Design and Control of a Flight-Style AUV with Hovering Capability

J. Liu[1], M. E. Furlong[2], A. Palmer [1],
A. B. Phillips [1], S. R. Turnock [1], S. M. Sharkh[1]
[1]University of Southampton, University Road, Southampton, SO17 1BJ, UK
[2]National Oceanography Centre, European Way, Southampton, SO14 3ZH, UK
Jiazhong.liu@soton.ac.uk

***Abstract*—The small flight-style Delphin AUV is designed to evaluate the performance of a long range survey AUV with the additional capability to hover and manoeuvre at slow speed. Delphin's hull form is based on a scaled version of Autosub 6000, and in addition to the main thruster and control surfaces at the rear of the vehicle, Delphin is equipped with four rim driven tunnel thrusters. In order to reduce the development cycle time, Delphin was designed to use commercial-off-the-shelf (COTS) sensors and thrusters interfaced to a standard PC motherboard running the control software within the MS Windows environment. To further simplify the development, the autonomy system uses the State-Flow Toolbox within the Matlab/Simulink environment. While the autonomy software is running, image processing routines are used for obstacle avoidance and target tracking, within the commercial Scorpion Vision software. This runs as a parallel thread and passes results to Matlab via the TCP/IP communication protocol. The COTS based development approach has proved effective. However, a powerful PC is required to effectively run Matlab and Simulink, and, due to the nature of the Windows environment, it is impossible to run the control in hard real-time. The autonomy system will be recoded to run under the Matlab Windows Real-Time Windows Target in the near future. Experimental results are used to demonstrating the performance and current capabilities of the vehicle are presented.**

***Index Term*—Autonomous Underwater Vehicle (AUV), Hybrid Control System**

I. Introduction

Autonomous underwater vehicles (AUVs) are now commonly used for long range underwater surveys, however, applications are being identified that require close observation of particular areas of interest, e.g. mines, coral reefs, offshore oil well head risers, or wrecks [1]. These applications require AUVs with increased low speed manoeuvrability. To provide this high level of manoeuvring especially at low speed it is necessary to add additional thrusters to a traditional flight style AUV [2]. These thrusters would allow the AUV to hover and manoeuvre at slow speed in a similar fashion to an ROV.

As part of the Student Autonomous Underwater Challenge – Europe (SAUC-E) [3], the Delphin AUV, shown in Fig. 1, was designed to both compete in the competition and to act as a test bed to evaluate hover performance in a flight style AUV. The design philosophy adopted during the development of the Delphin AUV was to produce a simple and robust vehicle maximising the use of commercial-off-the-shelf (COTS) components. To reduce the cost and duration of the development cycle, the autonomy system was implemented using Matlab/Simulink and image processing was performed using the commercial Scorpion Vision software. Both systems run within the MS Windows environment on a small low profile PC motherboard. This simplifies the programming task, by making use of available tool boxes, e.g. signal processing and control tool boxes and graphical programming tools, as it facilitates rapid software development.

Fig. 1: Delphin AUV on trial in Eastleigh lake, Hampshire

The paper describes the ongoing development work on the Delphin AUV, focusing on the challenges of developing the low speed control system for hovering and manoeuvring and also outlines the advantages and disadvantages encountered in using simple commercial computer packages to implement the control software for the AUV.

II.Low speed control for flight style AUVs

Flight styles AUVs generally have positive buoyancy to facilitate simple recovery as they float to the surface in the case of an emergency. This positive buoyancy is overcome in flight mode by using lifting surfaces on the body of the AUV to generate a hydrodynamic down force for a given pitch of the AUV. As the AUV slows greater pitch will be required to maintain depth. If the speed is further reduced a minimum speed is encountered at which control authority is lost and the

AUV can no longer control itself in flight mode. Although this lower limit can be reduced by changing the design of the AUV possibly by adding additional wings it cannot be removed. To allow a flight style AUV to stop and hover at very slow speeds it is necessary to add thrusters to overcome this buoyancy. Thrusters are also required to manoeuvre in the horizontal plane.

Fig. 2 shows an example rim driven tunnel thruster [4], [5] that was used on Delphin AUV. The analysis of the performance of tunnel thruster has shown that their energy efficiency is reduced when the vehicle is moving [6], as illustrated Fig. 3: the tunnel thruster jets $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ are deflected in the direction of the ambient flow as a function of the relative strength of the jet flow and the ambient flow. The interaction between the jet and ambient flow results in suction forces $\boldsymbol{F}_{s1}$ and $\boldsymbol{F}_{s2}$ on the vehicle downstream of the thrusters' exit which acts against the desired thrust force $\boldsymbol{F}_T$, hence reducing the effectiveness of the thrusters.

Fig. 2: TSL rim driven tunnel thruster

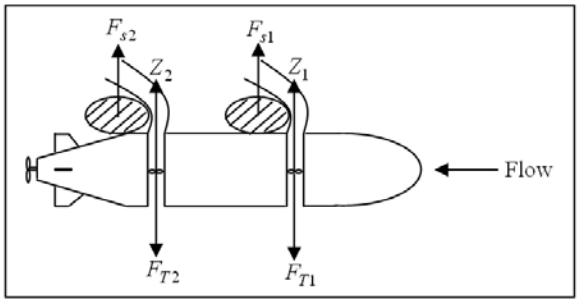

Fig. 3: Tunnel thrusters flow and forces

The low speed and hovering control aspects of a flight style AUV using tunnel thrusters are also different from those of an ROV. Whereas an ROV has significant roll and pitch stability arising from its layout, a flight style AUV does not. In an ROV the horizontal and vertical motion effectively decoupled whereas in a flight style AUV this is not possible. During flight mode the AUV needs to be able to pitch to overcome its positive buoyancy and hence only has a limited pitch and roll stability. This stability issue means that two thrusters are required for depth control to allow control of pitch unlike typical ROVs which use a single vertical thruster [7].

