UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Cooperation Among Robots: A Behavioural Approach Supported On Group Theory

João Fernando Cardoso Silva Sequeira
(Mestre)

Dissertação para a obtenção do Grau de Doutor em
Engenharia Electrotécnica e de Computadores

Janeiro de 1999

Tese realizada sob a supervisão de

Doutora Eng$^{\underline{a}}$ Maria Isabel Lobato de Faria Ribeiro

Professora Associada do

Departamento de Engenharia Electrotécnica e de Computadores

INSTITUTO SUPERIOR TÉCNICO

[To feel is to create] (excerpt).

To feel is to think without ideas, and therefore, feeling is understanding, given that the Universe has no ideas.

Fernando Pessoa

Acknowledgements

While working in this thesis a multitude of experiences enriched my life. Many people constructively influenced this work, either directly or indirectly. I will only name a few, but I am deeply grateful to them all.

My familly and close friends gave me the necessary support during this period.

Prof. M. Isabel Ribeiro, the thesis supervisor, provided constant encouragement and advice during this working period.

The early ideas present in this thesis, together with an important part of the experimental work were developed at the Sensor Based Applications Unit of the Institute for Systems, Informatics and Safety of the European Commission Joint Research Centre, in Ispra, Italy. Dr. João Gonçalves, Head of the Sensor Based Applications Sector, of the Reliable Information Technologies Unit, provided all the necessary support during the two periods I stayed at the JRC. Many fruitful discussions contributed to make my visits to the JRC intense periods of my life that I will never forget.

I had some tough discussions with Dr. José del Millan Ruiz, senior researcher at JRC, that resulted in valuable ideas.

Prof. Michael Athans, Professor of Electrical Engineering (Emeritus) at MIT and Visiting Scientist at ISR/IST, introduced me to Viability Theory, where I found some important connections with the thesis subject.

Prof. Maria João Rendas, from Laboratoire I3S - CNRS, France, read some drafts of the work and provided valuable criticisms and suggestions.

On several ocasions, Prof. Rui Loja Fernandes, from the Department of Mathematics of IST, advised me on algebraic structures.

Vítor Sequeira and Pedro Cunha were true friends. While in Italy, both were “victims” of my passion about cars.

Several institutions provided generous contributions to this work, granting the facilities and the financial support for the participation in international conferences and visits to research institutions.

Instituto Superior Técnico and Instituto de Sistemas e Robótica provided a place to think. A major part of the experimental work was also developed using the facilities at the ISR Mobile Robotics Laboratory. Prof. João Sentieiro, Director of the Lisbon pole of the Instituto de Sistemas e Robótica, was always receptive to any needs related with this work.

The Joint Research Centre of the European Commission in Ispra, Italy, through Mr. Fernand Sorel, Head of the Reliable Information Technologies Unit, of the JRC's Institute for Systems, Informatics and Safety, provided me with all the kind attention during the two periods at the JRC.

The INVOTAN commitee, in Portugal, granted financial support for my longest stay in Italy. Their assistance on several details was very much appreciated.

The PRODEP program and the Gulbenkian Foundation provided the support for attending several enlightening conferences and summer schools in France and Italy. Part of the expenses of my first period with the JRC were also supported by this program.

The Project NATO-Science for stability, PO-ROBOT, shares some of the ideas developed in the thesis for the control of mobile platforms and gracefully supported the participation in some international conferences.

The last part of this work was financily supported by Project PRAXIS/2/2.1/TPAR/2087/95 - COOPERA. Part of the future work referred in the thesis is already scheduled within this project.

Resumo

Esta tese apresenta um enquadramento formal para o controlo, baseado em comportamentos, de conjuntos de robots heterogéneos, actuando em cooperação. Do referido enquadramento resulta um conjunto de blocos fundamentais constituintes de uma arquitectura de controlo.

O controlo por comportamentos tem a sua principal motivação na complexidade do controlo de sistemas com um número elevado de graus de liberdade, característico de robots antropomórficos, bem como de conjuntos de robots onde seja possível identificar relações de interdependencia entre os seus elementos. A complexidade do controlo de robots com as características de locomoção e manipulação dos seres humanos continua a motivar a procura e desenvolvimento de abordagens alternativas. Numa perspectiva de aplicação de robots em ambientes reais é desejável a existência de uma interface capaz de simplificar a interacção com os utilizadores.

No que respeita à interface entre seres humanos e robots, o trabalho constante desta tese propõe o uso de máquinas de estados finitas, por forma a manter uma característica de linguagem de programação. Relativamente ao controlo, o enquadramento propõe o uso de um espaço de estados em que cada elemento contem informação relativa à intenção de movimento do robot, expressa num espaço dito de *acções*, bem como às respectivas condições iniciais, e suporta a eventual existência de incertezas no processo de programação utilizado pela interface. Cada estado representa, simultaneamente, múltiplas trajectórias (eventualmente em número infinito) no espaço de configuração do robot.

As características inerentes ao movimento de um robot enquanto sistema físico permitem uma abordagem constructiva no desenvolvimento do enquadramento, conduzindo de forma natural a uma estrutura de grupo para o espaço de acções. A extensão para uma estrutura de semi-anel não próprio conduz à simplificação de aspectos de implementação. A extensão ao controlo de múltiplos robots considera uma abordagem descentralizada suportada na controlabilidade do conjunto quando encarada na perspectiva do enquadramento proposto.

Um conjunto de resultados experimentais, obtidos em testes reais e de simulação e considerando robots singulares (plataformas móveis) e sistemas com múltiplos robots (manipuladores móveis) permite uma avaliação do desempenho da estratégia de controlo por comportamentos, suportada no enquadramento proposto. As experiências apresentadas ilustram a cooperação entre robots na prevenção de colisões entre plataformas móveis e no transporte de uma barra por dois manipuladores móveis.

Palavras chave: Cooperação, Estrutura de grupo, Comportamentos, Controlo por comportamentos, Robótica móvel, Agente.

Abstract

This thesis presents a novel framework to model the behavioural control of multiple heterogeneous robots acting cooperatively. The basis building blocks for a control architecture are identified within the framework.

The behavioural approach is motivated by the complexity of the control of systems with multiple degrees of freedom. This is a characteristic of anthropomorphic robots and of multiple robot systems operating simultaneously. Furthermore, from a perspective of application to real world problems, the simplicity of operation of these complex robotic systems requires a user friendly interface and a control approach able to filter spurious information that may disturb the operation of the system.

In what concerns the interface between humans and robots, the framework presented in the thesis relies on a programming approach, based on Finite State Machines. As for the control, the framework proposes a state space that copes with the existence of uncertainty in the programming process, this representing an innovative contribution of the thesis. Each element of the state space contains information about the current motion trend of the robot, included in the so called *actions space*, and the corresponding initial conditions. Moreover, each state represents a large (eventually infinite) number of possible trajectories in the robot's configuration space, this being the main tool to cope with the aforementioned uncertainty.

The physical nature of the robot control problem allows a constructive approach in the development of the framework that is shown to induce a group structure on the aforementioned actions space. An additional operation extends the group structure to a non-proper semiring structure that proves useful in the computational implementation of the framework. The extension of the control of a single robot system to a multiple robot system acting cooperatively is made by a decentralized approach, supported on a controllability argument.

The proposed behavioural framework is assessed in a series of experiments involving single robots (mobile platforms) and multiple robot systems (mobile platforms and manipulators). Real and simulated robots were used in the experiments. Single robot experiments test the emergence of collision avoidance behaviours. Multiple robot experiments consider the control of a single mobile manipulator (mobile platform carrying a manipulator) and of two mobile manipulators carrying a large bar. These experiments illustrate the cooperation between the manipulators and their respective mobile platforms, and between the two mobile manipulators in relevant operations.

Keywords: Cooperation, Group structure, Behaviours, Behavioural control, Mobile robotics, Agent.

Contents

Chapter 1

Introduction

1.1 Motivation

The study of cooperation between heterogeneous entities, e.g, humans, mechanical robots or software agents, is a research area with multidisciplinary characteristics encompassing a large number of scientific areas, from the human and social sciences to the exact sciences. This was already recognized by authors such as Brocket, [Brocket, 1993]. Figure 1.1 presents a schematic representation of such distinct areas with a common intersection named as *behaviours*[1]. Within the Psychology, Maslow's Theory of Motivation considers human behaviours as outcomes of a hierarchy of necessities, [Klausmeier, 1977]. As for Engineering, multiple definitions of the *behaviour* concept can also be given, the diversity due to the different formal systems supporting the research in each of its scientific areas (see [Willems, 1991, Albus and Meystel, 1996, Brooks, 1986a] for some examples). A common point in this diversity is the fact that a *behaviour* is an outcome of some experiment, and, in the particular case of robotics, this outcome has a counterpart in the motion of the robot(s) in some physical workspace. *Behaviour* based paradigms in robotics represent an attempt to explore a connection between Engineering sciences and Psychology. Without advocating formal identifications between the two so distinct scientific areas, this thesis bears motivation from everyday life problems where anthropomorphic characteristics are present to define a framework for the cooperation among a set of robots.

The motivation is oriented towards the fulfillment of the objectives described bellow.

- Definition of a robot programming scheme to perform the interface with end users. This should be independent of the kinematics to maintain the complexity of operation within acceptable limits.

- Definition of a control architecture to support the programming of the robot. This should

[1]Throughout the thesis the word behaviour will be italicized when in the robotics context. Otherwise, the current character font is used.

encapsulate the functionalities that depend on the robot's kinematics into building blocks not accessible through programming.

These two objectives can be identified with two levels of control. The narrow level, identified with the first objective, encompasses everything that requires user support, namely the specification of some of the necessary functionalities for robot control. The contents of the narrow level may be changed, depending of the objectives assigned to the robot by the user. The wide level, identified with the second objective, encompasses the user independent aspects, namely the definition of the basis building blocks of the control architecture necessary to the operation of the robot. The contents of the wide level are common to a wide class of problems and hence, once defined they are kept unchanged. Throughout the thesis, these two objectives will be jointly considered and named as the *user's perspective* of the *behavioural* control.

This thesis focuses on the robotics field, exploring a *behaviour* concept aiming at defining a methodology for robot control using the aforementioned user's perspective. A novel framework, able to model concepts with an intrinsic degree of uncertainty such as the *behaviour* concept, is used to define a set of basis building blocks of a control architecture for single heterogeneous robotic devices. The extension to the multiple robot case, acting in cooperation, is also derived. The performance is assessed through the results obtained in a set of missions using mobile manipulators operating either isolated or cooperating in the transportation of a bar.

Any discussion on the merits/demerits of *behavioural* approaches must consider the reactiveness issue. The degree of combination between reactive and non-reactive approaches is a subject of difficult evaluation that depends mainly on the particular problem under study and on the kind of information available to make decisions. Reactive architectures are in general identified with fast responses in dynamic environments and have been associated with *behaviours*, [Brooks, 1986b]. Non-reactive architectures often embed some kind of conceptual hierarchy that mirrors the well known paradigm of "decreasing precision to increasing intelligence" proposed by Saridis, [Valavanis and Saridis, 1991].

The extrapolation of concepts from biological and social sciences to robotics relies, to some extent, in the use of systems able to communicate through symbols. The assignment of a meaning to a

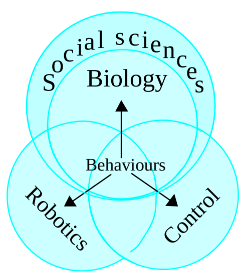


Figure 1.1: Interdisciplinary *behavioural* relationships.

symbol ("symbol grounding" in Artificial Intelligence terminology) seems very much dependent on the particular problem being solved. Emphasizing, to some extent, the importance of symbolic processing techniques, some of the robot path planning techniques have been recognized as over-simplifications of the more general problem of defining a goal for single or multiple robot systems, [Brocket, 1993]. Hybrid systems (i.e., systems with continuous dynamics depending also on one or more discrete index(es)) seem promising in what concerns symbol grounding as they provide, *per se*, discrete variables to which meanings can be associated. Moreover, hybrid systems are common in the robotics field. For instance, complex interactions between robots and environment may require the use of multiple models, each one valid within particular conditions.

Regardless of the nature of the models to be considered (continuous, discrete or hybrid), the synthesis of the framework to describe *behaviours* begins with the choice of suitable mathematical structures to model the phenomena involved. At the core of the *behaviour* paradigm developed in this thesis are the mathematical concepts of *set* and *group*. *Sets* are the mathematical objects at the very basis of the paradigm as, by definition, they are able to encapsulate abstract elements. *Groups* endow sets with ability to "jump" among its elements, through the corresponding group operation and thus support a basis requirement of any robotics system: motion generation.

The methodology followed in the development of the framework presented in this thesis is adapted from the general problem of robot control, in the perspective of dynamic systems. Robot control is a problem in the area of non-linear control where the theory of Lie groups and Lie algebras is extensively used (recall that a Lie group is a group structure on a smooth manifold, [Olver, 1993], and that the configuration space of a robot is a smooth manifold, [Latombe, 1992]). For example, the set of all vector fields defining motion directions in a smooth manifold (i.e., the manifold's tangent space) has an algebra structure under the Lie bracket operation. Using the geometrical interpretation of the Lie bracket as a limiting motion direction, the dynamic system control problem can be interpreted as the choice of a suitable vector field from a set of Lie bracket combinations among a set of vector fields on the robot's configuration space.

1.2 Overview of the research on robot control, *behaviours* and cooperation

The concept of *behaviour* can be traced back to the communications field, embedded in the basis ideas of adaptation and error recovery proposed and/or implemented for a long time, namely in the use of telephonic line concentrators with a capacity lower than the maximum number of telephones, in the assignment of priorities in systems handling queues and in the use of adaptive algorithms for packet routing over communication networks (for instance using stochastic learning automata, [Narendra and Thathachar, 1989], Chapter 9).

As widely acknowledged by several authors, the control of a single robotic device is still a challenging problem due to difficulty in the adaptation of the control schemes to the variety of environments arising in the real world, [Antsaklis et al., 1994, Košecká et al., 1997, Ram et al., 1997]. The Sojourner vehicle used in the NASA's Mars Path Finder mission is possibly the most recent example to illustrate these difficulties. Other examples can be found in the class of anthropomorphic robotic devices, like mobile manipulators, due to the large number of degrees of freedom involved and the constraints imposed by the kinematics, sensors and actuators. Even for single robots, physical limits are often not considered in the mathematical models, due to the increase of complexity they bring to the analysis methods. Such is the case in mobile platforms with the limitation in the linear and angular velocities and in robotic manipulators with limitations in the joints values imposed by mechanical constraints. A practical example is given by the feedback controller developed in [Sordalen, 1993] for a nonholonomic mobile platform. This controller, based uniquely in the kinematics of the vehicle, often generates linear and angular velocities with values above the maximum allowed for some of the presently manufactured mobile platforms. The scaling of those values can not be used systematically since some platforms have minimum values for the controls and tend to behave poorly when near to minimum values are used, due to precision problems arising with the specific electronics design (see Appendix D for a particular example). In the sequel, a brief overview of important works related with the control of single and multiple robots is presented.

Albus proposed an axiomatic theory of intelligent machines (using five axioms), named Real-time Control System architecture (RCS), [Albus, 1996]. Each of the levels in the hierarchical control architecture derived from the axiomatics is formed by five types of objects: *behaviour generation*, *knowledge databases*, *world modeling*, *sensory perception* and *value judgement* modules, implemented as Finite State Machines.

Based in the RCS axioms, Meystel and Albus, in [Albus and Meystel, 1996], consider the emergence of the phenomenon of intelligence as an outcome of the application of three operators, named *Grouping*, *Focusing Attention* and *Combinatorial Search*. These operators are the components of an entity, named Generalized Subsistence Machine (GSM), allegedly the minimal intelligence unit in the system. Three additional postulates on GSMs, defining what can be done with them, complete the framework.

Saridis proposed the three level hierarchy, *Organization, Coordination, Execution*, as the structural model for an intelligent machine. For each of the three levels, different implementation principles can be used, namely, neural networks for the *Organization*, Petri nets for the *Coordination* level, with the *Execution* level containing all the required hardware. Entropy is used as a control measure to integrate the information in the three levels. See [Saridis, 1996] for a review of this approach.

Yamamoto and Yun, in [Yamamoto and Yun, 1993], considered a cart mobile platform (see Appendix C) and a holonomic manipulator, with three revolution joints. The task assigned to the system is defined as having the manipulator end-effector following an arbitrary trajectory. Due to the holonomic nature of the manipulator, the task trajectory can be easily mapped into an adequated trajectory for the manipulator's base. The manipulability measured is maximized to determine the preferred configuration for the manipulator, such that, for each position of the end-effector, the preferred position for the manipulator's base is used as a reference for the mobile platform to follow. The nonholonomic nature of the cart implies that not all trajectories can be followed accurately. Therefore, the manipulator is kept in the preferred configuration provided that the mobile platform is able to follow the desired trajectory.

In robotics, as recognized in [Brocket, 1993], "purposeful motion involves some interaction with physical objects having uncertain positions and characteristics". Thus, motion control must consider the compliance between mechanisms. Brocket proposed formal models of hybrid systems, i.e., dynamical systems involving both continuous and discrete variables, as general tools to generate compliance. Brocket's framework introduces the idea of *instruction set* as the tool to generate a "time" variable to command the evolution of the systems (the equivalent of an *event* in systems modeled by Discrete Event Systems). Dynamical systems using gain scheduling controllers can be modelled using this idea.

Willems developed a behavioural approach to the theory of dynamical systems [Willems, 1991], in which a *behaviour* is defined by the trajectories produced by a model $(\mathbb{U}, \mathfrak{B})^2$, with $\mathbb{U}$ the set of all the variables involved and $\mathfrak{B} = \{u \in \mathbb{U} : f_1(u) = f_2(u)\}$ represents the so called *behavioural equations*, for some mappings f_1, f_2 (inequalities can also be considered within this framework). Every model, $M \in \mathbb{M}$, generating a pair $(\mathbb{U}, \mathfrak{B})$, can be described in terms of a mapping $\pi : \mathbb{P} \mapsto \mathbb{M}$, where $\mathbb{P}$ is a parameter space, e.g., the space of matrices or the space of polynomials. Equivalent models, i.e., models generating identical *behaviours*, are determined by an equivalence relation in the parameter space defined by $\{p_1 \hat{\mathcal{E}} p_2\} \Leftrightarrow \{\pi(p_1) = \pi(p_2)\}$. The corresponding equivalence classes can be computed using the notion of transformation group of the parameter space (see Appendix A.7). Consider the transformation group on $\mathbb{P}$ denoted by $\Theta = (S_g, g \in \Theta)$, where S_g is a bijection on $\mathbb{P}$. The orbit of Θ at $p \in \mathbb{P}$ (see [Olver, 1993], pp. 22-23) is the set $\mathbb{P}' = \{p' : p' = S_g(p), g \in \Theta\}$. If the equivalence relation induced by Θ coincides with $\hat{\mathcal{E}}$ then the elements in the same orbit generate identical models. For example, consider the dynamical systems family described by $\tilde{\Sigma} = (\mathbb{T}, \mathbb{W}, \mathfrak{B})$, where $\mathbb{T} = \mathbb{Z}$ is the time axis, $\mathbb{W}$ is the space where the dynamical systems have values and $\mathfrak{B}$ some *behaviour*. Consider the group $(\mathbb{Z}, +)$ and σ^t the backward shift operator. The backward shift operator defines the transformation group of backward time shifts on dynamical systems. The orbit defined by $(\mathbb{Z}, \mathbb{W}, \mathfrak{B}) = (\mathbb{Z}, \mathbb{W}, \sigma^t \mathfrak{B})$ represents the class of time invariant dynamical systems.

Further classical methodologies, for single and multiple robots, can be found in [Sheu and Xue, 1993] and [Latombe, 1992].

[2]The notation was kept as close as possible to the original.

