ON THE CONTROL OF AUTONOMOUS MOBILE ROBOTS

J. Sousa, MSc, F. Pereira, PhD, A. Martins, MSc, A. Matos, MSc, J. Almeida, MSc, N. Cruz, MSc and S. Cunha, MSc.
Instituto de Sistemas e Robótica-Porto e Faculdade de Engenharia da Universidade do Porto,
Rua dos Bragas, 4099 Porto Codex, Portugal
e-mail: flpereir@garfield.fe.up.pt

E. Pereira da Silva, MSc.
Instituto de Sistemas e Robótica-Porto e Instituto Superior de Engenharia do Porto,
R. de S. Tomé, 4200 Porto, Portugal

ABSTRACT

In this article, we describe the effort being carried out in the analysis, design and implementation of the Vehicle and Mission Management Systems of a mobile platform for autonomous transportation, surveillance and inspection in structured and semi-structured industrial environments. The Vehicle and Mission Management System is based in a hierarchical structure organized linguistically permitting the real-time parallel execution of tasks. This architecture is composed by three levels, Organization, Coordination and Functional, structured according to the Increasing Precision with Decreasing Intelligence Principle.

1. INTRODUCTION

This paper describes the analysis, design and implementation of a mobile robotic system with increasing degrees of autonomy to operate in semi-structured industrial environments. Design requirements were driven by the following major applicational goals:

Platform for research activities.

Educational platform.

Demonstration for industrial applications.

The analysis and design phases were conducted in such a way as to explicit the main guidelines associated with it in an attempt to extract a systematic approach. This turned out to be a large effort in the beginning of the project which was targeted not to implementation but rather to the definition of the main architectural concepts, the main interfaces, and to the development methodology. However, this turned out to be a sound investiment as it not only led to the selection of engineering systems methodologies which were instrumental in the success of the project, but also provided a solid basis for future refinements. The absence of systematic analysis, design and implementation techniques is the first major difficulty encountered.

Another important issue to consider is the clear relation between human intervention and autonomy. The full specification of autonomy requirements leads to a structure which may naturally accommodate human intervention at various levels. Furthermore, an appropriate design of human intervention plays an important role in the success of operational systems. This constitutes a valuable contribution to the dissemination of the technology and a motivation for the development of more complex systems.

The remainder of this paper is organized as follows: After presenting system components, in section 3, the software implementation is described with detail in section 4. In section 5, the user interface is presented emphasizing the main operating modes of the system. The paper concludes with some remarks concerning future work conclude the paper.

2. REQUIREMENTS AND DESIGN OPTIONS

Now, we describe the main architectural concepts involving Navigation, Guidance, Dynamic Control and the Vehicle and Mission Management Systems. These functions were implemented in a commercially available system consisting of a mobile platform ROBUTER running the real-time operating system ALBATROS [2], and a UNIX SUN station.

The implementation of a control system for an intelligent machine [3] involves: definition of design requirements, decomposition of design requirements into implementable pieces, definition of the hardware and functional processing so that the requirements are met, mapping of the functional processing into the hardware, and global verification.

Requirements

Operation in an industrial environment represents the main motivation behind the definition of the following design requirements for the system:

1. Two main operating modes: teleoperation and mission execution. While in the first operating mode the operator directly controls the motors of the platform, in the second, the operator commands the execution of pre-specified missions. Besides a high level language [4] to support mission specification, mission templates can be used for typical missions.

2. Mission execution involves navigation in a semi-structured environment and precise docking at pre-defined locations to accomplish given tasks.

3. Two types of users: Operator and system administrator. While the first is responsible for operating the vehicle, the second will be in charge of defining and maintaining the world model, parameter values and implicit execution rules which constitute the system´s doctrine.

4. User interface should be supported by a graphical environment. Simplicity of operation is the main design issue in what concerns operator´s interface.

The use of this platform poses additional constraints on the system´s requirements. These constraints are related to the decomposition of design requirements into implementable pieces and respective mapping of the functional processing into the hardware. The above mentioned requirements constitute the basis for the main design options [4]:

Tri-level hierarchic control architecture

The control architecture is composed of three levels: Organization, Coordination and Functional Layer, [5]. This option corresponds to the need to have well defined planning and control mechanisms which are at the heart of hierarchical approaches [7], [8]. The emergence of reactive behaviors at the Functional Layer is enabled or disabled by the coordination level.

