Proceedings of the 10th Mediterranean Conference on Control and Automation - MED2002
Lisbon, Portugal, July 9-12, 2002.

A FRAMEWORK FOR THE AUTOMATION OF A REMOTELY OPERATED VEHICLE

Sérgio Loureiro Fraga, João Borges Sousa, Fernando Lobo Pereira

* Laboratório de Sistemas e Tecnologias Subaquática
Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal
e-mail: {slfraga,jtasso,flp}@fe.up.pt
http://www.fe.up.pt/~lsts

Keywords: Remotely Operated Vehicle, Trajectory Generation, Vehicle Automation, Manoeuvres, Underwater Inspection

Abstract

A framework for the automation of a Remotely Operate Vehicle (ROV) is presented. This framework entails a three-layered control architecture, a principled approach to design and implementation within the architecture, and hybrid systems design techniques. The control architecture is structured according to the principle of composition of vehicle motions from a minimal set of elemental manoeuvres that are designed and verified independently. The principled approach is based on distributed hybrid systems techniques, and spans integrated design, simulation and implementation as the same model is used throughout. Hybrid systems control techniques are used to synthesize the elemental manoeuvres and to design protocols that coordinate the execution of elemental manoeuvres within a complex manoeuvre. The architecture is fault-tolerant by design since it uses verified manoeuvres.

This work is part of the Inspection of Underwater Structures (IES) project whose main objective is the implementation of a ROV-based system for the inspection of underwater structures.

1 Introduction

Remotely Operated Vehicles (ROV) are small, tethered submersibles. There are numerous applications for ROVs, such as oceanographic surveys, operations in hazardous environments, underwater structure inspection, and military applications. Admittedly, ROV control presents many difficult challenges. However, recent advances in navigation, power and communication systems offer the appropriate technology to use ROV for data gathering in the coastal and open ocean very feasible.

Here, and in this context, we will concentrate on the development of a framework for the automation of a ROV. By a control framework, we mean the organization of the problems faced by the ROV as manoeuvres and manoeuvre switching. In our experience, the design of the control and software framework is one of the most critical phases in automated vehicle development. The architecture helps us understand what the system does, understand how the system works, be able to think and work on pieces of the system, extend the system, and reuse parts of the system to build another one.

We foresee two levels of automation, one in which "high-level" manoeuvre commands are sent by the operator and one of complete automation. The literature is abundant in low-level control techniques for ROV and/or ship applications. These techniques are mainly tailored to solve low level control problems (the level of control that directly interfaces with actuators such as auto-pilots, etc...) that are formulated in the framework of continuous time and differential equations. However, the high-level manoeuvre automation encompasses the realm of logic, and of discrete event models interacting with differential equations that model ROV dynamics. To reach the next level of

automation, it will be necessary to consider hybrid systems and new control techniques, in which continuous time and discrete events interact.

We organize the ROV motions in terms of manoeuvres. We are faced with a whole range of manoeuvres, of which some are simpler, and others more complex. We strive to define a basic set of "elemental manoeuvres", from which all the manoeuvres can be derived. Once we have found a minimal set of elemental manoeuvres, we can verify their design for safety. We then compose the complex ROV manoeuvres, using the elemental manoeuvres as building blocks. This enables us to always design correct manoeuvres, that is, manoeuvres that meet the given specifications, which may include safety and ensured results, even in the presence of disturbances. The novelties of the framework are the introduction of concepts and theories from distributed hybrid systems, and the use of systems engineering principles for architectural design [1]. We illustrate this approach with a case study from the project Inspection of Underwater Structures (IES). This project concerns the design and implementation of an advanced low cost system for the inspection of underwater structures based on a ROV.

This paper is organized as follows. In section 2, we describe the IES project. Section 3 presents our principled approach to design and implementation. Section 4 contains an overview of the control framework. Finally, section 5 deals with some concluding remarks, and future work.

