Internet mission control of the ROMEO Unmanned Underwater Vehicle using the CORAL Mission Controller

Ga. Bruzzone, R. Bono, M. Caccia, G. Veruggio
Consiglio Nazionale delle Ricerche
Istituto Automazione Navale
Via De Marini, 6
16149 Genova, Italy
E-mail: {gabry, ric, max, gian) @ian.ge.cnr.it
C. Ferreira, C. Silvestre, P. Oliveira, A. Pascoal
Instituto Superior Técnico
Institute for Systems and Robotics
Av. Rovisco Pais
1096 Lisboa Codex, Portugal
E-mail: {cpdf, cjs, pjcro, antonio} @isr.ist.utl.pt

Abstract

This paper addresses the problem of mission control of Unmanned Underwater Vehicles (UUVs) through the Internet network. Its main focus is on the integration of a Petri Net based mission control system developed by the Instituto Superior Técnico using the CORAL environment, with the control system of ROMEO, a prototype ROV developed by the Robotics Department of CNR-IAN. The system has been evaluated controlling ROMEO's missions directly from Lisbon in the underwater virtual world (UVW) in the IAN lab and in a pool in Genoa. In particular, the reliability of the Internet connection has been verified and the constraints introduced by communication delays have been examined. This research and development effort aims at contributing to the development of reliable mission control systems for the operation of robotic ocean vehicles at distance, over communication channels that may experience considerable delays. This is a subject of great relevance, in view of the widespread interest in the development of systems to allow a scientific end-user to program, execute, and follow the state of progress of robotic vehicle missions at sea from the comfort of his/her laboratory.

I. Introduction

In the Nineties the requirements of unmanned underwater vehicles (UUVs) in terms of capabilities of managing uncertainty and possibility of reducing development and trial costs induced the development of hierarchical intelligent control architectures (Byrnes et al., 1993; Le Rest et al., 1994; Pascoal et al., 1997; Wang et al., 1993) and virtual environments for underwater robots (Bono et al., 1997; Brutzman, 1995; Leonard et al., 1995). The result was the implementation of open, flexible architectures for developing robotic techniques and fostering co-operation between research groups, ultimately promoting vehicle exploitation (Healey et al., 1996; Coste-Manière et al., 1996).

At the same time, numerous novel robotics systems that employ the infrastructure of the Internet to extend current human abilities have been developed (Paulos and Goldberg, 1999), allowing remote interactions with actual mobile robots connected to the Web.

The high level of maturity achieved by this technology in space robotics applications (Backes et al., 1998), together with the resolution of basic problems in UUVs navigation, guidance and control (Fossen, 1994; Fryxell et al., 1996; Healey and Lienard, 1993; Whitcomb et al., 1999), has suggested its extension to the field of underwater robotics in order to allow a scientific end-user to program, execute, and follow the state of progress of robotic vehicle missions at sea from the comfort of his/her laboratory.

In order to allow Internet users to interact with places far away from their home or test control algorithms on a real robotic platform, mobile robots on the Web have to satisfy some basic specifications (Siegwart and Saucy, 1999): the robot system requires a high degree of autonomy to face any large time delay; a minimal data transfer should always indicate the instant status, events and robot position; the control strategy of the robot should be as intuitive as possible, and the update rate of the transmitted video images should be as high as possible to provide a good feeling to reality. It is worth noting that autonomous underwater vehicles, which can communicate with the human supervisor and/ or the mission controller through a very narrow-band acoustic link satisfy the first three requirements, while good quality video feedback can be guarantee by data compression techniques in the case of Internet control of a remotely operated vehicle.

