An Expert Navigator for an Autonomous Underwater Vehicle

Gerardo Acosta [1] — Hugo Curti [2] — Oscar Calvo [3] — Javier Mochnacs [3]

Investigador CONICET / INTELYMEC – Fac. de Ingeniería – Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina.
ggacosta@fio.unicen.edu.ar
INCA – Fac. de Cs. Exactas – Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina.
hcurti@exa.unicen.edu.ar
[3] Departamento de Físicas – Facultad de Cs. Exactas – Universitat de les Illes Balears, Spain.
{oscar|mochnacs}@galiota.uib.es

Abstract. This article describes considerations and design hypothesis in the implementation of the strategy to generate trajectories for an autonomous vehicle. The application problem consists of an autonomous underwater vehicle (AUV) tracking a pipeline in the seabed. To solve this problem, a real time expert system (named EN4AUV) was proposed to be included in the on-board AUV central processing unit. EN4AUV takes trajectory control decisions based on a number of variables, arranged around the concept of scenarios. For different scenarios, the expert system is able to suggest trajectories. Although this work is still undergoing, in this paper some incipient results over computer simulation are shown. The article is concluded with some conclusions and a snapshot of future work.

1 Brief introduction to underwater pipeline inspection

As a starting definition, in the context of this work, the pipeline tracking in general with an autonomous underwater vehicle (AUV) will comprise the pipe detection itself, as well as the feeding of this information to the vehicle control system. This will guide the AUV along the pipe at a predefined offset, without human intervention, in a way similar as drawn in figure 1. The main interest on solving this problem is to acquire data regarding the state of these pipelines, lying over the seabed, for maintenance purposes. If the AUV stops acquiring reliable data, that is, its sensors acquires data below a predefined certainty, it is necessary to correct the trajectory or asses a new running hypothesis (i.e., the pipe is buried or there is an obstacle in the pipe's trajectory).

Fig. 1. The AUV is tracking a pipeline.

Nowadays, remote operated vehicles (ROV), driven from platforms or ships carry out most of the pipeline inspections, with many drawbacks. Among them there is a strongly perturbed control, for the umbilical-vessel interaction, and a high cost for renting or support a ship with its crew in the sea all along the pipeline. Both features increase as the depth of exploration does. An AUV suppresses these drawbacks giving also the possibility of higher quality inspection data due to a smoother navigation. The only limit for improvement is the pressure the AUV can stand for, but this appears at greater depth than usual tracking [1].

In the present application, and for simulation tests, there were established two ways of acquisition, related to two types of different sensors:

- Acoustic track, based on a multibeam echosounder (MBE)
- Magnetic track, based on a magnetic sensors system (MAG)

Both kinds of sensors and a file containing historical data about the original pipeline location act as input to a sensor fusion module. This module yields an idea of the relative AUV/pipeline position. These historical data are of much help for they allow determining a zone in which the pipeline would be found with higher probability. This zone is named corridor [2].

2 The Expert System Approach

A first approach to manage this problem was developed as a state machine (SM) implemented in Linux with C++ language. Simulation results were promising, as reported in [3]. However, in a real underwater mission, complex situations might appear in which the SM might probably mislead. One of the final objectives of the whole project is to provide an AUV with the enough "intelligence" to cope with real situations in the marine world. Although these wet tests are still pending, an alternative and more powerful approach was developed, and is presented in next paragraphs. In order to face unexpected situations, like the sudden appearance of a fishing net, or a detour due to obstacle detection, and others, a piece of the experience of expert ROV users was elicited and codified in the form of a real time expert system.

This expert system, named ***EN4AUV*** (Expert Navigator for Autonomous Underwater Vehicles), was constructed using CLIPS, a C based shell [4]. The classical steps for its construction were followed: 1st) problem identification, 2nd) conceptualisation, 3rd) formalisation, 4th) implementation, and 5th) evaluation. As it is known these phases are progressive and there is a dynamic feedback among them during system development.

The problem to be solved is the trajectory generation for the vehicle, based on a sensors' module giving position points (in co-ordinates (***x, y, z***)) and a confidence in the measurement of such co-ordinates, called certainty error. The sensors' module implements a fusion of among the multibeam echosounder (MBE), the magnetic sensor (MAG) and the historical data (HD). Thus, EN4AUV shall propose a desired trajectory, formed by ***n*** number of points to be reached and surpassed by the submarine. At this stage, the desired trajectory must be in global co-ordinates of latitude, longitude and altitude.