2 The IES project

The project Inspection of Underwater Structures (IES) concerns the design and implementation of an advanced low cost system for the inspection of underwater structures based on a Remotely Operated Vehicle (ROV). The project started in 1999, has a total duration of 3 years, and is funded by PROGRAMA PRAXIS XXI - MEDIDA 3.1B, Portugal. IES is a collaborative project that involves the Associação dos Portos de Douro e Leixões (APDL)[1], Faculdade de Engenharia do Porto and Instituto de Sistemas e Robótica – Pólo do Porto.

A summary of the main inspection requirements for the IES project is presented in Table 1, while a typical operational scenario is depicted in Figure 1.

Main requirements		
Inspection objective	*Features*	*Main difficulties*
Evaluation of the state of corrosion of submerged steel plates	Metal plates are part of the pier structure. Critical corrosion spots: links between consecutive plates	Low visibility due to pollution and marine growth. Wave induced motions. Unavailability of good models of thruster-wall interactions.
Evaluation of the state of conservation of underwater structures	Fissures on the metal surface. Misalignment between consecutive metallic curtains.	Sensing small fissures and corrosion. Size of the area to be inspected.
Concrete pillars maintenance	Fissures on concrete pillars.	Wave-induced motions. Currents produced by tide variation and Leça river mainstream.
Additional requirements		
Verifying the state of the tetra pods that protect the harbour.	The locations of the tetra-pods change due to the action of the sea.	High waves. Currents. Operation from a vessel.
State of ship hulls.	Inspection should be made during loading and unloading operations.	Low visibility due to pollution and low luminosity.

Table 1: Inspection requirements

The main innovations of the IES project with respect to commercially available ROV solutions are:

- On-board power and computer systems. This physical configuration minimizes the number of wires in the tether cable thus minimizing drag and improving performance.
- Two modes of operation: tele-operation and tele-programming. The tele-operation mode is a standard feature in ROV systems. The tele-programming mode enables the operator to program automated operations, such as trajectory or path tracking.

[1] Harbor Authority

- Integrated navigation. The IES navigation system integrates data provided by an external acoustic system and by internal sensors for better control performance and position accuracy.
- PC-based control. Easy to use, COTS technology, low development and maintenance costs.
- Advanced control systems. The ROV control system includes advanced automated operation modes that relief the operator from tedious tasks, and do not require extensive training.
- Open system. The project uses standard software development tools and principles. The software architecture allows for the easy integration of code developed by third parties.

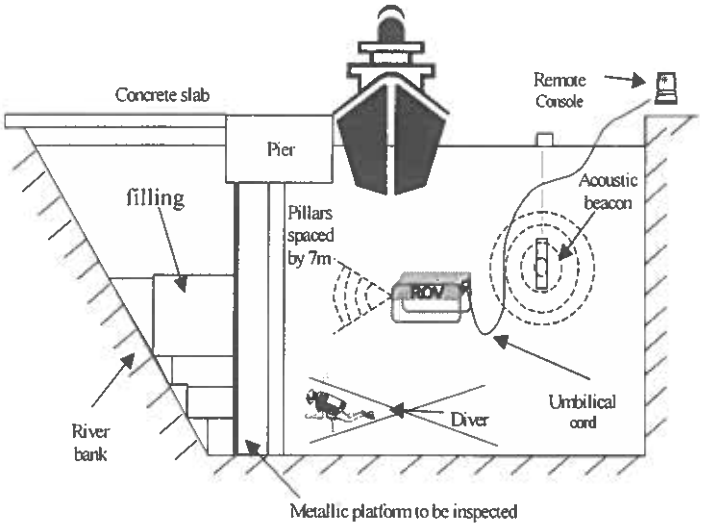

Figure 1: Typical operational scenario

The IES project integrates the following innovative technologies and systems developed at the Laboratório de Sistemas e Tecnologias Submarinas (LSTS) from Porto University:

- Acoustic navigation system (see [2,3]).
- Advanced control systems.
- Power and motor control.

Except for the ROV frame, hull and thrusters, all the other components and systems were designed and implemented at LSTS. The ROV frame, hull and thrusters are a customized version of the Deep Ocean 500 S model from Deep Ocean Engineering (Figure 2). The main difference with respect to the standard model is an additional cylinder that houses electronics and sensors.