In the research reported in this paper, a pool mission of Romeo, the prototype ROV developed by the CNR-IAN, has been controlled by a Petri Net based mission controller developed by the IST-ISR using CORAL, through an Internet link between Genoa and Lisbon. According to the hierarchical control architecture paradigms, the ROMEO's control system provides a set of basic navigation, guidance, and control task functions, which can be coordinated and activated by an automatic mission controller connected to the vehicle surface network. Task activation and coordination depend on the occurrence of external discrete-events such as operator commands and events triggered by sensor readings. This motivated the development of a methodology for mission control that builds on the theory of Petri nets, which are naturally oriented towards the modeling and analysis of asynchronous discrete-event systems with concurrency. Mission control is thus performed by specifying mission programs that are embodied in Petri net structures. A mission control development environment named CORAL, designed by IST, allows for graphically constructing the required Petri nets and executing them in real-time on a CORAL software Engine that runs on a PC. The operator can visualize the state of progress of the mission by observing the evolution of tokens in the corresponding Petri net structure. CORAL allows the execution of mission programs for general robotic vehicles. The only extra effort required for a specific application is the definition of the list of commands/replies to and from a given robotic system. This was done in the case of ROMEO by defining the packet format for commands/replies between CORAL and ROMEO on the basis of the basic tasks adopted. Furthermore, Romeo's original control system was adapted by adding a state machine capable of handling the command/reply mechanism and of triggering the execution of specific tasks in parallel, in order to exploit the Petri Net's capabilities in concurrency management.

Romeo networked architecture and basic task functions are described in section 2, while a short description of the Petri Net-based CORAL mission controller is given in section 3. The experimental setup, i.e. the physical integration of Romeo control system and CORAL mission controller, is described in section 4, where experimental results are reported and discussed.

II. ROMEO

Romeo's architecture basically consists of three Ethernet LANs (surface, on-board and lab LAN), which can be connected to the world-wide web for scientific cooperations. The surface LAN connects a net manager computer (IPER) and a multi-machine distributed Human Computer Interface (HCI), which allows a number of different operators to interact with the robot at various levels of the control system. The conventional HC1 for scientific applications consists of three interfaces for the pilot, who tele-operates the vehicle, the supervisor, who supervises the plant behavior and resources' allocation, and the marine scientist, who examines real-time images and sensors data to detect areas of interest. The IPER machine acquires all the vehicle surface sensors, as, for instance, acoustic positioning system, ship GPS and gyro-compass, and manages communications from the human interfaces and the robot control system on-board the vehicle, dispatching the telemetry data and collecting, by solving conflicts, the user commands. The surface LAN can also support the connection of an external supervisory/mission control module, which automatically manages a vehicle mission, as in the case of CORAL shown in Figure 1. In addition to the computer running the vehicle control system, the on-board LAN can also connect advanced end-user devices carried by the vehicle. Furthermore, the lab LAN supports the Underwater Virtual World simulation facilities (6 d.o.f. vehicle dynamics, environment and sensors) and graphics interfaces.

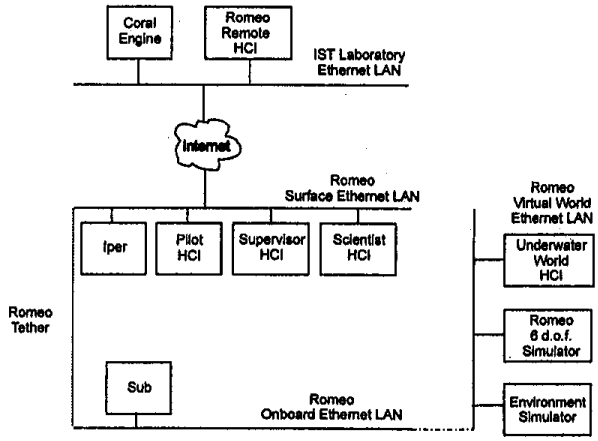

Fig. 1. CORAL-ROMEO networked architecture

Romeo control system is based on a hierarchical dual-loop architecture (Caccia et al., 1999) providing the human operator/mission controller with a set of basic guidance task functions performing auto-heading, auto-depth, auto-altitude and automatic maneuvering on the horizontal plane. The. sub-set of commands handling the task functions used in the CORAL-ROMEO mission are reported in Table 1.

