2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

ICTINEU^{AUV} Wins the First SAUC-E Competition

D. Ribas, N. Palomeras, P. Ridao, M. Carreras and E. Hernàndez.

Abstract— **A pioneer team of students of the University of Girona decided to design and develop an Autonomous Underwater Vehicle (AUV) called ICTINEU^{AUV} to face the Student Autonomous Underwater Challenge - Europe (SAUC-E). The prototype has evolved from the initial Computer Aided Design (CAD) model to become an operative AUV in the short period of seven months. The open frame and modular design principles together with the compatibility with other robots previously developed at the lab have provided the main design philosophy. Hence, at the robot's core, two networked computers give access to a wide set of sensors and actuators. The Gentoo/Linux distribution was chosen as the onboard operating system. A software architecture based on a set of distributed objects with soft real time capabilities was developed and a hybrid control architecture including mission control, a behavioural layer and a robust map-based localization algorithm made ICTINEU^{AUV} the winning entry.**

I. INTRODUCTION

From 1990, the Association for Unmanned Vehicle System International (AUVSI) has promoted the design and development skills of Autonomous Underwater Vehicles (AUV) by means of an annual competition for the american students [1]. Inspired by this competition, the Defence Science and Technology Lab (DSTL), the Heriot Watt University and the National Oceanographic Centre of Southampton have organized the first Student Autonomous Underwater Challenge Europe (SAUC-E) [2]. SAUC-E is a competition for european students to foster the research and development of underwater technology. In January 2006, a team of students collaborating with the Underwater Robotics Lab of the University of Girona decided to form the VICOROB-UdG Team to face the challenge [3]. Given the short period of time to invest in the project, our team decided to overlap the hardware and the software development (concurrent engineering) taking advantage of a hardware in the loop (HIL) simulator Neptune [4]. This paper describes the ICTINEU^{AUV} as an entry to the 2006 SAUC-E competition (additional references to similar vehicles can be found in [5], [6]). The paper is organized as follows. The mechanical, hardware and software design are explained in sections II to IV. Section V explains the map-based navigation system. Finally, sections VI and VII present the mission and the results respectively before concluding in section VIII.

Manuscript received September 15, 2006. This work was supported in part by the Dirección General de Investigación of Spain under project DPI2005-09001-C03-01.

The authors are with the Dept. Electrònica, Informàtica i Automàtica, Universitat de Girona, 17071 Girona, Spain (dribas@eia.udg.es, npalomer@eia.udg.es, pere@eia.udg.es, marcc@eia.udg.es, emilihb@eia.udg.es)

II. MECHANICAL DESIGN

The SAUC-E mission takes place in a small confined area in which a high manoeuvrability is required. In this situation a hover-type vehicle propelled and steered by thrusters is the most desirable configuration. The classical open frame design, commonly adopted by commercial ROVs, together with a modular design of the components conveniently housed in pressure vessels, is probably the simplest and most reliable approach. Although hydrodynamics of open frame vehicles is known to be less reliable than the hydrodynamics of close hull type vehicles, it is simpler and cheaper, as it is very easy to upgrade and maintain. For all these reasons ICTINEU^{AUV} has adopted the open frame design (see Figure 1). Our robot is propelled by four thrusters. It can move in the heave and sway directions depending on the composition of forces generated by the vertical thrusters. On the other hand, horizontal thrusters are used to move forward (surge DOF) as well as to change the heading (yaw DOF). Hence, the prototype is a fully actuated vehicle in four DOF (surge, sway, heave and yaw), while being passively stable in Roll and Pitch as its meta-centre is above the centre of gravity. The robot chassis is made of Delrin material. Three pressure vessels are used for holding the electronics. The two bigger cylinders are made of aluminium while the smaller one is made of Delrin. They all have a cover with all the connectors and use a conventional O-ring rubber for water-tight sealing. One of the cylinders houses the computers, another the thruster controllers and the batteries, and the last encapsulates the Motion Reference Unit (MRU).