Figure 2: The IES remotely operated vehicle

3 The Approach

The general problem of vehicle control automation is not a trivial one. ROV control automation is not an exception. We envision two levels of automation: 1) automation of the *basic manoeuvres* and, 2) full automation, from planning to operations.

From the control perspective, the main difficulties for automating ROV control come from the first level of automation, that of the elementary manoeuvres. The problem is that traditional control techniques, that form the current practice in ROV control, are not adequate for this purpose. These techniques are mainly tailored to solve the low-level control problems. The problem of design for the second level of automation is not trivial, but it relies on the services provided by the first level. Let us look at what is involved in the design of a basic manoeuvre controller to understand why we need new techniques.

Formal methods. During manual control the operator uses some control logic to command the operation of the ROV. We need to be able to capture this logic, to express it mathematically, and

to check it for consistency and correctness. This is why we need formal methods from computer science [4].

Models. The realm of the low-level control problems is that of continuous time and differential equations. Here, we enter the realm of logic, and of discrete event models interacting with differential equations that model ROV dynamics.

Automation is not mimicking manual operation. First, it may prove difficult to encode all the control logic in a compact representation. Second, even if this is done, there is no assurance that there are no flaws in the control logic. Third, automation enables complex quantitative reasoning that can be used to optimise operations or to implement complex control routines that ensure safe operations, even in the presence of disturbances.

Obviously the realm of interactions between continuous time dynamic models and discrete events requires the consideration of advanced control techniques. Let us discuss what is required from control design.

Logic based control. The actions of each module follow some control logic. This control logic is complicated because it not only involves discrete event behaviour, but also complex continuous dynamics and interactions with other modules. The traditional practice of `if-then-else` programming is no longer adequate. It is not required to be an expert programmer to realize that the amalgamation of `if-then-else` statements is very difficult to verify, and often leads to unpredictable behaviour.

Safety and predictability. Two important requirements for automated operation are predictability and safety. The controller has to perform according to what it is expected to do while respecting safety constraints. Disturbances play against predictability and safety. The question is then how to design predictable and safe controllers. This entails designing not only control laws, but also regions for safe operation.

In our experience, a lot of control code is designed and implemented using techniques that are not particularly adapted to this level of automation. The problems are: 1) expressiveness and, 2) tool integration. We need compact representations to express the models and relations described above, and we need to interface the code with tools that facilitate design and verification.

It is now clear that control design at this level requires: 1) **formal models** that span the design and implementation process; 2) a framework where we can study the overall structure and properties of control design that are not appropriately addressed within the constituent modules -**control architecture**; 3) **a principled approach** to design and implementation; 4) **new control techniques**.

4 Control Framework

This section describes the key elements of our approach. For the sake of clarity we opted to skip the mathematical details that can be found, for example, in [5].

4.1 Principles and techniques

We organise the operations of the ROV as a sequence of manoeuvres. First we define a basic set of "elemental manoeuvres", from which all the manoeuvres can be derived. Once we have found a minimal set of elemental manoeuvres, we can verify their design for safety. We then compose the complex ROV manoeuvres, using the elemental manoeuvres as building blocks. This enables us to always design correct manoeuvres, that is manoeuvres that meet the given specifications (which may include safety and ensured results), even in the presence of disturbances. From the operator's perspective, this means having at his disposal a set of commands with which a complex mission can be planned and executed. The set of commands is designed to comply with the operational requirements while ensuring proper termination, or adequate fault handling. For the purpose of modularity, tele-operation is defined as an elemental manoeuvre. The other manoeuvres are tele-programming primitives.

Definition 1 [Elemental manoeuvre]. *Prototypical solution to a class of ROV motion problems that cannot be obtained from the composition of other manoeuvres. It is characterized by:*

- *Objective.*
- *Hard and soft constraints.*
- *Information sets.*
- *Dynamic model and controllers.*

Figure 3 and Table 2 sketches the basic structural ingredients.