Algorithmic details about depth, heading and horizontal maneuvering task functions can be found in (Caccia et al., 1999) and (Caccia et al., 1998) respectively. Here it is sufficient to remark the meaning of the basic horizontal maneuvering tasks:

- Short Range Maneuvering (SRM): the vehicle hovers the target with the desired orientation;
- Long Range Maneuvering (LRM): the vehicle heads the target;
- Go to (GOTO): the vehicle moves to the target point switching between LRM, medium range maneuvering, i.e. car driver-like guidance, and SRM according to the target range;

- Docking (DOCKING): go to the docking point, recorded executing the GET_DOCK command

```
Depth control:
      DEPTH(z*)
      DEPTH?(z*,TIME_OUT)
Heading control:
      HEADING(ψ*)
      HEADING?(ψ*,TIME_OUT)
Horizontal maneuvering:
      LRM(x*,y*)
      XY?(x*,y*,TIME_OUT)
      SRM(x*,y*, ψ*)
      GOTO(x*,y*, ψ*)
      XYPSI?(x*,y*, ψ*,TIME_OUT)
      DOCKING
      DOCKING?(TIME_OUT)
Horizontal motion estimation:
      FIX_XY()
      GET_DOCK()

x,y,z: vehicle position
ψ: vehicle heading
```

Table 1. ROMEO task function commands

The ?-marked commands implement the *space-trap* queries, which evaluate if the corresponding task function has been executed by a specified time-out. A suitable evaluation criteria is the following: indicating by $\underline{x}^{(k)}(t)$ the k-th time derivative of the generic signal $\underline{x}(t)$, «$\underline{x}(t)$ tracks $\underline{x}^{*}(t)$ of order n» if $\left\|\underline{x}^{(k)}(t)-\underline{x}^{*(k)}(t)\right\| < \varepsilon_k, \quad \forall t \in [t_0 - T, t_0], \; 0 \le k \le n$, where $\| \|$ represents the Euclidean-norm. Tracking of order 1 is usually adopted, except in the case of LRM tracking of order 0.

III. The CORAL Development Environment. Implementation issues

This section introduces CORAL as a software environment for the design and implementation of Petri net structures and explains its interfacing to the System Control level of the CNR-IAN ROMEO vehicle. We assume the reader is familiar with Petri Net theory; see (Cassandras, 1993) for a lucid presentation of the subject. See also (Oliveira et al., 1998) and the references therein for the development of CORAL as a tool for mission programming and execution tool for the MARIUS AUV.

The organization of CORAL can be explained in very simple terms with the help of Figure 2, which illustrates how the design of a subset of a generic Petri net is done, and how the equivalent CORAL language description is obtained. In order to understand the figure and the design methodology adopted, two basic concepts are required:

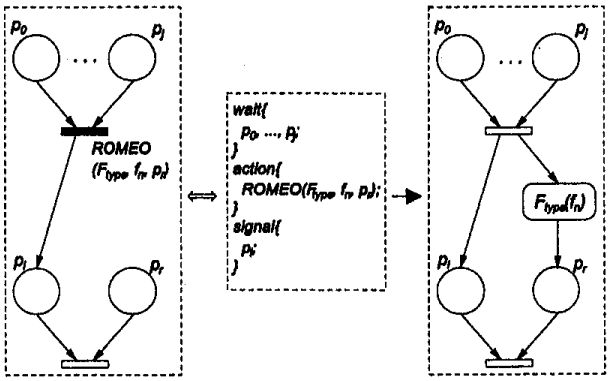

Figure 2: CORAL/System level Interface.

I) System Control Level Calling Header - The firing of a generic transition will start the execution of a System Control Level command, which is evoked through an header with the structure

$$\text{ROMEO}(F_{type}, f_n, P_r)$$

where ROMEO specifies a System Control Level function interface presented in the previous section, Ftype identifies the type of function or particular algorithm to be executed, and f_n are the parameters of the function to be called. The last calling parameter set P_r indicates a finite set of places in the Petri Net that will be marked depending on the type of messages received from the ROMEO system control level.

II) Wait, Action and Signal keywords - to describe a Petri net, the CORAL language uses three basic keywords: *wait, action,* and *signal.* The formal equivalence between the textual description of a Petri net using these keywords and its underlying Petri net graph can be easily understood by examining the input and output sets of a particular transition t_K. The following equivalence relationships follow immediately:

$$I(t_k) \Leftrightarrow \text{wait}\{p_0,\ldots, p_j\},$$
$$O(t_k) \Leftrightarrow \text{action}\{\text{ROMEO}(F_{type}, f_n, p_r)\}\ \text{signal}\{p_i\}.$$

where $I(t_K)$ and $0(t_K)$ are the input and output sets of places to and from the transition t_K, respectively. In this case, the function called has only one output event, and its occurrence will activate the marking of place p_r. The extension to more complex Petri Net structures is obvious.