III.The Delphin AUV

A. Mechanical Design

The Delphin AUV is a scaled version of the National Oceanography Centre's Autosub6000 [8]. The length of the hull is 1750mm, and the length of the propulsion unit is 200mm, resulting in an overall vehicle length of 1950mm. A modular design such that the hull can be split into three sections: a nose, a tail and a cylindrical parallel mid-body. Vaccum formed plastic covers were used to provide a free flooded fairing for the nose and tail sections. The mid-body is made of an acrylic cylindrical pressure vessel with piston sealed end caps that are used to contain the electronics and batteries. The electronic components are mounted on the shelves attached to the end caps. The front part of the pressure vessel contains the battery and power management system while the rear section contains the computer and internal sensors that will be discussed in the following sections.

B. Central Controller System

The core of central controller is a Kontron 986LCD Mini-ITX motherboard with 2GB of RAM and a processor upgraded to a 2.2 GHz Core 2 Duo. The Kontron Mini-ITX board has many advantages for this vehicle. It has a small 170 x 170 mm area. It runs Windows software and has four serial port devices, eight USB ports, two firewire ports and an eight channel GPIO port. This connectivity allows all the components to be connected directly to the motherboard. The board is powered by an M2-ATX power supply and uses a 60GB Hitachi TravelStar 5K120 disc drive for data storage. Wifi is also provided by a mini PCI express board installed on the motherboard. The overall electronic system on board Delphin is illustrated in Fig.4.

C. Propulsion System

Delphin is designed to use a combination of a main propulsion motor, four 70mm, 50N rim driven tunnel thrusters and four rear control surfaces to manoeuvre the vehicle. These thrusters are used to provide the vehicle with forward, lateral, and vertical translations, as well as yaw and pitch rotations. The tunnel thrusters (Fig. 2) were provided by TSL technology Ltd, as mentioned earlier. They are controlled by a six-channel motor driver board also supplied by TSL Technology, which interfaces to the central control system via two serial ports.

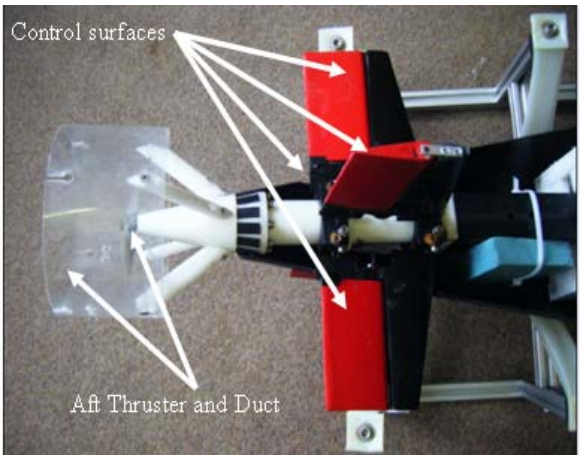

Fig. 5: The aft thruster and control surfaces

To facilitate manoeuvring the vehicle at higher forward speed the vehicle four movable control surfaces (Fig. 5) were designed [9]. The control surfaces have a NACA0015 section shape with a leading edge slope of 12º and are composed of a fixed skeg and a flap manufactured from high density foam and skinned with a layer of fibre glass and epoxy. The control surfaces are driven using HITEC low profile servos to adjust the plane angles. The servos are controlled by a Parallax USB servo driver board which is connected in turn to the main control motherboard. The main propulsion unit uses a Maxon brushless DC to drive the propeller. The Maxon motor is controlled by a Barracuda 80 Electronic Speed Controller (ESC) coupled to a Parallax servo driver board.

