Autonomous submarine robotic system

Fabrice LE BARS, Jan SLIWKA

Laboratory for Extraction and Exploitation of Information in Uncertain Environments
Department of New Technologies
ENSIETA, 2, Rue François Verny, 29806 Brest, France
lebarsfa@ensieta.fr, sliwkaja@ensieta.fr

Miroslav ŠEBELA

Department of Air Defence Systems
University of Defence, Kounicova 65, Brno, 612 00, Czech Republic
miroslav.sebela@unob.cz

Abstract:

This paper deals with the design of an AUV (Autonomous Underwater Vehicle) for the SAUC-E (Student Autonomous Underwater Challenge – Europe) competition. The main originality of this robot is the combination between the enforcement of the KISS (Keep It Simple, Stupid) principle with the use of COTS (Commercial Off The Shelf) components and simple computer algorithms associated with robust methods using interval analysis to solve movement, localization and detection problems. This ensures the reliability, reproducibility and modularity of each part of the system (with COTS components) as well as guaranteed results (with interval analysis).

Keywords:

AUV, SAUC-E, robot, KISS, COTS, interval, RSIVIA

1 Introduction

This robot (Figure 1) was created to take part in the SAUC-E competition, which has been held for the last four years in main European countries.

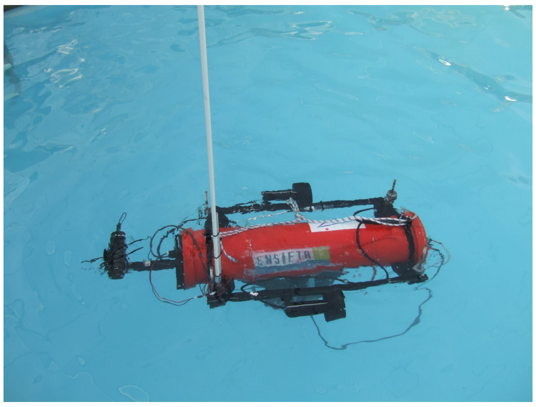

Figure 1: The submarine

The goals of this competition are to advance the state-of-the-art of AUV by challenging multi-disciplinary teams of students and engineers to perform an autonomous mission in the underwater environment and to foster ties between young engineers and the organizations involved in AUV technologies. It is designed as a mini-grand challenge for the autonomous underwater community which will create a suitable environment for interdisciplinary interactions between academic researchers.

The competition takes places in a pool where submarine robots must perform autonomous detections and localizations missions such as passing through 3 aligned gates avoiding the middle one, follow a moving orange ball target, follow a non-linear wall, dock into a box… These missions, already difficult on the dry land, are a lot more complex in water due to a reduced visibility or watertightness and communication problems.

Firstly, we will detail the physical architecture (mechanical and electronic) of our robot. Then, we will explain how we try to solve the issues of autonomy and mission planning with specific algorithms.

2 Physical description

In this part, we will describe the robot construction.

External architecture

The robot is based on an aluminum tube, which is a good choice for its resistance to pressure, its amagnetism, its resistance to corrosion and its facility to prepare a watertight environment.

The watertightness of the tube is made by 2 aluminum plaques (Figure 2). There are waterproof connectors on each plaque for the various sensors and actuators of the robot. The watertightness is provided by three stainless fastener screws. The fixation of the screws is done with a pawn centre. The extraction of plaques is done with three extraction screws. Only one tape must be removed to extract the electronic part of the robot. This allows an easier maintenance.

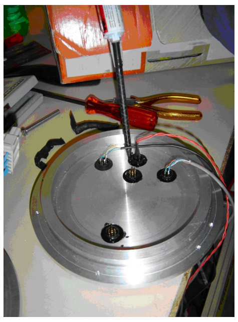

Figure 2: A plaque with its waterproof connectors (interior side)

A special structure (Figure 3) was made to carry the horizontal thrusters.