Brooks, in [Brooks, 1986a], proposed the so called *subsumption architecture* for the control of devices with multiple sensors and actuators using a net of functions, each implementing what was called a *behaviour*. The hardwiring between the elements of this net defines complex *behaviours*. The *subsumption architecture* evolved to a programming paradigm widely accepted by some research groups with emphasis on those working in underwater robotics (see [Valavanis et al., 1997] for a brief survey of the current architectures for the control of underwater vehicles). The *subsumption architecture* was able to illustrate the emergence of rational individual and group macroscopic *behaviours* in a set of robots, each of which controlled by low complexity algorithms (the so called robotic insects, in analogy with the living insects). The interactions between the robots, representing a form of competition, lead to the emergence of appropriate individual and/or group *behaviours* thus being also identified with cooperation. To the author's knowledge, Brooks' work was the first to recognize the importance of studying a control methodology for the multiple robot case not based in any form of planning, i.e., using no explicit world representation.

Environmental constraints may force the use of multiple robot controllers and the definition of the conditions of switching among them, i.e., a general model for a robot and the associated control scheme must be of hybrid type, with continuous and discrete systems interacting with each other (see [Brocket, 1993] for a number of classes of hybrid dynamical systems of importance in robotics). Hybrid systems naturally lead to control approaches supported in the Discrete Event Systems (DES) theory. Antsaklis *et al* in [Antsaklis et al., 1994] claimed that conventional control theory cannot address the design of controllers for complex dynamical systems. These authors studied the control of hybrid systems of the type *Controller* → *Interface* → *Plant*, with the *Controller* being implemented by a deterministic automaton and the *Interface* performing the transformations between discrete and continuous signals. The whole system (controller, interface and plant) behaves like a discrete event system so that standard results of DES theory (see [Ramadge and Wonham, 1987]) yield a controller design methodology. The basis underlying idea is to define an alphabet by a set of symbols each of which is identified with an event. Given a desired *behaviour* in the form of a finite string of symbols in the alphabet, the goal is to ensure that the language formed with the alphabet (i.e., the set of all strings, or words) is able to generate the desired string, this being equivalent to the controllability property. DES theory results ensure the existence of a controller leading to the desired *behaviour* provided that the language is controllable.

Biologically inspired control methodologies were analyzed in [Bekey, 1996], through examples with robotic hands, helicopters and colonies of small mobile vehicles. Bekey concluded that the present status of these biologically inspired methodologies is such that no formal theory has been derived from them.

Wark and Arkin, in [Ward and Arkin, 1994], describe a reactive approach to the control of a mobile manipulator. Motor schemes are the basis units to build *behaviours*. Each active motor schema receives data from the appropriate perceptual schema and generates a motion direction using an approach inspired in potential fields, i.e., both the arm and the base are guided using artificial forces generated by the goal and the obstacles. A move-robot schema combines the output of all the active schemes to yield reference pseudo-torques for each joint of the manipulator. These pseudo-torque values are further adapted using damping functions to produce natural, human like, motion. By preshaping the manipulator while the mobile platform moves towards a goal, a cooperation effect is achieved.

Arkin and Balch, in [Arkin and Balch, 1998], presented a reactive *behavioural* approach, using also schemas, to the control of a trash-collecting team of robots.

Reignier *et al*, in [Reignier et al., 1995], used a supervised learning approach, together with a *grow and learn* principle applied to neural networks, to control a single mobile platform.

Millan, in [Millan, 1996], proposed a connectionist reinforcement learning architecture to control a single mobile platform. The connectionist controller (a modular two layer network where each module codifies a set of reaction rules) is used to generalize the past knowledge to the current situation. Whenever the controller fails the generalization, the adequate action is chosen among a set of basis reflexes, hardwired in a subsumption mode. Each module has associated a value that measures how well the module matches the current perceived situation. Actions are computed after the module with the highest value. Sensory situations are registered in the units forming the modules. The output layer uses the total reinforcement expected by each of the modules in the input layer and the cost of performing a particular action to define the weights assigned to each unit of the winning module. The controller generates a prototypical action and a variation on this prototypical action defined by the winning module.

Ahmad and Luo, in [Ahmad and Luo, 1989], describe a hierarchical architecture for the coordination of multiple manipulators for welding. Roughly, the welding problem is solved as the control of an intermediate point of the closed kinematics formed by the robotic devices involved: a seven degree-of-freedom manipulator and a welding table. The inverse kinematics of a six degree-of-freedom manipulator is used to generate the motion of the weld torch which is coordinated with the welding table. The remaining degrees of freedom are used to keep the manipulator outside the low manipulability regions and the welding table far from the physical limits.

Yun, in [Yun, 1993], describes the handling of an object, without grasp, by two manipulators. Using results of the differential geometric control theory, the nonlinear system of equations is decoupled into a linear model and standard linear feedback control is then used. The same problem has been considered in [Liu and Arimoto, 1996] using a distributed control approach, based in the model-based adaptive control.

Çela and Hamam, in [Çela and Hamam, 1992], proposed a solution to the optimal control of multiple robot systems in the presence of obstacles, based in nonlinear programming and a decomposition and coordination procedure of the optimal control problem based in a particular function that measures distances among the robots and between a robot and the obstacles.

Li *et al*, in [Li et al., 1991], modeled a system with two manipulators in cooperation as a closed kinematic chain and formalized the control problem as an optimization problem subject to the dynamic equations, the joint driving torque limits and the maximum internal force/torque that the object the arms are in contact with can support.

Pin *et al*, in [Pin et al., 1994], studied the problem of planning the configurations of a mobile manipulator in such a way that the system may switch between tasks without constraint violations. An example is given by having the manipulator reaching an adequate configuration such that the grasp of an object may take place without violating the torque limit of any joint. Their approach encompasses two steps: forecast the optimal commutation configurations and, given the initial and final configuration for each task, solve the corresponding two-boundary value path planning problem.

Košecká *et al*, in [Košecká et al., 1997], used DES theory to create a framework for the description of *behaviour* based systems. The support argument is the partition of the framework in three components: the *action component*, which associates a number of elementary motion strategies to each actuator, the *perception component*, which associates to each sensor a set of strategies for the acquisition and processing of sensor data, and *the computation component*, comprising any additional computational procedures such as path-planning. Each elementary strategy or computation is considered an elementary unit (also termed *process* or *strategy*) of the framework and is modeled as a finite state machine (FSM). Elementary strategies represent simple *behaviours.* A supervisor (or controller) defines the events that determine the transitions between states of the FSMs. Simultaneously, the supervisor must select the appropriate operators, out of a set of six operators, to use in the composition of the elementary processes. This composition enables the definition of processes of higher complexity than the elementary ones.

Multiple robot systems are also found in highly automated plants (e.g., automotive industry) and in flexible manufacturing plants (as some industries dealing with the assembly of electronic components in printed circuit boards). In these highly structured environments it is possible to predict almost all the events that can affect the behaviour of the system. Therefore, DES based approaches are, in general, adequate to solve these problems.

Fukuda and Ueyama, in [Fukuda and Ueyama, 1994], proposed an innovative cellular robotics concept. Different robotic devices used learning techniques, as genetic algorithms, to group the robots into more complex devices able to perform a specified task.

Economic considerations are also a source of motivation for the development of multiple robot control techniques. Cohen, in [Cohen, 1996], presented a team of inexpensive robots to map an *a* *priori* unknown planar environment while searching for a goal. The approach is based on two types of robots: navigators and cartographers, each of which executing dedicated algorithms. For both types of robots it is assumed that none of them can determine its absolute position. Cartographers are assumed to be capable of traveling randomly through the environment, detecting the goal location and communicating with other robots within a short range. Navigators have the additional capability to track the goal or any other robot within a short range and in line of sight. The basic idea is to have the cartographers wandering until one of them detects the goal. At that point it broadcasts the information to the near robots which in turn broadcast to far located robots. The navigator uses the broadcasted information to find the path to the goal. The final map may not be complete and the path found may not be optimal.

Matarić, in [Matarić, 1994], proposed a basis of six *behaviours*, named, *safe-wandering*, *following*, *aggregation*, *dispersion* and *homing*, through which more complex *behaviours*, as *flocking* or *foraging*, in multiple robot systems can be obtained. The generation of complex *behaviours* is addressed using reinforcement learning methods to choose, at each event, the best *behaviour*.

Parker, in [Parker, 1998], proposed a *behaviour* based approach to the control of multiple, loosely coupled, robots. This architecture, named ALLIANCE, is supported in Brooks' subsumption architecture, with three main levels of control. The highest level, called *Motivational behaviours*, models the *impatience* and the *acquiescence* of the robot. *Impatience* measures the attitude of a robot towards the other robots present in the system, increasing when the performance of the other robots is such that the task assigned to the system is not being accomplished. *Acquiescence* measures the attitude of the robot towards itself, increasing when the robot is taking too much time to accomplish its own task. The intermediate level of the architecture contains a group of sets of *behaviours*, with mutually exclusive *behaviours* placed in different sets. At each time, a single set of *behaviours* is activated by the *Motivational behaviours* level. The lowest level is composed by the basis layers of competence and is always active. This level handles, for instance, the obstacle avoidance. The access to the actuators by the *behaviours* is made using the supression and inhibition mechanisms, already defined in the subsumption architecture, on the lowest level of the architecture. The ALLIANCE architecture was tested in a laboratory version of the hazardous waste cleanup experiment.

Recent works in the cooperation area focused on having large data structures shared by a number of robotic clients. Alami *et al*, in [Alami et al., 1998], describe the control of a fleet of trucks operating in harbor container transportation. The whole system is managed by a server accessed by each client (truck) using a protocol to read/write access to these structures.

Among *behaviour* design methodologies, the approach proposed in [Colombetti et al., 1996], named *Behaviour Analysis and Training*, consists in seven steps: *application description and behaviour requirements*, *behaviour analysis*, *specification*, *design*, *implementation and verification of the nascent robot*, *training* and *behaviour assessment*. These steps are supported on a set of rules of thumb with the argument, taken from [Colombetti et al., 1996], that "There is no general rule about how to carve simple behaviours out of more complex ones". The presented examples, navigation tasks by mobile platforms and a grabbing task by a manipulator, showed some limitations in the methodology. The authors recognized that an "appropriate set of behavior metrics" is necessary for the *behavioural* approach to be used in industrial applications.

1.3 Introductory relations between groups, *behaviours* and cooperation

This section briefly introduces a number of topics relevant to the understanding of the work presented in this thesis. In some sense, the relations briefly established among these topics are a counterpart of the methodology followed in the development of the *behavioural* framework presented.

1.3.1 Groups and *behaviours*

The vast majority of the robot models of interest are nonlinear. Geometrical control theory, [Jurdjevic, 1997], uses multiple ideas from differential geometry to lend a constructive approach to the nonlinear control problem.

Roughly, nonlinear control studies the controllability and observability properties and the synthesis of appropriate controllers for dynamical systems of the form $\dot{x} = f(x, u)$, with $x \in M$, M being a m-dimensional manifold in $\mathbb{R}^n$ and $f(x, u)$ being an analytic vector field on M. Using the geometrical approach, a dynamical system is interpreted as a family of vector fields. As vector fields specify motion directions in M, the study of which vector fields can be generated from the particular family specified by the dynamical system is thus identified with the general control problem.

Without lack of generality, a *behaviour* can be considered as an entity capable of specify motion in some space and thus it shares a connection with vector fields. As it is reasonable to question, within the *behavioural* framework, how can new *behaviours* be generated from old ones (the subject of the generation of complex *behaviours* from a set of basic ones is central to this approach, [Matarić, 1994, Arkin, 1995, Millan, 1996]), the same question is pertinent when a set of vector fields is considered. The answer to this last question is given by the Lie bracket operation, or commutator, on vector fields[3].

Given two vector fields on M, represented by $\mathbf{v}$ and $\mathbf{w}$, the Lie bracket operation defines an algebra, the so called Lie algebra of vector fields on M. When a sequence of controls is such that

[3] See Appendix A.

the vector fields $\mathbf{w}$, $\mathbf{v}$, $-\mathbf{w}$ and $-\mathbf{v}$ define motion directions during an infinitesimal time instant, the net motion direction is given by the Lie bracket, usualy represented by $[\mathbf{v}, \mathbf{w}]$. This is the well known geometrical interpretation of the Lie bracket (see Appendix A.8) that expresses the fact that everything that can be done in M is contained in its Lie algebra of vector fields, [Olver, 1993].

By a recursive use of the geometrical interpretation of the Lie bracket, the successive applications of Lie brackets generate the complete set of maneuvers that a robot can perform. Alternatively, successive Lie bracket applications can be seen as determining the control sequences defining every maneuver. As an example, consider the system $\dot{x} = f(x)u_1 + g(x)u_2$. Typical elements of the set of motion directions available at x are given by $[f, \dots, [g, \dots, [f, g] \dots] \dots]$. Assuming that $u_1, u_2 \in \{0, 1\}$, the control sequence $(1,0) \rightarrow (0,1) \rightarrow (-1,0) \rightarrow (0,-1) \rightarrow (1,0) \rightarrow (0,1) \rightarrow (1,0) \rightarrow (0,-1) \rightarrow (-1,0) \rightarrow (-1,0)$, where each control is applied for an infinitesimal period of time, drives the system towards the direction defined by the vector field $[[f, g], f]$.

Besides the elegancy, the complexity of defining a control by an explicit use of the geometrical interpretation of the Lie bracket is obvious. The analysis of the dimension of the space of vector fields generated by all the Lie bracket combinations at every point $x \in M$ is fundamental to determine the dimension of the submanifold of M that can be reached from x. The dimension of this submanifold is identical to the dimension of the space of vector fields generated by the Lie bracket combinations and is closely related with the controllability issue, [Sussmann and Jurdjevic, 1972, Hermann and Krener, 1977][4]. However, finding a basis, at each point of M, of vector fields in the space generated by the Lie bracket combinations is a non-decidable problem, [Laumond, 1993], pp. 157.

The resemblances between the Lie bracket techniques used in nonlinear control theory and a framework supporting *behaviours* are not immediately established. For instance, vector fields can be associated with differential equations to generate trajectories whereas *behaviours* can not (at least *a priori*) unless risking loosing generality. However, this argument suggests the use of a group structure as the core of the framework, to represent the basis functionalities necessary to robot control. Other algebraic structures, such as algebras, define other characteristics that give additional insight to the physical interpretation of motion generation, e.g., as in the physical interpretation of the Lie bracket of two vector fields. This argument suggests the subsequent extension of the group structure to more specialized algebraic structures.

1.3.2 *Behaviours* and cooperation

The robotic systems studied in the nonlinear control field are mainly of continuous nature. Instead, cooperation among multiple robotic devices, or, more generally, between multiple entities, consid-

[4] A simple interpretation to the well known controllability tests for dynamical systems can be derived from the the physical interpretation of the Lie bracket. See [Hermann and Krener, 1977] for details on how to obtain the usual controllability test for linear systems.

ers hybrid systems, exchanging information at non-uniformly spaced instants of time (also called *events*). Whenever necessary, a robot sends requests for information to the other robots, having no warranty that an answer will come back. Also, there is no assumption on the synchronism of the request-answer protocol, which means that a device may send a request to a second robot and, in the meantime, receive information from a third robot. This discrete nature is a characteristic that rends difficult a straightforward application of nonlinear control methods to hybrid dynamical systems (in general) and cooperation problems (in particular).

In game theory parlance, cooperation is related with the analysis methodologies used in decision problems. The cooperative theory is concerned with decision problems in spaces of *outcomes* of a system and not with the details leading to those *outcomes*, (see the note by Aumann in [Hart and Mas-Colell, 1996]). This surjective characteristic of the mapping *system state* $\mapsto$ *outcomes* means that part of the information contained in the state space is lost by the mapping. This reduction in the level of information in the problem (in general leading to a reduction in complexity) was already present in Brooks' work in the observation that the *behavioural* paradigm tends to simplify the interactions with the humans, [Brooks, 1986b]. In addition, Košecká *et al*, in [Košecká et al., 1997], recognized the importance of a parallel development between frameworks describing *behaviours* and programming methodologies.

1.4 Main contributions

The main contribution of this thesis is the framework that leads to the definition of a set of basis building blocks of a *behaviour* based control architecture. The methodology used in the development of the *behavioural* framework encompasses the following steps:

- definition of an appropriate state space,
- definition of an operator to induce a group structure in a subspace of the state space,
- definition of the basis building blocks for a control architecture,
- generalization of the control architecture to multiple robot systems.

This framework has the following innovative characteristics (ranked by relevance):

1. The framework models a class of systems characterized by the ability to generate multiple trajectories (or aggregates of trajectories) under similar initial conditions. The state space considered is thus a space of aggregates of trajectories in a robot's C-space.

Different trajectories performed by a robot can be considered as equivalent in a sense that it is simultaneously useful to the control of the robot and coherent with the usual notion (as used in [Reignier et al., 1995]) of *behaviour* as a *Perception* $\mapsto$ *Action* mapping.

A group structure superimposed on the state space leads to the definition of a *behavioural* architecture for the control of single robots. The extension of the group structure to a non-proper semiring structure is shown to simplify some aspects of the computational implementation.

2. Since the early stages of development, the *behavioural* framework leads to controllability concepts that are intermediate of those current in the context of control theory.

The use of the *behavioural* architecture in multiple robot systems is supported on an argument on the dependence of the controllability of decentralized systems on the exchange of information among the robots.

3. The framework presented in this thesis shares some common points with the Viability Theory, [Aubin, 1991], namely the concepts of *tube* and *contingent cone* (Appendix B presents a brief introduction to Viability Theory). The utility of the first concept will be apparent from the beginning of the formal exposition in Chapter 2. As for the concept of *contingent cone*, it is used as the basis property leading to a form of *behaviour* convergence in the algorithms proposed to implement the framework.

The concept of *event*, as the time mark in the evolution of systems, is borrowed from DES theory. The use of Finite State Machines, as decentralized supervision devices, is also a common point with other works in this area, some of which using DES theory.

Saridis, in [Saridis, 1996], claim that Artificial Intelligence (AI) techniques lack the rigor that analytic techniques provide. The framework presented herein contributes to narrow the gap between the *behaviour* concept, originated in the AI field, and the analytic techniques.

1.5 Outline of the thesis

This thesis is organized in six chapters. The first three chapters introduce the problem and define a framework to model cooperation problems under a behavioural approach. The next two chapters detail the implementation aspects of the proposed framework for a particular set of robots composed of manipulators and mobile platforms, and present the experimental results of several case studies. The last chapter presents the conclusions of the thesis.

Chapter 1 presents the motivation for the work described in the thesis, along with a review of some relevant work in behavioural control and cooperation. Primary connections between the areas of research explored in the thesis are also presented, together with the methodology followed to fulfill its objectives.

Chapter 2 presents the definitions of the main concepts and their basis properties, together with the assumptions and terminology adopted throughout the thesis. A group structure is superimposed on a subspace of the state space, established to support behavioural control modelling, and is used in the definition of the basis building blocks of a control architecture. The need for the group structure follows from the basis functionality common to every purposeful robot: moving, in a controllable way, in its configuration space. The identification of fundamental characteristics of *behaviour* based approaches with the mathematical concepts of equivalence relation and group structure forms the core of the framework. The kinematic constraints of common robots are used to motivate various aspects of the work.

Chapter 3 begins by showing the connections between a *behaviour* approach supported on vector fields and the framework developed in Chapter 2. The basis structure of the *behavioural* architecture proposed in Chapter 2 is extended with an additional operation and the implementation details, tailored to the mobile platforms and manipulators, are presented. The extended structure, which is shown to be a non-proper semiring, is constrained to the detection of adequate events in the execution of the additional operation. A discussion on the controllability of the multiple robot system supports the extension of the *behaviour* framework to the multiple robot systems. The chapter ends with a summary of the complete framework.

Chapter 4 describes the operation of the basis building blocks in the control architecture, in terms of the experiments presented in the thesis. Motor and sensory aspects of the robots considered and the general software architecture used to obtain the experimental results are detailed.