Fig. 1. ICTINEU^{AUV}, the VICOROB-UdG Team's entry for the SAUC-E 2006 competition.

The thrusters were built using MAXON DC motors of 250 Watts, using planetary gears and contained in stainless steel housings. Three blade brass propellers are linked to the motor by a stainless steel mechanically sealed shaft, providing around 14.7/14.2 Newtons of forward/backward thrust. Buoyancy is provided by a cover of technical foam with 10.5 litres volume and a weight of 0.6 Kg.

III. HARDWARE DESIGN

A. Computer Module

Two PCs, one for control and one for image and sonar processing connected by a 100 MBs switch, form the core of the robot's hardware. The control PC is an AMD GEODE-300MHz powered by a 50 W power supply module. The PC104 stack also incorporates an A/D and digital I/O card with 8 analogue input channels, 4 analogue output channels and 24 digital I/O. The mini-ITX computer is a Via C3 1 GHz Pentium clone and is used to process the data from the imaging sonar and the cameras. A cheap PCTV110 from Pinnacle is used for image processing.

B. Power module

The power module contains the four power drivers for the thrusters as well as a pack of 2 cheap, sealed lead-acid batteries. A DC-DC converter is included to provide a stabilized voltage to the rest of components. There is also a simple relay circuit which commutes between the internal and the external power. External power, supplied through an optional umbilical, is very useful for running long term experiments before the competition. It is worth noting that the batteries have been dimensioned for the short time experiments to be done during the competition days. Moreover, when the robot works with external power it can recharge the internal batteries.

C. Sensor Suite

The vehicle is equipped with a complete sensor suite composed of a forward-looking color camera, a downward-looking b&w camera, a MRU MTi from XSens Technologies, a Miniking imaging sonar from Tritech, an echo sounder, a transducer for acoustic device detection and an Argonaut Doppler Velocity Log (DVL) from Sontek which also includes a compass/tilt sensor. Additional temperature and pressure sensors and water leak detectors were installed into the pressure vessels for safety purposes.

D. Actuators

In addition to the four thrusters previously mentioned, the vehicle is also equipped with an actuator to release the markers when the bottom target is detected (See mission description in Section VI)

IV. SOFTWARE DESIGN

The software architecture has the task of guaranteeing the AUV functionality. The real-time POSIX, together with the ACE/TAO CORBA-RT ORB, have been extensively used to develop the architecture as a set of distributed objects

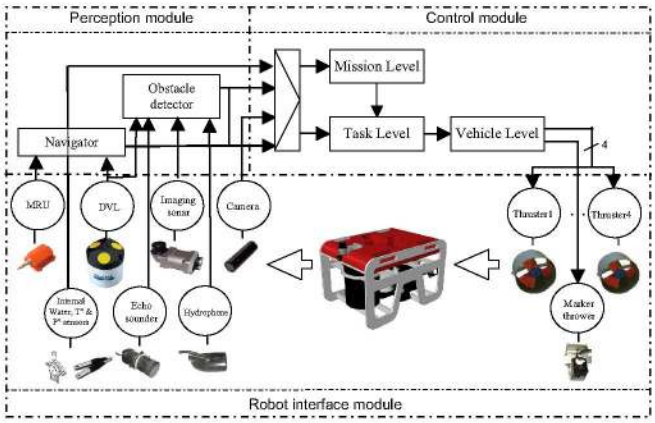

Fig. 2. Schematic of the ICTINEU^{AUV} software arquitecture.

with soft real time capabilities. These objects are distributed among the two onboard PCs and the external PC. The last one is only used during the experiments in the lab, being connected to the robot through the umbilical cable for monitoring purposes. The architecture is composed of a base system and a set of objects customized for the desired robot. There are classes providing soft real-time capabilities, this guarantees the period of execution of the periodic tasks such as the controllers or the sensors. Another important part of the base systems are the loggers. A logger system is used to log data from sensors, actuators or any other object component. Loggers do not execute in real time, they are background processes which receive the data from real time objects. Their role consists of packing the data and saving them into files. It is worth noting that, although loggers do not run in real time, the data has a time-stamp corresponding to the gather time. Moreover, all the computers in the network are synchronized by means of the NTP (Network Time Protocol) and hence, all the data coming from different sensors can be time related. The software architecture is divided between three modules (fig. 2): Robot interface module, Perception module and Control module.