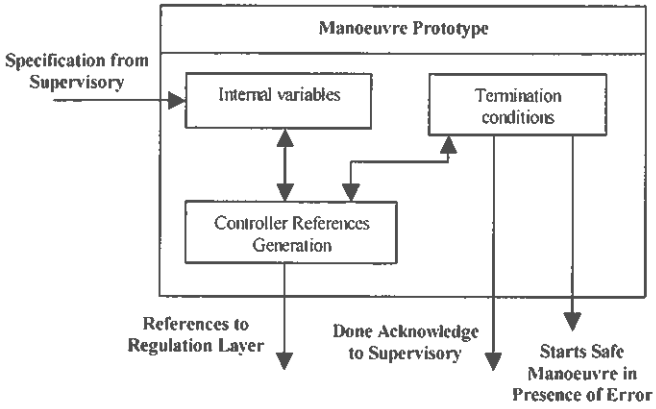

Figure 3: Manoeuvre prototype

The design of elemental manoeuvres is incremental. First, each elemental manoeuvre is developed and verified independently. Then, the more complex manoeuvres are formed and tested. For each elemental manoeuvre, we first synthesise a least restrictive controller – one that provides us with "maximal sets", or "windows" of possible actions. When safety or final objectives are not at stake, we are free to select one of the controls. This is convenient, since in controls, precision is costly, yet the ocean environment cannot be quantified exactly. To handle safety and performance issues, we then complement the design with rules or controllers that incorporate some logic. Our controllers ensure that, if feasible, the manoeuvre's objective is indeed attained. Hence, the control architecture is fault-tolerant since it uses correct elemental manoeuvres.

The overall concept of operation is better explained with recourse to Figure 4. The operator uses the console to command the execution of manoeuvres. The ROV computer runs a supervisor that accepts (or rejects) commands from the console and controls their execution. Basically, there are three modes for the supervisor: executing command, idle and error. By default, and in the absence of any command sent by the operator, the supervisor is in the idle mode that instantiates a special manoeuvre - the safe manoeuvre. Upon receiving a command from the console, the supervisor either starts its execution, or does not accept the command. Command execution either terminates as programmed, or is aborted when an error occurs. In this case, the transition to the error mode invokes error-handling procedures.

Specification	
Feature	Description
Inputs	Type of manoeuvre. Parameters of the manoeuvre. Parameters include acceleration profiles, time-outs, final position, etc.. User commands accepted during the execution.
Outputs	References to regulation layer. Termination and error events..
Internal Variables	Used to evaluate termination conditions.
Termination Conditions	Specification of the normal and error conditions for the termination of a manoeuvre.
Controller	Algorithms for trajectory generation and control.

Table 2: Manoeuvre specification

This control framework is based on recent developments in distributed hybrid systems. Informally, a distributed hybrid system is a collection of dynamic systems – each of which includes both continuous time activities and discrete-event features – that interact through the exchange of data and messages. Figure 5 describes some of those interactions.

4.2 Architecture

The control architecture addresses the overall structure and the global properties of control systems, hence providing a focus for certain aspects of design and development that are not appropriately addressed within the constituent modules (see [6] for a discussion on software architectures). This structure is formalized in terms

of layers and interfaces [7]. We consider a tri-level control architecture (Figure 4).