Chapter 5 presents the experimental results using real and simulated robotic devices. All the tasks considered in the thesis refer to mobile manipulators. The experiments include the execution of missions by single and dual mobile manipulators. The single mobile manipulator experiments demonstrate the cooperation between the manipulator and the mobile platform, corresponding to a situation of tight coupling between the robots. In addition, the dual mobile manipulator experiments demonstrate the cooperation between two mobile manipulators in the transportation of a large bar. In these experiments one of the mobile manipulators grasps the bar at a fixed point whereas the other is allowed to slide the grasping point, corresponding to a situation of loose coupling between robots.

Chapter 6 presents the overall conclusions of the thesis and foreseen developments.

The basis mathematical concepts used in the thesis are presented in Appendix A. Appendix B presents a brief overview on some topics of the Viability Theory that are relevant to the thesis. Appendix C describes the kinematics of two common mobile platforms, including the one considered in this thesis, for which details of the control are also presented. Some engineering problems with the robots considered are described in Appendix D. A geometrical trajectory planner used by some examples in the thesis is described in Appendix E. Finally, Appendix F presents a list of the sources of several MPEG video files obtained during the experiments.

Chapter 2

Basis framework and general concepts

2.1 Introduction - Basis definitions

This chapter describes the basis of a framework to model a *behaviour* based architecture for the control of robotic devices. Definitions of the basis concepts, motivated by practical examples, along with terminology used in the thesis, are presented in Section 2.1. The concept of agent is used as the entity able to cooperate while executing an assigned task. The control of the agent is driven by events, representing the time instants where the decisions influencing the behaviour of the agent are taken. The cooperation problem is then defined as the problem of driving a set of agents so they fullfil their assigned tasks. Section 2.2 describes the meaning of the state variable chosen to represent the motion of an agent. This state variable, aiming at simplifying the exchange of information among multiple agents, is composed of two elements, a *configuration* and an *action*. Section 2.3 defines the natural equivalence relation in the state set, that is used in Section 2.4 to define a group structure on the actions space. Sections 2.5 and 2.6 detail some important aspects of the state space that appear as a consequence of the group structure previously defined. The chapter is concluded in Section 2.7 with the description of a *behaviour* based control architecture supported in the framework developed. Part of the material in this chapter was first presented in [Sequeira and Ribeiro, 1997].

2.1.1 Single agents

Definition 2.1.1 (Agent) *An agent is composed by a robotic device*[1] *and whatever is necessary to have it exhibit a specified behaviour, e.g., controllers, sensors and actuators.*

[1] The generic term *robot* will also be used. In this thesis it is assumed that each agent has a robot of its own and each robot belongs to a unique agent.

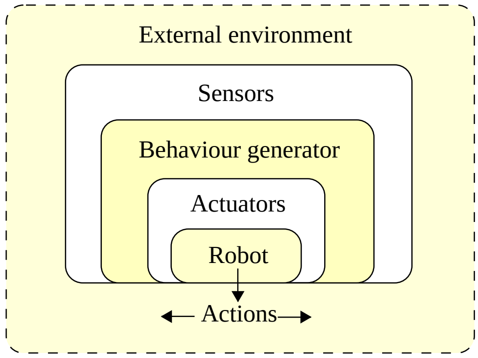


Figure 2.1: Abstract robotic agent.

The agent[2] is the basis unit of a multiple robot system and enables the encapsulation of most of the aspects common to the control of multiple heterogeneous robotic devices. Figure 2.1 shows the agent's components enclosed by the solid lines.

A robotic agent interacts with the environment through the associated robot, e.g., avoiding obstacles or even modifying the characteristics of the environment through manipulation. From a classical control perspective, this interaction between agent and environment is made by the control of the actuators commanding the mechanical components of the robot. From a *behavioural* perspective, the interaction is made through the generation of high level commands, each of which is then translated to references to be followed by the actuators. The approach considered in this thesis assumes that a finite number of these high level commands is sufficient to drive the agent towards its goals, with an acceptable performance. As motivated ahead in this chapter, these high level commands are called actions.

The interactions between the agents components and the external environment, implicitly represented in Figure 2.1, are formally defined by Diagram (2.1),

$$\begin{array}{ccc} S & \xrightarrow{M_{ST}} & T \\ & {}_{M_{QS}}\nwarrow & \downarrow {}_{M_{TQ}} \\ & & Q \end{array} \tag{2.1}$$

where the sets are defined in Table 2.1. Hereafter, the word *set* will be used to designate a collection of elements without an explicit form. Ahead in the thesis, the word *space* will be used to designate a set with some kind of superimposed structure[3].

The environment, S, represents all the information that is perceived by the agent. S contains the information from the environment surrounding the agent and from internal sensors, e.g., encoders. Robots usually operate through the actuation of a mechanical device. The space used to represent the degrees of freedom of the mechanical part is called the configuration space and is represented

[2]This is a concept with roots in the Artificial Intelligence community. Nowadays, it has been extended to Internet applications, e.g., the Internet robots, also known under a variety of names like *Web Crawlers*, *Web Wanderers*, *Spiders* and *WebAnts*.

[3]This interpretation follows [Simmons, 1963], pp. 5.

S	The agent's sensing information, e.g., battery level, ultrasound data.
Q	The robot's configuration space, or C-space.
A	The set of every possible motion strategy.
$A^p \subset A$	The set of the user defined motion strategies.
$T = Q \times A$	The agent's state set.

Table 2.1: Basis agent sets.

by Q. The set A represents all the possible motion strategies, or actions, that can be defined in the C-space. The set A may have a finite or infinite number of elements.

Each of the elements in A^p expresses a motion strategy for the agent to follow. The set A^p, which represents a subset of A, is *a priori* defined by an entity (or supervisor). The term *user* is used in Table 2.1 to denote this supervisor, which is external to the agent. Other terms, such as *programmer*, can also be used to emphasize the high level nature of the control methodology. Following the discussion in Section 1.1 on the definition of two control levels, the set A^p represents the narrow control level (i.e., the problem dependent level) and aims at reduce the complexity of the problem of choosing among the actions in A.

Whatever the agent is requested to perform can be described in terms of the sets in Table 2.1. The following definition introduces two terms that simplify the exposition.

Definition 2.1.2 (Mission and Task) *A task is defined as an element of T. A mission is a sequence of tasks.*

A pair (configuration, velocity) is called a state in Control Theory (and a phase in Physics). The motion skills expressed by each element of A implicitly define the corresponding motion directions, or velocity vectors. Thus, the final element in Table 2.1, the pair (Q, A), is a state and T is called the state set of the agent.

The mappings M_{ST} and M_{TQ} in Diagram (2.1) model, respectively, the perception of the environment and the actuation into that same environment of the agent. M_{QS} models the influence of the actuation of the agent into the perceivable information.

The user's perspective, mentioned in Chapter 1 as one of the thesis objectives, assumes that the elements of A^p may be specified without full knowledge of the robots and environment characteristics and leads to a *decoupled* control methodology, with two phases[4]:

[4] The alternative is a *centralized* control scheme, e.g., a path planner, to plan a free trajectory, in Q, corresponding to the task execution. The terminology is borrowed from [Latombe, 1992], pp. 23.

1. Specification of a basis set of actions, A^p, along with the corresponding trigger conditions. A typical approach is to define a number of actions that solve the *a priori* expected problems arising during the execution of a task. This phase is related with the class of tasks the agent is prepared to perform. The precise definition of this class of tasks is of fundamental importance, as it seems not a realistic assumption to design an agent able to execute all sorts of tasks.

2. Specification of a supervisor, or controller, to choose and monitor the execution of the actions.

This user's perspective has a counterpart in the two phase path planning methods under nonholonomic kinematic constraints, namely planning ignoring the nonholonomy followed by a transformation of the holonomic path to account for the nonholonomy (see [Latombe, 1992], Chapter 9).

Recognizing the relevant characteristics of the environment where the agent operates and incorporating this knowledge into the actions in A^p is, in general, a difficult task. Therefore, the behavioural control architecture must cope with the poor performance of some of the actions in particular circumstances, caused by the incomplete specification of the elements of A^p or by technological limitations. A relevant example is given by the type of obstacles that may be perceived during the execution of a path by a mobile platform. Low resolution sensors may need a considerable amount of time before detecting an obstacle, or may detect only obstacles of certain shapes and volumes. A lower limit on the turning radius is another example of a technological constraint that is common is some mobile platforms[5]. The resulting constraints on the feasible trajectories may yield difficult maneuvers in obstacle avoidance situations. The strategy used to overcome the physical limits of a robot also depends on the environment. Highly cluttered environments may not be compatible with some classes of obstacle avoidance trajectories. For example, a mobile platform may not able to perform a wide motion around an obstacle lying in a narrow corridor. Instead, this obstacle avoidance may require some maneuvering.

In what concerns the execution of a task, a fundamental property is that of being able to follow a specific order by the user. This property can be informally identified with the concept of controllability, common in the control systems field (see [Hermann and Krener, 1977] for the formal definition of controllability of a dynamical system, supported on the *accessibility* concept, and other related concepts). The pairwise structure of the state variable inherently expresses the capacity to represent two of such properties, each related with different elements of the pair, denoted as: *point-to-point controllability* property and *action controllability* property. Point-to-point controllability expresses the ability of the agent to reach every element of Q, starting from an arbitrary state, using sequences of actions in A^p. The action controllability concept corresponds to the ability to move, according the strategy specified by each single element of A^p, such that any configuration in a bounded region of the configuration space can be reached up to some small neighbourhood. As it will be exemplified ahead, the kinematics of robotic devices often impose constraints on the way

[5]This is the case of the ROBUTER mobile platform, manufactured by Robosoft, France, that will be considered in Chapters 4 and 5.

some configurations can be reached.

The following example clarifies the differences between the aforementioned controllability concepts and the classical controllability concepts. The length of a path connecting two points arbitrarily close to each other and lying along "forbidden" directions of motion, is not necessarily small. As an example, consider the case of a mobile platform described by[6],

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.2}$$

where (x, y, θ) is the robot's configuration and with the turning radius verifying $\mid v/\omega \mid \geq L$. Consider the initial configuration $q_i = (0,0,0)$ and the final configuration $q_f = (\epsilon_x, \epsilon_y, \epsilon_\theta)$, arbitrarily close to $(0,0,0)$, as illustrated in Figure 2.2.

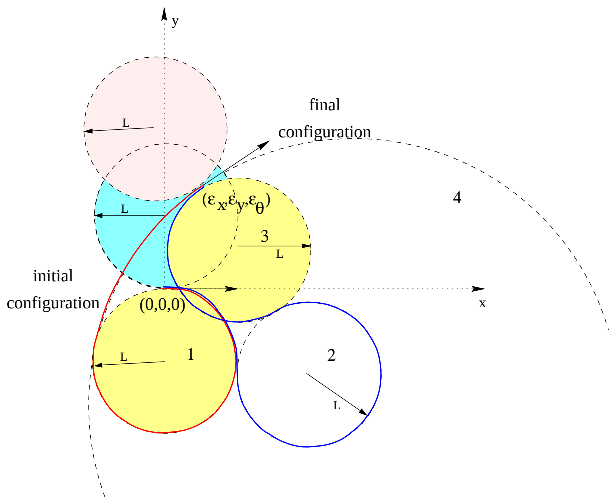


Figure 2.2: A "small task" that leads to a "long" path.

All the small circumferences have radius L, with the shadowed ones representing configurations that can not be reached when the vehicle is either at the initial or final configurations. Two possible trajectories between q_i and q_f are represented, one using arcs of the circumferences marked as 1 and 4, and the other using arcs of the circumferences 1, 2 and 3. This graphical representation is enough to show that the length of a smooth path can be quite large when compared with the distance between $(0,0)$ and (ϵ_x, ϵ_y). A reduction in the euclidean distance between the initial and the final positions will not necessarily reduce the number of maneuvers. Instead, paths with the same characteristics of the previous case are obtained, this being the case if the final configuration is, for instance, $(\epsilon_x/2, \epsilon_y/2, \epsilon_\theta)$.

[6]This is a well known model in the robotics field, usually called *cart*, [Sordalen, 1993], or *unicycle*, [Lafferriere and Sussmann, 1993]. It also describes the motion of a roling disk in a plane if θ, v and ω represent, respectively, the heading angle, and the steering and rolling velocities, [Gurvitz and Li, 1993].

The existence of short paths between two close points is a consequence of a *local controllability* property, while the weaker property of the existence of a path, not necessarily short, between any two points is related to the *global controllability*, [Hermann and Krener, 1977]. Clearly, the cart, with the constraint on the turning radius, is globally controllable but not locally controllable.

The particular motion skill generated by each action is expressed by a trajectory to be followed by the agent. The maneuvering shown by Figure 2.2 demonstrates that the cart robot, with the constraint on the turning radius, can not follow arbitrary trajectories and hence, for some A^p sets, the cart is not action controllable. However, the cart is point-to-point controllable, as there is, at least, a set of feasible actions such that any configuration can be reached from any other by using sequences of actions in the feasible set.

Table 2.2 shows the relations between the controllability concepts common to dynamical systems and the ones proposed above. Roughly, the concepts are weak when the accessibility between any two points is a reflexive and transitive but not a symmetric relation.

local controllability	$\Rightarrow$	global controllability
$\Downarrow$		$\Downarrow$
action controllability	$\Rightarrow$	point-to-point controllability
$\Downarrow$		$\Downarrow$
local weak controllability	$\Rightarrow$	global weak controllability

Table 2.2: Relations between controllability concepts.

Local and global controllability account for the motion constraints expressed by the robot's kinematics. The two additional concepts (action and point-to-point controllability) account for the constraints imposed by the control methodology. Action controllability requires that any configuration be accessible by the agent's robot using a single action in A^p. Note that the user specifies the actions in A^p eventually without full knowledge on the kinematics and thus the trajectories generated by the actions may not be feasible. Local controllability requires the accessibility of every configuration in a neighbourhood of q and hence is a stronger concept than action controllability, which requires only the accessibility of the configurations belonging to trajectories generated by the actions in A^p. Local weak controllability requires the existence of a path between any two close configurations, but the inverse path is not required to exist. The existence of this inverse path, at least between the configurations of some subset of Q, is a desirable functionality for the control of the agent. As demonstrated ahead in this chapter, this functionality is part of the algebraic structure of the actions space. Hence, the action controllability property is stronger than the local weak controllability property.

Point-to-point controllability indicates that, by using the motion strategies defined by sequences of actions in A^p, any two configurations are accessible from each other. Global controllability requires the accessibility of any configuration without concerns about the motion strategy used to generate the corresponding trajectory and hence is a stronger concept than point-to-point controllability, as the feasible actions may not be able to generate some trajectories. Global weak controllability removes the requirement on the existence of an inverse path between any two configurations. As referred above, the existence of such an inverse path represents a useful requirement that strengthens the concept to the point-to-point controllability.

From the above discussion yields that if an agent has the action controllability property, it has the ability to "follow the orders" of the supervisor, in the decoupled control approach, independently of their sequence. If only point-to-point controllability is considered, the supervisor must search for an appropriate sequence of actions in A^p (assuming that A^p contains the adequate actions).

In what concerns the execution of a task, the interest lies on a bounded subset of A so that the controllability properties are preserved when the actions are restricted to be in this subset, i.e., the agent has the ability to execute any task using uniquely the bounded set of actions. Moreover, from the *behavioural* approach perspective, it is desirable to use only a finite number of actions in order to maintain the complexity of the choice problem at a fair level, i.e., A^p is a countable and finite set. In the sequel, it is shown that this constraint imposed on A^p does not lead to any lack of generality.

In a multiple agent system, the information travels in bursts along the lines connecting the agents, that is, at discrete, randomly spaced, time instants. When the information arrives at an agent it is a local decision whether or not to process it immediately. The decision of postponing the processing of the information just arrived is equivalent to the introduction of a delay in the interconnection lines[7]. Since the arrival of information is a totally asynchronous process, these delays are not *a priori* known. Moreover, different agents will have processors sampling their input buffers at different rates[8]. These characteristics classify the multiple agent systems as event driven systems. Single agents can also be included in the class of event driven systems as they are likely to make decisions at discrete instants of time. The following definition characterizes an event as the time mark in robotic agent systems.

Definition 2.1.3 (Event) *Assume that the relevant conditions, either internal or external, sensed by the agent are placed in some buffer*[9]*. The decision of processing one of the conditions in the buffer is called an event.*

[7]For instance, it may depend on the instantaneous computational load on the processors available in the agent.

[8]In the robotics field, typical digital controllers are designed to run with fixed sampling rates, varying from a few Hertz to several thousands.

[9]In a real situation this buffer has a limited size, and this is a fact that must be taken into account by the procedure that extracts the events from the buffer.

Figure 2.3 shows a pictorial representation of a sample of the events detected by an agent during the execution of a task. The configuration of the agent when the event i happens, is denoted by q_i. Along the thesis, the time axis variable will be represented by k. To represent specific time instants, e.g., events, a subscript is used. At event i, the agent takes an action that drives it along the C-space.

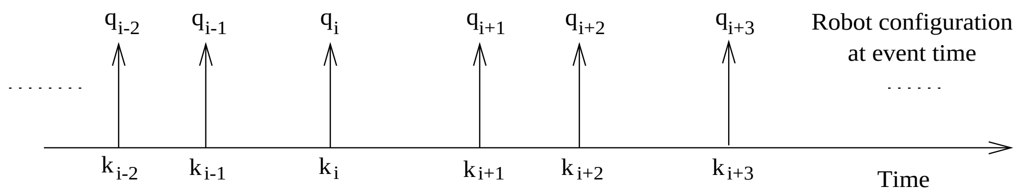


Figure 2.3: Representation of a sequence of events.

The discrete nature of the events leads to their association with symbols. Human intelligence associates symbols and meanings and thus it becomes natural to associate the term *intelligent* to a system able to do symbolic processing. Artificial Intelligence techniques usually claim that a link between meanings and symbols, referred as *symbol grounding*, exists in every intelligent system. Semiotics, considered in [Meystel, 1996] as "a science of dealing with symbolic systems" is a vast framework in which multiple intelligent systems architectures can be represented (an example is the celebrated principle of *increasing precision with decreasing intelligence*, proposed by Saridis, [Saridis, 1996], which is a counterpart of the multiresolutional architectures proposed by Meystel, [Isik and Meystel, 1988]). As a consequence, the research directions in agent control architectures will have to consider symbol processing methodologies or, at least, some analogue of them. *Behaviours* seem to provide the aforementioned link between *meanings* and *symbols*.

Each event is defined by a set of preconditions on the state and environmental data. Whenever the set of preconditions is verified the event is *detected*. The ability of the agent to detect the necessary and sufficient events for the mission completion corresponds to an *observability* concept. For example, Chapter 4 presents a set of collision avoidance experiments by a real mobile platform using a simplified algorithm based on ultrasound data. The detection of the event that triggers the obstacle avoidance action is made relying only on the minimum of the set of raw measurements returned by the ultrasound sensors on the mobile platform. The multiple reflections of the ultrasound beams on the obstacles can deceive the algorithm and represent a situation of undetectability of the correct event, i.e., the corresponding agent is not observable. Similarly to the controllability, the observability has also local/global and weak/strong forms, [Hermann and Krener, 1977]. An exaustive analysis of the influence of the observability on the performance of the agents will not be considered in this thesis. Nonetheless, some basis requirements on the detection of specific events will be considered as they are relevant in the development of the framework.