There are other sensors' inputs for the expert system that determines its final behaviour. EN4AUV is clearly a reactive expert system, behaving in such way for such situation, taking into account for instance the pipeline status, the type of survey, the different mission settings, and others. Then to classify different situations, the concept of scenario is used.

2.1 Scenarios

A scenario is defined as a set of input monitoring variables that describe a situation. The AUV shall react in different ways to one scenario than to another. This way of representing information is also called data abstraction. The following 10 scenarios were identified for the present application.

1st Scenario

The AUV is tracking an exposed pipeline and is navigating on top, at a fixed offset smaller or equal to 2 meters. Both the MBE and the MAG detect it, and the sensors' module reports that the AUV is tracking the pipe normally.

2nd Scenario

The AUV is tracking a buried pipeline on top, at a fixed offset smaller or equal to 2 meters. The MBE can not detect it, but the MAG does.

3rd Scenario

The AUV is tracking an intermittent (intermittently exposed and buried) pipeline at a fixed offset. This is a sequence of alternative appearance of scenarios number one and two.

4th Scenario

The AUV is tracking an exposed pipeline at a fixed offset greater than 2 meters. The pipe is tracked only based on MBE readings, which may be detecting the pipe itself or the trench. This situation is close similar to the first pipeline approach when starting the survey.

5th Scenario

The AUV is tracking a pipeline in the presence of one or more pipes (like infield pipelines) or other magnetic objects in the area, with indication of pipeline detected, from the sensor's module.

6th Scenario

The AUV is tracking a pipeline in the presence of one or more pipes (like infield pipelines) or other magnetic objects in the area, but with indication of an incident and pipeline detected, from the sensor's module.

7th Scenario

The AUV is tracking a pipeline avoiding an obstacle. In this scenario the error certainty may increase beyond its threshold, but EN4AUV "knows" where the pipe is and ignores the pipe_lost flag. The path planner module (PPM) outputs a flag indicating this condition and the EN4AUV may query the HD to confirm the existence of an exclusion zone.

8th Scenario

The AUV is searching a buried pipeline.

9th Scenario

The AUV is searching a pipeline in the presence of one or more pipes/cables (like infield pipelines) or other magnetic objects in the area, with indication of pipeline detected, from the sensor's module. Previously the EN4AUV was going through scenarios 5 or 6.

10th Scenario

The AUV is going to a fixed position. Although the EN4AUV is reading messages from sensors' module, it ignores them since it is in a special situation, like skipping from present searching position, or changing the pipe to track, or returning to a known position when failing to acquire the pipe, and others.

As mentioned, each scenario will have different actions as outputs. There were identified three main tasks to be carried out to achieve the objective of trajectory generation once the scenario is assessed: TRACK, SKIP and SEARCH, each one with corresponding subtasks, as will be introduced next. When as the pipeline is located and identified, it should be closely followed to acquire useful data for inspection. This is carried out within the **Track** task. It is assumed that while this task is executing, the certainty error remains below the predefined threshold. Several subtasks were defined: Track normally, Track with MBE, Track with MAG, Track multiple pipelines, and Track avoiding obstacle. The **Search** task is the procedure in charge of pipe localisation when it is considered to be lost. A basic (strong) assumption is that if the pipe is visible for the MBE, it will never get lost. The pipe will be considered not detected when the error certainty on positions coming from sensors' module supersedes a certain threshold. Then EN4AUV will decide if the pipe is lost based on these measures and previous scenarios. The two subtasks for Search are searching a single pipeline and searching multiple pipelines. There are also situations in which the AUV shall reach a certain position (latitude, longitude, altitude), not within the search or track tasks (i.e., when diving, skipping or changing the survey from one pipe to another). This is implemented in the **Skip** task.

2.2 Rule base and objects

CLIPS allows the knowledge representation in the form of rules and frames (COOL or Clips Object Oriented Language). These formalisms are used in the knowledge base (KB) to represent the involved knowledge. Thus, there is a set of rules devoted to pipeline's layout determination, if it is or not detected, also if it is buried or freespan. Once this is established, rules determine the AUV follow status as regards as the pipeline. These follow status may be avoiding an object, pipeline found, pipeline lost, pipeline intermittent. Then rules assess which scenario is present among the set of ten scenarios presented in the former paragraph, and then select the corresponding action. These actions are modularly implemented as C++ routines. In figure 2, a rule for determining pipeline's layout in the CLIPS syntax is presented and in figure 3, the class definition for the concept of "*working scenario"* is also shown.

```
(defrule R03.1

 (INPUT_TRAJ)

 ?ps <- (object (is-a SURVEY))

 ?ws <- (object (is-a WORKING_SCENARIO))

 (test (< (send ?ps get-Certainty_Threshold) (send ?ws get-
Error_Budget)))

 (test (= 1 (send [SFM_traj] get-Detected)))

=>

 (send [OBJ_STUDY] put-Present_Layout_Status DETECTED)

       (printout t "CLIPSMACHINE: R3.1 Present Layout Status
is " (send [OBJ_STUDY] get-Present_Layout_Status) crlf)

    (printout t "CLIPSMACHINE: Input Traj Detected Flag is "
(send [SFM_traj] get-Detected) crlf))
```

