The traversability principle suggests banning those directions where an obstacle has been detected (figure 2(a)) and, in case the GoTo behaviour response results banned, generates two alternative obstacle-free directions, generically labelled as left and right (figure 2(b)). On the other hand, which direction, left or right, is taken by the robot is decided according to the tenacity principle, in the sense that it tenaciously suggests taking the same direction that was selected the last time. These two principles, although simple in concept, have proven fully effective to solve trapping situations. As a consequence of the application of the T2 principles, a global behaviour emerges by which the robot circumnavigates the contour of an obstacle once it is aware of its existence.

3. Three T2-based Algorithms

The application of the T2 principles require making two decisions during the navigation: (1) in which direction the contour of an obstacle is followed (i.e. which direction is taken the first time the tenacity principle is applied), and (2) at which moment the robot decides to leave the contour of an obstacle to continue with the navigation towards the goal point.

Random T2, Connectivity T2 and Bug-based T2 are three strategies deriving from T2 which make the two aforementioned decisions in different ways, giving rise to different performances and capabilities. They are briefly described in the following:

• In Random T2, the robot chooses randomly the direction to follow the contour, while the contour is left as soon as the direction towards the goal point becomes allowed.

• Connectivity T2 decides the direction to follow the contour according to a minimum turn criterion and remembers which direction is taken, in order to explore the other direction in case the goal point is not reached. The contour of the obstacle is also left as soon as the direction towards the goal point becomes allowed.

• Bug-based T2 also chooses the minimum turn direction to follow the contour. As for the leaving to occur, either: (1) the navigation filter must indicate there is a free-obstacle path towards the goal point and this must be the first time the robot leaves the obstacle at approximately that same position, or (2) the robot trajectory must cut the line joining the start and goal points (M-line) and the distance from the robot to the goal has to be shorter than the one associated with the last time this condition was satisfied. Each time (1) is satisfied the M-line is redefined using the current robot position as the start position.

4. Experimental Results

In order to show the effectiveness of the T2 strategies, figure 3 shows simulation results for a maze-like environment, while table I provides a comparison of path lengths with a well-known motion planning algorithm called Bug2 [3] for a set of up to seven different environments.

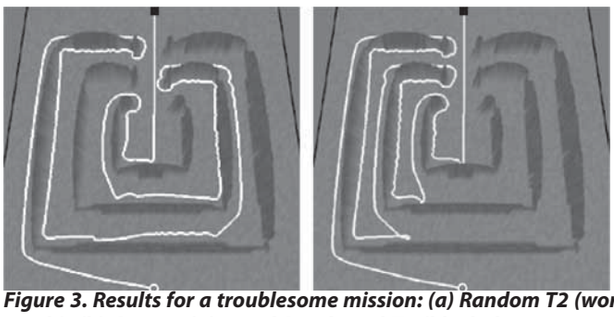

Figure 3. Results for a troublesome mission: (a) Random T2 (worst result), (b) Connectivity and Bug-based T2 (black dot = start, red dot = goal).

Mission	RT2	CT2/BT2	Bug2
1	313.62	281.54	388.00
2	420.65	420.65	426.00
3	623.00	405.17	578.00
4	286.83	368.55	421.00
5	883.47	883.47	934.00
6	1586.71	1484.91	1672.00
7	663.82	246.85	797.00
Total (m)	4778.10	4091.19	5216.00

Table I. Comparison of the path lengths of Random T2 (RT2), Connectivity T2 (CT2), Bug-based T2 (BT2) and Bug2.

5. Conclusions

A novel family of geometric algorithms of motion planning based on potential fields and the traversability and tenacity principles have been put forward. The T2 variant paths have resulted, on average, significantly shorter than the ones of other algorithms for a representative set of missions. An extension of T2 strategies to three dimensions is under development at the moment to take advantage of the 6 DOF of AUVs.

6. References

[1] O. Khatib. "Real-time obstacle avoidance for manipulators and mobile robots," International Journal of Robotics Research, vol. 5, no. 1, pp. 90–98, 1986.

[2] Y. Koren and J. Borenstein. "Potential field methods and their inherent limitations for mobile robot navigation," in Proc. 1991 Robotics and Automation Conf., pp. 1398–1404.