2.1.2 Multiple agents

In a multiple robot system, each agent has its own task, which is derived from the global task assigned to the system. The subdivision of a global task into the individual agent tasks has been recognized as an important issue, [Brocket, 1993]. Inadequate task assignments may compromise the cooperation among the agents. The notion of cooperative game, as described by Aumann in [Hart and Mas-Colell, 1996], can be used to better explain the meaning of the expression "adequate task assignment". In a cooperative game, some of the players may not contribute significantly for the expected outcome. Likewise, in a multiple robot system, the tasks assigned to some of the agents may not be relevant to achieve the global task. An example is given by a set of three mobile manipulators carrying a bar between two shelfs of a warehouse. The need for the third mobile manipulator depends on the performance of the two first mobile manipulators, which, in general, depends on the environment characteristics, thus making it necessary in some (and not all) executions of the same task. The nowadays popular robotic teams of football, [Veloso et al., 1997], are another potential source of examples of cooperative behaviour among multiple robots. As in most human teams, besides the difficulty of having the players performing according to the coach's tactics, it is common to have players that do not contribute significantly to the global performance (or game outcome).

Definition 2.1.4 summarizes the above discussion and bounds the class of problems that are considered, in this thesis, as cooperation problems.

Definition 2.1.4 (The multiple agent cooperation problem) *A set of agents operating as in Diagram (2.1) is said to display* <u>*cooperative behaviour*</u> *if, assuming an adequate task assignment, the global task assigned to the set is fulfilled.*

Following Definition 2.1.4, multiple robot control using standard control methods, namely the mobile manipulator in [Yamamoto and Yun, 1993] or the welding setup in [Ahmad and Luo, 1989], referred by the authors as *coordination* problems, are also cooperation examples.

2.2 The agent state variable

This section describes and details the state variable of each agent, assigning to it a physical meaning adequate to the agent control problem.

Definition 2.2.1 (Action) *An action is a mapping* $a : Q_s \times \mathbb{R} \mapsto Q$*, where* $Q_s \subseteq Q$ *is the space of all possible configurations where to start the the execution of the action and* $\mathbb{R}$ *is the time axis.*

Definition 2.2.1 is a direct consequence of the definition of action of a group on a set taking $\mathbb{R}$, the time axis, as the group acting on Q, see [Hungerford, 1996], pp. 88). Therefore, an *action* is a trajectory generator in Q that, as referred in Section 2.1.1, represents a motion skill of the agent.

Between any two consecutive events, the action taken by the agent, the initial configuration where it is applied completely determines the trajectory of the robot, in the C-space. This results in the following definition for the state variable.

Definition 2.2.2 (The agent state variable) *Between any two consecutive events the agent state is defined by the element, $(q, a) \in T$. This state variable is interpreted as a bundle of trajectories generated by the action a when applied at a neighbourhood of configuration q (the initial configuration of the state).*

The current endeavour of the agent is represented by the trajectories generated by the current state. The configuration $q \in Q$ acts as a landmark in a neighbourhood of which the action $a \in A$ is applied, e.g., push a button (the action) starting the push movement in the vicinity of the button location (the landmark).

The notation $a(q)$ is preferred to represent the elements of T as it suggests a relationship between q and a, as opposed to (q, a) that does not suggest any relation between its elements. The first notation is, for instance, common in functional analysis to represent a transformation, a, applied at a point, q, of its domain space, [Kreysig, 1978]. It can also be interpreted as a flow generated by some vector field on Q[10]. More generally, it represents the evolution of a more complex process, e.g., an hybrid system able to generate bifurcations in the trajectories. The notation $a(q)|_k$ is used to represent a particular configuration of the trajectory generated by the state $a(q)$, at event k. Following Definition 2.2.2, $a(q)\big|_{k=0} \in V(q)$,[11] for some neighbourhood of q, $V(q) \subset Q$. The nature of $V(q)$ will be detailed in Section 2.5.

Physically, Definition 2.2.2 is appealing because it contains all the information about the current motion strategy of the agent and also some memory of the close past, embedded in the initial configuration of the action. If an agent is fed back with the state, $a(q)$, of other agent[12], it means that it knows what the other is doing, i.e., what kind of motion strategy the other is taking. That information can be useful when taking its own decisions.

[10] For example, $a(q_0)$ can represent a solution of a differential equation $\dot{q} = f(q)$, for some vector field $f(\cdot)$ and initial condition q_0.

[11] Any trajectory generated by $a(q)$ is assumed to start at $k = 0$. If a time shift is considered, any other value could be used.

[12] In the game theory terminology this state variable represents the player's present configuration and the current game rules. The exchange of information among agents using this type of state variable provides every agent with the best possible information about its opponent.

To better illustrate the advantages of chosing a state set with the characteristics of T, consider the example of a manipulator, with three revolution joints, represented in Figure 2.4.

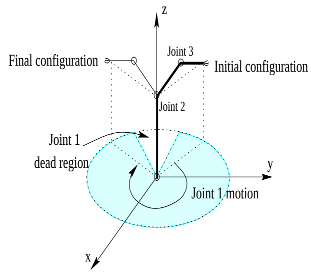


Figure 2.4: Manipulator avoiding a dead zone in the base joint.

The shadowed area in the xy plane indicates the area within the physical limits of the base joint. The small circles indicate the position of the revolution joints. To move the end-effector between the initial and the final configurations the controller of the manipulator must choose the path that causes the greatest displacement in the base joint. If the manipulator is part of a system with multiple agents with which it must interact, it is not enough to send them the present configuration, q. Unless some prediction strategy is used, the behaviour of the manipulator may not seem appropriate to an external "observer", as it starts by going away from the goal position. This may induce the observers, e.g., other agents, to try to compensate this apparently unappropriated behaviour, eventually generating other sources of error. The solution to this problem is to send to all the external observers the "reason" of such odd trajectory, which means, the action taken by the manipulator, or (which is equivalent) what the manipulator expects from this behaviour. This qualitative information is contained in the agent state variable. Using again the example of Figure 2.4, when the manipulator is moving without concerns about the joint physical limits, it can be considered as executing an action which has associated a specific meaning, for instance, *the arm is moving without concerns about joint limits.* When receiving this state information, another agent may decide that its motion does not need to obey to any particular constraint, given the particular state of the manipulator. Through this association between meanings and actions, the information exchange between agents can be identified with an exchange of grounded symbols and the agents act as symbol processing systems.

2.3 The agent state set

This section defines the basis structure superimposed on the state set. The state space created with this structure enables the definition of a number of basic building blocks for a *behaviour* based control architecture. Part of the defined concepts are illustrated through abstract diagrams with representations in Q.

Definition 2.3.1 (The set T) *Let $\mathbb{I}$ be a connected set and $A = \{a_i : i \in \mathbb{I}\}$*[13] *be the set of all possible actions, during the execution of a task. Let $a_i(q) = \{a_i(q)\big|_k, k \geq 0\}$ represent the trajectory generated by the action a_i in the d-dimensional manifold Q. This trajectory verifies*

$$\begin{array}{l} a_i(q)\big|_{k=0} \in V(q) \\ \forall a_i(q) \in T, \;\; \exists B_{a_i}(q) \subset Q \;:\; a_i(q) \subset B_{a_i}(q) \\ V(q) \subset B_{a_i}(q) \end{array}$$

where $V(q)$ is a neighbourhood of q, and $B_{a_i}(q)$ a path connected subset of Q, bounded by a $d-1$-dimensional submanifold of Q.

The path connectedness requirement for the $B_{a_i}(q)$ sets is a consequence of the physical nature of the robotic motion problem. A subset $B_{a_i}(q)$ is easily perceived as a bounded *region*, *volume*, or a *tube*[14], in Q, with the configuration q marking a neighbourhood of the origin of all the trajectories that can be generated by $a(q)$. If an action is executed during a limited period of time, then $B_{a_i}(q)$ is bounded by a closed subset of Q. Otherwise it may prolong itself indefinitely throughout Q, depending on the trajectories generated[15].

Following the discussion in Section 2.1.1 on the role of the actions within the *behavioural* framework, actions can be specified without full knowledge of the agent and/or the environment. Even in the case of the existence of full knowledge one may be interested only in a qualitative control of the agent and not in the exact following of a specific trajectory. A straightforward example is given by a mobile platform facing the unique small obstacle in the middle of a wide room. All the infinite avoidance trajectories express the same qualitative property (the obstacle avoidance). A qualitative control scheme commands the mobile platform to *avoid the obstacle*, instead of specifying an avoidance trajectory. Therefore, the sets bounding the states express both the need for some freedom in the definition of the actions, due to imperfect information, and the idea of qualitative control as a tool that simplifies the design of control procedures. In practice, these sets contain the necessary and sufficient space to account for:

- environmental constraints,
- kinematics constraints,
- user defined constraints in the accessible subspaces of Q.

The environmental constraints include environment specific conditions such as the minimal clearance between obstacles. The range of some of the parameter dependencies of the actions in A^p can also be considered in this class, e.g., parameters that can be tuned in real-time, like the curvature

[13] For the sake of simplicity, discrete values for the indexes will be used hereafter.

[14] The term *tube* is borrowed from Viability Theory. Further connections between the *behavioural* framework and Viability Theory will emerge along the thesis.

[15] For the sake of readability, in some statements referring to sets bounding states, the initial configuration of the state is omited.

of a trajectory made by a mobile platform to avoid an obstacle or the velocity with which a manipulator, mounted over the mobile platform, must react in order to compensate for the perturbations introduced by the platform. Nonholonomic kinematic constraints are taken into consideration by an appropriate specification of the region spanned, in Q, by the robot when it moves between two close configurations.

By Definition 2.3.1, multiple trajectories generated by $a_i(q)$ are enclosed in the same subset $B_{a_i}(q)$. The $B_{a_i}(q)$ is a *viability domain* of the state $a_i(q)$, i.e., it represents the portion of Q where the trajectories generated by $a(q)$, starting in a neighbourhood of q, exist (see Appendix B for a brief outline of the foundations of Viability Theory). *Retraction* based path planning techniques use a similar principle. Roughly, in a first phase, planning is made in a subspace of Q (the *retract* of Q[16]), and extended to the whole Q in a second phase. For example, when planning a trajectory for a mobile platform moving in (x, y, θ), it is common to first plan the motion using only the subspace spanned by (x, y), considering at a later stage the θ variable. Representing by $\mathbb{P}_{xy}(a_i(q))$ the projection of the trajectory generated by the state $a_i(q)$ in the xy plane, the set bounding the trajectories is defined by $\mathbb{P}_{xy}(a_i(q)) \times [0, 2\pi[$. Figure 2.5 illustrates an example of such a situation.

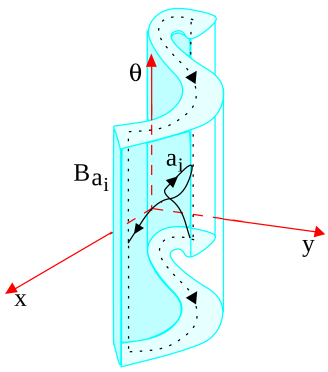


Figure 2.5: Example of a set bounding a trajectory on the C-space.

To account for the characteristics of the state variable the following natural equivalence relation in T is proposed.

Definition 2.3.2 (Equality relation in T) *Let $V_1(q)$ and $V_2(q)$ be neighbourhoods of q, verifying $V_1(q), V_2(q) \subset B_{a_i}(q_i) \cap B_{a_j}(q_j)$ and $q_i, q_j \in V_1(q)$, for some q. Two states, $a_i(q_i)$ and $a_j(q_j)$ are said to be* **equal**, *the relation being represented by $a_i(q_i) = a_j(q_j)$, if and only if the following conditions are verified.*

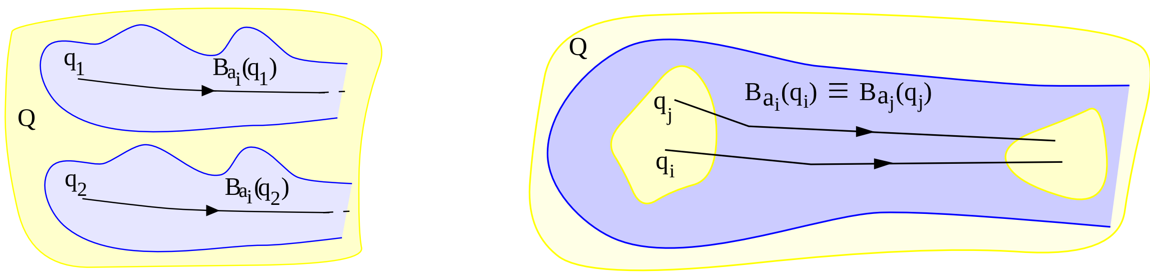
[16]Following [Massey, 1977], pp. 65, a *retract*, C, is obtained through a continuous mapping, r, such that $r(c) = c$ for any $c \in C$.

Spatial similarity $a_i(q_i), a_j(q_j) \subset B_{a_i}(q_i) \cap B_{a_j}(q_j)$, *that is, any of the trajectories generated by* $a_i(q_i)$ *and* $a_j(q_j)$ *lies in the same bounded region of the robot's configuration space.*

Time similarity $\forall k_j \geq 0,\ \exists k_i \geq 0 : a_i(q_i)\big|_{k_i} \in V_2(a_j(q_j)\big|_{k_j})$, *that is, if both actions are applied at time* $k = 0$ *in a neighbourhood* $V_1(q)$, a_i *reaches (in an unspecified time* k_i*) a neighbourhood* $V_2(a_j(q_j)\big|_{k_j})$ *for an arbitrary time* k_j.

In general terms two states are equal if and only if **all the possible trajectories** generated by their corresponding actions produce the same qualitative effect. Definition 2.3.2 expresses this idea using the similarity in terms of the space spanned by the trajectories generated by the state and of their temporal evolution.

Figure 2.6a presents an example of unequal states due to the lack of spatial similarity. In Figure 2.6b, although the differences in the produced trajectories, the two states are equal as both the spatial and the time similarities are verified. In practical terms, if two states are *equal* they have identical submanifolds bounding the trajectories they can generate. As can be perceived from Figure 2.6, the converse is not true as it would destroy time similarity. In regions where $B_{a_i}(q_i) \cap B_{a_j}(q_j) = \emptyset$ the trajectories generated by each action may be of the same kind and still represent different states, as in Figure 2.6a.



(a) Example of unequal states with equal actions.

(b) Example of equal states with equal actions, though with different initial conditions.

Figure 2.6: State characterization.

Given an arbitrary state, $a_i(q_i)$, the reflexiveness expression

$$a_i(q_i) = a_i(q_i)$$

indicates that any aggregate of different trajectories verifying the spatial and temporal similarities can be considered as belonging to a particular state. Thus, the state $a_i(q_i)$ can be understood as a label representing different trajectories. The symmetry expression

$$a_i(q_i) = a_j(q_j) \Rightarrow a_j(q_j) = a_i(q_i)$$

indicates that any two aggregates of trajectories verifying the spatial and temporal similarity conditions can be labeled either $a_i(q_i)$ or $a_j(q_j)$, i.e., the labels are interchangeable. The transitivity expression

$$a_i(q_i) = a_j(q_j) \ \wedge a_j(q_j) = a_r(q_r) \ \Rightarrow \ a_i(q_i) = a_r(q_r)$$

extends the labeling interchange to every possible aggregate of trajectories exhibiting similar temporal and spatial evolutions.

It is worth to remark that Definition 2.3.2 does not specify an equivalence relation between trajectories in Q. However, from the above three properties, the equality relation of Definition 2.3.2 is an equivalence relation on aggregates of trajectories in Q. The relation obtained by the removal of the symmetry condition is still relevant for the framework. This relation, weaker than the natural equivalence relation, means that, after some time, the label of the state $a_i(q_i)$ can be substituted by the label of the state $a_j(q_j)$, whereas the reverse may not be valid.

Definition 2.3.3 (State substitution) *Given any two states $a_i(q_i)$ and $a_j(q_j)$, the state $a_j(q_j)$ is a substitute for state $a_i(q_i)$, the relation being represented by $a_i(q_i) \leftarrow a_j(q_j)$, if and only if*

$$\exists K_j \geq 0 \ : \forall k_j \geq K_j, \begin{cases} a_j(q_j)\big|_{k_j} \in B_{a_i}(q_i) \\ \exists k_i \geq 0 \ : \ a_i(q_i)\big|_{k_i} \in V(a_j(q_j)\big|_{k_j}). \end{cases}$$

for some neighbourhood $V(q) \subset B_{a_j}(q_j)$.

After some time instant, the trajectories generated by the substitute state are similar to the ones generated by the substituted one.

Definition 2.3.4 (Equality relation in A) *Two actions a_i and a_j are said to be* **equal**, *the relation being represented by $a_i = a_j$, if and only if*

$$\forall q \in Q, \ a_i(q) = a_j(q).$$

By Definition 2.3.4, two identical states are not necessarily generated by identical actions. It suffices to have the corresponding actions generate trajectories in different bounding sets for two different initial conditions. In summary, two actions are equal if and only if, applied in identical[17] circumstances, they generate equal states in the sense of Definition 2.3.2, i.e., maintaining both the spatial and time similarities.

[17]This statement appeals to intuition. The formal definition of the term "identical" will be postponed until Section 2.5.

2.4 The basis structure for T and the agent control problem

The agent control problem is concerned with both the generation of the appropriate trajectory in T and the transitions between the states in this trajectory. The basis structure for T defined in this section represents the mechanism to generate sequences of states (or trajectories) in T.

Consider a mobile platform, moving in the plane xy, and assume that it has limited locomotion capabilities, namely either it moves in a straight line or rotates by $\pm\pi/2$ rad about a point located in its body[18]. This mobile platform is not able to follow an arbitrary trajectory in the xy plane. Using a *decoupled* approach, a path planner may be used, in the first phase, to generate a discrete set of positions (or via points) in the workspace without having into account the vehicle constraints. The second phase of the *decoupled* approach corresponds to the switch between the current motion direction and the direction heading to the next via point. At this phase, the present and the future states are "composed", using pure rotations, to change the heading angle. Although not producing a visible effect in the xy workspace, this composition operation (i.e., the pure rotation) generates a continuous path in the C-space between the terminus of the trajectory generated by one state and the beginning of the next state.

Figure 2.7 illustrates two situations using schematic diagrams. In both situations, the agent is assumed to be in a state $a_j(q_j)$ when an event occurs, forcing it to switch to state $a_i(q_i)$. The $*$ symbol indicates the configurations where the agent starts to execute the actions, the shadowed regions represent the sets bounding the states. In Figure 2.7a the agent starts the action a_i at q_i, while in Figure 2.7b the beginning of the trajectory generated by a_i is at configuration s, ahead[19] of q_i. The event at k_j determines $a_i(q_i)$ as the agent next state. A link path between the configurations $r = a_j(q_j)\big|_{k=k_j}$ and $a_i(q_i)\big|_{k=k_i}$ assures the continuity of the trajectory between the present and the next states (in the previous example with a low cost mobile platform the link trajectory is simply the rotation to align the robot between two straight line motions).

For some dynamical systems it is not possible to define a direct path, e.g., a straight line in a cartesian space, between any two configurations, this being related with the local/global characteristic of the controllability properties of the robot. For the situation described in Figure 2.2, the case where changes in the sign of the linear velocity of the robot are not allowed is illustrated in Figure 2.8a, whereas the case where changes are allowed is illustrated in Figure 2.8b[20]. The configurations $q_i = (x_i, y_i, \theta_i)$ and $q_f = (x_f, y_f, \theta_f)$, represented by vectors, can be joined by a line, in $\mathbb{R}^3$. However, the limit in the turning radius of the mobile platform prevents the direct path to be feasible.

[18] A typical example are the low cost toy mobile platforms with highly simplified motor controllers.

[19] Time is implicit here, i.e., if a_i starts generating a trajectory at q_i then, at some future time, the trajectory will be near s.

[20] These changes usually characterize the term *maneuvering*.