Fig. 2. A knowledge-base rule from EN4AUV.

```
(defclass WORKING_SCENARIO (is-a SYM_VAR)

          (role concrete)

          (pattern-match reactive)

          (multislot Movie (create-accessor read-write))

          (slot Navigation_type (type SYMBOL) (create-accessor
read-write))

          (slot Incident_point (create-accessor read-write))

          (slot Error_Budget (type FLOAT) (create-accessor
read-write))

          (slot Avoiding (type INTEGER) (create-accessor read-
write))

          (slot Tracking_status (type INTEGER) (create-
accessor read-write))

          (slot Follow_status (type SYMBOL) (create-accessor
read-write))
```

Fig. 3. A framework representing a Working Scenario.

In this early stage, the KB is conformed by 25 rules. They are forward chained as usual in data driven real time expert systems. The inference rule used is based on the Rete's algorithm. The objects used for formalisation were: Symbolic_Variable, Waypoint, Type of Survey, Survey, Trajectory, Input Trajectory (is a Trajectory), Output Trajectory (is a Trajectory), Object of Study (is a symbolic variable set to pipeline in this first approach), Working Scenario (is a symbolic variable).

2.3 The runtime environment of EN4AUV

The KB system runs as an embedded application. Although it is a core piece to drive the AUV, it needs from the assistance of other modules, as it may be seen in figure 4, where the full system architecture is presented. In this figure, the KB system module is drawn in red and it is called Autotracker.

Over a GNU Linux running in personal computer type CPU, some C++ modules were constructed for communication purposes. This is because every module in the system has an input/output data flow based on messages put on a UDP channel and broadcasted for the remaining ones, using TCP/IP protocol.

As it may be seen, EN4AUV receives position estimation from the sensors' module, can access to historical data and can receive information from a (low level in the control scheme) path planner. Signals from sensors are converted into an absolute depth, latitude and longitude within the Sensor Fusion Module (***x, y, z***). This triplet is transmitted to the Autotracker module within a message. Once received, the content of this message is stored in a "Waypoint" object. According to the present scenario assessed by EN4AUV, and based on the waypoint received, it provides a desired trajectory formed by four waypoints. Then, the path planner combines what is desired (the trajectory proposed by the Autotracker module) with what is possible (contour conditions from an obstacle detection system).

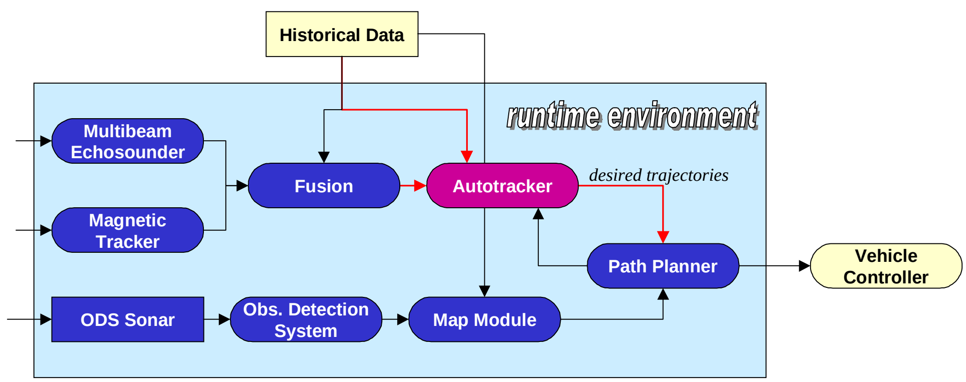

Fig. 4. Runtime environment for EN4AUV.

3 Simulation results

The final version is not ready now, because the last evaluation step will be done over an experimental AUV prototype, and this is still pending. However, some preliminary results were obtained in simulation using the environment reported in [3], and they showed a correct development line. One of these cases is presented in this section.

