigation module on where to go. All modules communicate using the OceanSHELL system developed in the Ocean Systems Laboratory. This is a lightweight packet based communications protocol for distributed, modular systems. The software modules used in the Nessie II vehicle will be described in more detail in the following sections.

B. Cameras

Two USB webcams (Logitech QuickCam 4000 Pro, Figure 8) are used to grab 320 x 240 pixel forward and downward looking images with a frame rate of 10. Each forward image is searched to find the mid-water target (the orange ball), and the dummy mid-water target (the green ball), while each downward image is searched to find any of the four expected bottom targets (the circular target with the cross and beacon at its centre, the tyre, the cone, and the pipe).

Since in the forward images the objects are colour objects, the image is converted from the RGB (Red-Green-Blue) colour space into the HSV (Hue-Saturation-Value) colour space, and the specific ranges of the hue are used to segment the image and highlight the orange or the green ball. After the segmentation, an open morphological transformation using disk-shape structuring element is applied which is an erosion followed by a dilation. The effect of this transformation is a preservation of foreground regions that have a similar shape to this structuring element, or that can completely contain the structuring element (in our case the orange or green ball), while eliminating all other regions of foreground pixels. The binary resulted image is then passed into a Hough transform circle detection algorithm to detect the circle surrounding the orange or the green ball. Figure 13 shows a mid-water target image grabbed by the forward camera in the Ocean Systems Laboratory tank highlighted by the detected circle.

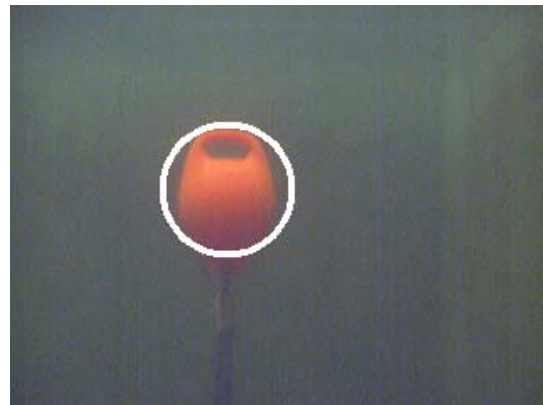


Figure 13: The mid-water target detected by the orange ball algorithm.

Three factors are used to detect the circular bottom target with the cross and the flashing light in its centre: the light detector, the cross detector, and the circle detector. Each of these three detection algorithms are applied to the downward image after thresholding it with different criteria. Each down image is also investigated to see if there are two circles approximately concentric, and one of them has approximately double radius of the other, and in this case a tyre will be detected in the image.

Moreover, a similar detection algorithm to detect the orange ball in the forward images is used to detect the cone in the downward images based on its significant colour in comparison with the other expected objects at the bottom of

the tank. The image in Figure 14 is a downward image with the three bottom targets detected by the downward images algorithms. Finally, to detect the pipe the down looking image is thresholded and morphological operators are applied to remove the noise and preserve just the pipe object. Then, based on the vehicle height from the bottom of the tank and thickness of the pipe, decision can be made to know if what we detect is the pipe or other similar lanes at the bottom of the tank.

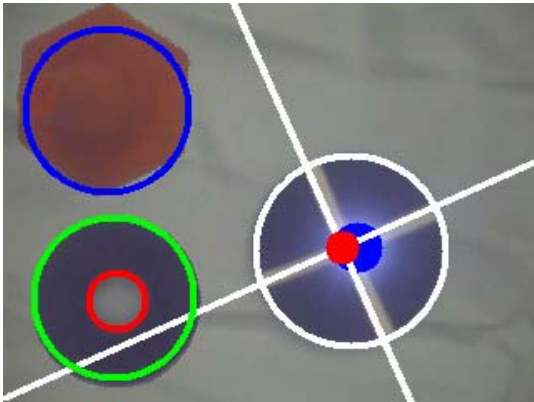


Figure 14: Three bottom targets detected in the same image by the down looking camera algorithms.