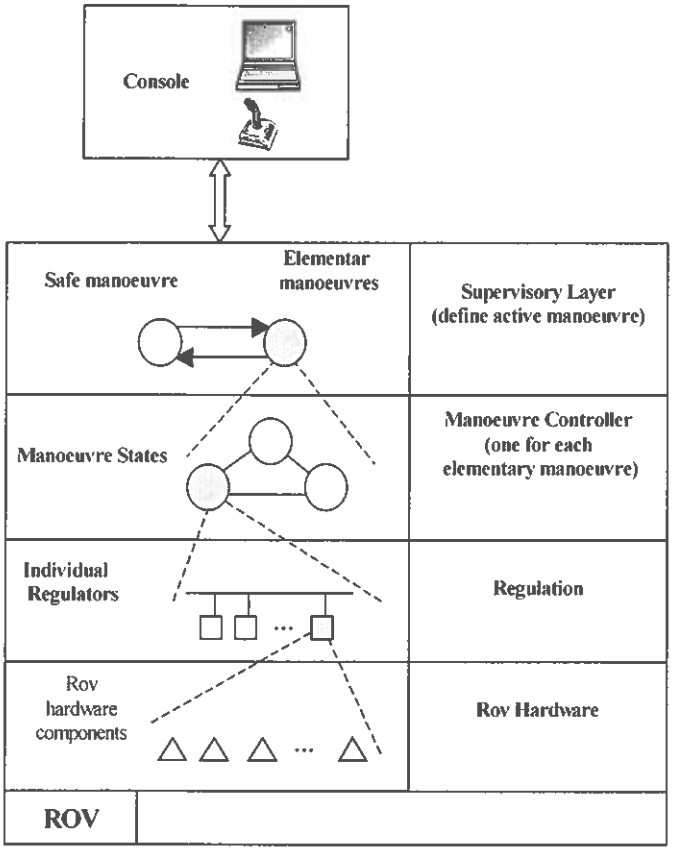

Figure 4: Concepts of operation

Regulation Layer - the automated vehicles. The dynamical models of the vehicles are given in terms of non-linear ordinary differential equations. This level deals with continuous signals, and interfaces directly with the vehicle hardware. Control laws are given as vehicle state or observation feedback policies for controlling the vehicle dynamics. Control laws at this level correspond to low level commands such as course keeping, turning, etc…

Manoeuvre Layer - control and observation subsystems responsible for safe execution of manoeuvres - the first level of automation. The supervisory layer commands the execution of elemental manoeuvres according to some motion plan. Interactions with the regulation layer are mediated by the elemental manoeuvres. Each elementary manoeuvre sends low-level commands to the regulation layer and receives events concerning their completion or failure. Elemental manoeuvre control is given in terms of hybrid automata. The current design uses protocols in the form of finite state machines.

Supervisory Layer – controls and coordinates the execution of manoeuvres.

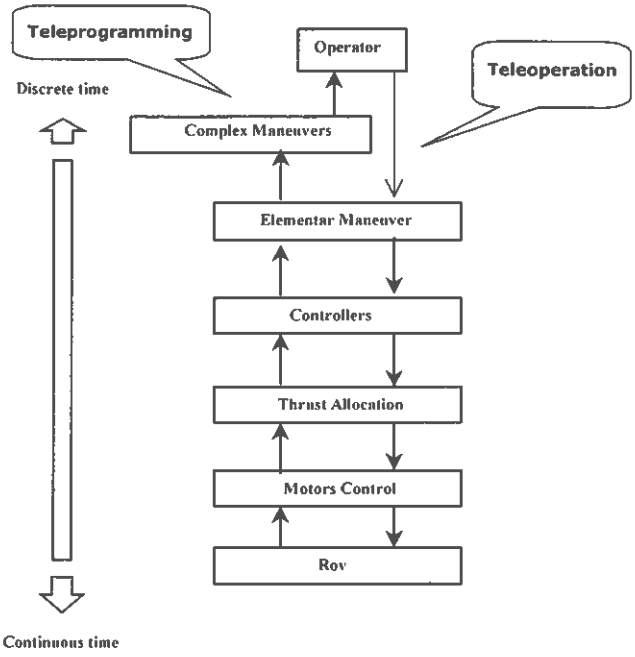

Figure 5: Control interactions

4.3 List of manoeuvres

The Manoeuvre class aggregates the common attributes and methods of all manoeuvres. Specific manoeuvres inherit all these attributes and methods and contain additional parameters or methods. A brief description of all of the elementary manoeuvres follows:

Hovering. Hover inside some prescribed boundaries while maintaining ROV orientation. Examples: inspection of a target with high precision.

Plane. Stabilize with respect to a vertical or a horizontal plane with bounded errors. Examples: inspect a wall while moving in a vertical plane parallel to it.

Orientation. Change ROV orientation in terms of its vertical axis. Examples: change ROV orientation in order to track some visual target.

Trajectory. Follow a prescribed trajectory, with bounded errors. Examples: inspect a wall in a yo-yo motion with a specific time parameterisation.

Path. Follow a prescribed path, with bounded errors. Examples: follow a prescribed path inspection pattern while allowing the operator to decide on the speed of the ROV.

TrackTarget. Track a target acquired with the visual system within bounded error. Examples: inspection a moving object.