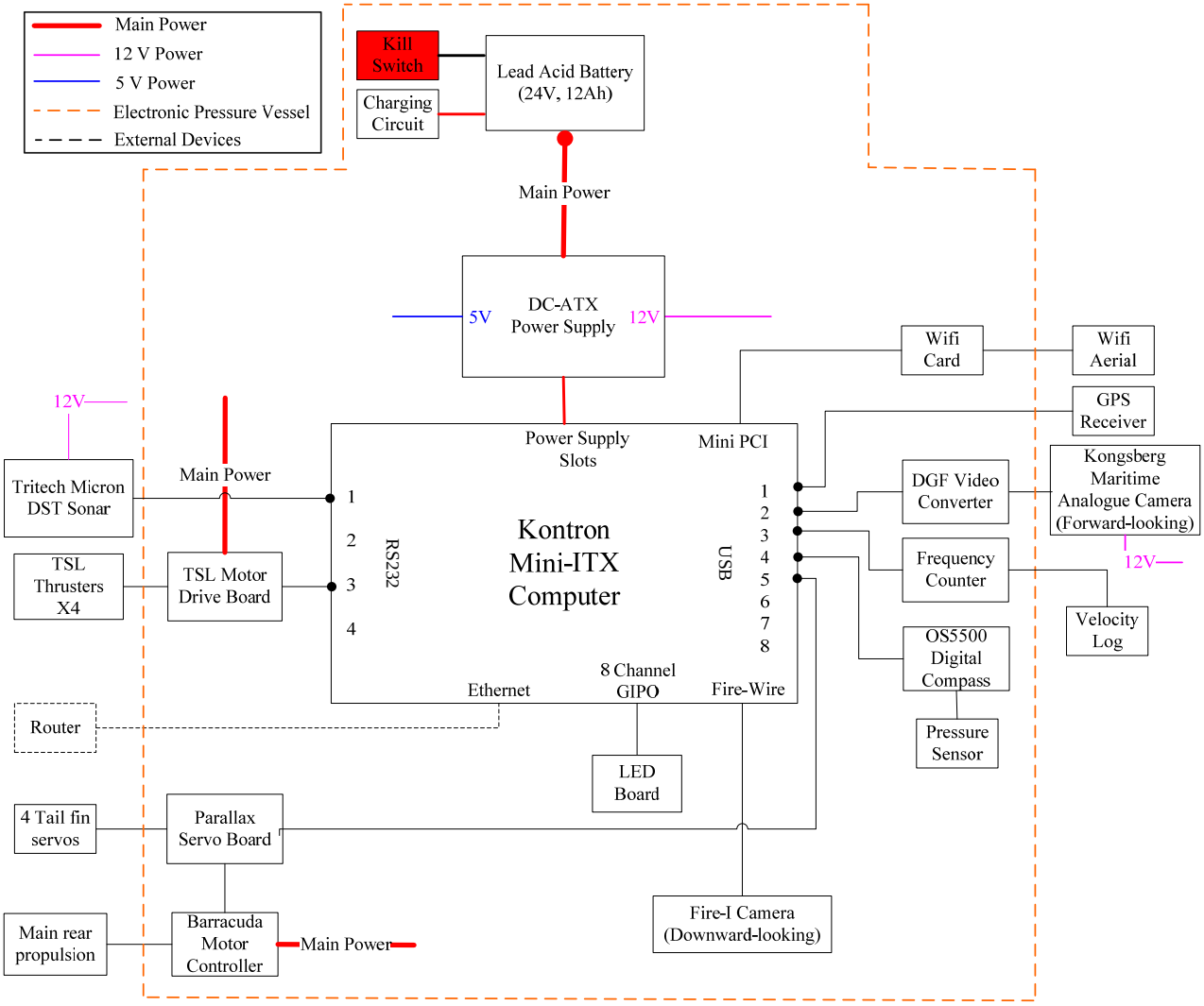

Fig 4: The overall electronic system onboard Delphin

D. Sensor Suite

The vehicle is equipped with two wide angle cameras and one imaging scanning sonar for the perception system, shown as Fig. 6. The imaging system consists of a forward-looking Konsberg Maritime OE14-110 CCD analogue underwater camera and a Unibrain fire-i firewire digital downward-looking camera. There is also an option to install a third upward looking camera. The forward-looking camera has a 73º angle of view in water and is contained within its own waterproof housing with a maximum operating water depth of 3000 meters; it is connected to a DFG video-to-USB frame grabber to convert the analogue output to digital signals for Scorpion Vision imaging processing. The obstacle avoidance and far off target detection system uses a Tritech Micron DST sonar which is a 675 KHz mechanically scanned sonar and normally used on small ROVs. The sonar is interfaced to the main computer system via an RS232 port. A digital compass from OceanServer OS5500 was used which provides a heading resolution of 0.1º. The compass consists of a three axis MEMS magnetometer and a 24bit A/D mounted on a 1" square PCB. The compass package can be also used to measure depth by connecting it a pressure sensor (Sensortechnics CTE 9005AY7). The pressure sensor provides a pressure range from 0-5bar absolute thus it can be used to measure depth up to 40 meters. The digital compass is interfaced via a USB and is configured to output the vehicle's

heading, roll, pitch, yaw, as well as the angular rates, accelerations and depth. The forward velocity is measured using a flow velocity log, which drives a small synchronous generator. The frequency of the output voltage of generator is proportional to the turbine speed which is counted by a frequency counter interfaced via a USB connection. A GPS receiver is used only when the vehicle is near the surface.

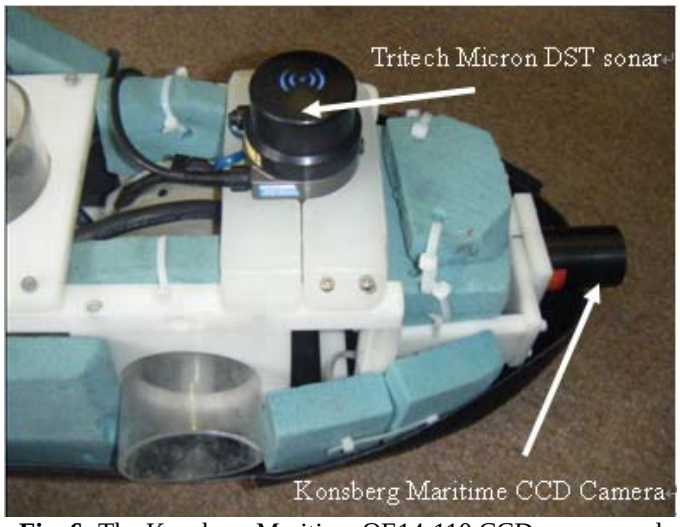

Fig. 6: The Konsberg Maritime OE14-110 CCD camera and Tritech Micron DST sonar

E. Battery Pack

The AUV battery pack consists of two 12V, 12 A.h lead-acid batteries arranged in series. The nominal voltage for the pack is 24V. The M2-ATX power supply board has a maximum input voltage of 24V with clamping occurring between 25-27V. The central computer and the tunnel thrusters operating at a typical rpm that could draw approximately 8-10A current, thus the battery pack enable submerged operation durations of over 1 hour. In order to indicate system status, an LED light board is fixed on the battery which consists of nine LEDs, one indicating power switch on/off status and the remaining eight LEDS are connected to the motherboard GPIO ports to indicate battery over current , low voltage, and the thrusters' operation status such as stall or serial port communication error.

IV. The Autonomy System and Control Software

The autonomy system runs within the MS Windows XP environment. Two software packages were used: Matlab/Simulink with a set of toolboxes and Scorpion Vision. These software packages have many advantages for the development of the autonomy system and control software. The Matlab/Simulink contains many toolboxes which provide a simple and quick way of developing, testing and debugging the control software, and it also supports the multi-tasking requirement by the vehicle. The computer vision system is implemented using Scorpion Vision, which is a high-level computer vision package containing many customizable frameworks for tools such as edge detections, blob, and colour/ texture matcher. These tools can be easily adapted and combined to detect features in images such as colour or shape. The Video and Image Processing Blockset within Matlab/Simulink could also potentially be used to perform the image processing tasks, however, this was found to require more computational resources than Scorpion Vision, which slows down the low level control loops leading to system instability due to large time lags.