C. Sonar

The purpose of Sonar Module in the Nessie II vehicle is to detect the walls of the pool in the tank and to detect the mid water targets. In this section our new methodology is discussed including four new concepts for the 2007 vehicle. The first is the automated tuning of the Sonar's parameters, the second is the performance estimation of the different image enhancement stages, the third is sonar focusing, and the fourth is the correction of the slant range distortion while calculating the coordinates of the mid water target. However, the base of the methodology is a Hough Transform used in searching for lines, and applying the problem constraints on the result of this transform.

1) Methodology

This year we are going to enhance the previous methodology, taking in to account the different conditions of the SAUC-E competition in 2007. In 2006 the sonar module was scanning the whole pool to detect four walls. This is not the case this year, because the new pool dimensions are 120x60m. However, not all of the pool will be used; just 30x20m will be within the competition area. Due to this, if we scan the whole pool we will decrease the resolution of the sonar's image. Moreover, scanning the whole pool will increase the scanning time significantly. In the 2006 sonar images we had a rectangle representing the pool's walls. This year we are going to have two perpendicular lines representing one corner of the pool. We can state the main ideas in 2007 methodology in these following points:

- Sonar tuning: Auto tune the sonar's gain depending on the noise estimation developed in this report

- Image enhancement: Apply different Image Enhancement techniques (Erosion, Opening, Closing, Threshold, Canny Filter ...)
- Performance Estimation: Measure the Noise reduction after each stage of the image enhancement to compare between different parameters set, and to compare similar techniques.
- Decision Stage: Determine the best chain of enhancements and the best parameters set depending on the Noise Reduction Estimation.
- Wall Detection: Build Hough Transform Matrix (Accumulator Array) and search for the brightest line, then search for the closest line to the sonar which is parallel to the brightest line (this will solve the reflection problem), then search for brightest and closest to sonar perpendicular line to the first found line . The intersection of these two lines will be the corner of the pool which we are searching for.
- Target Detection: the same procedure of determining the best techniques and the best parameters works here as well. A correcting procedure is applied on the target coordinates to increase the confidence in the target coordinates, this procedure is added this year to decrease the error produced by the slant range distortion in the sonar's image.

2) Tuning

There are two important parameters in the sonar which we have to determine correctly, Range and Gain. The Range is the maximum distance which will be scanned by the sonar. This value could be anything between (5-100m) in the Tritech Micron sonar used on Nessie II. The competition area will be 20x30m this year, which means that we have to scan 30m in maximum, so the suitable range value is 30m.

Unfortunately, determining the gain value which ranges from 0% to 100% is not as easy as determining the range parameter. Thus, two critical questions were raised: first, what is the best value for gain to get the best image and second, how to determine the best image?

Manual observation of the sonar's images was the solution. This year we are going to decide the best gain depending on the minimum noise in the images. In other words, searching for the best gain is transformed into searching for the minimum noise in the noise array. The noise array will be formed by changing the gain from 0% to 100%. Each time the Noise will be estimated in the image and stored in the corresponding cell in the noise array. The index of the minimum noise in the noise array will be the best gain.

Another issue that should be discussed is how to estimate the noise in the sonar's images. To address this, a simple noise estimation technique was developed.

The noise estimation technique presented here is designed for estimating image's quality which is obtained from sonar. The need for measuring the quality of sonar's image triggered this simple method, however, other applications for this

technique were found as well.

The noise estimation of the sonar's image can be defined as the normalized absolute difference between the sonar's image and its model. The model can be defined as the ideal sonar image which we can get from the ideal sonar working in the ideal conditions.

Thus, if we have the image called *SonarPic* and its model which is called *FirstModel* then we can estimate the total noise value with the equation:

$$Total\ Noise = \sum_x \sum_y |SonarPic(x, y) - FirstModel(x, y)|$$

where $x \in [0, MaxWidth]$, $y \in [0, MaxHeight]$

Clearly, the model has the same size as the sonar's image. From the total noise equation we can calculate the maximum noise value which we can get. The maximum noise value is the noise value between the model and its inverse version; in 8 bit images the maximum noise value can be calculated as following (given that the model is back or white):

$$Max\ Noise = 255 \times MaxWidth \times MaxHeight$$

Finally, the normalized noise estimation in the range of 0 to 10 can be calculated as following:

$$Noise = \frac{Total\ Noise \times 10}{Max\ Noise}$$

Calculating the noise estimation in the sonar's image requires the model image. This model can be obtained from the ideal sonar working in the ideal conditions. Unfortunately, ideal sonar doesn't exist in reality. However we can build the model manually. In other words, we will draw the model corresponding to a particular position of the sonar. Building the model manually is one of the disadvantages of this method and drawing the model can be one kind of a black art. On the other hand, this method gives us the opportunity to fairly compare the images (obtained using different gain values) rather than comparing between them using the naked eye. This comparison between images can be fair enough since they are measured using the same measuring tool which is the model in our case.

Noise estimation in the sonar's image can be used in three different arenas:

- To compare between the impacts of the different Filters on the same sonar's image.
- To compare between the impacts of different parameters of the same image enhancement technique, this can be different threshold values, different structuring elements, different filter kernels, ...etc
- To compare between sonar images obtained in the same position with different Gain values.

Figure 15 shows one of the sonar images its model. The noise in the image is: 1.81616.

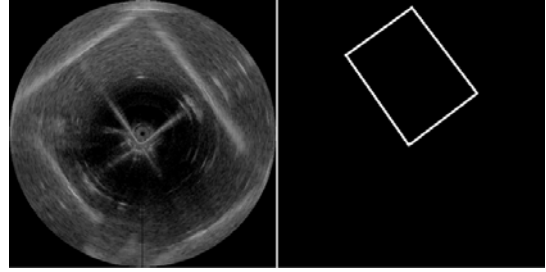


Figure 15: A sonar image on the left with the corresponding tank boundaries on the right.

All image processing applications include one stage for image enhancement. This year the best chain of procedures to enhance the image will be applied. This can be done by estimating the noise after each procedure and compare the noise value with the previous estimated value.

After applying one of the image enhancements we can estimate the improvements happened in the image by the equation:

$$EE = Noise_{after} - Noise_{before}$$

Where EE stands for enhancement estimation, $Noise_{after}$ is the noise estimation after applying the image enhancement technique. $Noise_{before}$ is the estimated noise before applying it. The previous equation shows that EE ranges between -10 and 10 depending on the noise definition in this report. The following table illustrates the program output after applying different image enhancement techniques. The noise value increased after applying the closing, this is because closing increases the overall brightness of the image. Figure 16 shows the closed image and the same image filtered with a canny filter (the source image is shown in Figure 15).

Original sonar noise	1.81616
Noise after closing	2.00854
Noise after Canny filter	0.00192515

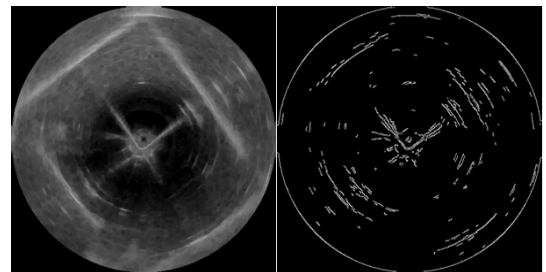


Figure 16: A sonar image on the left and the same image on the right with noise filtered out.

After performing Hough Transform and retrieving the

brightest line in the image and the closest to sonar, an algorithm should search for the brightest perpendicular line to it. The intersection of these two lines will be the corner of the pool which we are searching for. Figure 17 shows the detected lines in the original image (Figure 15).

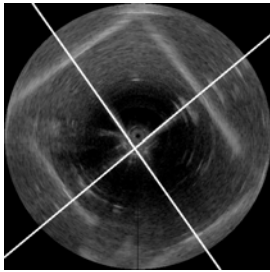


Figure 17: A sonar image showing the detection of the pool corner.

Image enhancements to detect the mid water target are done by applying a canny filter in the pool area, and searching for the contours inside it. Increasing the result confidence will be sustained through two stages, the sonar focusing stage and the correction stage.

Sonar focusing can be done after the first scanning of the environment with 360° and range equal to the length of the virtual pool (length of the competition area). In the first scan we can estimate the position of the mid water target. The next scan can be done with a range bigger than the estimated distance slightly, and the scanning sector can be 30° which includes the target. This procedure can enhance the resolution of the image since the range will be smaller than the first range and the noise will be less than the noise in the first scanned image.