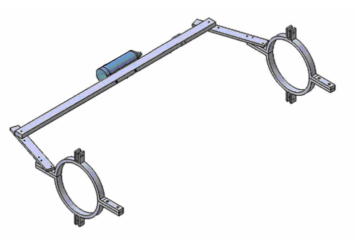

Figure 3: Structure that carries the horizontal thrusters

The roll and pitch of the submarine are not controlled because they remain stable thanks to a weighted keel that is also a support for the sonar and the vertical thruster. The keel is cut to put our vertical thruster in the center of the submarine, in order to keep symmetry (Figure 4).

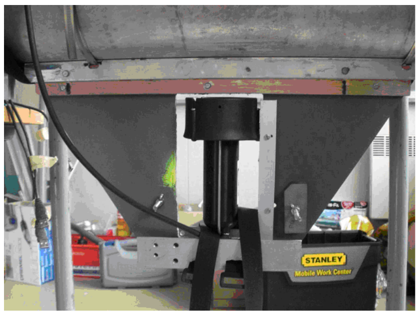

Figure 4: Vertical thruster centered in the keel

The weight of the ballast keel is currently of about 630 grams. Additionally, we have a system to adjust the overall ballast of the submarine: bored lead mass can be added on 4 rods placed in the 4 corners of the submarine so we can easily set the weight of the submarine to reach the limit zone of buoyancy. As a result we just need a very weak propelling force to make the submarine go under the surface, and when the vertical engine is shut down, it goes itself to the surface.

Internal architecture

Because dismantling or reassembling the submarine is a significant factor of loss of time and reliability, we have chosen to limit sharply the need to open one of the plaques that close the cylinder by bringing all the connections to the aluminum rear plaque of the submarine.

Rails with glue for aluminum (Figure 5) enable us to drag a Plexiglas plate of 6 mm width which is the main support base for the internal electronic devices of the submarine.

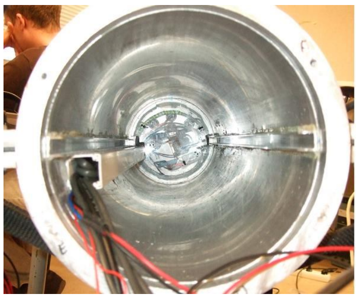

Figure 5: Rails inside the tube

Below the plate, another sliding support (Figure 6) contains the batteries. Thus, we can readily access the batteries without having to touch any of the other electronic devices. These are put above the main Plexiglas plate.

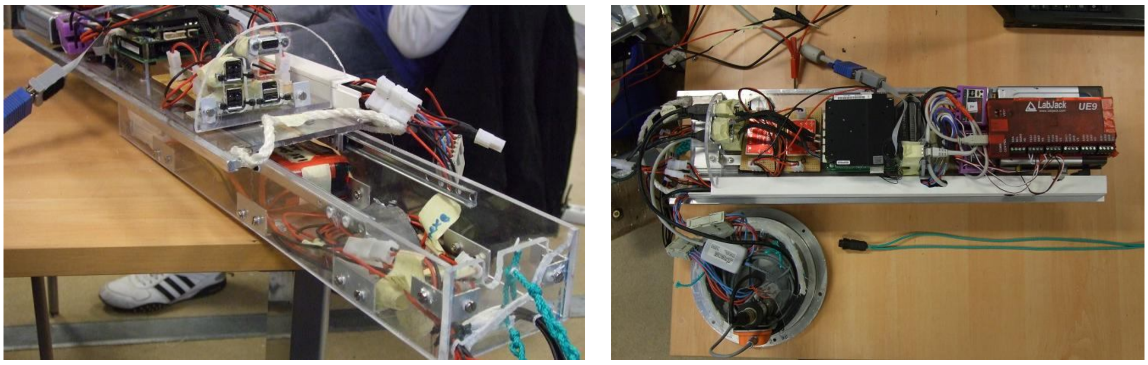

Figure 6: Sliding support and internal devices

Electronic architecture

The core of this system (Figure 7) is a PC104 embedded computer which collects data from 2 webcams, a sonar, an IMU (Inertial Measurement Unit) and a pressure sensor. We tried to avoid the use of home made circuits because our experience showed that most of the problems we had came from these circuits. Therefore, we used professional devices, which are very reliable and that we can easily change in case of a problem without loosing time. We do not need to fully know about the internal parts of the devices to be able to use them. The internal parts of the devices are not important to us because we can trust the manufacturers.