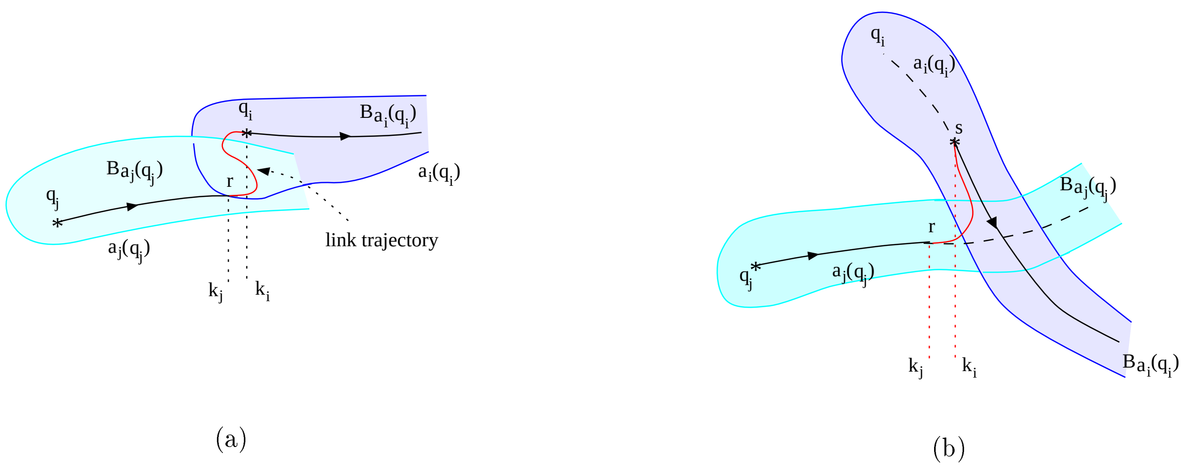


Figure 2.7: State switching. The diagrams are representations in Q.

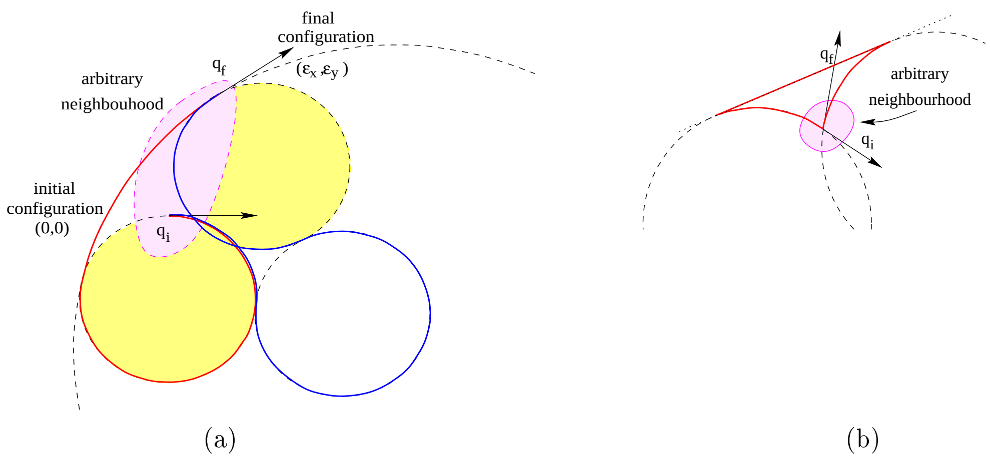


Figure 2.8: Small neighbourhoods not containing a path between q_i and q_f. The diagrams are representations in the xy plane.

Figure 2.8b uses a construction due to Laumond, [Latombe, 1992], pp. 424-425, to generate a path equivalent to a pure rotation, in the sense that the initial and final configurations differ only in the orientation. The area enclosed by the trajectory made by the vehicle is the minimum area necessary for the execution of the equivalent of a pure rotation. An increase in the value of the minimum turning radius leads to the expansion of this area.

Definition 2.4.1 introduces an operation that characterizes the link trajectories as the mechanism providing the switching between states. In some sense, this operation endows T with a motion generator.

Definition 2.4.1 (State composition) *Consider an agent in state $a_j(q_j)$. At time k_j the agent is required to switch to the state $a_i(q_i)$, which will start to be executed at an unspecified time k_i. The operation $\circ\ :\ T \times T \to T$, $a_{i\circ j}(q_j) = a_i(q_i) \circ a_j(q_j)$, (to be read as "$a_i(q_i)$ after $a_j(q_j)$" or "$a_i(q_i)$ composed with $a_j(q_j)$") verifying*

$$\begin{array}{ll} \left.\begin{array}{l} a_{i\circ j}(q_j) \subset B_{a_i}(q_i) \cup B_{a_j}(q_j) \\ a_{i\circ j}(a_j(q_j)\big|_{k_j})\big|_{k_i} \in B_{a_i}(q_i) \\ a_{i\circ j}(a_j(q_j)\big|_{k_j})\big|_{k_j} \in B_{a_j(q_j)} \end{array}\right\} & \text{if } B_{a_i}(q_i) \cap B_{a_j}(q_j) \neq \emptyset \\ \text{no meaning} & \text{otherwise} \end{array}$$

is called the state composition operation.

The trajectories resulting from the composition $a_{i\circ j}(q_j)$ are equivalent, in the sense of Definition 2.3.2, to those generated by the sequence $a_j(q_j)\big|_{0\leq k\leq k_j}$, $a_i(q_i)\big|_{k\geq k_i}$ and, if necessary, a path linking $a_j(q_j)\big|_{k_j}$ and $a_i(q_i)\big|_{k_i}$.

The *no meaning* case in Definition 2.4.1 leads to state compositions requiring the use of intermediate states. As a practical example, consider an agent operating in $\mathbb{R}^2$ and assume the existence of no constraints on the robot's motion and hence its local controllability. Moreover, consider the agent in a state $a_j(q_j)$, where a_j uniquely generates straight line segments. At some event the agent is required to switch to state $a_i(q_i)$, where a_i is also constrained to uniquely generate straight line trajectories. The sets bounding each of the states can be made arbitrarily close to the straight lines of each trajectory, leading to a situation where no intersection between the two exists. Nevertheless, the controllability property of the robot ensures that a linking path exists. Together with the initial assumption on the absence of any motion constraints, an arbitrary number of intermediate states can be defined, each providing a piece of the path connecting $a_j(q_j)$ and $a_i(q_i)$. Figure 2.9 illustrates an example of state composition using intermediate states.

The local characteristic of the composition operation[21], expressed in the *no meaning* situation of Definition 2.4.1, matches the notion of global controllability in the sense that a globally controllable

[21]Throughout the thesis the term *local* (and the corresponding adverb) is used in two different senses: (i) in the context of control theory, namely when referring to controllability, meaning that a short length path exists between

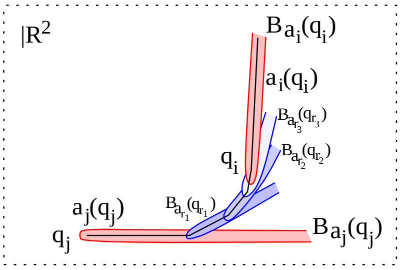


Figure 2.9: A state composition using intermediate states.

robot can not execute some trajectories, or, equivalently, that a minimum area for maneuvering is required to reach some configurations.

As in any topological space, it is possible to define a group structure on the set of all closed paths (or loops) passing through an arbitrary point $q \in Q$, with an adequate path composition operation, [Massey, 1977]. This group is called the *fundamental group* of Q, usualy denoted by $\pi(Q, q)$[22]. If a path between two arbitrary points $q_1, q_2 \in Q$ exists, it is also part of an element of $\pi(Q, q_1)$ (and of an element of $\pi(Q, q_2)$), and thus, through each point in the set of points accessible from q_1 passes an element of $\pi(Q, q_1)$. Therefore, since the controllability property is equivalent to the accessibility condition between any two points of Q, the existence of the fundamental groups is a necessary condition for the existence of the controllability property. By Theorem 3.5 in [Massey, 1977], pp. 62, any two fundamental groups in a path connected space are isomorphic. Since Q is path connected, one concludes about the existence of the referred isomorphisms and hence the controllability property at one point is identical to that at any other point.

In a pragmatical manner, given the aforementioned connections between controllability and group structures, the interest is to obtain a group structure applicable to the robotics problem considered in this thesis, i.e., such that it can be used to extract the basis functionalities that must be present in any implementation of a control architecture.

The state space structure can be further motivated by intuitive examples such as the docking of a mobile platform. Geometrical techniques for planning feasible paths for the common mobile platforms, either holonomic or nonholonomic, often assume the existence of no external disturbances. Non modeled dynamics and unexpected obstacles are two examples of major sources of perturbations that eventually require the replanning of the trajectory to execute the docking. The ability to replan requires the existence of functionalities, supplied by the systems designer, allowing, for instance, the interruption and inversion of the current trend of the robot, these being easily identified with properties of algebraic groups, as it will be shown in the sequel.

two close configurations, (ii) in the context of the framework being presented, meaning that two configurations lie in some bounded region, thus not being related with short length paths between configurations. Whenever necessary, the proper context will be made explicit.

[22] This notation follows [Massey, 1977].

For the sake of emphasizing the role of the aforementioned functionalities, the group structure is defined by two separate propositions. Given Definition 2.4.1, Proposition 2.4.1 presents the conditions on the elements of A for the existence of the group structure. These conditions are then used by Proposition 2.4.2 to clarify an additional condition on the state composition operation of Definition 2.4.1.

Proposition 2.4.1 (The A group of aggregates of feasible trajectories) *Under the state composition operation specified by Definition 2.4.1, A is a local group of transformations of Q.*

Demonstration: The demonstration follows by a direct verification of the conditions under which the group properties hold, given the state composition operation of Definition 2.4.1.

The null element, a_e

The null element of A is the unique action that does not generate any purposeful motion. In terms of natural language, this means that, at each event, it must be always possible *to do nothing*, or, *to do something that does not compromise future decisions*, independently of the configuration of the agent at the event.

Using Definition 2.4.1,

$$a_e(q_e) \circ a_i(q_i) \subset B_{a_e}(q_e) \cup B_{a_i}(q_i),$$

and, in order to verify

$$a_e(q_e) \circ a_i(q_i) = a_i(q_i),$$

it is required that, for all $q_i, q_e \in Q$ such that the state composition is meaningful, $B_{a_e}(q_e) \subseteq B_{a_i}(q_i)$, independently of the trajectory generated by $a_e(q_e)$. Moreover, the expressions

$$(a_i(q_i) \circ a_e(q_e)) \circ a_j(q_j) = a_i(q_i) \circ (a_e(q_e) \circ a_j(q_j)) = a_i(q_i) \circ a_j(q_j),$$

must hold for arbitrary states $a_i(q_i), a_j(q_j)$, also for meaningful state compositions, yielding

$$B_{a_e}(q_e) \subseteq B_{a_j}(q_j) \quad \wedge \quad B_{a_e}(q_e) \subseteq B_{a_i}(q_i)$$

and thus

$$B_{a_e}(q_e) \subseteq B_{a_j}(q_j) \cap B_{a_i}(q_i)$$

which, extended over the whole T, yields[23]

$$B_{a_e}(q) = \min_{Q,i,j}\{B_{a_j}(q_j) \cap B_{a_i}(q_i)\}. \tag{2.3}$$

For any $q \in Q$, $a_e(q)$ always generates trajectories not moving "too much" away from the current configuration, so that any subsequent action has the same effect (in the sense of Definition 2.3.2) as it would have if a_e had not been applied.

[23] The intersection of all of the bounding regions at an arbitrary configuration q corresponds to the common bounding region that is spanned by every combination of actions at that point. The minimum value is obtained by chosing over the whole Q.

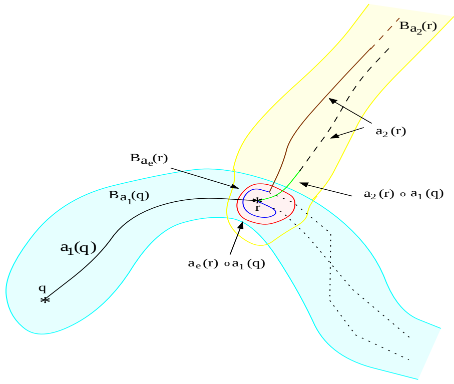


Figure 2.10: Null action example.

Figure 2.10 illustrates a possible result of the execution of a_e. Action a_1 is applied at configuration q. When it reaches configuration r, the null action is applied keeping the agent in a close neighbourhood, $B_{a_e}(r)$ (the B_{a_e} region represented in the figure was chosen merely as an example). The region $B_{a_2}(r)$ contains all the trajectories generated by the state $a_2(r)$. The dashed line represents a valid alternative trajectory generated by $a_2(r)$ without any influence of the trajectory generated by the null state $a_e(r)$.

The existence of a null element also means that taking any action after having taking a null action must lead to

$$a_i(q_i) \circ a_e(q_e) = a_i(q_i),$$

for all q_i, q_e such that the state composition is meaningful. Applying Definition 2.4.1, yields

$$a_i(q_i) \circ a_e(q_e) = a_i(q_e)$$

with $a_i(q_e) \subset B_{a_i}(q_i) \cup B_{a_e}(q_e)$. To maintain the time similarity condition, expression (2.3) defines $B_{a_e}(q_e)$ as a bounding set, for arbitrary q_e. Thus, $a_i(q_i)$ must be applied in a neighbourhood of q_e for the state composition to be valid, leading to

$$a_i(q_e) = a_i(q_i).$$

Therefore, the state space supports the existence of a null element, $a_e \in A$.

Inverse element, a^{-1}

Consider an agent executing action a_i. At event k_1 a new action, that will be denoted by a_i^{-1}, is chosen to start execution, aiming at driving the agent back to a configuration such that a subsequent application of a_i, at event k_2, would have had the same effect as if no a_i^{-1} would have been applied. Using Definition 2.4.1, this statement is represented by

$$a_i(a_i^{-1}(a_i(q)\big|_{k_1})\big|_{k_2}) \circ (a_i^{-1}(a_i(q)\big|_{k_1}) \circ a_i(q)) = a_i(q), \tag{2.4}$$

which, to be verified, requires that the substitution

$$a_i^{-1}(a_i(q)\big|_{k_1}) \circ a_i(q) \leftarrow a_e(q) \tag{2.5}$$

holds for arbitrary k_1, such that

$$a_i(a_i^{-1}(a_i(q)\big|_{k_1})\big|_{k_2}) = a_i(q)$$

is verified for some k_2. For k_1 close enough to 0 (the initial time) the state substitution in expression (2.5) tends to the state equality. By Definitions 2.3.3 and 2.4.1, expression (2.5) requires that

$$B_{a_e}(q) \subset B_{a_i^{-1}}(a_i(q)\big|_{k_1}) \cup B_{a_i}(q)$$

and, by the null element property,

$$B_{a_e}(q) \subset B_{a_i}(q) \quad \wedge \quad B_{a_e}(q) \subset B_{a_i^{-1}}(a_i(q)\big|_{k_1}). \tag{2.6}$$

Defining

$$B_{a_i^{-1}}(a_i(q)\big|_{k_1}) = B_{a_i}(q)$$

yields

$$a_i^{-1}(a_i(q)\big|_{k_1}) \subset B_{a_i}(q)$$

for arbitrary k_1, i.e., a_i^{-1} lies in the same region spanned by a_i. Using the substitution $a_i(a_i^{-1}(q)\big|_{k_1}) \circ a_i^{-1}(q) \leftarrow a_e(q)$, expression (2.4) with a_i and a_i^{-1} interchanged is also verified.

Figure 2.11 illustrates a typical situation of an inverse action application.

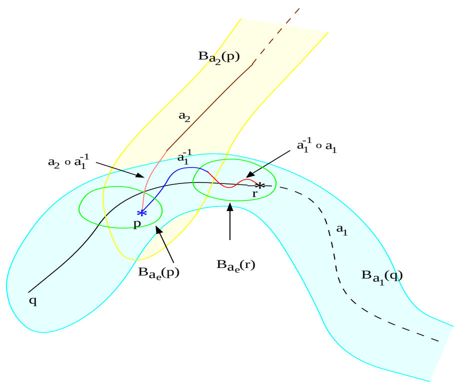


Figure 2.11: Inverse action example.

Action a_1 is applied at configuration q. When it reaches configuration r the inverse action is applied, keeping the trajectory inside the region $B_{a_1}(q)$. An event at configuration p determines the execution of a_2. Each configuration of the inverse trajectory lies in a neighbourhood, $B_{a_e}(a_1(q)\big|_k)$, for some k.

Associative property

Consider the compositions given by

$$(a_3(q_3) \circ a_2(q_2)) \circ a_1(q_1) = a_{3\circ 2}(q_2) \circ a_1(q_1) = a_{(3\circ 2)\circ 1}(q_1). \tag{2.7}$$

Using Definition 2.4.1, the composition $a_{3\circ 2}(q_2) = a_3(q_3) \circ a_2(q_2)$ generates trajectories lying in the region $B_{a_3}(q_3) \cup B_{a_2}(q_2)$ and $a_{(3\circ 2)\circ 1}(q_1)$ lies in $B_{a_1}(q_1) \cup B_{a_2}(q_2) \cup B_{a_3}(q_3)$.

Similarly, for the compositions given by

$$a_3(q_3) \circ (a_2(q_2) \circ a_1(q_1)) = a_3(q_3) \circ a_{2\circ 1}(q_1) = a_{3\circ(2\circ 1)}(q_1), \tag{2.8}$$

the trajectories generated by the state $a_{3\circ(2\circ 1)}(q_1)$ lie in the region defined by $B_{a_3}(q_3) \cup B_{a_2}(q_2) \cup B_{a_1}(q_1)$.

From Definition 2.3.2, the equality between states requires that the bounding regions be identical. Moreover, temporal similarity is verified as any trajectory generated by the composed states resulting from (2.7) and (2.8) is composed by the same sequence of trajectories. This amount to say that the relation

$$a_{(3\circ 2)\circ 1}(q_1) = a_{3\circ(2\circ 1)}(q_1)$$

is verified and the operation $\circ$ is associative. This concludes the demonstration.

$\square$

The local characteristic of the A group of transformations is due to the fact that A acts locally to each point Q since the group operation is not defined whenever the state bounding sets do not intersect themselves. Moreover, some of the elements in A may not be defined everywhere in Q.

Proposition 2.4.1 requires a minimum amount of space for the null element. This requirement has a counterpart on the minimum overlap between the sets bounding any two states being composed. This is expressed by the following proposition.

Proposition 2.4.2 (Minimum overlap between states) *Consider the group structure defined by Proposition 2.4.1 and let $a_i(q_i)$ and $a_j(q_j)$ be two arbitrary elements of T, such that $R_{ij} = B_{a_i}(q_i) \cap B_{a_j}(q_j)$ is a non empty set. Then, the overlap between the regions bounding the states in the composition $a_i(q_i) \circ a_j(q_j)$ requires that $R_{ij} \supseteq B_{a_e}(q_e)$, for some $q_e \in R_{ij}$.*

Demonstration: Suppose that R_{ij} was strictly contained in $B_{a_e}(q_e)$ and $q_e \in R_{ij}$. From the group properties

$$\exists q_w \in R_{ij} : \; a_i(q_i) \circ a_j(q_j) = a_i(q_i) \circ a_e(q_w) \circ a_j(q_j).$$

Using Definition 2.4.1, some trajectories generated by $a_i(q_i) \circ a_e(q_w) \circ a_j(q_j)$ contain points in $B_{a_i}(q_i) \cup B_{a_e}(q_w) \cup B_{a_j}(q_j)$ belonging to $B_{a_e}(q_w) \cap B_{a_i}(q_i)$ and $B_{a_e}(q_w) \cap B_{a_j}(q_j)$, but not to $B_{a_e}(q_w) \cap B_{a_i}(q_i) \cap B_{a_j}(q_j)$. Those trajectories would violate the spatial similarity with $a_i(q_i) \circ a_j(q_j)$ and thus the null element property, resulting in a contradiction with the initial assumption on the existence of the group structure and hence R_{ij} must be at least equal to $B_{a_e}(q_e)$.