The AUV is left in a starting position and it goes to the initial point defined in the mission settings (scenario 10). This is a basic working assumption: the AUV is always left by humans in a point near the mission start (distance of meters) and the first task it exhibits initially is a search to precisely determine the pipeline allocation. Then, in this point the EN4AUV starts a Search task (scenario 8), in order to estimate the pipeline direction. As the pipeline is successfully detected, the AUV navigates under the Track task (scenario 4). A depiction of the submarine's followed trajectory is given in figure 5 as a dot-line. Note that when dots are red the pipeline is freespan. As soon as the pipe is visible over the sea-bed (black dots) the scenario becomes number 1. When the pipeline is buried (green dots), the system assess scenario 2.

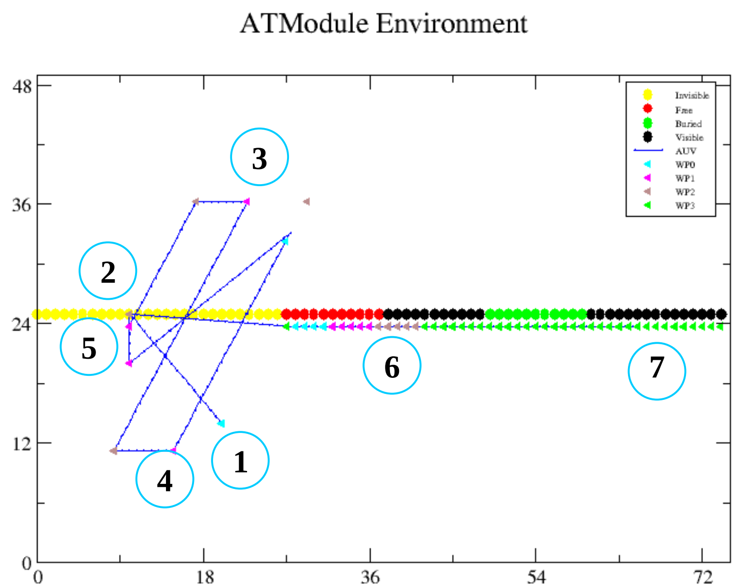

Fig. 5. AUV followed trajectory for a case study consisting of a straight pipeline. Dots in different colours represent different pipeline layout states. Little triangles in different colours represent the waypoints generated by EN4AUV.

The starting point for the mission is 1. The AUV navigates to point 2 where the pipeline is supposed to be, but as it is buried, EN4AUV starts a search with a predefined pattern (points 2, 3, and 4). With a pair of detection by the MAG it assess the right direction, so the AUV returns to the starting point (5) and tracks in the determined direction (points 6 and 7). The navigation depth is supposed to be fixed for these simulation trials.

4 Conclusions and future work

Although the test presented in this article is quite simple, the whole developing and validating environment is ready to cope with more difficult cases as the knowledge-base of EN4AUV enlarges and incorporates more case experiences. This is one of the most outstanding features of this approach, comparing it to the state machine one. With the last method a new case is added to the software needing a redesign of many modules and a new compilation of every one. The addition of new cases in the KB approach instead implies the addition of new rules. Then the attention must be focused on the KB consistency, as with any expert system. Also, this system design allows developing modular KB (i.e., one KB for pipeline status determination, one for scenario determination, one for action calling), with a great debugging facility during implementation. Software's reusability of can be enhanced by this approach. Effectively, the same system may be used to submarine cable tracking, to coastal studies, and other applications simply interchanging different KBs. This is one of the other aims of this project.

As it may be seen, the work presented here is a first step in the development of the knowledge-based system. From now on, different use cases, defined by potential users of this technology, will be faced in simulation. Finally, when a testing AUV were available, wet trials in the sea will be carried out. Hopefully the simulated use cases shall be a good starting point to validation in the real underwater world. Surely, the most difficult but fascinating phase to enrich the KB with new knowledge will start then.

5 Acknowledgements

Part of this work was developed under the European project AUTOTRACKER (GDR1-2000-25150). This article was written also thanks to the support of Universitat de les Illes Balears, while author Acosta was working there as guest Professor.

6 References

1. Bjerrum, A. – Slater, T. (2001): Autonomous tracking of submarine pipelines and cables, Hydrographic Society HYDRO 2001.
2. Allen, S. (2001): Pipeline Inspection, Hydrographic Society HYDROFEST 2000.
3. Curti, H. – Acosta, G. – Calvo, O. – Maximiliano Suárez (2003): Estrategia de búsqueda y seguimiento de objetos sobre el lecho marino con un vehículo submarino autónomo", V Workshop de Investigadores en Cs. Computación WICC '03, Tandil, Argentina, Julio 2003.
4. Giarratano, J.: "*CLIPS User's Guide*", NASA LB Johnson Space Center – Software Technology Branch, 2002.