[3] V. Lumelsky and A. Stepanov. "Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," Algorithmica, vol. 2, pp. 403–430, 1987.

DEFINING A MISSION CONTROL LANGUAGE

Narcis Palomeras, Pere Ridao and Marc Carreras

Univ. de Girona, Campus de Montilivi, Edifici P4 – Girona
npalomer@eia.udg.es

1. Introduction:

A mission controller is the part of the control architecture that is in charge of defining a thread of tasks to be carried out in order to fulfil a mission. Each task can be executed by means of some vehicle primitives often referred as basic robot commands or behaviours. The mission controller must define how the mission is divided into a set of tasks and how primitives are combined to fulfil each task. The development of a Mission Control System (MCS) for an Autonomous Underwater Vehicle (AUV) lies at the intersection of a Discrete Event System (DES), in charge of enabling/disabling basic actions when some events are produced, and the Dynamic Control Continuous System (DCS), used for every action to achieve a specific goal [1]. The main objective of this project is defining a language able to describe AUV missions in a simple but versatile way.

The Petri net formalism [2] has been chosen to describe the missions that the MCS is going to execute, but instead of using graphic tools to describe these Petri nets, our approach uses a Mission Control Language (MCL), which compiles into a Petri net. The adoption of this formalism helps us evaluating the net properties at compilation time to detect, for instance, if possible inconsistent states can be reached. Hence corrective actions can be adopted before actual execution.

This work is structured as follows, section 2 describes our proposal

for a MCS. Section 3 presents how the MCL works. Finally, section 4 presents the conclusions.

2. A Generic Mission Control System:

Our intention has been to design the MCL as generic as possible and to allow for an easily tailoring to different control architectures. To achieve this goal our MCS presents a clear interface with any particular vehicle control architecture based on Actions and Events. Between the MCS and the vehicle control architecture there is an Architecture Abstraction Layer (AAL) that adapts these Actions and Events for every particular architecture and provides, moreover, provides an expressions evaluator (with accessibility to all the vehicle internal variables) as well as a set of timers (
Figure 1).

Actions in the MCS, provoke the execution of vehicle primitives within the vehicle architecture or the evaluation of an expression or the initialization of a timer within the AAL. Events, are used to notify any thing that happens inside the vehicle architecture or the AAL (vehicle primitive achievements, timeouts, vehicle exception, evaluation results, …)

The AAL depends on the control architecture being used allowing the MCS to remain architecture-independent. With the AAL, it is possible use this MCS approach in different vehicles with different control architectures.

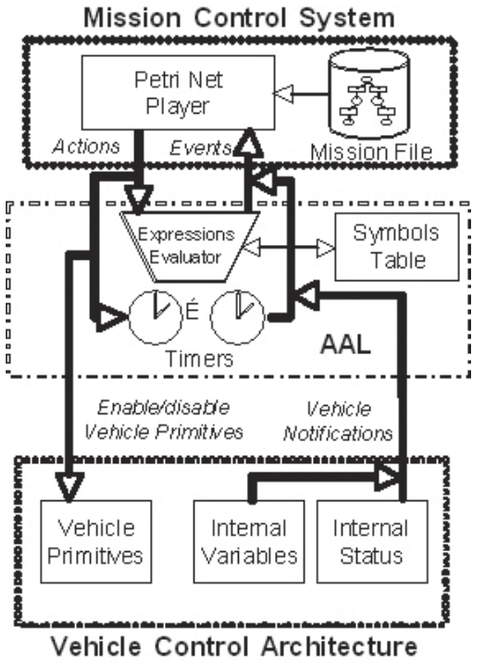

Figure 1: MCS connected to a particular Vehicle Architecture through the AAL.