A CORAL Engine has been developed that accepts Petri net descriptions and executes them in real-time. Figure 3 shows a schematic representation of the CORAL Engine data structure and the communication mechanisms that implement a Petri Net. The CORAL Engine accepts input messages corresponding to the markings of the Petri net being run, checks for the current set of enabled transitions, and issues output messages that correspond to the new markings determined by the firing of those transitions. In practice, this is done by executing a CORAL Engine synchronous loop described by the following sequence of actions:

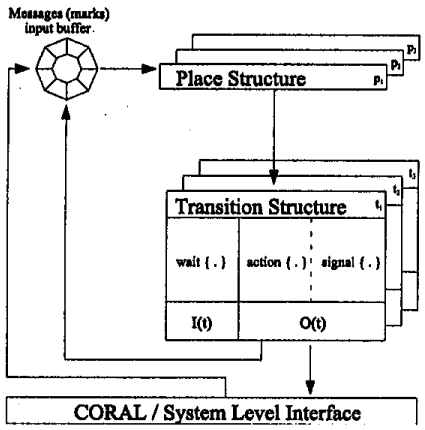

Figure 3: CORAL Implementation Structure.

for each message in the input buffer,
(1) update the number of marks in the corresponding place.
(2) for the current state, check for the set of enabled transitions.
(3) choose one transition from the set of enabled transitions.
(4) update the number of marks in the set of input places

$$I(t_k) \Leftrightarrow (\text{wait}\{...\}).$$

(5) issue messages in order to update the number of marks in the set of outputs places,

$$O(t_k) \Leftrightarrow (\text{action}\{...\}\ \text{signal}\{...\}).$$

(6) repeat (2) through (5) until the set of enabled transitions has been exhausted.

This cycle is repeated until the input buffer is empty.

IV. Experimental setup and results

The integration of the CORAL mission controller and ROMEO control system has been carried out in the lab thanks to the IAN Underwater Virtual World facilities (Bono et al., 1999). The execution of a set of real-time hardware-in-the-loop virtual missions allowed to verify the communication protocol, and to check the correctness of the events' management in the actual experimental conditions, including the communication bandwidth constraints due to the 9600 bps mobile phone serial data link between the ROV operating site and the network in the IAN lab. Once the system set-up was completed in the lab, the same set of missions has been executed with the actual vehicle in a swimming pool in Genoa, located at about 5 km from the IAN lab. The research scientists in IST lab in Lisbon followed the mission in real-time by means of a graphics representation of the vehicle basic telemetry (position and heading) sent through Internet using datagram BSD sockets. It is worth noting that for mission control commands/acknowledges transmission, according to the sequence shown in Figure 4, the more reliable stream BSD sockets have been used.

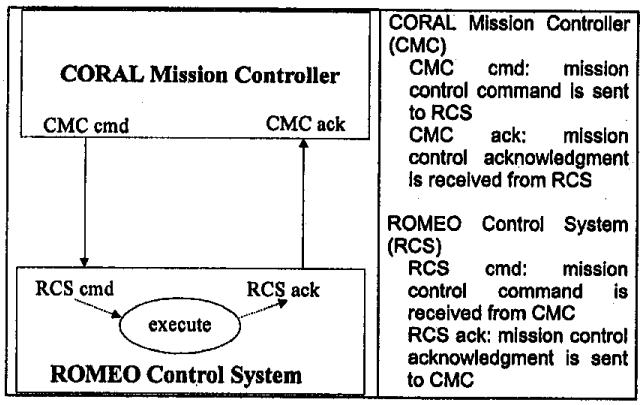

Figure 4. Right sequence of each mission control command/acknowledge.

The most significant mission consists of three phases:
1. go to operating site
   Romeo reaches the operating depth and a suitable orientation to initialize its horizontal position estimate by sonar ranges from the pool walls, records its position to dock there at the end of the mission and moves to the operating site;
2. way-point navigation
   Romeo navigates through four way-points, placed in the corners of a square, executing a long range maneuvering guidance task;
3. docking
   Romeo hovers the operating site, moves to the docking point and emerges.