A. Robot interface module

This is the only module that containing software objects that dialog with the hardware. There are basically two types of objects: sensor objects responsible for reading data from sensors and actuator objects responsible for sending commands to the actuators. Sensor objects for ICTINEU^{AUV} include a DVL, an imaging sonar, an MRU, two cameras, a depth sensor, and an echo sounder. There are also objects for the safety sensors like water leakage detectors and internal temperature and pressure sensors that allow for the monitoring of the conditions within the pressure vessels. Actuator objects for the ICTINEU^{AUV} include the thrusters, and the marker thrower.

B. Perception module

This module contains two basic components: the Navigator and the Obstacle Detector. The Navigator object has the goal of estimating the position of the robot. To accomplish

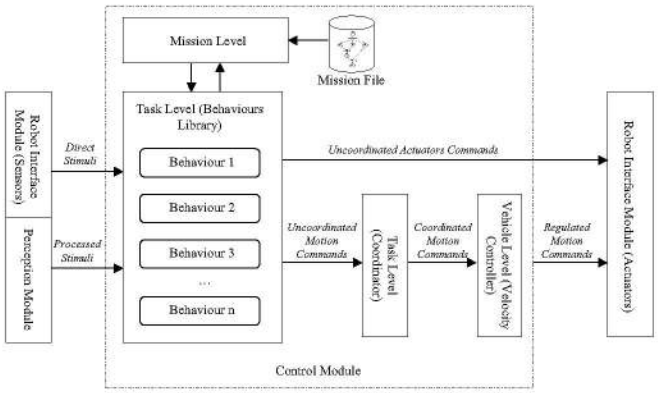

Fig. 3. Schematic of the ICTINEU^{AUV} control architecture.

this task, there is an interface called NavigationSensor from which all the localization sensors (DVL, MRU, depth sensor) inherit. This interface provides all these sensors with a set of methods to return the position, velocity and acceleration in the six DOF together with an estimation of the quality of these measurements. The Navigator can be dynamically connected to any NavigationSensor and, using the quality factor, fuses the data to obtain a more accurate position, velocity and acceleration. Furthermore, the Navigator can also access the imaging sonar to implement the navigation method described in section V. The Control module uses the navigation data provided by the Navigator keeping the behaviours independent of the physical sensors being used for the localization. The Obstacle detector uses the same philosophy to provide obstacle position in the world fixed frame. The Obstacle detector is also used to detect the distance between the vehicle and the bottom of the pool. Detecting frontal obstacles is possible using the echo sounder or the imaging sonar, and the pool bottom obstacles can be detected with the DVL sensor.

C. Control module

The control module receives sensor inputs from the perception module and sends command outputs to the Actuators residing in the Robot Interface Module (fig. 3). Since task and behaviours are words that are interpreted in different ways for different authors in the literature, hereafter, we describe how they are interpreted within our project. A behaviour is a function that maps the sensor input space (stimuli) into a velocity setpoint (behaviour response) for the robot's low level controller. The behaviour response is chosen in a way that drives the robot towards its corresponding goal. In this way, the goal corresponding to the KeepDepth behaviour is considered to be achieved when the robot is within an interval around the desired depth. A task is a set of behaviours that are enabled together to achieve a more complex goal. For instance, KeepDepth and MoveTo2D can work together to allow for planar navigation. The control module follows the principles of the hybrid control architecture organized in three layers: Vehicle Level, Task level and Mission level.