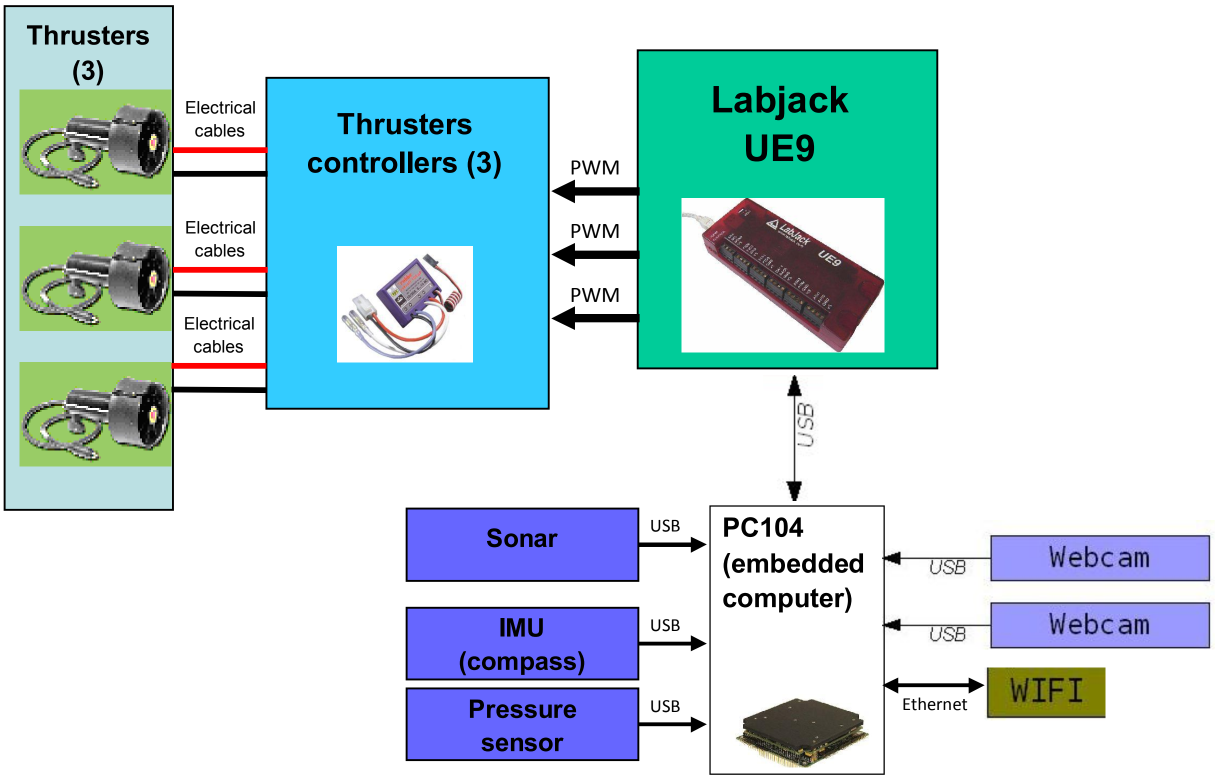

Figure 7: Electronic architecture

We use 3 thrusters SBT150 from SEABOTIX, an american manufacturer specialized in ROVs (Remote Operated Vehicles) to make the robot move:

- 1 vertical thruster to adjust the depth of the submarine
- 2 horizontal thrusters to control the speed and the direction

To control the thrusters with electronic signals, we use a servo controller Robbe Rokraft (Figure 8). The power sent to the thrusters (and therefore their speed) depends on the PWM (Pulse Width Modulation) signal.