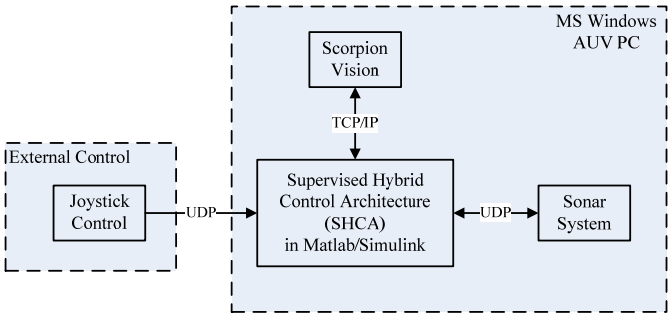

Fig. 7: The autonomy system software structure

The autonomy system and control software can be divided into four parts as illustrated in Fig. 7: 1) the main controller is implemented within the Supervised Hybrid Control Architecture (SHCA) block. All codes associated with mission planning, low-level controllers and hardware interfacing are developed and executed within this block. The Stateflow toolbox is used to implement the mission controller and Matlab/Simulink blocks and S-functions are used to implement the low level control and hardware interfacing. 2) The Joystick controller which sets up a joystick operation mode that is used for remote control vehicle via a joystick; 3) The sonar system software, which is designed to execute independently of the main controller, performs a target detection and obstacle avoidance algorithm and sends outputs decisions to the main controller. This system is implemented in Matlab/Simulink; 4) Scorpion Vision was used to implement the computer vision system which processes the input image frames from both cameras and outputs information to the main controller.

Two communication systems are used in the autonomy system. The communication with the joystick control and sonar system was setup via the Instrument Control toolbox contained within Matlab/Simulink, and the communication with Scorpion Vision was setup using the ActiveX Control in Matlab.

A. The Supervised Hybrid Control Architecture

The SHCA is hybrid deliberative-reactive control architecture similar to that described in [10] [11] [12]. In this section, the design criteria of the SHCA will be first summarized. This is then followed by a description of the hardware/software configuration and implementation of the SHCA.

Design Criteria of SHCA

The design criteria of SHCA for Delphin AUV can be summarized as following [10]:

- Modularity: The control architecture need to be modular allowing the addition of third party hardware or software modules.
- Real-Time Performance: The control system need to be

having a fast real-time response to enable the vehicle to respond quickly and appropriately to varying environmental conditions.

- Reliability: This is important for the vehicle because it needs to perform without human intervention.

The solutions of hardware and software to satisfy these criteria are presented in the following section.

Software Configuration

The software configuration of SHCA is illustrated in Fig. 8. The architecture can be divided into following three layers:

- Mission Planning Layer: provides high-level control of the vehicle during a mission and is responsible for the mission planning, execution and supervision.
- Control Layer: is responsible for the low-level control of the vehicle behaviour.
- Interfacing Layer: comprises of the sensor and actuator groups within the vehicle and related electronics and interfacing hardware.

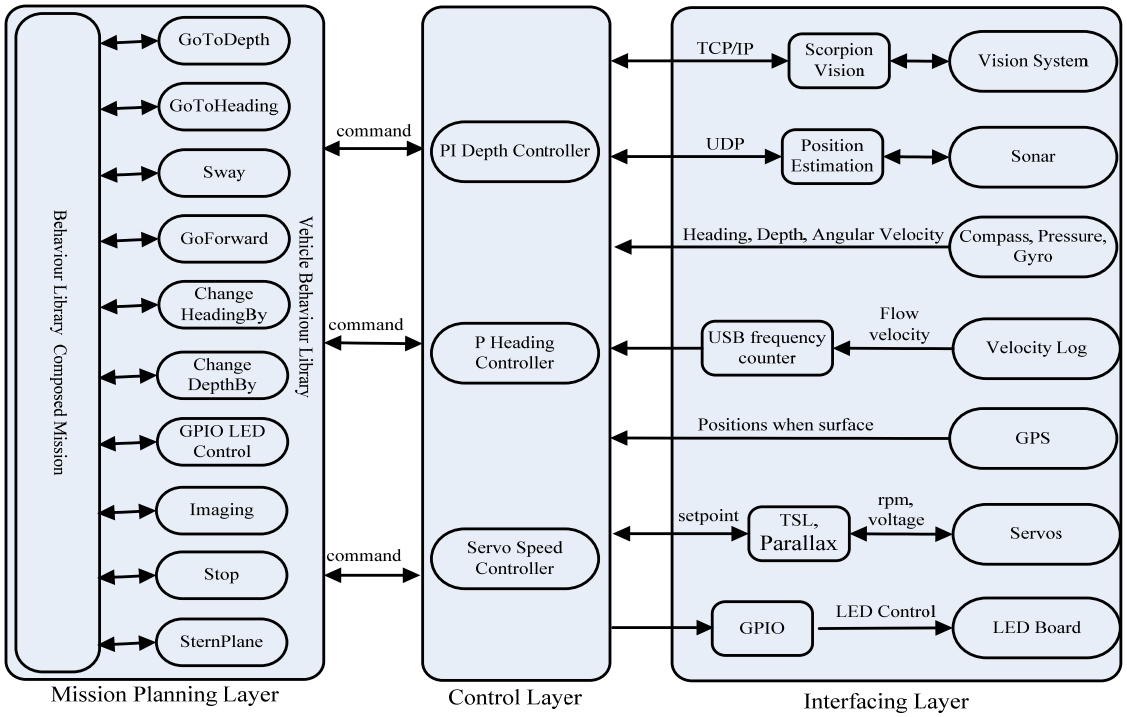

Fig. 8: The system architecture and data flow

Mission Planning Layer

The Mission Planning Layer is responsible for high-level system control, which contains three different modes: constant value setpoint mode, used for testing and development of the low level controller; a joystick operation mode, used for receiving the joystick signals for remote control the vehicle, and an autonomous mode, used for performing predefined missions as defined using by a Matlab Stateflow toolbox state flow diagram.