Vehicle Level

The vehicle level is composed of a MIMO PID velocity controller for each DOF. This object reads the vehicle velocity from the Navigator object and receives the velocities set points from the Coordinator Object. This level also includes a simple control allocator strategy based on the pseudo inverse of the thruster configuration matrix [7].

Task level

The Task level is a conventional behavioural layer [8] including a library of behaviours that can run alone or in parallel. Each behaviour has a particular goal. The input of a behaviour can be taken from any object of the software architecture (sensors, perception module...). The output, called behaviour response, contains:

- Velocity setpoints for every DOF normalized between -1 and 1.
- Activation level for every DOF normalized between 0 and 1 indicating how important it is for the behaviour to take control of the robot.
- Blocking term (Boolean) set stating if the behaviour must block the execution thread of the mission level.

To initialize a behaviour, apart from setting its particular parameters, it is necessary to specify the following attributes:

- *Enable:* Boolean variable that indicates if the behaviour is activated or not and if its output will be considered by the Coordinator.
- *Priority:* Priority stating the relative importance of each behaviour.
- *TimeOut:* The time out indicates when the behaviour will block the execution thread. If TimeOut<0, the behaviour blocks the execution thread until its goal is fulfilled. If TimeOut$=0$, the behaviour doesn't block the execution thread. If TimeOut>0, the behaviour blocks the execution thread until TimeOut seconds or until its goal is fulfilled.

During the execution of a mission, more than one behaviour can be enabled simultaneously. Hence, a coordinator module is used to fuse all the responses corresponding to the enabled behaviours into a single response to be sent to the velocity controller (Vehicle Level).

Each degree of freedom is considered separately since not all the behaviours act over all the DOF. To combine all the behaviour responses, the Coordinator sorts all the responses by their priority and combine the responses for every DOF, two by two, from the highest priority to the least. To combine the responses, the activation level and a k factor are used. Equation 1 show this process. a_1, a_2 and s_1, s_2 correspond to the activation level and the desired setpoints for the highest priority and the least priority behaviour respectively, while s corresponds to the final coordinator response.

$$s = \frac{a_1 s_1}{a_1 + a_2(1-a_1)^k} + \frac{a_2 s_2 (1-a_1)^k}{a_1 + a_2(1-a_1)^k} \qquad (1)$$

The Coordinator output, after combining all active behaviours, is a vector as large as the number of the robot's DOFs. Each value corresponds to a normalized velocity [9].

Mission level

Finally, the upper layer (mission level) is responsible for the sequencing of the mission tasks, selecting for each mission phase the set of behaviours that must be enabled as well as their parameters.

The mission controller was built with a Petri network in which the sequence of tasks is defined. Since the vehicle moves in an unstructured environment, unexpected situations have to be taken into account by the mission designer. According to the network, some nodes will become active. Each node represents a behaviour that will be executed on the task controller. There is a library of behaviours that are used to define a mission. Each one has a simple goal such as move to point, keep depth, search a target, etc. Therefore, the mission controller has the work of defining the task that the robot is accomplishing at each moment by activating or deactivating behaviours with the final goal of fulfilling the mission. The mission controller does not determine the actions that guide the robot, it only determines the active behaviours and its configuration which, through the task controller, will be coordinated to guide the robot.

In our Petri net, every place corresponds to one behaviour with a particular configuration. When a place has a token, this behaviour is enabled. When all places that go towards a transition are enabled, and their behaviours do not block the execution thread, the transition is ready to be fired. When a transition is fired, a token is removed from each of the input places of the transition and a token is generated in each output place of the same transition. The control mission algorithm starts on the initial state, checks fired transitions, applies the previously explained procedure, and repeats this process until it reaches the final state [9].