The events' sequence, including the corresponding timestamps, has been recorded by both the Romeo Supervisor HCI and CORAL engine machines. The logged data have been merged off-line to analyze the system performance. The starting time has been fixed when the first command has been sent by CMC.

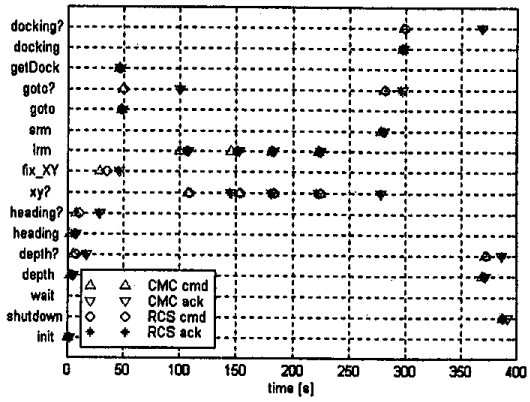

Figure 5. Romeo and CORAL pool mission events' sequence

Defining t_0^{cmc} and t_1^{cmc} the instants, in the CMC clock reference, when CMC sends and receives the «INIT»

command/ack, and t_0^{RCS} and t_1^{RCS} the instants, in the RCS clock reference, when RCS receives and sends the «INIT» command/ack, the time offset t_{offset} between the CMC and RCS recorded events' sequences has been approximated as $t_{offset} \approx \frac{t_1^{CMC} - t_0^{CMC}}{2} - \frac{t_1^{RCS} - t_0^{RCS}}{2}$.

The resulting events' sequence is plotted in Figure 5. The Petri net corresponding to the «go to operating site» phase is depicted in Figure 6, while the system behavior is reported in Figure 7, where the parallel execution in the presence of non-deterministic and time-varying delays of auto-depth and auto-heading tasks before estimating the vehicle horizontal position is pointed out.

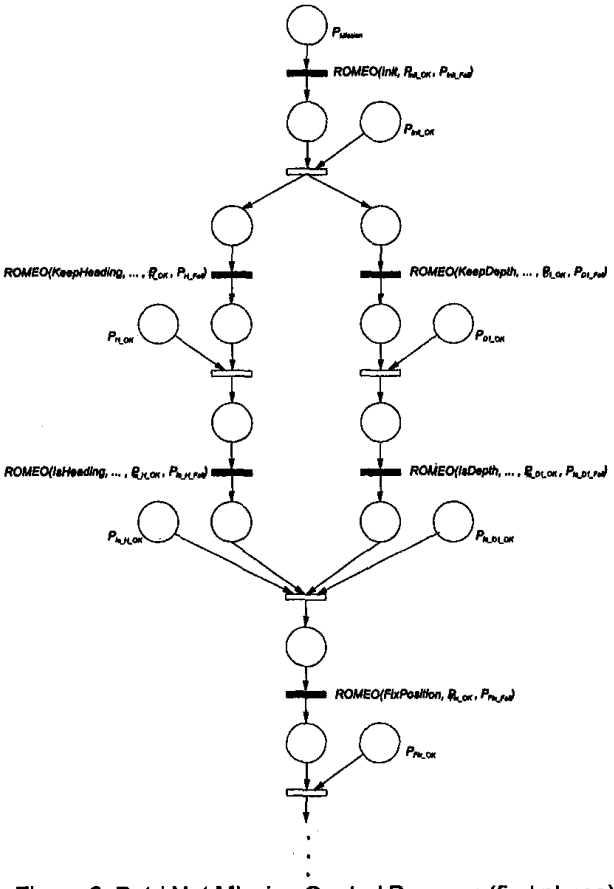

Figure 6: Petri Net Mission Control Program (first phase)

Although the DEPTH and HEADING commands are sent by CMC contemporaneously, the latter takes a longer time to reach RCS (see the time interval between 0 and 10 seconds in the top picture). Anyway, the CORAL engine waits for the successfully execution of both the tasks before sending the FIX_XY command, which takes 10 seconds to be executed in order to allow the correct initialization of the extended Kalman filter for motion estimation (see the time interval between 30 and 50 seconds in the top picture). At that time, the GET_ DOCK command is executed in order to record the point where to dock at the end of the mission. It is worth noting that, executing the GOTO task function, the ROV maneuvers on the horizontal plane in order to approach the target point with the desired orientation (see the bottom picture after about 50 seconds).