Linguistic approach

The hierarchic structure will rely on the definition of independent linguistic units for each level [10]. Besides accomplishing parallel and independent design and programming for the hierarchic architecture, this approach is also particularly suited to support the coordination activities. Furthermore, it provides a common framework to deal with concepts from both control and information theories.

Mission Composition from Primitive Functions

Maximum flexibility is obtained when mission components are designed as the composition of elementary capabilities. The composition mechanisms are supported by the hierarchic encapsulation of the linguist structure which is well suited to the incremental development of behaviors and functionalities.

Error handling and recovery procedures

An additional hierarchy of error handling devices is considered. A global error handler takes care of errors which were not solved by the corresponding module error handling device.

Definition of basic motion primitives

These requirements led to the choice of continuous path following and to the definition of the path as a set of basic motion primitives. These options are well adapted to the implementation in a system composed of two computers connected through a serial link. The pilot module (at the workstation) takes the path segment data and passes each segment in turn to the motion controller (at the platform computer) which converts the sparse spatial data of the plan into detailed temporal data ([11] [12]).

Distributed processing

The software components with stringent real-time requirements reside on the platform computer. The remaining software components reside at a UNIX station where special attention is paid to real-time issues.

3. SYSTEM COMPONENTS

Figure 1 shows the main hardware components for the system in the initial configuration.

Workstation

Graphical UNIX workstation supports the user interface, the communication server and the mission management system.

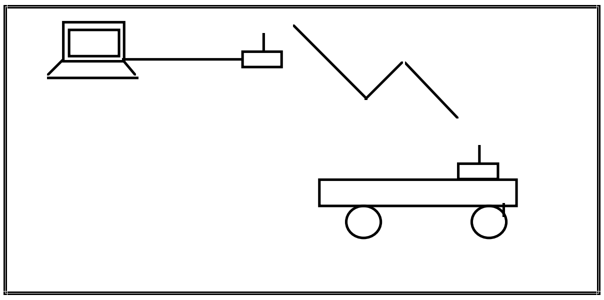
Figure 1

Mobile platform:

With a payload up to 120kg and running a real-time operating system ALBATROS [2] specifically designed for multi-axis control, the platform is composed of the following:

a) Differential steering system based on two independent motors.

b) Optical encoders for platform positioning.

c) Network of ultrasonic sensors for range sensing.

d) Bumpers.

e) VME bus with 5 slots, 68020, 16 Mhz.

f) 9600 baud serial link.

g) Software package for developing and downloading C application programs from a host machine.

4. SOFTWARE IMPLEMENTATION

Initial control software implementation relies on the interconnection of two distinct environments whose communication is made via a serial link. This solution is described on the next two sub-sections. Early in the development stages, it was clear that the limited onboard computational capabilities and the low communication rate between both environments, were a serious drawback to address any realistic application scenario. In subsection 4.3, we describe the design effort undertaken introduce a new onboard operating system.

4. 1. UNIX environment

Figure 2 describes the main software components of the system running on the workstation.

Communication Server

The communication server manages interprocess communication at this level. Rapid prototyping and dynamic routing in an Ethernet environment represent the requirements underlying the main implementation options. Sockets are at the basis of communication. A host table and a post table are used for the purpose of routing messages. The server supports broadcast and point to point communication modes.