V. NAVIGATION

An essential component of an AUV is a reliable navigation system [10]–[12]. In applications in which a previous knowledge of the scenario is available, the localization problem can be addressed by using the information from the exteroceptive sensors together with an *a priori* map [13].

The navigation system proposed hereafter makes use of a mechanical scanning imaging sonar and a compass for the navigation on the horizontal plane, while the depth of the vehicle can be easily measured with a pressure sensor. Mechanical scanning sonars perform scans in a 2D plane by rotating a sonar beam through a series of small angle steps. For each emitted beam, distance vs. echo-amplitude data is returned forming an acoustic image of the surroundings (Figure 4). The objects (walls) present in the environment appear as high echo-amplitude returns. Hence, it is possible to extract features from which it is posible to determine the initial position of the vehicle inside the map [14]. In the context of the SAUC-E competition, where the orientation, dimensions and shape of the water tank are known, a very simple but effective procedure based on the Hough transform [15] can be used. In our particular case, we don't search for features but go directly for the vehicle position. The Hough voting space is defined as a discretization along the

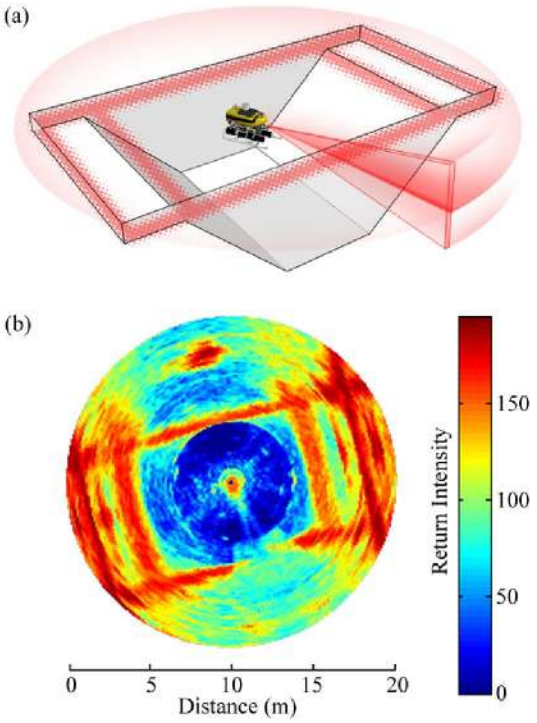

Fig. 4. (a) Schematic representation of the environment where the example sonar data were gathered. The highlighted zones represent the expected sonar returns. (b) Image generated from acoustic data

XY coordinates of all the possible vehicle positions inside the water tank playground (Figure 5). Each time a single beam is available, its highest intensity return is selected as the most likely evidence of the presence of the boundary walls (Figure 5b). Each measurement defines a particular zone in which the vehicle can be (Figure 5c). By voting on these zones and accumulating information in a complete 360° scan, the position of the vehicle can be determined. As can be seen in Figure 5d, the zone with the highest number of votes is easily identifiable and matches the vehicle's position. The imaging sonar takes about 6 seconds to complete a scan. However, in order to increment the rate of position estimates, the algorithm maintain a measurement pipeline so the last complete scan data can be recovered at any moment. In our final implementation, it was set to search for the position every 0.5 seconds. Finally, in order to reduce the computational cost of the system and make it more robust, a resolution of 0.5 meters was set for the voting space.

VI. THE MISSION

The SAUC-E competition takes place in a water tank environment of 20 meters by 10 meters and a depth of 6 meters. The mission consists of (see Figure 6):

1) Moving from a launch/release point and submerging.
2) Passing through a 3x4 meter validation gate.
3) Locating a cross situated on the bottom of the pool and dropping a marker over it.
4) Locating a mid-water target (an orange buoy) and contacting it with the AUV.