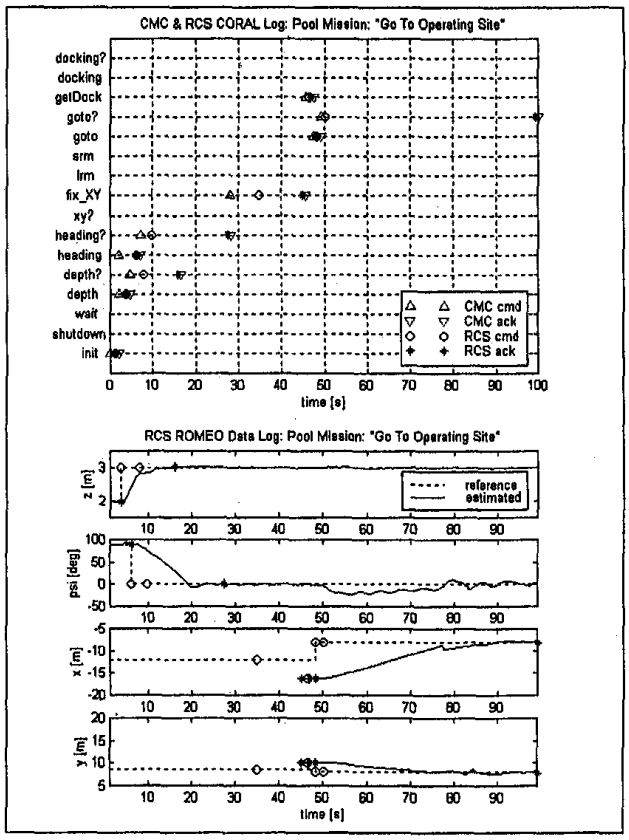

Figure 7. «go to operating site»: Romeo and CORAL events' sequence and ROV telemetry

The system behavior during the «way-point navigation» phase is reported in Figure 8, where remarkable network delays, i.e. of about 7 seconds, in the transmission of the first two LRM commands are visible. The maneuvering behavior consisting in heading the target is shown by the Romeo telemetry plots in the bottom picture.

Figure 9 shows the system behavior in the «docking» phase. After hovering the target with the desired orientation, the ROV reaches, with a complex maneuver, the docking position and emerges. The jump in the vehicle x position after more than 380 seconds (see the bottom picture) is due to bad sonar measurements obtained when the sonar was too close to the surface.

V. Conclusions

This paper described an experiment whereby a Petri Net based mission controller developed by the Instituto Superior Técnico using CORAL, was integrated with the control system of ROMEO, a prototype ROV developed by the

Robotics Department of CNR-IAN. The system was evaluated controlling ROMEO's missions directly from Lisbon in the underwater virtual world (UVW), in the IAN lab, and in a pool in Genoa The relative simplicity with which an example mission was jointly programmed and run demonstrates that true inter-group cooperation on the subject of underwater vehicle mission control is within reach. This example is but a small step in the process of developing systems that will allow scientific end-users to program, execute, and follow the state of progress of robotic vehicle missions at sea from the comfort of their laboratories.