3. Mission Control Language:

From the point of view of the MCS, a mission is just a Petri net that evolves when some Events are received and being able to execute a set of desired Actions. To describe a mission, MCL uses two different sections: the architecture-link and the mission itself. In the architecture-link all the Actions and Events as well as Vehicle Variables or Resources that are available from the MCS have to be defined together with their correspondence (Vehicle Primitives, Notifications, Variables…). Tasks have to be defined also in this section. A task is defined as a flow of states, where each state is characterised by a set of enabled Vehicle Primitives and timers pursuing a set of related goals. When these goals are achieved, the vehicle architecture notifies the AAL that sends different Events. As a result, MCS changes its state and executes a set of Actions. In our approach, this control execution is modelled with an ordinary Petri net extended as follows: (1) Every transition has a set of events and a set of actions associated; (2) a transition t is enabled if each input place p of t is marked with 1 token (we use ordinary Petri nets) but can fire only if its associated events are received. Finally, (3) when a transition t is fired, in addition

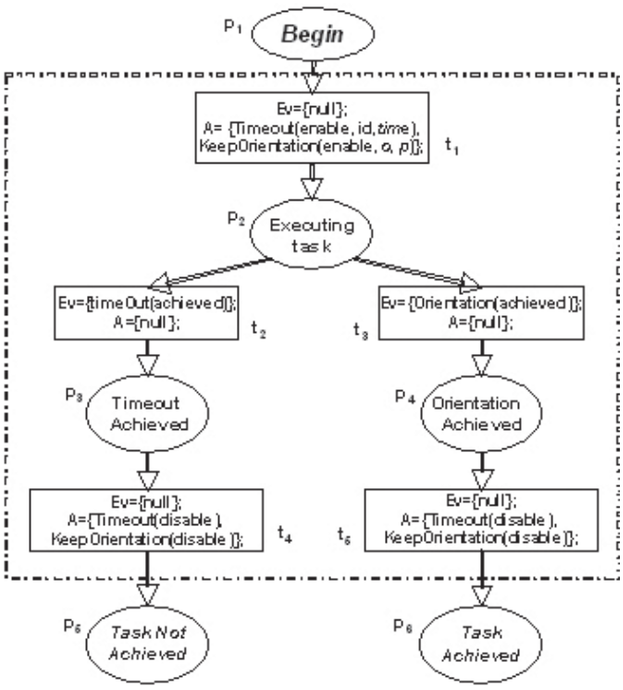

Figure 2: Example of an AutoHeading Task.

to removing 1 token from each input place p of t, and adding 1 token to each output place p of t, the set of associated actions is executed. With these extensions it is possible to disambiguate the conflicts in the Petri net by firing only the transitions whose events have been received.

In order to illustrate how a task is implemented, let us consider the Autoheading task shown in Figure 2. This is a very simple task involving only one Vehicle Primitive (KeepOrientation) and a timer. The corresponding Petri net has a transition to enable the timer and the KeepOrientation primitive (t1) and two transitions more (t2 and t3) waiting for the events timeout expiration or orientation achievement. Places p3 and p4 are pure indicatives of the state in which the Petri net is. Transitions t4 and t5 are used to deactivate the timer and the KeepOrientation primitive.

To link different tasks between them, control structures are introduced. Control structures are Petri nets that define the tasks control

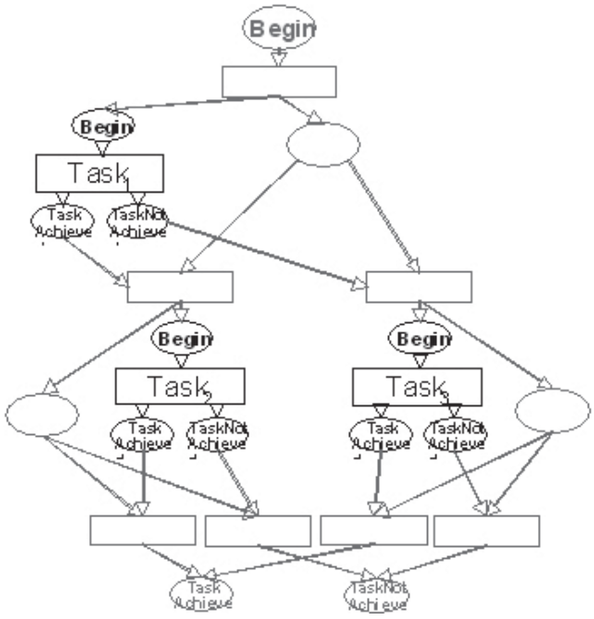

Figure 3: Example of a control structure: If(Task$_1$) then Task$_2$ else Task$_3$.