After detecting the mid water target and measuring the distance between the target and the sonar, a correction parameter should be applied on this distance. The SLAM Module is expecting the mid water coordinates (retrieved from sonar) to be in the xy plane. Unfortunately, this is not the case. Figure 18 shows the measured distance and the real distance in the xy plane.

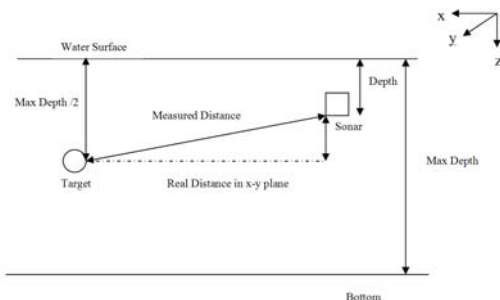


Figure 18: The real and the measured distance between the target and the sonar.

To decrease the error in the distance value, we can approximate the mid water target depth to be fixed in (Max Depth/2), since we know the sonar's depth (from the pressure sensor) we can estimate the real distance more accurately

through the equation:

$$real\ dis = \sqrt{(measured\ dis)^2 - \left(\frac{max\ depth}{2} - depth\right)^2}$$

However, the mid water target is not in the depth (max depth/2). Nevertheless, this calculation will decrease the error while estimating the real distance with the measured one. Figure 19 shows the estimated value in the case where the mid water target is below the max depth/2 level. The estimated distance value is closer to the real distance value than the measured one.

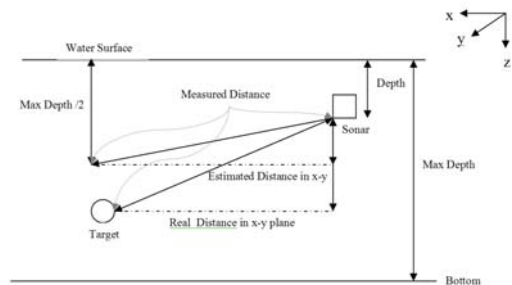


Figure 19: Real, measured, and the estimated distance between the target and the sonar.

D. Integrated SINS and SLAM

1) General SLAM

In this section, the concept and the research evolution of Simultaneous Localisation and Mapping (SLAM) (SLAM is referred as Concurrent Mapping and Localisation (CML) in some of the literature) will be introduced in detail. Figure 20 shows the structure of SLAM. The first consistent solution of SLAM came from Randall Smith, Matthew Self and Peter Chesseman [1]. Following that book, SLAM research has become more and more popular in the autonomous vehicle area because self localisation is a critical question to an autonomous vehicle. An autonomous system must answer three questions: "Where am I?" "Where am I going?" and "How do I get there?" The first one can be thought as the most important one of these three questions [2].

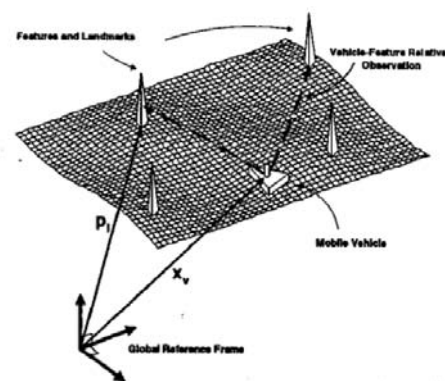


Figure 20: The structure of the SLAM problem.

Randall Smith presented a general theory for estimating the

uncertain relationships between the vehicle and the features in the environment in his paper [1]. This theory also allows the estimation of the boundary of the uncertainty. It is the beginning of SLAM research. After that, many papers [3] [4] started to discuss SLAM: Anastasiosz Mourikis analysed the positioning uncertainty [5], Momotaz Begum introduced a novel idea that combined Fuzzy logic and genetic algorithms for SLAM research [6], Ioseba Tena Ruiz [7] showed the experimental results of SLAM on underwater vehicles using side-scan sonar, Robert N. Capenter used a forward looking sonar in SLAM [8]. This section provides a general introduction for SLAM theory.