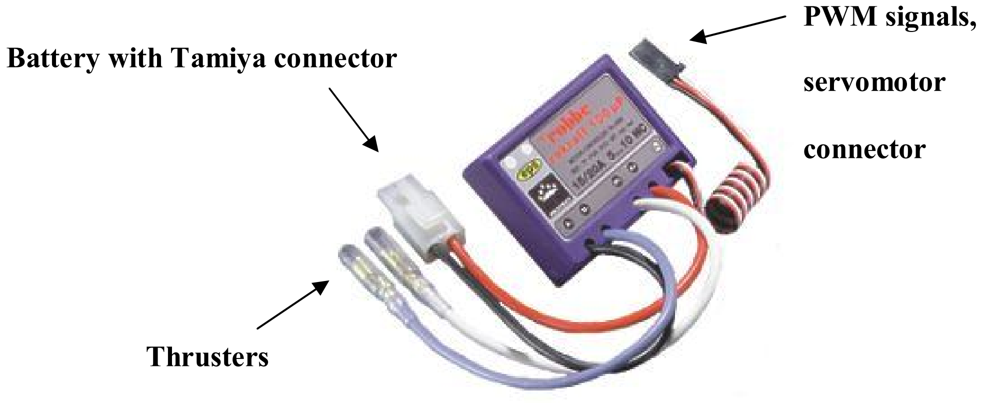
Battery with Tamiya connector
PWM signals, servomotor connector
Thrusters

Figure 8: Thruster controller

To generate these PWM signals from computer programs, we need an interface module between the computer and the servo controllers: the Labjack UE9. It is a professional USB (or Ethernet) computer device that provides several IO pins to connect to electronic devices. Alternatively, the Labjack UE9 could have been replaced by Parallax or Polulu modules.

The embedded computer is a PC104 from EUROTECH with a Pentium M 1.4 GHz CPU and 512 MB of RAM (Figure 9). The operating system and the programs are stored on a 2.5” hard drive of 40 GB. 8 USB, 1 Ethernet, 2 RS232 and 1 VGA ports provide all we need to connect external devices and communicate with the computer.

Figure 9: PC104 CPU module and power supply module for the CPU

A power supply module (compliant with the PC104 standard) that provides 3.3, 5, +12 and -12 V has been added to power it directly from 12 or 24V batteries (Figure 9).

To detect the objects in the pool we have 2 standard webcams Logitech Quickcam Pro 9000 that can get pictures with a very high resolution (up to 1600x1200). Moreover, the common defaults in webcam pictures such as distortions and light or color problems are automatically handled by its integrated filter. Our 2 webcams were made waterproof by putting them in plumbing pipes with a Plexiglas window (Figure 10).

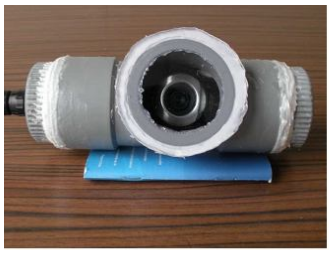

Figure 10: Webcams

To get the depth of the submarine, we use a professional pressure sensor Keller PAA33X connected to the computer with a RS485 to USB converter. The sensor is fixed on the back plaque of the submarine. To get the orientation of the robot in the pool we use an IMU Xsens MTi lent by the GESMA (Groupe d'Etudes Sous Marines de l'Atlantique). It has a built-in fusion filter that uses magnetic data and gyroscopes to get a correct orientation even in case of magnetic disturbances. It is connected to the embedded computer via a RS422 to USB converter. The main sensor to get the position of the robot in the pool is the MiniKing imaging sonar from Tritech. It is connected to the embedded computer via a RS232 to USB converter (Figure 11).

Figure 11: Pressure sensor, IMU and sonar

A wireless access point DWL G700AP (Figure 12) in combination with an external antenna of 1m enables the robot to communicate with us (via a laptop) when it is near the water surface.

Figure 12: Wireless access point

The power supply is divided into 3 parts:

- The engines are powered by a 12 V battery.
- The sonar needs a specific battery of 24 V.
- The PC104 and the wireless access point (via the 5 V provided by the power supply PC104 module) are powered by an additional 12 V battery.

All the other devices (pressure sensor, sonar, IMU, Labjack, webcams…) are powered via the USB ports of the computer.

Currently we use Windows as operating system on the embedded computer for the following reasons:

- Windows has built-in remote desktop connection software.
- Most of the devices we use have official drivers and programs for Windows whereas the manufacturers do not always provide support for Linux. As a consequence, we can use official tools to test our devices in real conditions (e.g.: test the sonar and the inertial measurement unit in a pool) without having to develop our own software to try to find all the capabilities of our devices.
- In combination with the remote desktop software, it is easy to develop directly a GUI on the robot's computer with Visual Studio, C++ Builder...