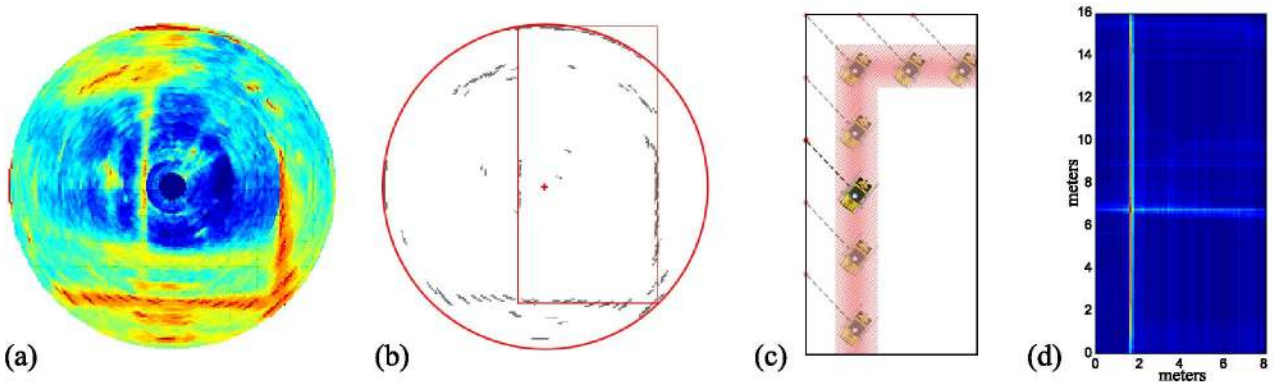

Fig. 5. Determining the robot's position. (a) Example of a challenging sonar scan with some spurious data and the boundaries of the water tank only partially observable. (b) Resulting image after selecting the highest intensity return from each single beam. The boundaries of the water tank are also represented. (c) Diagram of the zone of possible vehicle positions inferred from a single sonar return. (d) Resulting Hough voting space.

5) Surfacing at designated recovery zone marked by an acoustic device.

The mission starts facing the validation gate. The position and depth of this gate, the two targets and the recovery zone are roughly known but not sufficiently to impact the buoy, drop a marker over the cross or surface in the recovery zone with any accuracy. For these reason it is necessary to implement several behaviours to find the targets.

A. Behaviour library

To accomplish the mission, nine behaviours were implemented. They are divided into two main groups: navigation and perception behaviours.

Navigation Behaviours:

- *KeepDepth:* Keeps a constant depth.
- *KeepOrientation:* Maintains the vehicle on a determined heading.
- *MoveTo2D:* Moves the vehicle using a Line Of Sight trajectory to a 2D point inside the water tank.
- *CrossGate:* Follows a straight trajectory at a specific depth and heading until the end of the water tank.
- *LawnmowerMove:* Moves the robot following a lawnmower pattern. This behaviour is used together with the FindCross to localize the cross.
- *Search:* Rotates around its own Z axis at different depths and positions. This behaviour is used together with the FindBuoy to locate the mid-water target.

The current depth and yaw are given by the DVL sensor. The X and Y position used by the MoveTo2D are taken from the algorithm described in section V and the end of the water tank is detected using the echo sounder.

Perception Behaviours:

- *FindCross:* Uses the B&W bottom camera when the robot is near the cross after approaching with the lawnmowerMove behaviour, centers the cross and drops a marker. The following steps are used to perceive the center of the cross: First the image resolution is reduced, the bottom background of the water tank is subtracted, a binarization and an erode filter are applied and finally, the mass center is calculated.
- *FindBuoy:* Uses the RGB forward looking camera when the robot is facing the buoy to track and impact it. The following steps are used to perceive the buoy: First the image is converted into the HSV color space, then a LUT is used to segment the image. Then, a region with an area above a certain threshold conforming with a certain shape is located. In the following frames the operation is repeated with a tracking window around the object to speed up the computation.
- *FindPinger:* Makes the robot surface when the onboard hydrophone detects that the robot is over the acoustic device.