The Kalman Filter's state includes the relevant states of the vehicle and the landmarks in the environment in which the vehicle is moving. So, the state vector x has the form:

$$x = [x_V \quad x_1 \quad \cdots \quad x_n]^T$$

where x_v is the state of the vehicle, $x_i, i=1 \dots n$ hold the states of the landmarks. The estimated error covariance for this system is P :

$$P = \begin{bmatrix} P_{VV} & P_{V1} & \cdots & P_{Vn} \\ P_{1V} & P_{11} & \cdots & P_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{nV} & P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

The state and covariance are updated by using a general Kalman Filter (KF).

$$\begin{aligned} \hat{x}_k^- &= \Phi_{k-1} \hat{x}_{k-1}^+ + B_{k-1} u_{k-1} \\ P_k^- &= \Phi_{k-1} P_{k-1}^+ \Phi_{k-1}^T + Q_{k-1} \\ K_k &= P_{k-1}^- H_{k-1}^T [H_{k-1} P_{k-1}^- H_{k-1}^T + R_{k-1}]^{-1} \\ x_k^+ &= x_k^- + K [z - H_k x_k^-] \\ P_k^+ &= [I - K_k H_k] P_k^- \end{aligned}$$

In the general SLAM theory, the landmarks are assumed to be fixed. The observed new landmarks which are not in the stochastic map will be added in the state to generate a new state vector. When a new landmark is added to the map, the new map state vector and covariance will be [9]:

$$x_{k+1|k+1} \leftarrow \begin{bmatrix} x_{k+1|k+1} \\ x_{N+1} \end{bmatrix}$$

$$\begin{aligned} P_{N+1|N+1} &= J_{x_r} P_{rr_{k+1|k+1}} J_{x_r}^T + J_z R J_z^T \\ P_{r_{N+1}} &= P_{N+1}^T = P_{rr_{k+1|k+1}} J_{x_r}^T \\ P_{N+1|k+1|k+1} &= P_{N+1}^T P_{r_{k+1|k+1}}^T = J_{x_r} P_{r_{k+1|k+1}}^T \end{aligned}$$

Where J is the Jacobians respect to the robot state x and to the

new landmark.

This is the basic theory of SLAM, based on the assumptions listed below:

- The vehicle is the only moving object in the environment.
- The computer can process any quantity of the landmarks which can be observed by the vehicle.
- There is only one vehicle in the environment.
- The landmarks can be distinguished from the background.

These assumptions adopted in general SLAM can simplify the analysis of the problem, but in some applications, the designers have to face situations where they do not hold up. The coming sections will show a simple example of SLAM based on these assumptions. The next section will introduce the methods and algorithms which focus on SLAM without one or more of the assumptions.

2) IMU and SINS

a) Mathematics underlying IMU

Inertial Measurement Units (IMU) is the key sensor of the Inertial Navigation System (INS). The designer of INS must understand how the IMU works and how the output of the IMU is disturbed by the environment noise. Usually the accelerometers and gyros are called IMU in INS. Because the price of an accurate IMU is very high and is not affordable to the student competition vehicles, a low cost IMU (Figure 7) is adapted in Nessie2. Mathematical foundations underlying the IMU are also very helpful to understand the INS theory.

The outputs of the IMU are the absolute acceleration and angle rate in the inertial frame projecting to the body frame. The earth rotation, interference between movement and rotation and environment noise are also included in the IMU readings.

In inertial frame, by assuming the mass is 1, the Newton's Second Law of Motion can be written as:

$$\bar{f} = \bar{a} - \bar{g}_m$$

$\bar{a} = \frac{d}{dt} \left[\frac{d\bar{r}}{dt} \right]_I$: absolute acceleration with respect to the inertial frame
\bar{g}_m	: gravitational acceleration due to the mass attraction, point to the centre of the earth.