The methodology adopted for the design of a mission control system in Stateflow was built on a series of basic building blocks, namely Vehicle primitive (VPs). Each VP is a predefined embedded function within Stateflow that defines basic actions of the vehicle. There are 10 VPs within the Delphin's mission controller such as VPs to go to desired depth/heading, go forward, and imaging of hovering area.

The implementation of the GoToHeading primitive using Stateflow is illustrated in Fig. 9. Two variables are specified: the variable *mode* is an enable/disable setting which switches ON/OFF of the vehicle heading controller, and the variable of *value* is related to the heading demand. Each VP is in charge of a specific objective and is performed through an enable/disable flag set by the mission control layer. When the autonomy system is running, the Planning Layer takes inputs from the navigational sensor, sonar system and cameras system, and then activates related VPs using this information to update the demands sent to the low-level control layer. This simple and flexible structure of building blocks allows complex missions to be constructed in a rapid manner alongside the development of the low level control.

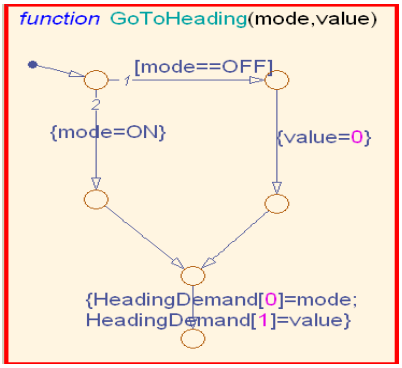

Fig. 9: The example GoToHeading vehicle primitive

Within Stateflow, the mission is decomposed into a number of sub-tasks. A sub-task module consists of and basic building blocks of VPs. This means that each sub-task module is designed to perform a well defined operation with clear goals and procedures. The execution of the sub-task is monitored by the Task Scheduler which controls the execution order of sub-task, checks the completion of sub-tasks, and directs the vehicle to move onto the next sub-task.

The Mission Planning Layer also contains a series of programmed housekeeping tasks. These include the provision of emergency procedures and obstacle avoidance. The emergency procedures account for occurrences of hardware failure, lost control authority such as overstep defined pitch angle and the maximum hull sustainable depth. The obstacle avoidance is performed by the execution of the sonar system, which measures the distances to the obstacle and feed back to the mission controller.

Interfacing Layer

The Interfacing Layer software module interfaces the software to the hardware. The interfacing between sensors and actuators with central computer has been presented in section III.

Control Layer

The Control Layer is implemented within the native Simulink environment. There are three primary low-level controllers: a low speed vertical controller, a low speed horizontal controller, and a forward speed controller. The Control Layer takes the demands from the Mission Planning Layer, and then the low level controller converts these demands into the thrusters' setpoints and activates appropriate actuators. The low level controllers send demands to the Interfacing Layer to drive required actuators. The following section describe the Low Level Control System

B. Low Level Control

This section discusses the challenges of implementing the low level control systems.

Low Speed Control

The tunnel thrusters are driven by brushless dc motors that are powered by a sensorless control board shown in Fig. 10, which can drive up to 6 thrusters independently, although only four are used on Delphin.

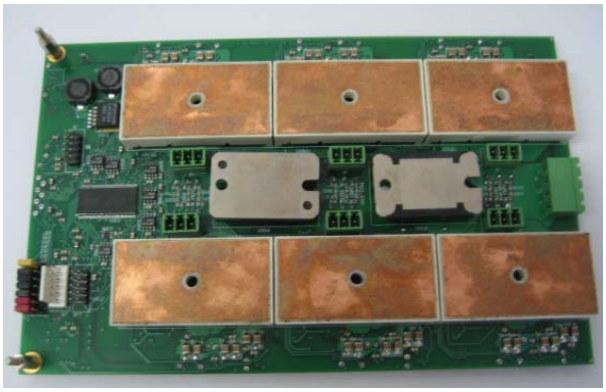

Fig. 10: TSL servo controller

Due to the nature of the sensorless control system, which relies on measuring the emf, it is not possible to control the motors at low speed in closed loop mode with position feedback when the emf (which is proportional to speed) is too small to measure. Low speed control is possible by operating the motors are stepping synchronous motors, which is the method used for starting. But this mode of control, which is less efficient than the closed loop mode, has not yet been implemented. The speed of the thrusters therefore exhibits a deadband with some hysteresis as illustrated in Fig. 11. Tests show that the minimum closed loop thrusters speed is approximately ±210 rpm, but the motor does not reliably start if the speed set point is set to this value. To ensure reliable starting, the speed setpoint is set to a higher Startpoint value and then the speed is gradually reduced to the desired value if it was lower than the Startpoint.