3 Autonomy and mission planning

Overview

The autonomy of the robot is based on the combination of the localization and the image processing algorithms. Indeed, the robot needs firstly to know its environment and its state in it to know were to go. Therefore, we have decided to work a lot on the localization of the robot in the pool, taking into account outliers due to sensors or objects in the pool.

First it is easy to get the depth of the robot thanks to its pressure sensor. This enables to have a depth regulation loop, which is most of the time independent from the position in the pool. The most difficult is to get the x,y position in the horizontal plane. Except for some missions, the compass (the IMU) and the sonar are used to get it.

The compass gets the orientation of the robot with respect to the North and then we get the orientation with respect to the pool (Figure 13). This enables the robot to follow a particular direction (for example, to go in a straight line through the 3 gates…). With the sonar, the distance to the first obstacle at a specific angle is got easily. But if we take into account outliers and consider that the orientation given by the compass is not always precise, the algorithm of localization becomes more complex. We use interval arithmetic [1] to deal with these problems and get guaranteed results (i.e. no solutions are lost, contrary to other methods such as those using probabilities).

Cameras are used to detect objects and to identify them (Figure 13).

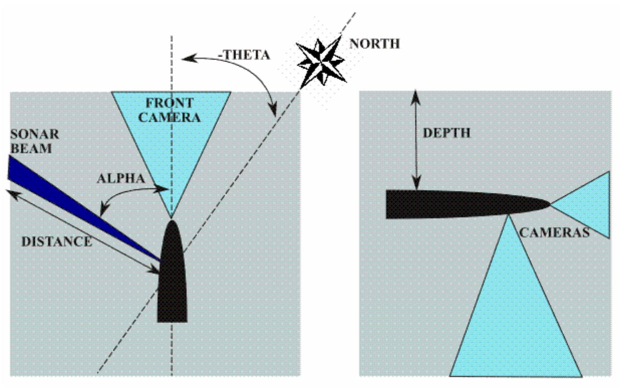

Figure 13: Localization and detection

Interval analysis

Introduction

To solve problems related to robotics as localization, the variables can either be represented by continuous sets, probabilistic distributions or a set of points (Figure 14).

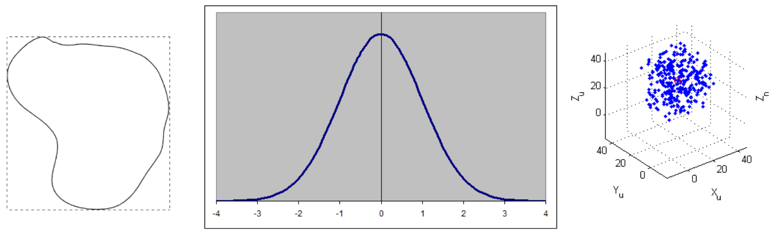

Figure 14: Different representations of variables

In our case, we chose to represent the variables by continuous sets i.e. intervals. In fact, sensor measures are often considered as bounded and the bounds are given by the manufacturer (the precision). This representation offers the possibility to create interval arithmetic, explained in more details in this part. This arithmetic is well adapted to non linear problem solving. Many methods based on this arithmetic were created. Those methods advantages are:

- Guaranteed results
- Powerful parallelizable algorithms
- Robustness to outliers

Interval arithmetic

An interval is a closed and bounded subset of R. For example, $[1,3]$, $\{1\}$, $]-\infty,6]$, R and $\emptyset$ are intervals whereas $]1,3[$ and $[1,2]\cup[3,4]$ are not. IR is the set of the intervals of R.

A box or interval vector $[x]$ of IR^n is a vector whose components are intervals:

$$[x]=[x_1^-,x_1^+]\times\ldots\times[x_n^-,x_n^+]=[x_1]\times\ldots\times[x_n]$$

The length $w([x])$ of a box $[x]$ is the length of its longest side. By convention $w(\emptyset)=-\infty$. If $w([x])=0$, $[x]$ is degenerated. In this case, $[x]$ is a singleton of IR^n and will be noted $\{x\}$.