Based on the Coriolis formula:

$$\frac{dR}{dt} \Big|_I = \frac{dR}{dt} \Big|_m + \omega_m \times R$$

The absolute acceleration can be rewritten in the form of:

$$\bar{a} = \frac{d\bar{V}}{dt} \Big|_N + \bar{\omega}_N \times \bar{V} + \bar{U} \times \bar{V} + \bar{U} \times (\bar{U} \times \bar{r})$$

in which, $dR/dT |I$ is the derivation of vector R in the inertial frame, and $dR/dT |m$ is the derivation of the vector R in m frame, m is arbitrary frame. ω_m is the angle rate of the m frame with respect to the inertial frame.

V is vehicle's velocity with respect to the Earth-fixed frame and \dot{U} is the Earth angle rate. If the speed is available, the accelerometer's reading can be calculated by the following function, this reading is the acceleration with respect to the inertial frame projected to the body frame. Usually the vehicle moves on the Earth surface, the gyro's reading is the angle rate plus the Earth angle-rate. This can be described by:

$$\omega_{ib}^b = \omega_{bb}^b + C_e^b \omega_{ie}^e$$

For each different IMU, the contribution of each error source is totally different [10]. Only main error sources are list below:

- Scale Factor: errors in the ratio of a change in the output signal to a change in the input which is to be measured
- Cross-coupling Error: outputs of one axis resulting from the movement or rotation in the orthogonal axis.
- Random Bias: caused by instabilities within the sensor assembly.

b) Strap-down Inertial Navigation System

The operation of inertial navigation system depends on the laws of classical mechanics as formulated by Newton. The inertial navigation is the process whereby the measurements provided by the accelerometers and gyros are used to determine the position, attitude and speed of a vehicle. Inertial navigation systems are self-contained within the vehicle.

The original applications of inertial navigation technology used stable platform techniques. In such systems, inertial sensors are mounted on a stable platform. Modern systems have removed most of the mechanical complexity of platform systems by having the sensors attached to the body of the vehicle, which is called strap-down navigation.

The basic principle of SINS is discussed in many books [11],[12],[13]. The main computing tasks, those of attitude determination, specific force resolution and solution of the navigation's equation, are indicated in appendix.

For the purpose of integration with other sensors: GPS, DVL, the error equations which are in the first following equation are used. It can be written in the form of a single matrix error equation as the second following equation.

$$\dot{\delta\theta} = \delta\varpi_{yb}$$

$$\begin{aligned}\delta f_{xi} &= f_{zi}\delta\theta + \delta f_{xb}\cos\theta + \delta f_{zb}\sin\theta \\ \delta f_{zi} &= -f_{zi}\delta\theta - \delta f_{xb}\sin\theta + \delta f_{zb}\cos\theta \\ \delta \dot{v}_{xi} &= \delta f_{xi} \\ \delta \dot{v}_{zi} &= \delta f_{zi} \\ \delta \dot{x}_i &= \delta v_{xi} \\ \delta \dot{z}_i &= \delta v_{zi}\end{aligned}$$

$$\delta \dot{X} = F\delta X + Gu$$

Where:

$$\begin{aligned}\delta X &= [\delta\alpha \quad \delta\beta \quad \delta\gamma \quad \delta v_n \quad \delta v_e \quad \delta v_d \quad \delta L \quad \delta l \quad \delta h]^T \\ u &= [\delta\varphi_x \quad \delta\varphi_y \quad \delta\varphi_z \quad \delta f_x \quad \delta f_y \quad \delta f_z]^T \\ G &= \begin{pmatrix} -c_{11} & -c_{12} & -c_{13} & 0 & 0 & 0 \\ -c_{21} & -c_{22} & -c_{23} & 0 & 0 & 0 \\ -c_{31} & -c_{32} & -c_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{11} & c_{12} & c_{13} \\ 0 & 0 & 0 & c_{21} & c_{22} & c_{23} \\ 0 & 0 & 0 & c_{31} & c_{32} & c_{33} \end{pmatrix}\end{aligned}$$

E. Mission Planning

The mission planner is the software module which makes the high level decisions throughout the mission. It decides the next action according to the current state of the pool and the actions taken so far.