□

The request for a minimum space in B_{a_e}, expressed by (2.3), is directly related with the nonholonomy of the robotic devices. For the case of a car-like robot, Figures 2.8a and 2.8b show that a minimum region around each configuration q is necessary for the robot to be able to reconfigure itself from q to an arbitrarily close configuration. As for a holonomic manipulator[24] any configuration is accessible from any other configuration, i.e., B_{a_e} can be reduced to a single point in Q. These examples show that the minimum space required for B_{a_e} depends of the particular characteristics of the robots involved, varying from "large" regions for nonholonomic devices to points for holonomic ones.

2.5 Initial configurations

In a composition $a_i(q_i) \circ a_j(q_j)$, Definition 2.4.1 does not specify the exact configuration where the execution of $a_i(q_i)$ starts. From Definition 2.3.1, any trajectory generated by the state $a_i(q_i)$ starts in a neighbourhood of q_i. Similarly, in the case of a composition the link trajectory enters $a_i(q_i)$ in a neighbourhood of q_i. Figure 2.12 illustrates such a case by displaying two alternatives for the link trajectories connecting to the identical states $a_i(q_i)$ and $a_w(q_w)$ (this representing an example of the usefulness of multiple labeling for identical states).

Proposition 2.5.1 *Let $a_i(q_i)$ and $a_j(q_j)$ be any two states of an agent. The trajectory generated by the composition $a_i(q_i) \circ a_j(q_j)$ enters the state $a_i(q_i)$ at any configuration, q_{entry}, verifying*

$$q_{entry} \in B_{a_e}(a_i(q_i)\big|_k)$$

for some event k.

Let $q_{\text{entry}} = a_{i\circ j}(q_j)\big|_k$ represent the configuration where the agent enters the state $a_i(q_i)$, for some event k. After k (with the agent already inside the region $B_{a_i}(q_i)$) the trajectory defined by $a_{i\circ j}(q_j)$ must be indistinguishable from any trajectory generated by $a_i(q_i)$, in the sense that

$$a_{i\circ j}(a_{i\circ j}(q_j)\big|_k) = a_i(a_i(q_i)\big|_{k_2}), \tag{2.9}$$

[24]Consider a common 6-dof manipulator with an anthropomorphic architecture and using only revolution joints.

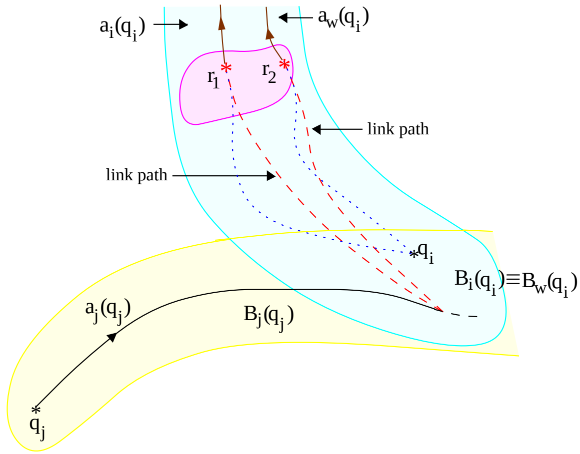


Figure 2.12: Initial configuration at a state transition.

for some k_2.

The group structure requires that

$$a_i(q_{\text{entry}}) \circ a_e(a_i(q_i)\big|_{k_2}) = a_i(a_i(q_i)\big|_{k_2}) = a_i(q_{\text{entry}}) = a_i(a_i(q_i)\big|_{k_2}) \circ a_e(q_{\text{entry}}),$$

By the minimum overlap requirement for the bounding regions

$$\begin{array}{l} a_i(q_i)\big|_{k_2} \in B_{a_e}(q_{\text{entry}}) \\ q_{\text{entry}} \in B_{a_e}(a_i(q_i)\big|_{k_2}) \end{array}$$

and thus the agent enters the state $a_i(q_i)$ at any configuration $q_{\text{entry}} \in B_{a_e}(a_i(q_i)\big|_{k_2})$, for some k_2 such that (2.9) is verified.

□

The nature of the set $V(q)$ in Definition 2.3.1 and the expression "identical circumstances", used in Section 2.3 to qualify the regions denoted also by $V(q)$, has now a definite meaning: if there is a configuration q_w such that $q_i, q_j \in B_{a_e}(q_w)$, the actions a_i and a_j, forming the states $a_i(q_i)$ and $a_j(q_j)$, are applied in identical circumstances, i.e.,

$$V(q) \equiv B_{a_e}(q).$$

2.6 Motion constraints inside a state bounding set

In this section the use of the state composition operation is extended to interpret the handling of the kinematic constraints inside a single state. While in an arbitrary state $a_j(q_j)$, the problem of following the trajectory generated by $a_j(q_j)$, i.e., the problem of generating appropriate references to the actuators, depends on the particular robot being considered and differs from the agent control problem, as considered in this thesis, since no change of state occurs in the process.

To account for any constraints imposed by the kinematics, the environment, or defined by an external user, each state must allocate a minimum amount of space for the corresponding bounding set. The state composition operation then generates any necessary maneuvering to perform any transition between states. However, while in a particular state, it may also be necessary to maneuver in order to avoid leaving the state and thus it is natural to use the state composition to additionally represent this situation.

Consider an agent based in a car-like robot (see Appendix C.2 for the kinematics) with a limited turning radius. Figure 2.13 shows the projection in the xy plane of the set bounding some state $a_j(q_j)$. The robot's configuration is assumed to be already q_j. For the sake of readability, the orientation of the robot at q_j is represented by the orientation in a fixed reference frame of the shown vector. The constraint on the turning radius is represented by the two circles superimposed on $B_{a_j}(q_j)$. The robot can not enter any of the configurations in the circles unless it performs some maneuvers.

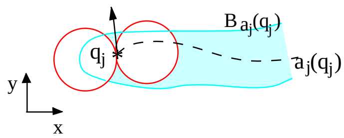


Figure 2.13: Constraint on a state bounding region. Representation in the xy plane.

In the example of the figure, the initial part of the trajectory generated by the action a_j (represented by the dashed line) lies inside one of the circles. Therefore, the robot is forced to maneuver in order to reach a configuration such that it can follow the trajectory specified by $a_j(q_j)$, always moving inside $B_{a_j}(q_j)$. In general, the feasibility of this maneuvering is limited by the amount of space available in $B_{a_j}(q_j)$ and depends on the robot being considered. While maneuvering, the robot's trajectory is constrained not to cross the boundary of $B_{a_j}(q_j)$. Otherwise, the agent leaves the state $a_j(q_j)$, violating the agent control premise, implicit in the decoupled control methodology defined in Section 2.1.1, that state changes are not decided by the robot itself but by a supervisor. The process of following a trajectory defined by an arbitrary state is thus composed by the interleaving of two phases,

1 - maneuvering, inside the state bounding set, to account for the robot's kinematic constraints, and

2 - following the trajectory generated by the state,

that are encapsulated in the following operator.

Definition 2.6.1 (The interleaving operator $\otimes$) *Consider an agent in state $a_j(q_j)$ that, at some event, is required to switch to the state $a_i(q_i)$. Represent by $W_a(q_j)$ the set containing the configurations locally weakly accessible from q_j and define $R_j(q_j) = W_a(q_j) \cap B_{a_j}(q_j)$. The interleaving operator $\otimes$ is defined by*

$$a_i(q_i) \otimes a_j(q_j) \equiv \begin{cases} a_j(q_j) \circ a_j(q_j) & \text{if } a_j(q_j)\big|_k \in R_j(q_j) \cap R_j(q_j)' \wedge a_j(q_j)\big|_{k+1} \in R_j(q_j)' \\ a_i(q_i) \circ a_j(q_j) & \text{otherwise.} \end{cases}$$

The first condition on the operator $\otimes$ expresses the existence of a link trajectory, generated by the state composition, to account for any maneuver required to avoid the violation of the kinematic constraints while in the state $a_j(q_j)$ and preparing to switch to $a_i(q_i)$. The emergence of a violation of the kinematic constraints is expressed by the robot configuration being on the boundary of $R_j(q_j)$ given that the next configuration the robot must reach lies inside the "forbiden" region $R_j(q_j)'$. The second condition in the operator $\otimes$ simply expresses the state composition for the required state transition that is to be performed as soon as the agent finds itself in a non constrained region. While in state $a_j(q_j)$, the $B_{a_j}(q_j)$ region remains static whereas the region $W_a(q_j)$ (and hence $R_j(q_j)$) changes with the agent configuration.

Using the operator $\otimes$, the following of a trajectory generated by an arbitrary state $a_j(q_j)$ is represented by $a_e(q_e) \otimes a_j(q_j) = a_j(q_j)$, for some $q_e \in B_{a_j}(q_j)$.

2.7 The control architecture

The local characteristic of the state composition operation supports the notion of *decoupled* agent control, already mentioned in Sub-section 2.1.1, and illustrated in Figure 2.14. The *Supervision* block defines a sequence of elements of T to be reached during the execution of a task. This sequence corresponds to the partitioning of the assigned task into a number of states such that the state composition is meaningful. Each of the states in the sequence is dispatched to the *Execution* block and acts as a reference to be followed by the *Execution* block. The *Execution* block encapsulates the concepts developed in this chapter to the control of an agent. The feedback path returns to the *Supervision* block the necessary information on the current task execution status.

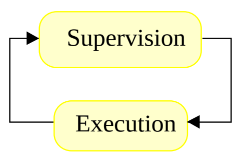


Figure 2.14: Conceptual control architecture.

In addition to choosing the action to be executed at each event, the *Supervision* block may also execute internal organization operations, e.g., reducing/expanding the set of actions to be searched at each event. An example where such a concept may be useful is a door traversal mission by a mobile manipulator. Consider an ultrasonic sensor mounted at the manipulator end-effector that detects the possibility of a collision while the mobile manipulator is approaching the door. Since the end-effector is most likely used to push the door, it is necessary to avoid use of the collision avoidance procedure by the manipulator. This can be accomplish be the temporary removal of the collision avoidance action from A^p, which is equivalent to not detecting the event that triggers the action.

Many of the current architectures for robot control can be placed under the conceptual scheme of Figure 2.14. Two representative examples of different paradigms are the *Organization, Coordination* and *Execution* hierarchy proposed by Saridis, [Saridis, 1996], and the subsumption architecture proposed by Brooks, [Brooks, 1986a]. In the first case, the *Supervision* block is composed by the *Organization* and *Coordination* levels. In the second case, it is represented by the hardwiring of the individual modules present in the subsumption architecture. In both of these two paradigms the basic mechanism of group operation is not identified. Instead, the concepts of *horizontal* and *vertical* decomposition, [Brooks, 1986a], are used to separate the architectures in two different classes. The state composition operation, introduced in this thesis, appears to be a unifying link among these two classes of architectures, as it leads naturally to a conceptual architecture that represents both classes. It is interesting to note that a recent approach to compiler design uses a *behavioural* technique which vaguely resembles the framework developed in this chapter (see [Knapp, 1996] for details). The technique consists in the definition of a number of fundamental operators, in the sense that they express the compiling process basis notions, which are then organized into an architecture.

The framework developed in this chapter introduced the concepts of action, state composition operation, state equality and of supervisor. The relations among these concepts and with the environment are expressed graphically, using two feedback loops, in Figures 2.15 and 2.16. In each of the figures, all the blocks with the exception of the *Supervision* correspond to the *Execution* block of Figure 2.14. The two architectures differ only in the $\ominus$ block. The *actions* architecture considers only the user defined actions, whereas the *behaviours* architecture is able to transform the corresponding states into equivalent ones, using the $\ominus$ blocks. The *Supervision* block represents the mechanism that, at each event, chooses the state the agent must switch to. Each *Action* block corresponds to an element of A^p. The equality blocks, represented by $\ominus$, transform the states formed with their input actions into equivalent ones, in the sense of Definition 2.3.2. At each event, the *Supervision* block sends to the $\otimes$ block, described in Section 2.6, the chosen next state which is then composed with the present state[25]. The inmost loop in the diagram, composed by the *Actions*, the *State composition* and the *State equality* blocks, generates the reference trajectory to be followed by the robot. The outmost loop in the diagram, working at a higher abstraction level than the inner loop, is composed by the *Supervisor*, the *State composition*, the rightmost *State*

[25]For the sake of simplicity, the $\otimes$ block is also referred as *state composition*. Whenever necessary the proper terminology is used.

equality block and the environment. It influences the reference trajectory sent to the robot through the choice of the state where to switch at each event.

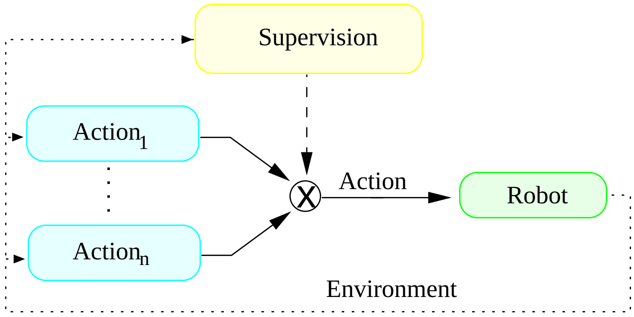


Figure 2.15: *Actions* based architecture.

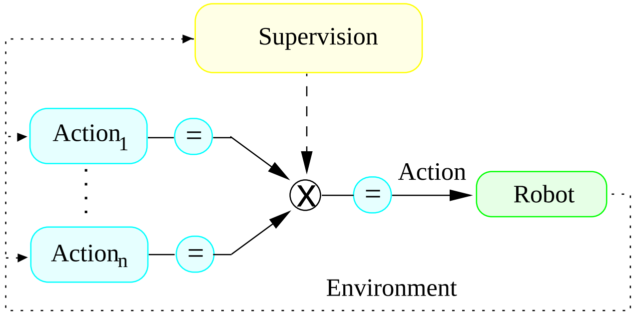


Figure 2.16: *Behaviours* based architecture.

Interestingly, the topology of the Figures 2.14, 2.15 and 2.16 agrees with recent biological studies on frogs. These studies shown that a small set of vector fields, identified in the frog's spine, generate all the motor behaviour of the frog, [Matarić, 1994]. Within the control field, the linearization of nonlinear systems around a set of relevant operating points and the design of a controller for each of the linear systems can be identified with the *actions* architecture of Figure 2.15. The switching between each controller is made at some neighbourhood of each operating point.

Chapter 3

Extending the state space structure

3.1 Introduction

The agent control problem, as it emerges from Chapter 2, assumes that a set of *a priori* user defined actions exists in such a way that, at every event, the supervisor block in the both the architectures of Figures 2.15 and 2.16 can choose an action among this set to execute. In general, the complexity of the environment, together with the kinematics of the robots involved, rend the definition of an *appropriate set of actions* into a difficult problem, eventually requiring a trial and error procedure.

The expression *appropriate set of actions* can be given a formal meaning using the concept of *generators of a group*. Following Massey, [Massey, 1977], pp. 89, a subset, S, of a group G, is said to generate G if every element of G can be written as a product of positive and negative powers[1] of elements of S. Therefore, the agent should be *a priori* equipped with a set of generators of the group A, or, given the local characteristic of the group, with multiple sets of generators, covering the whole set. In general, this is not the case and the set of user defined actions, A^p, does not coincide with a set of generators, thus enabling the generation of only a subset $A^g \subset A$, that includes all the states resulting from compositions containing elements of A^p. A^g may not contain enough elements to enable the robot to execute an arbitrary task.

In this chapter, the group structure of Chapter 2 is extended, aiming at overcoming the aforementioned potential problem of underspecification of the A^p set of actions and, simultaneously, aiming at defining a simple computational implementation of the proposed framework. In Section 3.2, an action is identified with an integral curve in Q and a linear combination of actions in A^p is used to expand the space of available actions. This particular form of expansion of A^p constrains A to be a linear space. A discussion on the relationships between the particular form of state composition induced by this linear combination and the more general definition of state composition proposed in

[1] The term *product* refers to the group operation. *Power* refers to the successive application of the group operation in a set of identical elements of the group.

Chapter 2 is presented. Some examples illustrate typical trajectories obtained with this method. In Section 3.3, the discussion on the state space expansion is generalized to the framework of Chapter 2. An extended structure, constrained to the detection of adequate events, is defined in A and its physical interpretation presented. The computational implementation of the architecture, for the case of a single agent, is presented in Section 3.4. Section 3.5 presents the decentralized cooperation model for multiple agents, supported in the single agent architecture previously defined. Section 3.6 presents a summary of the framework developed.

3.2 A linear space structure for A

This section discusses the increase in the number of available actions beyond the set A^p, *a priori* defined by the user, by a linear space structure.

The linear space structure proposed in this section is supported on two hypothesis that specify the nature of the actions and that introduce a simple computational procedure for their combination. These are:

- each action generates integral curves in Q (i.e., curves generated by vector fields in the tangent spaces of Q), and
- the actions in A^p are combined by a weighted sum to form new actions.

Proposition 3.2.1 (The linear space A^m) *Consider an agent with a set of user defined actions $A^p = \{\mathfrak{g}_i,\ i = 1, \cdots m\}$, where each action generates integral curves in Q. Moreover assume the linear independence of every pair of actions in A^p and that the manifold Q is embedded in $\mathbb{R}^n$. Define a new action by*

$$A^m = \{a^m : a^m = \sum_{i=1}^{m} \phi_i \mathfrak{g}_i\} \tag{3.1}$$

where ϕ_i are scalars (the sum is taken for identical values of the time parameter of the curves generated by the actions $\mathfrak{g}_i$). Then, A^m has a linear space structure.

Demonstration: The demonstration is divided in two parts. The first part considers the usual equality relation in $\mathbb{R}^n$ and follows from the definition of linear space. The second part considers the equality relation of Definition 2.3.4 and follows from the construction of a bounding region for the trajectories generated by the states using the actions defined by (3.1) for a fixed set of ϕ_i weights.

For arbitrary q, each of the $\mathfrak{g}_i$ actions generates trajectories inside the bounding region $B_{\mathfrak{g}_i}(q)$. From Definition 2.2.2, every $B_{\mathfrak{g}_i}(q)$ set is path connected and hence there exists a homotopy mapping between any two trajectories generated by $\mathfrak{g}_i(q)$. Representing the set of trajectories generated by $\mathfrak{g}_i(q)$ as $\{\mathfrak{g}_{i_k}(q),\ k=1,\dots\}$ and the homotopy mapping between any two trajectories $\mathfrak{g}_{i_r}(q)$ and $\mathfrak{g}_{i_s}(q)$ by $F_{rs}: T\times[0,1]\mapsto Q$ then $\mathfrak{g}_{i_r} = F_{rs}(\mathfrak{g}_i(q),0)$ and $\mathfrak{g}_{i_s} = F_{rs}(\mathfrak{g}_i(q),1)$.

Substituting the actions in (3.1) by homotopy mappings, a trajectory defined by $\sum_{i=1}^m \phi_i F_{rs}(\mathfrak{g}_i(q),\lambda_i)$ is generated, for every set of constants ϕ_i and λ_i. For each pair $(\mathfrak{g}_{i_r}(q),\mathfrak{g}_{i_s}(q))$, the continuous variation of each of the parameters $\lambda_1,\dots,\lambda_m$, spans a path connected region, denoted by $B^i_{rs}(q)$, that contains part of the trajectories generated by $\mathfrak{g}_i$. The region given by $B_{\mathfrak{g}_i}(q) = \cup_{rs} B^i_{rs}(q)$ contains the whole set of trajectories generated by $\mathfrak{g}_i$.

The trajectories generated by $a^m(q)$ are contained inside the smallest path connected region that contains $\cup_i B_{\mathfrak{g}_i}(q)$.

□

If A^p has enough linearly independent elements to form a basis for A then the whole A can be generated. In general, the $\mathfrak{g}_i$ actions are not the generators of the group $(A,+)$, as the ϕ_i weights may not be integers. However, a set of generators, eventually with infinite elements, can be constructed using elements of the form $\alpha\mathfrak{g}_i$, where $\alpha\in\mathbb{R}$.