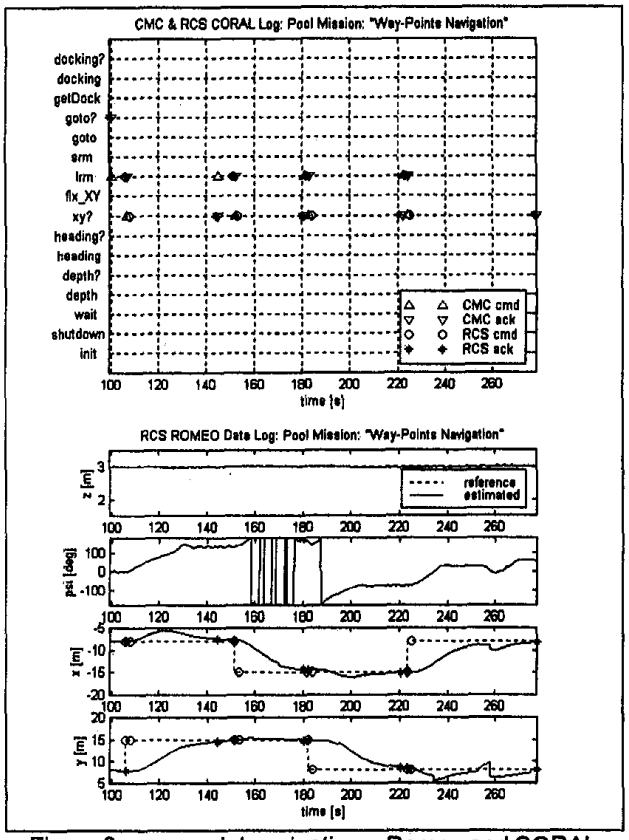

Figure 8. «way-point navigation»: Romeo and CORAL events' sequence and ROV telemetry

VI. Acknowledgments

This work has been performed in the framework of a bilateral agreement between the Italian Consiglio Nazionale delle Ricerche and the Portuguese Instituto de Cooperação Cientifica e Tecnológica Internacional.

VII. References

Backes, P.G., K.S. Tso, G.K.Tharp (1998). Mars Pathfinder mission Internet-based operations using WITS. Proc. of IEEE ICRA 98, pp. 284-291, Leuven, Belgium.

Byrnes, M.L. Nelson, S.H. Kwak, R.B. McGhee, A.J. Healey (1993). Rational Behavior Model: An Implemented Tri-Level Multilingual Software Architecture for Control of Autonomous Underwater Vehicles, Proc. of 8th International Symposium on Unmanned Untethered Submersible Technology, pp. 160-178.

Bono R., G. Bruzzone, M. Caccia, G. Veruggio, P. Virgili (1997). A Real-time Architecture for Development and Control of Unmanned Underwater Vehicles, Proc. of 4th IFAC Workshop on Algorithms and Architectures for Real-Time Control, Vilamoura, Portugal, pp. 58-63.

Bono R., G. Bruzzone, M. Caccia, G. Veruggio (1999). A flexible networked architecture for scientific applications and robotics research, Proc. of 11th International

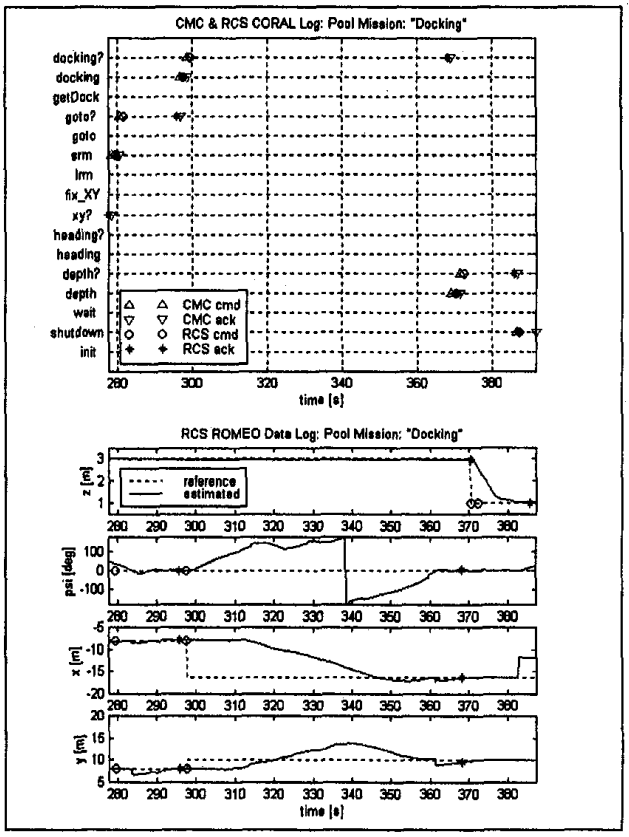

Figure 9. «docking»: Romeo and CORAL events' sequence and ROV telemetry

Symposium on Unmanned Untethered Submersible Technology, Durham, USA.