flow. Common control structures include parallel and sequential execution, repetitive structures (while, for, …), conditional structures (if-then-else, try-catch, …), etc.
Figure 3 shows a control structure If-then-else. Both tasks and control structures are embedded between the Begin, Task achieved and Task not achieved places. For this reason, the control structures can sequence not only tasks but even other control structures.
Once the architecture-link section is defined, programing a mission is as easy as writing a piece of code using the available control structures and the previous defined tasks (Algorithm 1).

```
Mission{
 yaw = 0;
 while(yaw < 2*PI){
   AutoHeading(yaw, 60);
   parallel{ConstantVelocity(0.8, 60);}
       or{Monitor(distance > 2);}
   yaw = yaw + PI/2;
   }
}
```

Algorithm 1: Example of a MCL mission.

4. Conclusions:
This is an ongoing project to design and implement a flexible MCL easy to be tailored to different AUV control architectures. A generic MCS based on Petri nets have been presented as well as the MCL used to define the missions. The main goal of this project consist on providing the end user with a simple but powerful language for describing the AUV missions.

5.References:
[1] Marco, D.B., A.J. Healey and R.B. Mcghee (1996). Autonomous underwater vehicles: Hybrid control of mission and motion. Autonomous Robots 3, 169–186.
[2] Murata, T. (1989). Petri nets: Properties, analysis and applications. Proceedings of the IEEE.

ELIMINATION UNITS FOR MARINE OIL POLLUTION (EU-MOP)

Jesús Carbajosa Waldo Rethemías

(CETEMAR Centro de Estudios Técnico Marítimos)

Introduction
This paper introduces a research project called EU-MOP, founded by EU Commission under the 6th Frame Work Programme (DG-RTD), that involves the design and evaluation of an intelligent robot system to respond to oil spills and new spill management tools.
The R&D team consist of 13 multidisciplinary European partners, everyone of which have an outstanding knowledge in their relevant areas.

Background
Oil pollution either from marine accidents or routine ship operations is one of the major problems that threaten the marine environment. Efforts in protecting the environment after an oil spill could cost billions of euros in cleanup and subsequent damage costs, often producing questionable results. The key factor for efficient cleanup operations is to develop an adequate structure focusing on the confrontation of oil when is into the sea and diminish the impact on nearby coasts.

Objectives
In fact there is a direct need for a renovation of anti-pollution methodologies and equipment. Such a goal must be incorporated at all hierarchical levels, taking the necessary legislative and surveillance measures, therefore in-situ techniques that allow for the control and elimination of spills become imperative.
1.Innovative concepts for oil spill management.
2.Novel devices for oil spill confrontation.
3.An integrated framework for oil spill management.

Expected Results
Analysis and assessment tools that complete an integrated framework for oil spill management system, including communications, logistical support and response tactics.
Design and proof of concept of autonomous elimination units for marine oil pollution
(EU-MOPs).
An advanced structure for the dissemination of oil pollution response policies.

ABOUT EXPERIENCES WITH DEEP SEA INTERVENTION DURING ONE DECADE

H. Gerber (1), G. Clauss (2)

Technische Fachhochschule Berlin – University of Applied Sciences
FB VIII Luxemburger Str. 10, D – 13353 Berlin
Technische Universität Berlin
Fak. V – ILS – Sekr. SG 17, Salzufer 17-19, D – 10587 Berlin

Introdcution
This contribution will deal with the long standing experience of a deep sea intervention system MODUS that has been developed by the authors groups for the deployment of heavy deep sea stations. This stations mainly operated by the partner INGV in Rome/Italy have the purpose to operate for a long term (one year) autonomously or to be linked to shore nodes.

Contents
It will be presented the complex process for the development of this technology with CAD design, CFD simulation, numerical simulation for the dynamic behaviour and the lab tests. This will be followed by the experiences with MODUS in the field. MODUS has been used within several European projects such as GEOSTAR1, GEOSTAR2, ORION, BIODEEP. Moreover in the Italian projects SN#1, APLABES and MABEL. During this projects we have had more than 70 dives down to 4000 m in different environments, like the Mediterrenean Sea and the Antarctic Sea. In the near future it will operate in the Atlantic Sea within NEAREST another European project.