FollowWall. Follow a prescribed wall, with bounded errors. Examples: inspection of a surface with unknown shape.

ManualOperation. Teleoperation. Assists the operator to drive the ROV from the GUI-based operator's console. This manoeuvre accepts operator commands during manoeuvre execution. The operator commands are filtered through a low-level control system that stabilizes the motion of platform. This fly-by-wire capability makes it possible for a novice operator with no special training to control the ROV with minimal effort. This feature is not available in commercial systems.

4.4 Trajectory generation

A trajectory generation module is included in the manoeuvre layer in order to obtain a good interface between the latter and the regulation layer (Figure 3). Each manoeuvre object defines desired values for the system state to complete a specific objective. These desired values should not be sent directly to the regulation layer in order to obtain better responses from the controllers. The role of the trajectory generation module is to process the desired values for the system and send reference values to regulators. The proposed trajectory generation architecture for the present framework is based in a model of two degrees of freedom. This model has two project components: a component that specifies a full trajectory for the system states and a component that is able to define the most suitable controllers to deal with uncertainty.

The two degrees of freedom model uses all the information available of the system resulting in a better performance than the one degree of freedom method, which makes references only for few system states. The two degree of freedom architecture [12,14] is depicted in Figure 6.

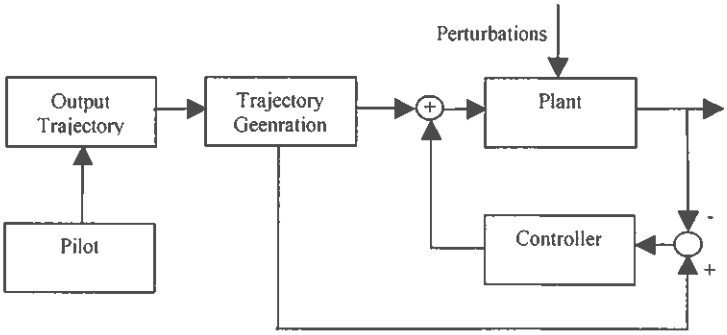

Figure 6: Two degree of freedom architecture

The model of the IES vehicle is a system differentially flat, which enables some good trajectory generation features.

Definition 2 [Differential flatness][12] *A system is said to be differentially flat if it is possible to define variables whose dimension is equal to the number of the inputs, such that all the system states and its inputs can be defined from these variables and its derivatives. Such variables are called flat outputs. The formulation of this problem is as follows:*

$$\begin{aligned} z &= \psi(x, u, u^{(1)}, \ldots, u^{(l)}) \\ (x, u) &= \phi(z, z^{(1)}, \ldots, z^{(l)}) \end{aligned} \qquad (1)$$

As said previously, the model of the ROV [13] is a system differential flat and is given by:

$$M_x(x)\ddot{x} + C_x(v,x)\dot{x} + D_x(v,x)\dot{x} + g_x(x) = \tau_x \qquad (2)$$

where x represents the position and the orientation of the ROV in the inertial frame, v is the velocity of the vehicle in the body fixed frame and represents the system input.

The flat outputs of this system coincide with its state. In this way, it is possible to define a trajectory of the system state enabling the determination of the inputs that implements the trajectory. The inputs are obtained by substituting the flat outputs (or in this case the system state) in the equation of the model (Equation 2). It is required that the flat outputs have at least the second order derivatives. The differentially flat systems allow a suitable way to generate trajectories for the system because, it

allows the definition of the desired trajectory in the space of the outputs and then easily both entire state and the input of the system become defined.

It should be noticed that the trajectory generation has requisites of real-time execution. This happens because, in the applications of the IES project, there is a pilot defining the desire outputs with some rate. For that reason the trajectory generation must be computed at the same rate as the inputs are defined. Despite this requisite, the trajectory generation is made several orders lower than the controllers' rate, allowing the realization of some complex computations. Due to the necessity of reducing computational times for trajectory generation an efficient method will be presented next.