B. Petri Net

To implement the mission controller it is necessary to define several aspects:

- The Petri Net with the sequence of behaviours to be executed.
- The parameters of every place/behaviour.
- The initial and final states in the Petri Net.

The Petri net can be displayed as a graph and implemented as a matrix. Figure 7 shows the mission graph corresponding to the mission file used in the SAUC-E final. For the sake of simplicity, the parameters of every place/behaviour are not included. The initial and final states correspond to the first and last place/behavior in the graph respectively.

VII. RESULTS

In Figure 8 the trajectory made by ICTINEU^{AUV} during the final run of the competition is shown. This plot has been obtained by the localization data logged in the vehicle during the mission (see Section V). As can be seen, the result is similar to what we can expect from the mission planning (see Figure 6). First, the vehicle went through the validation gate (until it detected the far end of the water tank) with only minor perturbations in the heading. Next, the vehicle moved to a waypoint and started the searching procedure for the bottom target. At the first sight of the target,

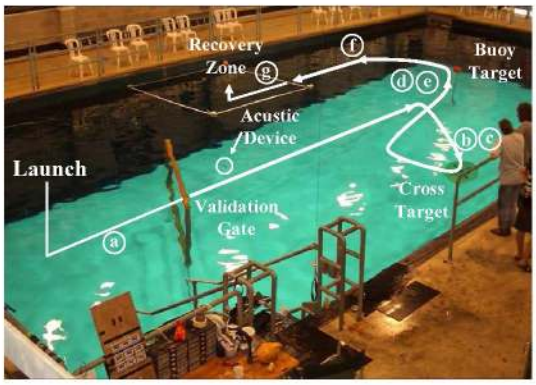

Fig. 6. SAUC-E final setup. Letters a, b, c ... correspond with the graph in figure 7.

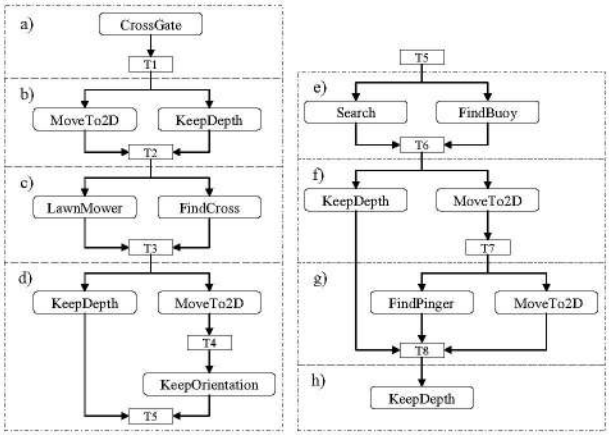

Fig. 7. a) Crosses the validation gate and goes until 5m from the end of the pool. b) Moves the robot closer to the cross at a constant depth. c) Performs a lawnmower movement until it finds the cross. Then, the robot tries to center it and drop a marker. d) Moves the robot closer to the buoy at a constant depth and orients the robot. e) Performs a searching move until it finds the buoy. When the buoy is perceived the robot tries to impact it. f) Moves the robot near the surface zone. g) Moves the robot in a straight line under the surfacing zone. When the acoustic device is perceived the robot surfaces. h) If all the timeouts have expired, the robot surfaces.

ICTINEU^{AUV} released one marker at 56 cm from the center. Unfortunately, while the vehicle was trying to make a second shot, it got stuck near a wall because of the peculiarities of the competition environment. The zone boundary between the black walls and the white bottom of the tank caused the vision algorithm to get confused. After the timeout expired, the vehicle proceeded with the mission going to the next waypoint. When ICTINEU^{AUV} found the buoy, it was too close. This made it harder to aim the target. As a result, the vehicle missed the target by a few millimeters. Finally, the vehicle moved to the recovery zone to end the mission. However, an error in the description file of the mission caused the last set of behaviours not to activate and hence the vehicle remained in the recovery zone without surfacing.