The mission planner works based on a list of conditions and a collection of tasks to achieve. Each task has a specific priority associated, a list of parameters specified for it and a Boolean expression formed with the list of Conditions mentioned before. The task selection algorithm is as follows: a list of tasks that have true conditions is created and the one with the highest priority is chosen. The current state of the list the conditions is updated constantly according by the world model. For our System, the list of conditions and tasks are:

Conditions:

1. Validation gate passed
2. There are still targets to find
3. There are targets that need classification
4. Mission complete
5. Mission time-out
6. Mid-water orange target discovered
7. Bottom target discovered
8. Recovery zone identified
9. Vehicle locked on to the bottom target
10. Vehicle locked on to the mid-water target
11. Vehicle locked on to the recovery zone
12. Mid-water target touched
13. Drop marker dropped

Task	Condition	Priority
Gate	(!1 & !5)	9
Map Pool	((1 & !5) & 2)	3

Examine Targets	(1 & (3 & (!5 & (!6 (!7 !8))))))	4
Lock on mid-water target	(1 & (!5 & (6 & (!9 & !12))))	5
Hit mid-water target	(1 & (!5 & (10 & !12)))	6
Lock on bottom target	(1 & (!5 & (7 & !13)))	7
Drop marker	(1 & (!5 & (9 & !13)))	8
Lock on recovery zone	(1 & (!5 & (8 & (!9 & (!10 & (12 & 13))))))	5
Surface	(((((1 & 11) & 12) & 13) 5) 4)	10

This information is taken by the mission planner from an init file, which is common for all the software modules. The init file makes it simple to change the behavior of the vehicle, without requiring a re-compilation.

In order to process the Boolean expression formed from the conditions, it was necessary to implement a parser. Every expression, after being parsed, is stored in a structure (Figure 21), that allows for simple evaluation:

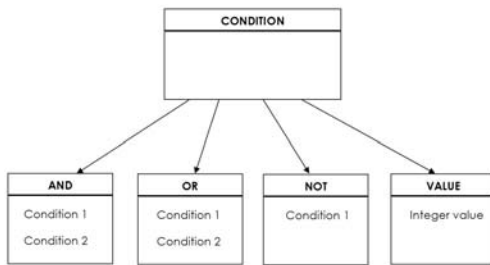


Figure 21: Structure of the mission conditions.

1) Communication

As with the rest of the software modules, the mission planner uses OceanSHELL Messages in order to communicate with the rest of software modules. The mission planner communicates with the following modules: vision, sonar, world model, navigation and PIC.

a) Vision

The mission planner switches the front and bottom cameras on and off. This will depend on the current task and global state of the mission. For example, as soon as both balls are classified and the orange ball is touched, it may not be necessary to keep the front camera on. Additionally, the vision module sends the mission planner a “Ready to Touch” message when Nessie II has the adequate position and heading to go forward and touch the orange ball.

b) Sonar

The mission planner can request for a sonar scan as soon as it requires it.

c) World Model

The Mission planner gets all the information about the environment from the world model. From the world model the mission planner gets the vehicle’s current position, heading, targets found so far and their type, etc. Thanks to this information, the mission planner periodically updates the

current condition states of the tasks. For example, if the mission planner read from the world model that the orange ball is one of the objects in the world, this would mean that the mission planner updated condition 6 to true, because the orange ball is seen.

d) Navigation

The mission planner needs to communicate with the autopilot because the autopilot is the module which receives the next way point from the mission planner. In this point it is necessary to clarify that the mission planner can order two types of movement: absolute and relative requests. With the mid-water target task, it is easier and better for the mission to order a relative movement when the vehicle is locked on the target. On the other hand, in order to lock on the bottom target and drop the marker, it is better to order an absolute way point because this last one is more accurate.

e) PIC

In order to drop the marker, sending a drop marker message to the PIC is necessary.

F. Navigation

1) Underwater Vehicle Modelling & Control

The motion of an underwater vehicle through the water is described by a combination of six velocities (surge, sway, heave, yaw, pitch & roll). These constitute the vehicle’s six degrees of freedom.

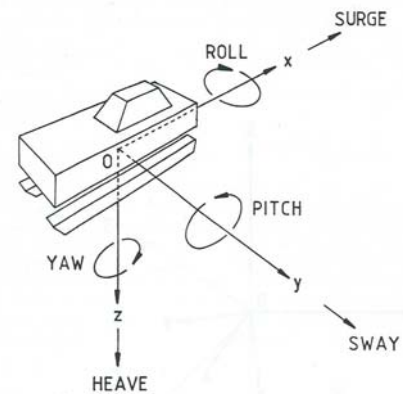


Figure 22: Components of motion for an underwater vehicle.