The linear combination of actions, extended to consider the equality relation of Definition 2.3.4, is included in the extended control architecture represented in Figure 3.1. The output of the $\boxed{\text{Action}_i}$ blocks are curves that can be transformed into other curves by the ⊜ blocks that follow the actions. The *Supervision* block chooses the adequate weights ϕ that are next combined by the addition operation $(+)$ in $\mathbb{R}^n$ in the $\oplus$ block. Further transformations can be included in the ⊜ block at the output of the $\oplus$ block.

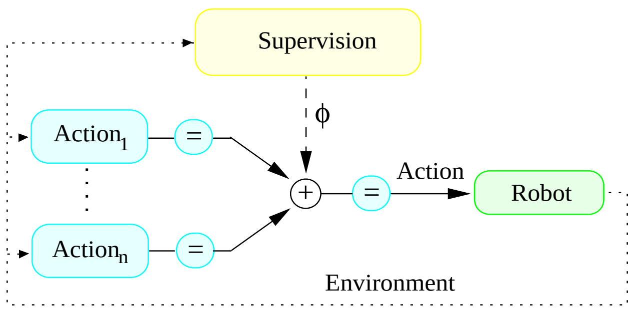


Figure 3.1: Control architecture supported in the linear combination of the actions.

Given an agent in a state $\alpha(q_\alpha) = \sum_i \mathfrak{g}_i(q_\alpha)\phi_i^\alpha$ and a state $\beta(q_\beta) = \sum_i \mathfrak{g}_i(q_\beta)\phi_i^\beta$ where to switch to, the supervisor simply changes from the ϕ_i^α weights to the ϕ_i^β weights. The simplification introduced by this structure in both the implementation of the *Supervision* block and the state composition hides complex problems, namely how to determine the set of the actions $\mathfrak{g}_i$ and how to choose the ϕ_i weights for a specific task.

In what concerns the architectures defined in Chapter 2 (see Figures 2.15 and 2.16), the implementation complexity lies mainly in the specification of the operation of the *Supervision* block and in the definition of the state composition operation, as the user is given freedom to define the actions in A^p. This discussion on the differences between architectures suggests the existence of a tradeoff in the distribution of the implementation complexity among the building blocks in each architecture. Further insight on this tradeoff is obtained by analyzing the relationships between the architecture resulting from the linear combination of actions (Figure 3.1) and the *behaviours* architecture (Figure 2.16). Consider a set $\{\mathfrak{g}_i,\ i = 1, \ldots, m\}$ of basis actions of A. The composition of any two states, $a_i(q_i)$, $a_j(q_j)$, described in terms of the basis actions of the linear space, is represented in terms of the framework defined in Chapter 2 by

$$a_i(q_i) \circ a_j(q_j) = \sum_{l=0}^{m} \mathfrak{g}_l(q_i)\phi_l^i \ \circ \ \sum_{l=0}^{m} \mathfrak{g}_l(q_j)\phi_l^j. \tag{3.2}$$

The left side of (3.2) re-written is terms of the basis actions yields

$$a_i(q_i) \circ a_j(q_j) = a_{i\circ j}(q_j) = \sum_{l=0}^{m} \mathfrak{g}_l(q_j)\gamma_l \tag{3.3}$$

for some scalars $\gamma_1, \ldots, \gamma_m$. Equating the right hand sides of (3.2) and (3.3)

$$\sum_{l=0}^{m} \mathfrak{g}_l(q_j)\gamma_l = \sum_{l=0}^{m} \mathfrak{g}_l(q_i)\phi_l^i \ \circ \ \sum_{l=0}^{m} \mathfrak{g}_l(q_j)\phi_l^j \tag{3.4}$$

which, introducing the operator $\chi_{\mathfrak{g}}(x, q) \equiv \sum_{l=0}^{m} \mathfrak{g}_l(q)x^l$, with $x = (x^1, \ldots, x^m)$, and $\phi_i = (\phi_1^i, \ldots, \phi_m^i)$, $\phi_j = (\phi_1^j, \ldots, \phi_m^j)$ and $\gamma = (\gamma_1, \ldots, \gamma_m)$, simplifies to

$$\chi_{\mathfrak{g}}(\phi_i, q_i) \circ \chi_{\mathfrak{g}}(\phi_j, q_j) = \chi_{\mathfrak{g}}(\gamma, q_j). \tag{3.5}$$

Without loosing generality, the *Supervision* block can be assumed to define the action, to execute at each event, through a binary operation, in $\mathbb{R}^m \times Q$, denoted by $\circledcirc$, such that

$$(\gamma, q_j) = (\phi_i, q_i) \circledcirc (\phi_j, q_j),$$

and the equality (3.5) can be written as

$$\chi_{\mathfrak{g}}(\phi_i, q_i) \circ \chi_{\mathfrak{g}}(\phi_j, q_j) = \chi_{\mathfrak{g}}((\phi_i, q_i) \circledcirc (\phi_j, q_j)) \tag{3.6}$$

which corresponds to the definition of homomorphism between the space of actions and a space of weights. Thus, the architectures of Figures 2.16 and 3.1 are homomorphic up to the operation $\circledcirc$. Moreover, expression (3.6) suggests the transfer of complexity from the state composition operation to the $\circledcirc$ operation and hence to the *Supervision* block, which implements it.

The idea underlying the linear combination of actions is the identification between actions and vector fields on M. As the *Supervision-Execution* diagram of Figure 2.14 suggests, the role of the

Supervision block is to adjust the commands to the *Execution* block, given the available information on the state of the agent and on its interactions with the environment. In general, the rate at which these adjustments are required prevents an action of lasting in execution for long periods of time and thus the corresponding trajectories can be considered as being generated by a vector field, defined in the neighbourhood of the state initial configuration. Therefore, the identification between actions and vector fields is supported on the assumption that the *Supervision* block acts as a controller, always being adjusting the controls that it generates.

The weighted sum of vector fields is a particular case of a general continuous dynamical system

$$\dot{x} = f(x, u), \tag{3.7}$$

where the state variable, x, lies in an m-dimensional smooth manifold in $\mathbb{R}^n$, M, $u \in \mathbb{R}^p$ with $p \leq n$, is the vector of control variables, and $f(.)$ is a vector field in the tangent bundle of M. In the sequel, system (3.7) is considered to be controllable, but not necessarily locally controllable.

Under the controllability assumption, the tangent space at each point x of M is spanned by a set of vector fields, $\mathfrak{g}_1(x), \cdots, \mathfrak{g}_m(x)$, [Sussmann and Jurdjevic, 1972], each being tangent to an integral curve passing at x. Thus, at each element x, it is possible to travel along any direction spanned by a combination of the vector fields $\mathfrak{g}_i$, i.e.,

$$\dot{x} = [\mathfrak{g}_1(x), \cdots, \mathfrak{g}_m(x)] \cdot \nu \tag{3.8}$$

where $\nu = [\nu_1, \ldots, \nu_m]' \in \mathbb{R}^m$ is a new vector of control variables. Alternatively, by an appropriate combination of these vector fields, it is possible to generate the integral curve defined by (3.7).

Using the controllability rank condition (see [Hermann and Krener, 1977]) and the geometrical interpretation for the Lie bracket, each of the $\mathfrak{g}_i(x)$ vector fields can be interpreted as a particular sequence of infinitesimal motions in M, along all the possible motion directions defined by $f(x, u)$ (one motion direction for each value of u). The set $\{\mathfrak{g}_i(x), i = 1, \ldots, m\}$ represents, at each point x, a basis for the smallest subalgebra of the Lie algebra of vector fields on M (known as the *Control Lie Algebra* (CLA) of vector fields, [Sordalen, 1993]).

Assuming the controllability of the robot, the correspondence between the elements defined above in the classical control setup and the elements defined in the context of the framework proposed in the thesis, is represented in Table 3.1. Figure 3.2 illustrates this correspondence in a feedback loop scheme.

Determining a basis for the CLA on Q is not compatible with the user's perspective of Chapter 1, in which the control methodology must cope with the uncertainty in the *a priori* knowledge about both the robot and the environment. Moreover, determining a basis for the CLA on Q is a non

Action	$\leftrightarrow$	Element of a CLA basis
State composition	$\leftrightarrow$	Sum in $\mathbb{R}^n$
Supervision	$\leftrightarrow$	Controller K
Task	$\leftrightarrow$	Reference x_{ref}

Table 3.1: Identification between a dynamical system and the presented framework.

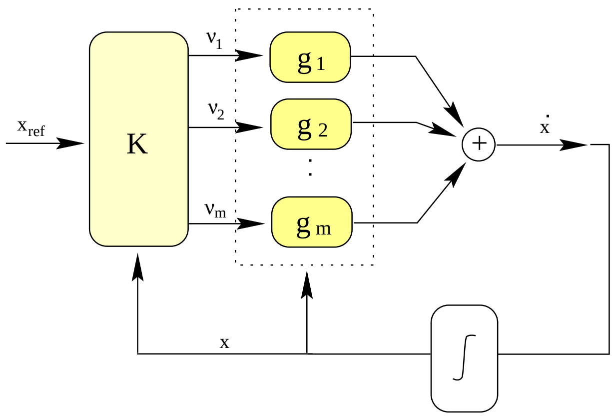


Figure 3.2: Feedback control loop derived from the linear combination of actions.

decidable problem, [Laumond, 1993], and hence the difficulty in the action synthesis remains when using the linear space structure. This represents a strong motivation to search for alternative ways to synthesize the actions.

As an application example of the linear combination of actions, consider the cart mobile platform to which a task of the form $a_e([x_{\text{task}}, y_{\text{task}}]')$, (recall that a_e is the null action) is assigned. The actions in A^p, specified as vector fields in the xy plane, are

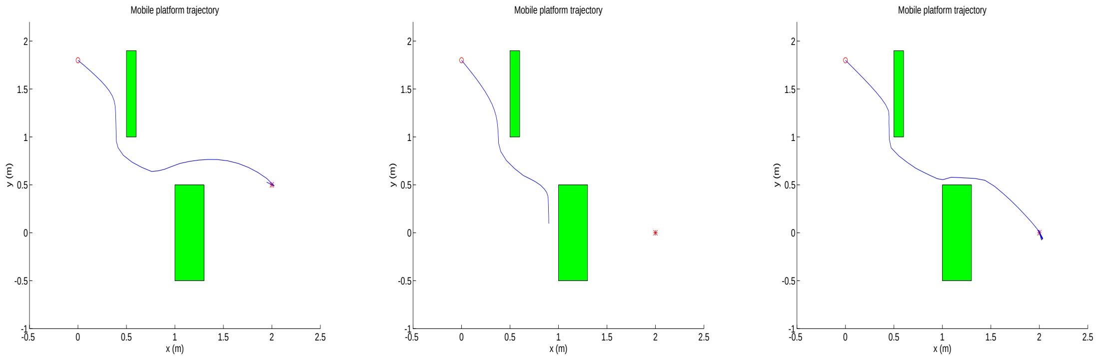
$$\textit{Go straight to goal} \qquad a_{\text{g}} = \begin{bmatrix} x_{task} - x \\ y_{task} - y \end{bmatrix},$$

where (x, y) is the current mobile platform position, and

$$\textit{Avoid obstacle} \qquad a_{\text{oa}} = \begin{bmatrix} -\alpha_x/d_o \\ -\alpha_y/d_o \end{bmatrix},$$

where (α_x, α_y) represents the direction of the minimum distance, d_o, to any obstacle.

Figure 3.3 shows three typical situations, obtained for two different tasks and different values for (ϕ_1, ϕ_2). Plot (a) refers to a task located at $(2, 0.5)$ whereas plots (b) and (c) refer to a task located at $(2, 0)$. The starting and goal locations are marked, respectively, with the symbols $\circ$ and $*$. The shadowed areas represent the obstacles. In this example the *Supervision* block is a single state Finite State Machine (FSM) as the weights ϕ_1 and ϕ_2 are constant during the entire execution of the task. The agent in this simulation has access to perfect information, i.e., it is implicitly assumed that the values of x, y, α_x, α_y and d_0 are available and are always perfect estimates of the correspondings real values. The robot has no physical dimensions.



(a) $(\phi_1, \phi_2) = (0.1, 1)$ (b) $(\phi_1, \phi_2) = (0.1, 1)$ (c) $(\phi_1, \phi_2) = (0.04, 1)$

Figure 3.3: Two different tasks, using fixed linear combination of the actions a_{oa} and a_{g}.

From cases (a) and (b) it is clear that a single set of action combination weights always is not appropriate as the task in (b) can not be completed (the agent enters a local minimum). As represented in (c), by a proper change of weights the agent is able to execute the task. The perfect information assumption, implicitly used when defining the actions, will seldom be verified when operating in a realistic environment. A number of factors, namely the complexity of the environment, the difficulty in the acquisition of proper sensorial data and the sensor characteristics, compete to make the definition of an appropriate set of ϕ weights a difficult problem.

In terms of the framework already defined in Chapter 2, each of the trajectories obtained in the experiments of Figure 3.3 corresponds to a single state of the agent. The *Supervision* block maintains the ϕ_i weights constant during the period of time necessary to the task execution. Whenever the trajectory generated by an action violates the kinematic constraints, the operator $\otimes$, defined in Section 2.6, forces the mobile platform to perform a pure rotation in order to maintain an adequate heading.

The next example uses a *Supervision* block based in a two state FSM without the action combination of Proposition 3.2.1. Each of the FSM states triggers one of the actions defined as in the previous example, $a_{\mathrm{g}} \equiv$ *go straight to the goal* and $a_{\mathrm{oa}} \equiv$ *avoid obstacle*, according to the following algorithm:

1 if the task is occluded from the actual location,

1.1 compute the smallest visibility polygon (a sequence of which is shown as solid lines in Figure 3.4) containing the obstacle (represented by the shadowed area), with the physical dimensions augmented by a safety margin (shown as a dashed line),

1.2 select the side of the polygon whose orientation is closer to the direction of the task location and use a dumping function to force a convergence between the vehicle orientation and this direction,

2 otherwise, use the direction of the task location as the reference in the control of the vehicle orientation.

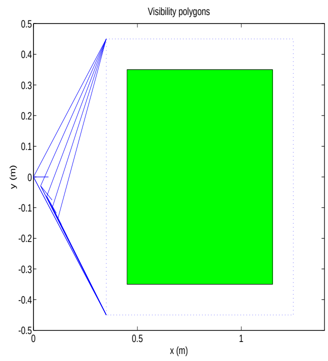
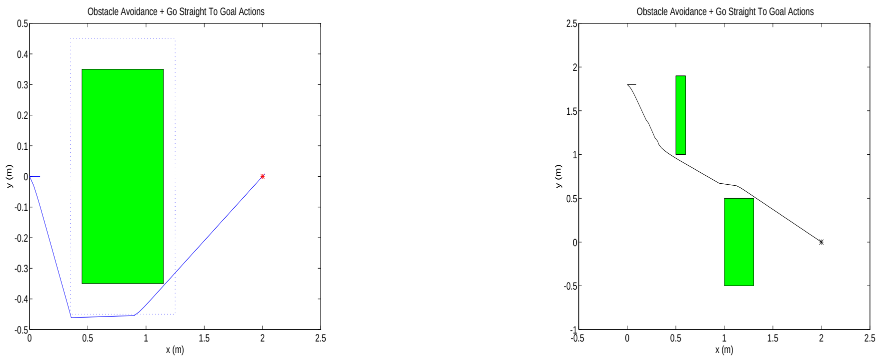


Figure 3.4: Sequence of visibility polygons for obstacle avoidance.

The initial configuration of each state sent to the *Execution* block by the supervisor corresponds to the configuration at the switching time instant. The link trajectories are pure rotations. Figure 3.5 shows examples of trajectories for two different tasks. As in the previous example, the robot has no physical dimensions and it is assumed that the agent has access to a perfect set of sensorial data. The goal is indicated by the $*$ symbol and the obstacles are represented by the shadowed areas. The short horizontal line segment at the beginning of each trajectory indicates the initial orientation of the robot. In the leftmost plot, the *obstacle avoidance* action determines the first two phases of the trajectory while the *go straight to goal* action controls the vehicle during the third, and last, phase. The changes of action during a task execution have a clear effect in the trajectory. In the rightmost plot, the same task of the examples (b) and (c) of Figure 3.3 is assigned to the agent. As in the leftmost plot, the changes of action are clearly visible in the trajectory.

A practical consequence of the use of an algorithmic structure for the *obstacle avoidance* action is illustrated by the non-smooth behaviour at the first break point of the trajectory in Figure 3.5a. Although smooth trajectories are desirable, emergency situations, requiring abrupt changes of motion direction, can always be created and hence smoothness can not be considered as a fundamental requirement for the trajectories to be performed by an agent.



(a) (b)

Figure 3.5: Examples of execution of two different tasks using two actions, a_g and a_{oa}.

3.3 Maneuvering in T - A constrained algebraic approach

In this section the maneuvering in T is addressed. In addition to the usual robotics meaning, expressed in Section 2.4, the term *maneuver* and its variants are used to indicate transitions between states, eventually including additional transitions between intermediate states. These transitions must be performed using the limited number of actions in the set A^p and hence it is of interest to determine the minimum number of actions necessary for an agent to be controllable. Using the identification between actions and vector fields, a comparative discussion between the algebraic structure used by Control theory to define a minimal set of vector fields that represents the motion capabilities of a robot and the group structure defined in Chapter 2 is presented. This discussion leads to the definition of an additional operation in A, through which the characteristics of the actions in A^p can be modified such that a specific task can be executed.

For the car-like mobile platform (see Appendix C.2 for the kinematics), with steering angle in the interval, $-\pi/2 < \Phi < \pi/2$, a minimal set is composed by straight lines and arcs of circumference (see [Laumond, 1993]). For the cart mobile platform, two examples of minimal set of actions are:

- pure rotations (clockwise and counterclockwise) together with pure translations (forward and backward) and,
- arcs of circumference and straight lines, with the shortest path between any two arbitrary configurations being a finite sequence of the actions in this minimal set, [Jacobs and Canny, 1993, Sordalen, 1993].

Figure 3.6 illustrates two examples of trajectories, for a cart robot, using arcs of circumference and straight lines. In both plots, two vectors (displayed as straight lines delimited by the symbols $\circ$ and $+$) represent the robot initial and final configurations. The region spanned by the maneuvers in this example may not be compatible with existing obstacles. An alternative is to use the minimal set with pure rotations and pure translations. In Appendix E a description of a planner using these concepts is presented, along with further examples of trajectories.

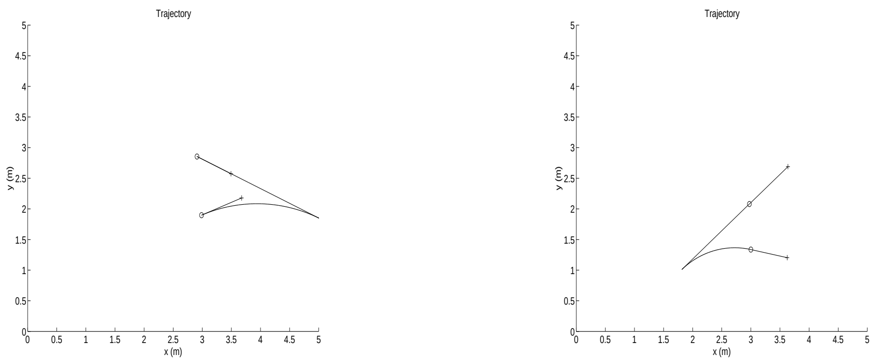


Figure 3.6: Examples of trajectories using arcs of circumference and straight lines.

The existence of a minimal set of actions, $\hat{a}_1, \ldots, \hat{a}_n$, implies that any state can be reached using a finite/infinite sequence of states, each defined using only the actions in the minimal set, and a finite/infinite set of initial configurations q_{ij}. This statement is expressed by

$$\begin{array}{rcl}\forall a \in A,\ q \in Q,\ \ \exists q_{11}, \ldots, q_{mn} \in Q, \quad a(q) & = & \hat{a}_1(q_{11}) \circ \hat{a}_2(q_{12}) \circ \ldots \circ \hat{a}_n(q_{1n}) \circ \ldots \\ & & \hat{a}_1(q_{21}) \circ \hat{a}_2(q_{22}) \circ \ldots \circ \hat{a}_n(q_{2n}) \circ \ldots \\ & & \vdots \\ & & \hat{a}_1(q_{m1}) \circ \hat{a}_2(q_{m2}) \circ \ldots \circ \hat{a}_n(q_{mn})\end{array} \tag{3.9}$$

for some finite n and finite/infinite m.