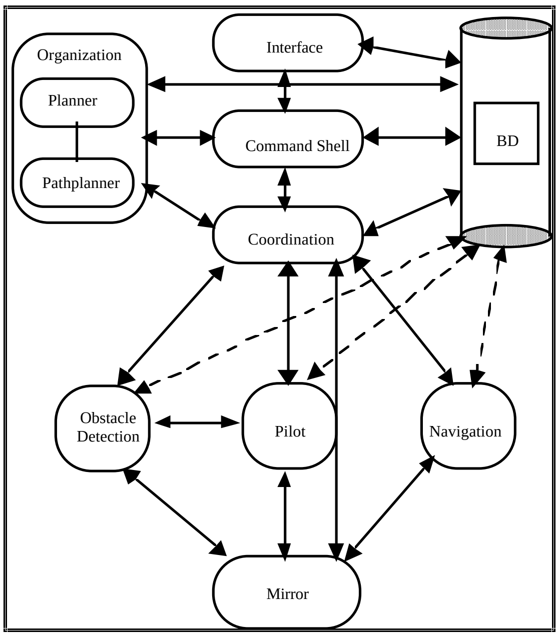

Figure 2

Communication Server

The communication server manages interprocess communication at this level. Rapid prototyping and dynamic routing in an Ethernet environment represent the requirements underlying the main implementation options. A version permitting transparent shared memory access is under development.

Planner

The generation of the mission plan from two distinct sources provides a convenient separation between the mission specification defined in terms simple for an operator to instruct the vehicle´s tasks and the definition of a set of rules underlying mission execution. The mission specification language fully supports the definition of missions to be accomplished in an industrial environment.

Command Shell

A user language [2] supports this command shell. Built-in mechanisms of the language support operation under a distributed computing system. This shell was not only particularly useful in the development phase since it was built incrementally, but also allows the dynamic augmentation of system´s functionalities. The underlying design options address the low-cost integration of the system in an industrial environment either in a standalone version or connected to another computing system.

Coordination

The coordination module is responsible for coordinating the activities of the whole system according to an execution guideline represented by the mission plan. For a given execution context, decisions represent the control actions taken as response to sensor and logical stimuli in order to accomplish the mission goals. By considering a rule based coordination structure, the coordination kernel maintains the coherence of the plan execution so that commands or tasks are not executed outside the respective scope.

Plan interpreter - dynamic compiler responsible for interpreting the rules and tasks constituting the mission and the associated doctrines. The net of rules and tasks can be altered by this module upon request of the mission supervisor.

Plan Supervisor - organizes, schedules and manages the tasks required for mission execution. It has to verify that tasks produce the expected effects and must react in case of discrepancy.

Task Coordinator and Refinement - implemented via the coordination formalism, constitute an abstraction of the coordination and associated refinement for each single task. Task refinement selects the best alternative, if available, for the task execution in terms of the current configuration. Implementation is supported UNIX utilities LEX and YACC.

Platform Mirror

This system mirrors the activity of the platform computer by maintaining a table representing the current platform state. The coordination activities are based on this state. Non-negligible transmission delays for the serial link and non-deterministic interprocess communication aspects of UNIX impose the need of temporal windows for command execution. Clock synchronization is fundamental at this point since all messages received from the platform are time-stamped.

The Path Planner, Pilot, Obstacle Detection and Navigation Modules are described in [4].

4.2. ALBATROS environment

Figure 3 describes the main software of the system running on the ROBUTER computer under the real-time multi-tasking ALBATROS operating system. The real-time management of processes are essential to get the most out of the available computational resources. Intensive experimentation and testing is required in order to define the optimal combination, in terms of the envisaged mission scenarios, of process priorities, durations and offsets.

Communication Server

This communication server maintains two independent buffers in order guarantee bi-directional communication with the platform. It is also responsible for routing messages from the workstation.

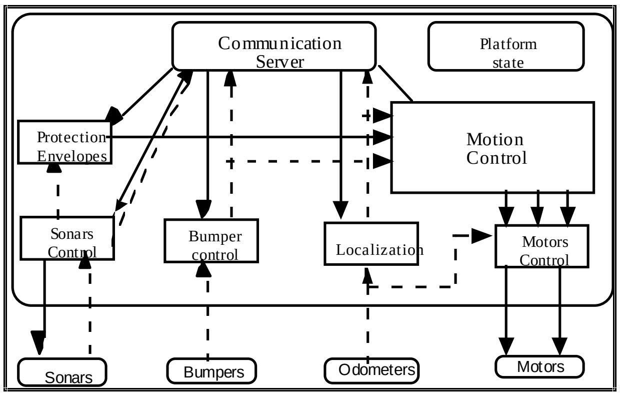

Figure 3

Sonars and Bumpers Control

This module controls the belt of ultra-sonic sensors. The belt is organized into a series of nodes which group non-interfering sonars. Every 100 ms the readings from a sonar node are sent to the workstation for obstacle detection and navigation purposes. Precise sonar description of a region can be obtained by changing the control configuration of the sonars.