ICTINEU^{AUV} probed its capability to undertake a preprogramed mission. It did two tasks and almost completed the other two, being the only entry of the competition able to link all the tasks. This performance gave the final victory to our team.

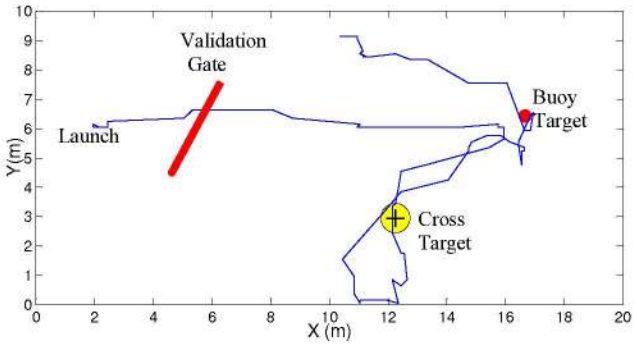

Fig. 8. Trajectory of the ICTINEU^{AUV} obtained with the navigation algorithm during the final run of the SAUC-E 2006 competition.

VIII. CONCLUSIONS

This paper has presented the current state of development of the ICTINEU^{AUV} robot designed by the VICOROB-UdG Team to face the SAUC-E challenge. The main principles of design (open frame architecture, modularity and backward compatibility) have been reported. The robot software is built as a distributed object oriented application. The control system is organized into three levels following the principles of the hybrid control architectures. It includes a low level velocity controller (vehicle level), a behavioural layer (task level) and a Petri Net based mission controller (mission level). The map-based navigation system based on the Hough transform proved to be very robust, presenting a bounded drift and being accurate enough for the mission at hand.

REFERENCES

[1] AUVSI, *http://www.auvsi.org/competitions/water.cfm*.
[2] SAUC-E, *http://www.dstl.gov.uk/news_events/competitions/sauce/*.
[3] VICOROB-UdG Team, *http://eia.udg.es/sauce*.
[4] P. Ridao, E. Batlle, D. Ribas, and M. Carreras, "Neptune: A hil simulator for multiple uuvs," in *Oceans04 MTS/IEEE*, Kobe, Japan, November 9-12 2004.
[5] AUVSI Journals, *http://www.auvsi.org/competitions/06competitors.cfm*.
[6] SAUC-E Journals, *http://www.dstl.gov.uk/news_events/competitions/sauce/06/journals.php*.
[7] T. I. Fossen, *Marine control systems*. Marine cybernetics, 2002.
[8] R. C. Arkin, *Behavior-Based Robotics*. MIT Press, 1998.
[9] N. Palomeras, M. Carreras, P. Riao, and E. Hernandez, "Mission control system for dam inspection with an auv," *IROS*, 2006.
[10] O. Bergem, "A multibeam sonar based positioning system for an AUV," in *Eighth International Symposium on Unmanned Untethered Submersible Technology (AUSI)*, September 1993.
[11] V. Rigaud and L. Marce, "Absolute location of underwater robotic vehicles by acoustic data fusion," in *Autonomous Mobile Robots: Perception, Mapping, and Navigation (Vol. 1)*, S. S. Iyengar and A. Elfes, Eds. Los Alamitos, CA: IEEE Computer Society Press, 1991, pp. 185–190.
[12] M. Carreras, P. Ridao, R. Garcia, and T. Nicosevici, "Vision-based localization of an underwater robot in a structured environment," in *IEEE International Conference on Robotics and Automation ICRA03*, Taipei, Taiwan, 2003.
[13] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.
[14] D. Ribas, J. Neira, P. Ridao, and J. Tardós, "SLAM using an imaging sonar for partially structured environments," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2006.
[15] R. Duda and P. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, 1972.