During the 1970s, a mathematical model was developed at Heriot Watt University that expresses the Hydrodynamic forces and moments applied to each of the vehicle’s six degrees of freedom. With this model the velocities of the vehicle can be determined and used to simulate the performance of the vehicle, both with and without compensation from an external control system. A simple co-ordinate transformation is required in order to convert the velocities of the vehicle into an absolute position and orientation of the vehicle relative to a fixed point. This enables control of the vehicle’s position and orientation within

makes the delivery of the messages are very quickly. That was the reason for starting to use a state diagram and we can develop all the operations in this way of the modules with only one thread.

This structure allows us to have efficient software which is easy to adapt beyond new changes.

2) PIC

The PIC receives messages from the PIC module. When a message is received an interruption is created in the entry port of the RS232. In this interruption the code checks to see if the message is valid by checking the first value. If it's not valid, it is discarded, otherwise it is processed. Messages are processed according to their type and the correct corresponding action is taken. This can include motor instructions, drop marker instructions, water alarm, etc.

IV. CONCLUSION

The Nessie II autonomous underwater vehicle combines the state of the art in both hardware and software design to create a robust, and fully autonomous vehicle. It's design will allow it both to succeed in the SAUC-E competition as well as grow as new technology is added.

ACKNOWLEDGMENT

Team Nessie would like to thank its sponsors and all the people in the Ocean Systems Laboratory. The generous support of these groups was pivotal to the success of the project and the team is eternally grateful.

REFERENCES

- [1] R. Smith, M. Self and P. Cheeseman. Estimating uncertain special relationships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*. Springer-Verlag, 1990.
- [2] P. Jensfelt, H.I. Christensen and G. Zunino. Integrated systems for mapping and localization. In J. Leonard and H. Durrant-Whyte, editors, *ICRA-02 SLAM Workshop*. IEEE, May 2002.
- [3] J.W. Fenwick, P.M. Newman and J.J. Leonard. Cooperative concurrent mapping and localization. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 1810-1817, May 2002.
- [4] G. Dissanayake, P. Newman, H. Durrant-Whyte and M. Csobra. A solution to the simultaneous localisation and mapping (slam) problem. *IEEE Trans. On Robotics and Automation*, 17(3):229-241, 2001.
- [5] A.I. Mourikis and S.I. Roumeliotis. Analysis of positioning uncertainty in simultaneous localization and mapping (slam). In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Japan, Sept. 2004.
- [6] M. Begum, G.K.I. Mann and R.G. Gosine. Concurrent mapping and localization for mobile robot using soft computing techniques.
- [7] I. Tena Ruiz, Y. Petillot and D.M. Lane. Concurrent mapping and localization using side-scan sonar. *IEEE Journal of Oceanic Engineering*, 29(2), 2004.
- [8] R.N. Carpenter. Concurrent mapping and localization with fls. In *Autonomous Underwater Vehicles, 1998. AUV'98 Proceedings of the 1998 Workshop*, pages 133-148, Cambridge, MA, USA, Aug. 1998.
- [9] G. Zunino and H.I. Christenesn. Simultaneous localization and mapping in domestic environment.
- [10] A. Lawrence. *Modern Inertial Technology: Navigation, Guidance and Control Second Edition*. Springer, 1998.
- [11] J. Farrell and M. Barth. *The Global Positioning System and Inertial Navigation*. McGraw-Hill, 1998.
- [12] D.H. Titterton and J.L. Weston. *Strap-down inertial navigation technology*. Peter Peregrinus Ltd. London, May, 1997.
- [13] O. Salychev. *Applied Inertial Navigation: Problems and Solutions*. BMSTU Press, Moscow, Russia, 2004.
- [14] E. Gat. "Three Level Architectures." *Artificial Intelligence and Mobile Robotics*, AIII Press / The MIT Press, Ch. 8, 1998.