The function $[f]: IR \to IR$ is an inclusion function of the real function $f: R \to R$ if and only if for all $[x]\in IR$:

$$f(x)\subset[f]([x])$$

Contractions and computations with intervals

To understand the basis of interval contractions, consider the following example (Figure 15):

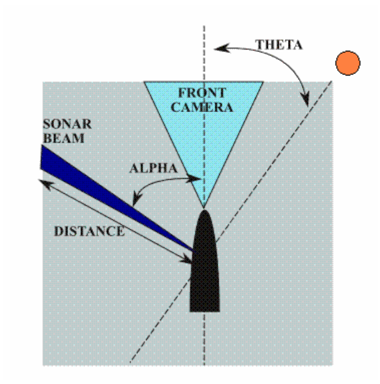

Figure 15: Simple robot with front camera and sonar

The robot has:

- a sonar with an angular accuracy of $\pm 1°$ and a distance accuracy of ± 0.5 m (in Figure 15, alpha is the angle of the current sonar ping and distance is the distance to the first obstacle encountered by the sonar ping)
- a front camera with an orange ball detection algorithm that can retrieve the angular position of such a ball when it is in the range of the camera (in Figure 15, theta) and its distance (assuming that we know the dimensions of the ball and all the characteristics of the camera) with accuracies of respectively $\pm 0.5°$ and ± 1 m.

If the sensors provide the following measurements:

$$\alpha=4.5,\ d_{sonar}=5.1,\ \theta=3.2,\ d_{camera}=5.7$$

you will note

$$[\alpha]=[3.5,5.5],\ [d_{sonar}]=[4.6,5.6],\ [\theta]=[2.7,3.7],\ [d_{camera}]=[4.7,6.7]$$

and you can conclude that

$$[\text{angular position of the ball}] = [\alpha] \cap [\theta]$$

$$[\text{distance to the ball}] = [d_{sonar}] \cap [d_{camera}]$$

i.e.

$$[\text{angular position of the ball}] = [3.5,3.7]$$

$$[\text{distance to the ball}] = [4.7,5.6].$$

And if you want to get the position of the ball in the coordinate space of the robot you can compute

$$[x] = [\text{distance to the ball}]\cos([\text{angular position of the ball}])$$

$$[y] = [\text{distance to the ball}]\sin([\text{angular position of the ball}])$$

always working with intervals in your computations (you can find inclusion functions for each elementary function, see [1] for more information).

Note that if one of the intersections is empty, you have an inconsistency. In real conditions, this often means that you made a mistake related to the accuracy of the sensors (they were less accurate than expected).

Algorithms

In order to execute the missions, we implement vision and localization algorithms (which are the most complicated part of the software) that we will briefly describe in the following parts.

Vision algorithm

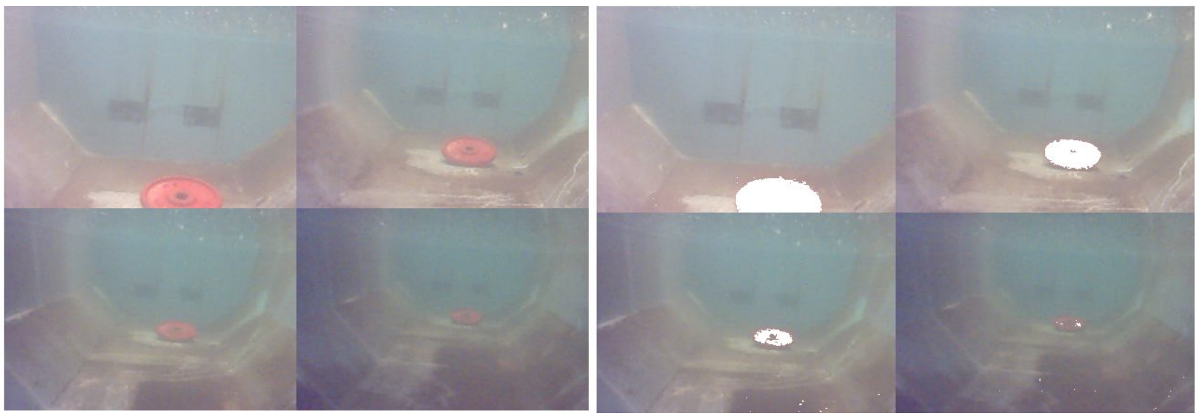

Figure 16: Underwater object detection based on color detection taking color absorption in water into consideration