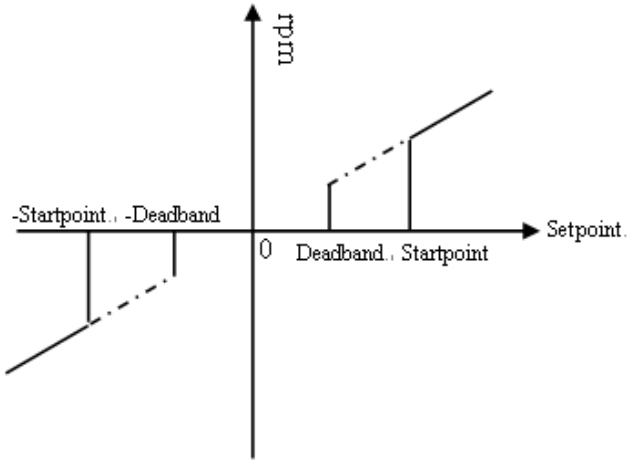

Fig. 11: The input output characteristic

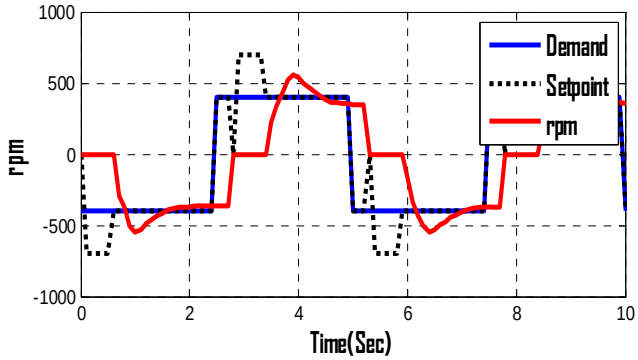

Fig. 12: The TSL thruster setpoint and rpm

Fig. 12 shows the TSL thruster's performance after control. The demand is a square wave which given as ±500 and the dash line represents the setpoint input to the thrusters after control. Once the controller detects the stall of thrusters, the setpoint will be reset to zero and then give a high Startpoint value, it can be seen that the stall issue of thruster has been solved.

Low Speed Vertical Control

For a flight style AUV, the control of depth and pitch are coupled. The low speed depth controller is performed with a

PI controller which assigns demands to the forward and aft vertical tunnel thrusters with the vehicle pitch controlled by a differential between the demands.

Fig. 13 illustrates the block diagram of depth control, with the pitch feedback in the inner loop and depth feedback in the outer loop. Due to the depth sensor being mounted on the end cap of the pressure vessel, the distances between forward (forward arm) and aft (aft arm) vertical tunnel thrusters to pressure sensor are used for compensating the pitch, which is multiplied by the pitch angle (in radians) to compensate for depth error. The saturating of the integrator is used for anti windup purposes. The error was also saturated to provide a gentle diving behaviour during control.

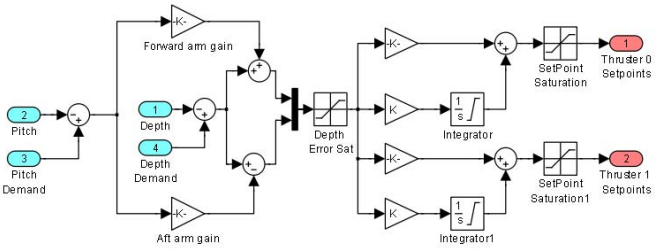

Fig. 13: The low-speed vertical control block diagram

Low Speed Horizontal Control

The low speed heading controller is a proportional controller based on the heading error and sends equal but opposite demands to the forward and aft horizontal tunnel thrusters. A sway demand results in identical commands being send to both thrusters.

Fig. 14 illustrates the block diagram of horizontal control. The horizontal control contains the heading controller and sway controller. Due to the thrusters speed hysteresis, it was noticed that there was considerable heading overshoot in the heading control. In order to reduce the chattering caused by the overshoot, a dead zone of ±4º of heading is preset. The sway controller directly assigns the demands to each horizontal thruster through a proportional gain.

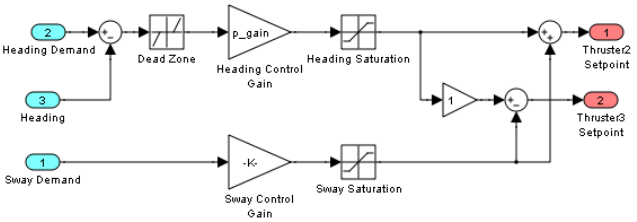

Fig. 14: The low-speed horizontal control block diagram

C. The Real Time Performance of the Control System

The Matlab based control approach is not a real time system in that it is not possible to guarantee that it will update the system at a given time. To facilitate an approximation to real time the Simulink model has a real time function added to it that pauses the simulation to maintain pace with real cpu time. This works adequately, but if the simulation step takes longer than real time the simulation is not paused and the simulation time will drift compared to the real time. This sets the minimum bound on the real time step seen using this approach. The minimum real time step was determined by seeing how long each time step would take and then setting the time step for the simulation to be longer than this.

Fig. 15 plots the distribution of the sampling time steps when the system runs only the whole mission controller. It shows a minimum sampling time step is 0.078 seconds and a maximum sampling time step is 0.625 second. The mean time step is 0.14645 second, which indicates the sampling frequency of the system is approximate 6.82 Hz. The figure also shows the system has 99.68% of the sampling time steps are less than 0.25 second in real time.

Fig. 16 plots the distribution of the sampling when running the mission controller and the Scorpion Vision software. The percentage of time steps below 0.25 seconds has dropped to 99.5534%.

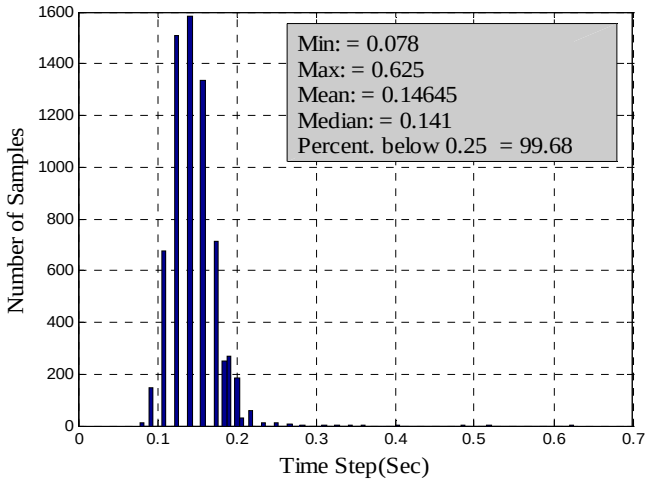

Fig. 15: Distribution of the sampling time step when running the mission controller