Problem 1 [Point-to-point steering]. *Point-to-point steering problem consists in finding a trajectory between two points in the state space ($x(t_0)=x_0$ and $x(t_1)=x_1$).*

This class of problems is irrelevant if the flat outputs are the variables that are intended to track because, for this problem, the initial and final state for two distinct time instants are defined. If the inputs and its derivates are also defined for the instants t_0 and t_1, it is possible to compute the flat outputs at these instants. This happens because these variables are a function of the system state and its inputs and derivatives (Equation 1). After flat outputs determination at initial and final instants it is possible to make a parameterization of these variables using a base of polynomials functions. The choice of this kind of function is made to facilitate the computation of derivation of the parameterized function. The parameterization results in the following equation:

$$z_i(t) = \sum_j A_{ij}\phi_j(t) \qquad (3)$$

where ϕ represents a base of functions. A linear combination of these functions, whose coefficients are given by matrix A, allows the determination of the flat outputs function. To get the values of the elements of matrix A is necessary to solve an equations system presented next:

$$\begin{aligned} z_i(t_0) &= \sum_j A_{ij}\phi_j(t_0) & z_i(t_1) &= \sum_j A_{ij}\phi_j(t_1) \\ &\cdots & &\cdots \\ z_i^{(l)}(t_0) &= \sum_j A_{ij}\phi_j^{(l)}(t_0) & z_i^{(l)}(t_1) &= \sum_j A_{ij}\phi_j^{(l)}(t_1) \end{aligned} \qquad (4)$$

This equations system has solution if the dimension of function basis is sufficient, i.e. $j \in [1..2(l+1)m]$. It is possible to overparameterize the flat outputs to achieve extra degrees of freedom that will allow the minimization of some cost function of interest. The trajectory generation problem is reduced to finding the coefficients of matrix A. After determination of matrix A it is necessary to compute a trajectory for the states and inputs from the flat outputs parameterization. It is necessary to compute a sufficient number of points for this trajectory to achieve a suitable accuracy in the system response. The choice of the number of points is a trade off between performance and calculation time. Increasing the number of points, better the accuracy is and bigger the computational time. The response of the system becomes better for large number of points because suitable input is applied more frequently in the system, enabling a more accurate tracking of the defined trajectory. The computation of states and the inputs of the system is made from flat outputs parameterization by solving a non-linear equations system. In ROV's model case the states determination is trivial and the inputs are determined using the Equation 2.

The discussion presented above leads with the following algorithm:

Algorithm 1 [Point-to-point steering][12] *Given the initial and final states, $x(t_0)=x_0$ and $x(t_1)=x_1$, and the system input and its derivatives at the same instants, the following steps should be performed:*

1. *Determination of the flat outputs at initial and final instants (Equation 1);*
2. *Parameterization of the flats outputs between the initial and final instants (Equation 4);*
3. *Computation of N points in the flat outputs parameterization;*
4. *Determination of the state and input system from the flat output points.*

Figure 7 shows the necessary steps for this algorithm.

The computation time for this trajectory generation algorithm is the sum of the time required to perform the parameterization and the time necessary to compute the points in the trajectory. Usually, computation of flat output parameterization takes less time than the trajectory *(x,u)* computation.

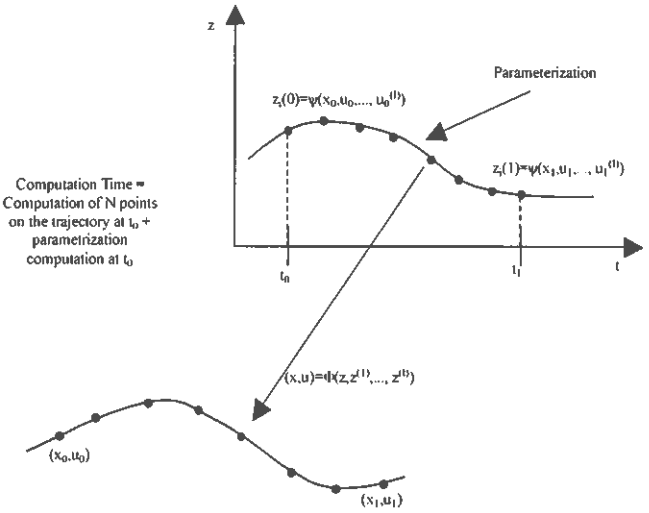

Figure 7: Point-to-point steering problem

The use of the method described previously allows a suitable way to generate trajectories for each manoeuvre.