Brutzman, D.(1995). Virtual World Visualization for an Autonomous Underwater Vehicle, Proc. of Oceans '95 MTS/ IEEE, San Diego, USA.

Caccia M., Casalino G., Cristi R., Veruggio G. (1998). Acoustic motion estimation and control for an unmanned underwater vehicle in a structured environment, IFAC Control Engineering Practice, Vol. 6, No. 5, September 1998, pp. 661-670, Elsevier Science Ltd., England.

Caccia M., G. Bruzzone, G. Veruggio (1999). Hovering and altitude control for open-frame UUVs. IEEE

International Conference on Robotics and Automation, pp. 72-77. Detroit, USA.
Cassandras C. (1993). Discrete Event Systems. Modeling and Performance Analysis, Aksen Associates Incorporated Publishers.
Coste-Manière E., H.H. Wang, S.M. Rock, A. Peuch, M.Perrier, V.Rigaud, M.J.Lee (1996). Cooperative research in underwater robot programming. Proc. of the 6th IARP'96 Workshop on Underwater Robotics.
Fossen, T.I. (1994). Guidance and Control of Ocean Vehicles, John Wiley & Sons, England.
Fryxell, D., P. Oliveira, A. Pascoal, C. Silvestre and 1. Kaminer (1996). Navigation, guidance and control of AUVs: an application to the MARIUS vehicle, IFAC Control Engineering Practice, Vol. 4, No. 3, pp. 401-409, Elsevier Science Ltd., England.
Healey, A.J. and D. Lienard (1993). Multivariable slidingmode control for autonomous diving and steering of unmanned underwater vehicles. IEEE Journal of Oceanic Engineering, Vol. 18, No. 3, pp. 327-339.
Healey, A., D. Marco, R.B. McGhee, P. Oliveira, A. Pascoal, V. Silva, C. Silvestre (1996). Implementation of a CORAL/Petri net strategic level on the NPS Phoenix vehicle, Proc. of the 6 th IARP'96 Workshop on Underwater Robotics.
Leonard, J. J., S. T. Tuohy, J. G. Bellingham, B. A. Moran, J. H. Kim, H. Schmidt, N. M. Patrikalakis and C. Chryssostomidis (1995). Virtual Environments for AUV Development and Ocean Exploration. Proc. Int. Symp. on Unmanned Untethered Submersible Technology, pages 436-443, New Hampshire.
Le Rest, L. Marce, V. Rigaud (1994). VORTEX-PILOT: a top-down approach for AUV's mission telerobotics language, Proc. of OCEANS'94, vol. 2, pp. 102-107.
Oliveira, P., A. Pascoal, V. Silva, C. Silvestre (1998). The Mission Control System of the MARIUS AUV: System design, implementation, and tests at sea. International Journal of Systems Science, Special Issue on Underwater Robotics, Vol. 29, No 10, pp 1065-1080.
Pascoal, A., C. Silvestre, P. Oliveira, A. Bjerrum, G. Ayela, J.-P. Pignon, S. Bruun, C. Petzelt (1997). MARIUS: An Autonomous Underwater Vehicle for Coastal Oceanography. IEEE Robotics and Automation Magazine, Special Issue on Robotics and Automation in Europe: Projects funded by the Commission of the European Union, pp. 46-59.
Paulos, E., and K.Goldberg (organizers) (1999). Current challenges in Internet robotics. Workshop IEEE ICRA 1999. Detroit, USA.
Siegwart, R. and P.Saucy (1999). Interacting mobile robots on the Web. Proc. of Workshop IEEE ICRA 1999, Detroit, USA.
Wang, H.H. , R.L. Marks, S.M. Rock, M.J. Lee (1993). Task-Based Control Architecture for Untethered, Unmanned Submersible, Proc. of 8th International Symposium on Unmanned Untethered Submersible Technology, Durham, USA, pp. 137-148.
Whitcomb, L., D. Yoerger, H. Singh (1999). Advances in Doppler-based navigation of underwater robotic vehicles. Proc. of IEEE ICRA 1999, pp. 399-406, Detroit, USA.