The activation of the bumpers may trigger a safety behavior implemented at the motion control module which consists in stopping the platform until the obstacle is removed and a continuation command is issued by the operator. The behavior can be enabled or disabled either through a menu option or by stating this option in the mission plan.

Protection envelopes

A protection envelope defines a region where the detection of a sonar return triggers the emergence of a safety behavior and issues a warning message to the operator. Presently, there are two velocity dependent protection envelopes. The first is used to generate a signal to decrease the platform velocity. The second is used to command the immediate stop of the platform.

Localization

The localization process runs every 50 ms. It is basically a dead reckoning procedure based on the readings from the encoders. The position and angle with respect to some fixed reference frame (x, y and θ) are calculated by integrating an equation [4] describing the relation between the positional increments of the platform and the measurements from the encoders. Localization data corresponding to the respective sonar readings is sent to the workstation to allow the navigation algorithm to avoid too large dead-reckoning errors.

Platform state

Transmission delays caused by the serial link and non-deterministic timing characteristics of inter-process communication at the UNIX workstation make explicit time dependencies for command execution difficult to manage. Motion control logic is managed by a finite state machine which is used to filter commands or messages from the UNIX workstation or from other processes running at the platform computer.

Motors control

This process runs at the same rate of the localization process. Independent velocity PI control algorithms are used for each motor. Differences between the two motors are properly taken into account by using experimental data to tune the gains of each control loop. Velocity feedback is obtained by differentiating the positional measurements from each wheel.

Motion Control

There are two basic control modes: direct drive of the motors (in the teleoperation mode) and execution of a motion primitive. Straight line and arc of circle are the basic implemented motion primitives. These motion primitives take as input initial and end points, type of segment and a signed velocity for that segment. The positional and heading errors of the platform with respect to the reference trajectory generate a velocity reference for each motor. The activities of this module are governed by the control module logic (Fig. 4)

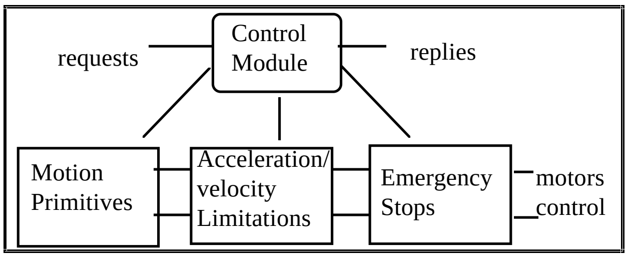

Figure 4

Motion primitives are implemented as self-contained behaviors [2] which are characterized by sets of pre-conditions, post-conditions and doctrine rules. These may include triggering alarms and eventually stop whenever the Euclidean distance from the reference trajectory exceeds a threshold level.

4.3. Migration from ALBATROS to V_X-Works

The replacement of the ALBATROS by real-time, multi-task operating system V_X-Works will overcome the limitations described at the beginning of this section for two reasons:

Higher computational power due to increased memory, better processor and higher clock frequency. These increased capabilities allow to run software in the onboard processor that was previously installed in the workstation.

Improved communications as the serial link will be replaced by a radio ethernet link.

These new capabilities will allow not only the design of a more efficient computational architecture, but also it will permit a much faster development. This is due to the fact that, besides being fully compatible with UNIX and having powerful development tools, V_X-Works makes it easier to define a configuration where the on-board computer is integrated in a simulation environment. This enables to support all the simulation activities with the code actually developed to the system, thus decreasing development costs and time.

5. USER INTERFACE

The user interface is fully supported by a multi-window system developed in MOTIF, Figure 5, describes the hierarchy of windows.

While the system administrator has access to the whole hierarchy, the operator can only use the functionalities which are related to the two main operating modes.