4.5 Control design

Here we discuss control design [11] for this architecture. For reasons explained before we will concentrate this discussion on control synthesis for the manoeuvre layer. The generic problem of elemental manoeuvre control synthesis can be described as follows: given a dynamic system, or a collection of interacting dynamic systems, synthesize a controller so that the system(s) satisfies(y) the manoeuvre specifications. The type of manoeuvre specification dictates the type of control formulation. Examples include:

1. The problem of invariance - staying inside some region [10];
2. The problem of attaining a given target set while the trajectories of the system remain inside some other set;
3. The problem of optimizing some criteria;
4. The problem of stabilizing a system.

Inherent to most of these control formulations is the problem of reach set computation -- the set of all positions that the ROV can reach from a given starting position [9]. In fact, given the reach set, it is quite simple to solve most of these problems.

5. Conclusions

This paper reports a framework for the automation of a ROV. The novelty of our approach stems from the consideration of systems engineering principles and of distributed hybrid systems techniques. The innovations are the introduction of safe elemental manoeuvres in a three level coordination and control hierarchy, in order to manage complexity. We formalize the notion of elemental manoeuvre and synthesize safe elemental manoeuvres using techniques from hybrid systems, differential games and viability theory.

Future work includes implementing a full set of safe elemental manoeuvres and enriching the manoeuvre switching logic to accommodate faults and automated planning.

Acknowledgements

The authors thank Prof. Anthony Healey for his comments and suggestions. This material is based upon work funded by the Programa PRAXIS XXI - Medida 3.1b) (IES project) and by Ministério da Defesa, Portugal.

References

[1] IEEE (1999). "IEEE standard for application and management of the systems engineering process." IEEE.

[2] A. Matos, A. Martins, N. Cruz, F. Pereira, "Development and implementation of a low-cost LBL navigation system for an AUV",

MTS/IEEE Oceans '99, Seattle, U.S.A, September 1999.

[3] N. Cruz,, L. Madureira, A. Matos and F. Pereira, "A versatile acoustic beacon for navigation and remote tracking of multiple underwater vehicles", MTS/IEEE Oceans '01, Honolulu, HI, U.S.A, November 2001.

[4] Jan van Leeuwen editor (1990). "Handbook of theoretical computer science". Elsevier.

[5] P. Varaiya and T. Simsek and J. Borges de Sousa (2001). "Communication and control in hybrid systems". Accepted as a tutorial session for ACC 2001.

[6] David Garlan. "Research directions in software architecture". ACM Computing Surveys (1995). Volume 27, number 2, pages 257-261.

[7] J. Borges de Sousa and F. Lobo Pereira and E. Pereira Silva (1995). "A dynamically configurable control architecture for autonomous mobile robots". Proceedings of the 34th IEEE Conference on Decision and Control.

[8] G. Booch, J. Rumbaugh, I. Jacobson. "The Unified Modelling Language User Guide". Addison-Wesley (1999).

[9] A. B. Kurzhanskii and P. Varaiya. "Ellipsoidal techniques for reachability analysis". Computation and control. N. Lynch and B. Krogh. Series "Lecture Notes in Computer Science". Springer-Verlag", (2000). Pages 202-214.

[10] Jean-Pierre Aubin, "Viability theory", Birkhauser (1991).

[11] M. Vidyasagar. "Nonlinear Systems Analysis", Pretice-Hall, Englewood Cliffs, NJ.

[12] Michiel J. van Nieuwstadt, Richard Murray. "Approximate trajectory generation for differentially flat systems with zero dynamics", In *IEEE Conference on Decision and Control, New Orleans.* (1995).

[13] Thor I. Fossen. "Guidance and Control of Ocean Vehicles", John Wiley and Sons. (1995).

[14] Sérgio L. Fraga, J. Borges Sousa, Anouck Girard, A. Martins. "An Automated Manoeuvre Control Framework for a Remotely Operated Vehicle", In *MTS/IEEE Oceans'01, Honolulu, HI, U.S.A.* (2001).