In the SAUC-E competition, we have to detect several objects: orange ball, black ball on a target, black box with lights… At least 2 methods of detection are available:

- shape detection
- color detection

Our vision algorithms are based on color detection taking color absorption in water into consideration (the red color is the most absorbed color in water depending on the distance). Indeed, the objects to detect are of different colors and sometimes have a similar shape. An easy formula from a Ph.D student [2] has been proven to be efficient (Figure 16).

Localization

For the localization, we use sonar and compass data to compute the position of the robot using set inversion methods from interval analysis [3][4].

The example below shows the overall principle of that kind of localization in a rectangular pool. The pings of the sonar (distances to first obstacle measured by the sonar) are represented in Figure 17 in the robot's workspace. In order to localize itself, the robot has to know the position of the pool in that workspace. The localization algorithm does that "matching" between the pool walls and the sonar images by bloc-testing of all the possible positions of the pool and rejecting the impossible solutions.

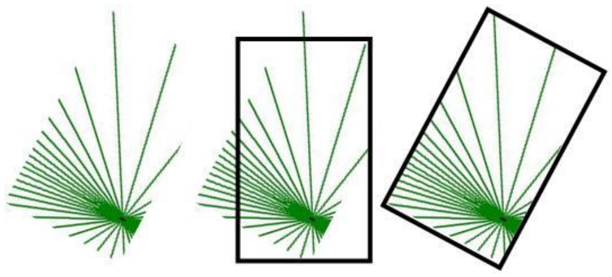
Figure 17: Matching between sonar pings and pool walls

In Figure 18 you can see a localization performed using simulated data and the RSIVIA algorithm [3][4]. In the figure on the right, the algorithm tests bloc by bloc and discards the impossible solutions leaving only the solution set (in black). This algorithm can also take into account outliers (bad distances to the first obstacle for some sonar pings, due to objects in the pool for example).

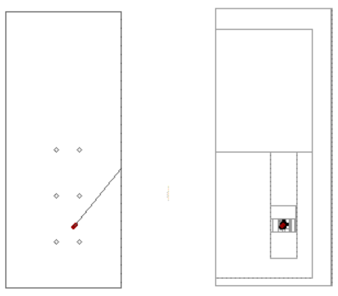
Figure 18: Simulation of the robot localization using RSIVIA algorithm

4 Conclusion

In this paper, we described an AUV built for the SAUC-E competition. This robot, made of existing commercial components rather than home-made ones, is one of the first autonomous submarines to use interval arithmetic for its localization and objects detections. We detailed the design of the robot, showing that it was a simple assembly of devices. Then, we presented the software part, introducing the use and the benefits of interval analysis for such a system.

5 References

[1] L. Jaulin, M. Kieffer, O. Didrit and E. Walter, *Applied Interval Analysis with Examples in Parameter and State Estimation*, Robust Control and Robotics, Springer-Verlag, 2001, ISBN: 1-85233-219-0, http://www.ensieta.fr/jaulin/publications.html

[2] S. Bazeille, *Vision sous-marine monoculaire pour la reconnaissance d'objets*, Ph.D. Thesis, Université de Bretagne Occidentale, 2008, and other related work, http://www.ensta.fr/~bazeille/fr/publications.html

[3] L. Jaulin, *Robust set membership state estimation*, Automatica, Volume 45, Issue 1, January 2009, Pages 202-206, http://www.ensieta.fr/jaulin/publications.html

[4] J. Sliwka, F. Le Bars, O. Reynet, L. Jaulin, *Reconnaissance de forme pour la localisation de robots*, submitted to RFIA 2010, 2009, France, http://www.ensieta.fr/sliwka/