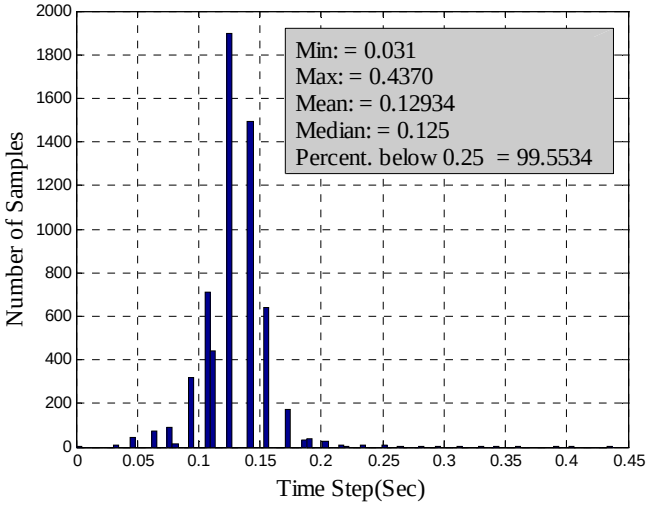

Fig. 16: Distribution of the sampling time steps when running the mission controller and Scorpion Vision

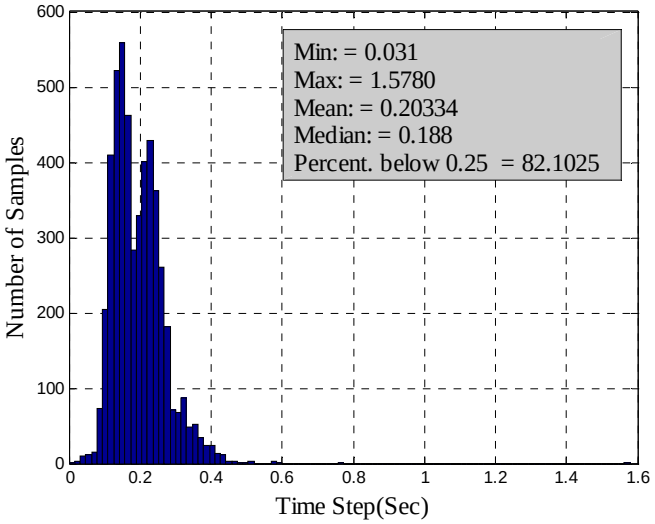

Fig. 17: Distribution of the sampling time steps when running mission controller, Scorpion Vision, and sonar system

Fig. 17 plots the time steps and the number of samples when running the mission controller, Scorpion Vision software and sonar system. It shows the maximum time steps of sample requires 1.5780 second and the mean sample rate is 0.20334 second, which indicates the system sample frequency drops to approximate 4.91Hz, this is due to the sonar system requiring more computational power to coordinate the position of the vehicle.

This approach does not guarantee real time processing, but has provided a reasonable approximation to it that is sufficient to run the AUV and to tests its algorithms.

V. Performance of the Autonomy System

In order to assess the performance of the autonomy and control system of the Delphin AUV, a number of tests were conducted within the Acoustics Tank at the National Oceanography Centre. Throughout the mission, a wifi antenna was mounted on the vehicle thus enabling radio communication between the vehicle when it surfaced and an on-shore station. The communication was setup via MS windows XP build-in tools of Remote Desktop.

Fig. 18 to 22 illustrate the data acquired during the course of the test, which show the commanded and measured depth, heading, pitch and the thrusters' activity, respectively.