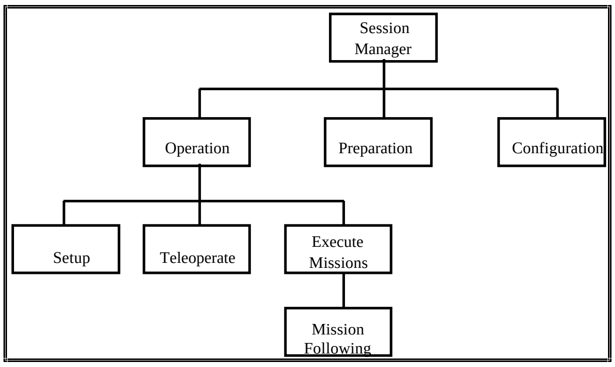

Figure 5

6. CONCLUSIONS

This paper presents the implementation aspects of a control architecture for an industrial mobile platform. An open system was developed so that a basis for future developments was established and a platform for research and development on the design of intelligent control systems is available.

Lessons learned include:

Cost effectiveness of a "hardware in the loop" simulation environment.

Realistic industrial applications can not be addressd due to the severe software and hardware constraints of the configuration adopted in the first phase.

V_X-Works real-time multi-tasking operating system and its development environment seems to fulfill the needed requirements.

ACKNOWLEDGEMENTS

Part of this work was developed in the PO-Robot Project funded under the NATO Science-for-Stability Programme.

REFERENCES

[1] G. Saridis, "Analytic Formulation of the Principle of Increasing Intelligent with Decreasing Precision for Intelligent Machines," Automatica, Vol. 25, nº. 3, 1989, pp.461-467.

[2] Albatros Reference Manual, Robosoft S.A., France, 1993.

[3] J. Borges de Sousa, F. Lobo Pereira and E. Pereira da Silva, "Software Architecture for Autonomous Mobile Vehicles: A Survey", Autonomous Underwater Vehicles, Kluwer Academic, in press.

[4] E. Pereira da Silva, J. Borges de Sousa, F. Lobo Pereira, J. Sequeira and I. Ribeiro (1994) "On the Design of the PO-ROBOT System" to appear in the Procs of the IEEE Intelligent Vehicles Symp.'94, Paris, France, Oct. 1994.

[5] J. Borges de Sousa, F. Lobo Pereira and E. Pereira da Silva, (1994) "A Dynamically Configurable Architecture for the Control of an AUV" to appear in the Procs. OCEANS 94 Conf., Brest, France, Sept. 1994.

[6] R. Chatila, R. Alami, Degallaix and H. Haruelle, "Integrated Planning and Execution Control of Autonomous Robot Actions". Procs of the IEEE Intern. Conf. on Robotics and Automation, Nice, France, 1992.

[7] R. Alami, R. Chatila, M. Ghallab,"Mission Planning for an Autonomous Mobile Robot", In Procs. of AIENG 93, Toulouse, (France), 1993.

[8] G. Saridis, "Architecture for Intelligent Machines", Technical Report CIRSSE-58, Center for Intelligent Robotic Systems for Space Exploration (CIRSSE), Rensselaer Polytechnic Institute, 1991.

[9] A. J. Healey, R.B. McGhee, R. Christi, F.A. Papoulias, S.H. Kwak, Y. Kanayama and Y. Lee,"Mission Planning, Execution and Data Analysis for the NPS AUV II Autonomous Underwater Vehicle," Procs of 1st IARP Workshop on Mobile Robots for Subsea Environments, Monterey, CA, 1991, pp. 177-186.

[10] G. Saridis and J. Graham, "Linguistic Decision Schemata for Intelligent Robots," Automatica, vol. 20, nº 1, pp. 12-126, 1984.

[11] W. Nelson and I. Cox, "Local Path Control for an Autnomous Vehicle", Autonomous Robot Vehicles, Springer-Verlag, pp. 38-44.

[12] T. Hongo, H. Arakawa, G. Sugimoto, K. Tange and Y. Yamamoto, "An Automatic Guidance System of a Self-Controlled Vehicle", Autonomous Robot Vehicles, Springer-Verlag, pp. 32-37.