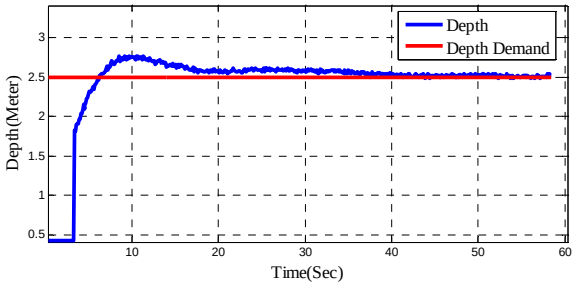

Fig. 18: Depth command vs. depth

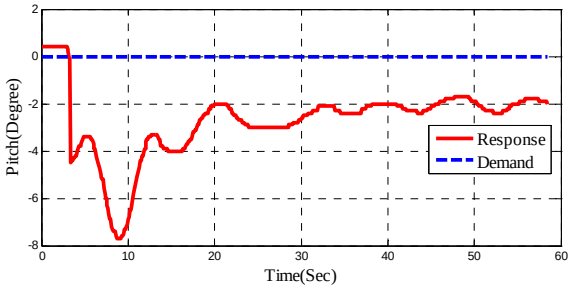

Fig. 19: pitch command vs. pitch

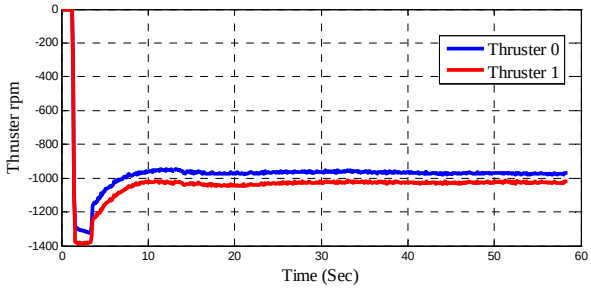

Fig. 20: Vertical thrusters' behaviour

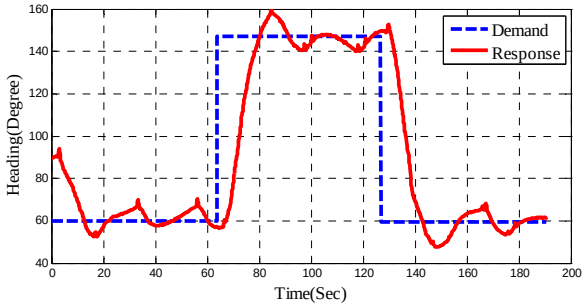

Fig. 21: Heading command vs. heading

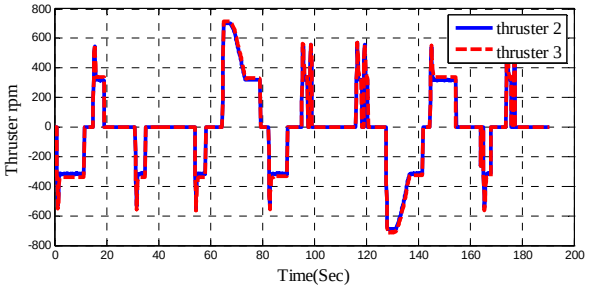

Fig. 22: Horizontal thrusters' behaviour

Fig. 18 shows the performance of the depth controller which has a 0.2 meter overshoot initially and drop to target depth after 30 seconds. Fig. 19 shows the pitch angle during the depth control. It varies between approximately 8 º and 2 º when the depth reaches steady state. Fig. 20 shows two vertical tunnel thrusters rpm during depth control. Fig. 21 plots the performance of heading controller. The demand of the heading was set first 60 º and then changed by 90 º within 60 seconds, and change back to 60º within 60 seconds afterwards. The response of heading has approximate approximately a 5 º overshoot, which is due to the thrusters speed deadband, which means that the thrusters could only be

run at a relatively high-speed thus becoming too powerful for vehicle heading control. The thrusters therefore have to reverse speed very frequently as illustrated in Fig. 22 which shows the thrusters' activities of the horizontal control.

Fig. 23 shows the Delphin AUV is performing a mission tested for tracking and hovering on a simulated bottom circle target. Fig. 24 shows the vehicle performing a tracking mission; the target is a floating orange coloured ball moving in the middle of the water.

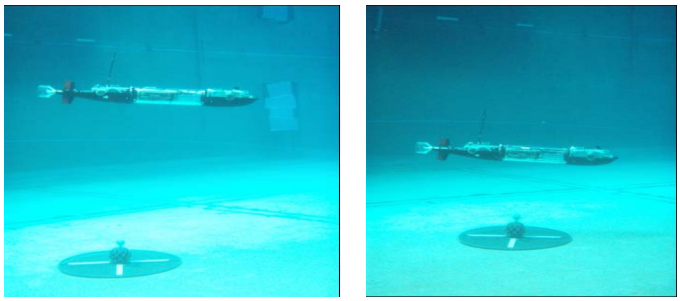

Fig. 23: Bottom circle target tracking test

Fig. 24: Orange ball target tracking test

VI. Conclusion

The paper introduced ongoing work on the development of a flight style Delphin AUV with hovering capabilities. The SHCA control system has been proven to be effective in controlling the motion of the AUV. Furthermore, Matlab/Simulink with MS Windows XP environment has been proven to provide a simple and accessible solution for designing the autonomy system of the vehicle. But a powerful computer is required to enable satisfactory real time operation. The thrusters' speed deadband provides an additional challenge, which causes significant overshoot and solutions are needed to either reduce or eliminate the deadband, or improve the control system to reduce the overshoot and improve control accuracy.

VII. Future Work

The focus of future work has been on the development of achieving hard real time control for the SHCA control architecture. To overcome this real time issue, the autonomy system will be recoded to run under the Matlab Windows Real-Time Windows Target and this will be implemented in the near future.

Acknowledgement

This research partly sponsored by National Oceanography Centre, Southampton and the School of Engineering Sciences, University of Southampton.

References

[1] Dunbabin, M.; Roberts, J.; Usher, K.; Winstanley, G.; Corke, P., "A Hybrid AUV Design for Shallow Water Reef Navigation," *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on* , vol., no., pp. 2105-2110, 18-22 April 2005

[2] Akhtman, J.; Furlong, M.; Palmer, A.; Phillips, A.; Sharkh, S.M.; Turnock, S.R. SotonAUV: the design and development of a small, manoeuvrable autonomous underwater vehicle, *Underwater Technology: The International Journal of the Society for Underwater Technology*, Volume 28, Number 1, 2008 , pp. 31-34.

[3] The DSTL (SAUC-E) Student Autonomous Underwater Competition – European http://www.dstl.gov.uk/news_events/competitions/sauce/09/index.php

[4]Abu Sharkh, S.M., Lai, S.H. and Turnock, S.R. Structurally integrated brushless PM motor for miniature propeller thrusters. *IEE Proceedings - Electric Power Applications*, 151, (5), 513-519. 2004

[5]Abu Sharkh, S.M., Turnock, S.R. and Hughes, A.W. Design and performance of an electric tip-driven thruster. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 217, (3), 133-147, 2003

[6] Palmer, A.R., Hearn, G.E. & Stevenson, P. Experimental Testing of an Autonomous Underwater Vehicle with Tunnel Thrusters. *Proceedings of the First International Symposium on Marine Propulsors*, pp569-575, Trondheim, Norway June 22-24 2009.

[7] SeaEye ROV website: http://www.seaeye.com/rovs.html

[8] Maaten Furlong, M. G., Henrik Hasselstrom, Jiazhong Liu, Alistair Palmer,, S. S. Alexander Philips, Muhammad Arif Sulaiman Hon,, et al. "Delphin: The University of Southampton Entry into the 2008 Student Autonomous Underwater Challenge - Europe." *The Student Underwater Challenge-European (SAUC-E), Brest, France*, 2008.

[9]Molland, A.F. and Turnock, S.R. Marine rudders and control surfaces: principles, data, design and applications, *Butterworth-Heinemann*, 414pp 2007.

[10]M. Meystel and J.S. Albus, "Intelligent Systems: Architecture, Design and Control", *Newyork: John Wiley & Sons*, 2002

[11]R.A. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE J. Robot. Auto.*, 2 pp.14-23,1986

[12]Gat, E. "Three Layer Architectures". *Artificial Intelligence and Mobile Robots, edited by The AAAI Press. The MIT Press*, ch. 8, 1998.

[13] J. Kim, C. I and H. Shin, "A New Task-Based Control Architecture for Personal Robots" *IEEE*, pp.1481- 1486 vol.2 ,2003