Identifying an action with a vector field leads to the use of standard concepts of control theory to define a minimal set of actions. Consider the smooth manifold, Q, and an arbitrary trajectory defined in it. As already expressed in Section 3.2, any trajectory in Q can be generated by a basis of vector fields for the Control Lie Algebra of vector fields on Q. The number of elements in this basis constitutes the minimum number of vector fields to generate arbitrary directions of motion on Q.

The following example illustrates the computation of the Lie algebra of vector fields with Q being the SE(2) matrix space in which the robot configuration is of the form[2]

$$q = \left[\begin{array}{ccc} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{array}\right]. \tag{3.10}$$

[2]The SE(2) space is a group under the usual matrix multiplication and a 3-parameter Lie group.

Every curve defined in Q (using the matrix representation (3.10)) can be generated by considering the possible infinitesimal changes for the elements of Q (the derivatives are taken relative to time), yielding,

$$\dot{q} = q \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\theta} + q \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{x} + q \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \dot{y} = q\hat{e}_1\dot{\theta} + q\hat{e}_2\dot{x} + q\hat{e}_3\dot{y}. \tag{3.11}$$

Each of the three terms in (3.11) contains the information on how the variations in each of the free parameters, (x, y, θ), affects the whole q. The expression (3.11) defines an algebra on a vector space defined on the space of 3x3 matrices of the form (3.10). The $\hat{e}_1, \hat{e}_2, \hat{e}_3$ vectors are elements of that vector space, and are called a basis of the vector space. Moreover, since the SE(2) space is a Lie group under the matrix multiplication, the above algebra is called a Lie algebra on SE(2).

The Lie algebra basis, $\hat{e}_1, \hat{e}_2, \hat{e}_3$, specifies the basis directions of motion in the SE(2) space. For more complex robots, such as the cart[3], given that $\dot{q} = q\hat{e}_1\omega + q(\hat{e}_2\cos(\theta) + \hat{e}_3\sin(\theta))v$, where ω and v represent the angular and linear velocities in the robot reference frame, some of these basis directions can not be chosen independently. In problems with these characteristics, i.e., where the motion directions can not be chosen arbitrarily, it may be necessary to consider sequences of displacements along the allowed motion directions to move the robot between any two configurations[4]. For the cart robot it is clear that sequences of pure translations and rotations are enough to drive the robot between any two configurations.

The basis of the CLA of vector fields on Q represents the minimal set of vector fields on Q for a robot to be globally weakly controllable. In the framework proposed in this thesis, the counterpart of the minimal set of vector fields is the minimal set of actions, constituting a set of generators of A.

The transition between arbitrary states, $a_j(q_j)$, $a_i(q_i)$, is triggered by the events that determine the agent's depart from $a_j(q_j)$ to start the link trajectory, and the depart from the link trajectory to enter $a_i(q_i)$. The positioning of these events in the regions bounding the states determines the four possibilities for the trajectories generated by $a_i(q_i) \circ a_j(q_j)$, represented in Table 3.2 as sequences of actions.

Assuming that $a_i(q_i)$ and $a_j(q_j)$ verify the minimum space requirements stated in Proposition 2.4.2, the existence of the $\tilde{a}_i, \tilde{a}_j$ actions is assured by setting (for example) $\tilde{a}_i = a_i$ and $\tilde{a}_j = a_j$, as any maneuver necessary to the link trajectories is contained inside the region bounded by $B_{a_i}(q_i) \cup B_{a_j}(q_j)$. In type 1 maneuvers, the execution of the action a_i begins at the configuration

[3] This is an example of a robot with a C-space in the $SE(2)$ class but with its velocity space being a 2-dimensional manifold, thus having constrained motion directions.

[4] This statement assumes, for the sake of simplicity, the controllability of the robot.

Type	Action sequence
1	$a_j \to a_i,$
2	$a_j \to \tilde{a}_i^{-1} \to a_i,$
3	$a_j \to \tilde{a}_j^{-1} \to a_i,$
4	$a_j \to \tilde{a}_j^{-1} \to \tilde{a}_i^{-1} \to a_i.$

Table 3.2: Sequences of actions to maneuver in T.

where the execution of action a_j ended, or the link path between the states can be executed using one or both of the actions a_i, a_j. In type 2 maneuvers, the agent enters $B_{a_i}(q_i)$ in a region such that it is necessary to "pull it back" to a proper neighbourhood of q_i. In type 3 maneuvers, the agent needs to be "pulled back" from $B_{a_j}(q_j)$ to enter $B_{a_i}(q_i)$. In type 4 maneuvers, the agent is "pulled back" from $B_{a_j})(q_j)$ till the trajectory enters $B_{a_i}(q_i)$ and again "pulled back" to have the agent entering the state $a_i(q_i)$ at a proper neighbourhood of configuration q_i.

Given the user's perspective defined in Chapter 1, the state bounding sets, allocated to each action, may not contain enough space to account for the constraints imposed by the maneuvers defined in Table 3.2. For example, consider a cart with a lower bound on the turning radius and an unexperienced user defining a set A^p containing tight curves. Furthermore, the supervisor may specify initial configurations such that the minimum overlap requirement is not verified. A first solution to have the generated trajectories behaving according the state composition operation is to define a number of intermediate states such that the generated trajectories always lie inside the corresponding bounding regions. When considered over the entire state space, this solution leads to the definition of the minimum number of actions necessary to such intermediate states. An alternative solution, that avoids the extensive state space search, is to expand parts of the region $B_{a_i}(q_i) \cup B_{a_j}(q_j)$ such that the maneuvers of Table 3.2 can be used.

The expansion of $B_{a_i}(q_i) \cup B_{a_j}(q_j)$ such that the state composition $a_i(q_i) \circ a_j(q_j)$ is meaningful (i.e., verifying the conditions in Definition 2.4.1 and the space requirements expressed by Proposition 2.4.2) is made around the region $B_{a_i}(q_i) \cap B_{a_j}(q_j)$. The expansion of the region bounding an isolated state, $a_i(q_i)$, is considered as the expansion of the region bounding the state composition $a_e(q_e) \circ a_i(q_i)$, for some q_e, and thus is made around $B_{a_e}(q_e) \cap B_{a_i}(q_i)$.

Definition 3.3.1 (The state expansion operation $\boxtimes$) *Let a_e denote the null action of the A group of transformations and let $a_i(q_i)$ and $a_j(q_j)$ be two arbitrary states such that $B_{a_i}(q_i) \cap B_{a_j}(q_j) \neq \emptyset$. The state defined by $a_k(q_j) = a_i(q_i) \boxtimes a_j(q_j)$ (meaning that $a_j(q_j)$ is expanded by $a_i(q_i)$) verifies*

$$a_k(q_j) \subset B_{a_i}(q_i) \cup B_{a_e}(q_k) \cup B_{a_j}(q_j)$$
$$a_k(q_j) \leftarrow a_j(q_j)$$

for some $q_k \in B_{a_i}(q_i) \cap B_{a_j}(q_j)$ and $a_k \in A$.

The first condition in Definition 3.3.1 specifies the additional amount of space introduced by the expansion. The second condition states that, after some time, the expanded state $a_k(q_j)$ can be substituted by the original state $a_j(q_j)$. Together, the two conditions indicate that the expanded state generates new trajectories that may travel in the additional space but that after some time behave as the trajectories of the original, unexpanded, state.

Everything that can be done in T is expressed by the operations of state composition and state expansion. State composition determines trajectories fitting inside the bounding sets. State expansion enlarges the bounding states so that a particular trajectory fits inside.

The user's perspective of Chapter 1 assumes that all of the user's relevant knowledge about the agent and environment is already incorporated in the actions and thus the role of the supervisor is to detect the adequate events and define the corresponding state transitions. Any exception condition, such as a lack of space to perform a state transition, is handled by the *Execution* block without any direct intervention by the user. The exact placement of the $\boxtimes$ operation inside the *Execution* block depends on its execution precedence properties relative to the state composition operation. The following proposition verifies two of such properties.

Proposition 3.3.1 (A non-proper semiring structure on A) *Constrained to the detection of adequate events, the state expansion operation is associative and distributive relative to the state composition and hence A has a non-proper semiring structure.*

Demonstration: The demonstration proceeds by verifying the referred properties for $(A, \circ, \boxtimes)$.

Given any three states, $a_i(q_i), a_j(q_j), a_k(q_k)$, the associative property of the state expansion operation, defined by

$$a_i(q_i) \boxtimes (a_j(q_j) \boxtimes a_k(q_k)) = (a_i(q_i) \boxtimes a_j(q_j)) \boxtimes a_k(q_k), \tag{3.12}$$

indicates that the order in which multiple state expansions are performed is not relevant.

The region bounding the left hand side of (3.12) is

$$B_{a_i}(q_i) \cup B_{a_e}(q_{w_1}) \cup B_{a_j}(q_j) \cup B_{a_e}(q_{w_2}) \cup B_{a_k}(q_k) \tag{3.13}$$

for some

$$\begin{array}{l} q_{w_1} \in B_{a_i}(q_i) \cap (B_{a_j}(q_j) \cup B_{a_e}(q_{w_2}) \cup B_{a_k}(q_k)) \\ q_{w_2} \in B_{a_j}(q_j) \cap B_{a_k}(q_k). \end{array}$$

The right hand side of (3.12) generates trajectories bounded by

$$B_{a_i}(q_i) \cup B_{a_e}(q_{w_3}) \cup B_{a_j}(q_j) \cup B_{a_e}(q_{w_4}) \cup B_{a_k}(q_k) \tag{3.14}$$

for some

$$\begin{array}{l} q_{w_3} \in B_{a_i}(q_i) \cap B_{a_j}(q_j) \\ q_{w_4} \in (B_{a_i}(q_i) \cup B_{a_e}(q_{w_3}) \cup B_{a_j}(q_j)) \cap B_{a_k}(q_k). \end{array}$$

The bounding regions (3.13) and (3.14) are identical provided that

$$B_{a_e}(q_{w_1}) \cup B_{a_e}(q_{w_2}) = B_{a_e}(q_{w_3}) \cup B_{a_e}(q_{w_4})$$

for which a sufficient condition is given by

$$\begin{aligned} q_{w_1} &= q_{w_3} \\ q_{w_2} &= q_{w_4}, \end{aligned}$$

and thus both of the two state expansions is each side of (3.12) are constrained to occur at well defined regions, which amounts to say that the *Supervision* block must detect an adequate event such that those expansions can be triggered.

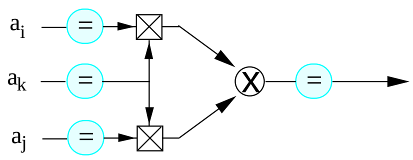
The temporal similarity required by the equality relation in (3.12) is verified as by Definition 3.3.1 both sides can be substituted by $a_k(q_k)$.

The precedence of the state expansion relative to the state composition operation is expressed by the left and right distributive properties, defined by

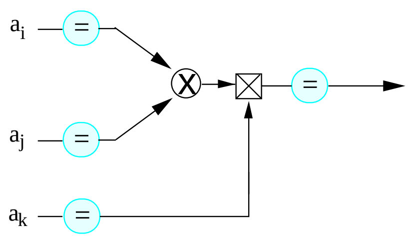
$$\text{Left distributive} \quad a_k(q_k) \boxtimes (a_i(q_i) \circ a_j(q_j)) = (a_k(q_k) \boxtimes a_i(q_i)) \circ (a_k(q_k) \boxtimes a_j(q_j)) \tag{3.15}$$

$$\text{Right distributive} \quad (a_i(q_i) \circ a_j(q_j)) \boxtimes a_k(q_k) = (a_i(q_i) \boxtimes a_k(q_k)) \circ (a_j(q_j) \boxtimes a_k(q_k)) \tag{3.16}$$

Figure 3.7 illustrates the details of two possible locations for the $\boxtimes$ operation inside the control architecture.



Case (a) $(a_k(q_k) \boxtimes a_i(q_i)) \circ (a_k(q_k) \boxtimes a_j(q_j))$



Case (b) $a_k(q_k) \boxtimes (a_i(q_i) \circ a_j(q_j))$ or $(a_i(q_i) \circ a_j(q_j)) \boxtimes a_k(q_k)$

Figure 3.7: Graphical interpretation of the distributive properties.

The left hand side of the left distributive property generates trajectories inside

$$B_{a_k}(q_k) \cup B_{a_e}(q_{w_1}) \cup B_{a_i}(q_i) \cup B_{a_j}(q_j) \tag{3.17}$$

for some

$$q_{w_1} \in B_{a_k}(q_k) \cap (B_{a_i}(q_i) \cup B_{a_j}(q_j))$$

whereas the right hand side generates trajectories inside

$$B_{a_k}(q_k) \cup B_{a_e}(q_{w_2}) \cup B_{a_i}(q_i) \cup B_{a_j}(q_j) \cup B_{a_e}(q_{w_3}) \tag{3.18}$$

for some

$$q_{w_2} \in B_{a_k}(q_k) \cap B_{a_i}(q_i)$$
$$q_{w_3} \in B_{a_k}(q_k) \cap B_{a_j}(q_j).$$

The bounding regions (3.17) and (3.18) are identical provided that

$$B_{a_e}(q_{w_1}) = B_{a_e}(q_{w_2}) \cup B_{a_e}(q_{w_3})$$

for which it is necessary and sufficient that q_{w_1}, q_{w_2} and q_{w_3} lie in the same region. This is accomplished by making, for example, $q_{w_1} = q_{w_2} = q_{w_3}$, meaning that the three expansions considered take place around the same point. Once more, the detection of an appropriate event, indicating the location of the expansion, is required.

The temporal similarity required by the equality relation is verified as, by Definition 3.3.1, both sides of (3.15) can be substituted by $a_i(q_i) \circ a_j(q_j)$.

Similarly, the right distributive property leads also to the conclusion that the three state expansion must take place at the same region requiring the detection of the corresponding triggering event.

Given the group structure of $(A, \circ)$ and the semigroup structure of $(A, \boxtimes)$, $(A, \circ, \boxtimes)$ is a non-proper semiring, assuming the detection of the adequate events to indicate the location for the state expansion.

$\square$

Therefore, assuming that the constraints on event detection referred in the above proposition are verified, the *behaviours* architecture can be further specialized as detailed either in Figure 3.7a or in Figure 3.7b. Even though the regions bounding the trajectories are identical in the two situations the trajectories may differ. In Figure 3.7a the state composition is performed on the states with the expanded bounding regions, whereas in Figure 3.7b the state expansion is performed to fit the bounding region to the trajectory obtained from the state composition.

The associative property of the $\boxtimes$ operation expresses the invariance of sequences of state expansions relative to the order in which they are performed. Thus, it can be used by the supervisor during the planning of a task execution.

3.4 Computational implementation

This section describes the computational implementation of the basis building blocks of the architecture of Figure 2.16, supported on the framework with the extended structure of Section 3.3. This implementation is tailored to the robotic devices detailed in Chapter 4 and used in the experiments performed in Chapter 5.

The user's perspective of Chapter 1 requires that an easy-to-use methodology for the definition of the state bounding regions is available. Furthermore, a simple computational implementation is a desirable requirement, in order to rend possible a real time application. The approach to develop this methodology is composed by the following topics:

1. Consider the robot C-space represented by the direct sum of two subspaces, $Q = E \oplus U$. Denoting by $\mathbb{P}_E : Q \mapsto E$ the projection mapping onto E, assume that the state bounding set $B_{a_i}(q_i)$ is defined such that $B_{a_i}(q_i) \subset \mathbb{B}_{a_i}(q_i)$, where

$$\mathbb{B}_{a_i}(q_i) = \mathbb{P}_E(B_{a_i}(q_i)) \times U. \tag{3.19}$$

   By using $\mathbb{B}_{a_i}(q_i)$ instead of $B_{a_i}(q_i)$, the definition of the actions in A^p is simplified, as the number of degrees of freedom is reduced. The space E is problem dependent. For the cart mobile platform considered in the thesis $E = \{(x, y)\}$ and $U = \{\theta\}$.

2. Given an arbitrary state $a_i(q)$, define a covering $\{B^p(q^p),\ p = 0, \dots\}$ for $\mathbb{B}_{a_i}(q)$, where each $B^p(q^p)$ is a user defined subset, path connected and bounded by a closed subset of Q. For instance, if an agent is in state $a_i(q)$, the $B^0(q^0), B^1(q^1), \dots$ sets verify, for $p = 0, 1, \dots,$

$$\begin{array}{l} q^p \in \mathbb{B}_{a_i}(q) \\ \exists\, r \neq p \ \ : \ B^p(q^p) \cap B^r(q^r) \neq \emptyset \\ \cup_p B^p(q^p) \supseteq \mathbb{B}_{a_i}(q). \end{array} \tag{3.20}$$

3. Complex interactions between the agent and the environment may require state bounding sets which might be difficult to visualize *a priori*. Using the covering procedure of step 2, the state bounding set can be incrementally defined by sequentially joining the $B^p(q^p)$ subsets. Each time the agent goes outside an element $B^p(q^p)$ into a region not yet covered, a new element $B^{p+1}(q^{p+1})$ is joined to the covering. During a multiple task mission, a new covering is started each time a task is finished and the agent starts moving towards the next task.

4. The trajectories generated by $a(q)$ can be partitioned into a sequence of small parts, denoted by $a^p(q^p)$, each of which is contained in a subset $B^p(q^p)$. While inside $B^p(q^p)$, the agent follows the trajectory generated by $a^p(q^p)$, switching to the trajectory generated by $a^{p+1}(q^{p+1})$ when it enters the subset $B^{p+1}(q^{p+1})$. The $a^p(q^p)$ are called pseudo-states to emphasize the fact that they represent only parts of trajectories that eventually may not be completely contained inside the $B^p(q^p)$ subsets.

The state composition operation is also used to represent the trajectories generated by sequences of pseudo-states. This does not imply any lack of generality as the $B^p(q^p)$ can be augmented with the necessary space to verify the minimum overlap requirement. As for the state expansion, given any two pseudo-states $a^p(q^p), a^s(q^s)$ such that $B^p(q^p) \cap B^s(q^s) \neq \emptyset$, the pseudo-state $a^{p \boxtimes s}(q^s) = a^p(q^p) \boxtimes a^s(q^s)$ generates trajectories always inside $B^p(q^p) \cup B^s(q^s)$ (a consequence of the pseudo-state definition). However, if necessary the trajectories can occupy the expanded region $B_{a^p}(q^p) \cup B_{a_e}(q_w) \cup B_{a^s}(q^s)$ that bounds the pseudo-state $a^{p \boxtimes s}(q^s)$, for some $q_w \in B^p(q^p) \cap B^s(q^s)$. In addition to the state expansions defined by the supervisor, during multiple executions of the same task a process similar to state expansion may occur. In fact, different coverings contain trajectories that lead to the execution of the assigned task. If a trajectory is contained in two different coverings then each of the coverings can be considered as an expansion of the other.

Table 3.3 summarizes the algorithm outlined in the above topics.

1 Set the index of the covering element the agent is currently in, $p = 0$, and represent by n_c the number of elements in the covering (at the start of the first run $n_c = 0$).

2 If first run, define $a^p(q^p) = a(q)$, where $a(q)$ is the agent's present state.

Otherwise,

- 2.1 $\exists\, w \leq n_c : a(q)\big|_0 \in B^w(q^w)$,
- 2.2 let $p = w$.

3 Define the mapping $F : Q \mapsto A$ such that $F(q^p) = a^p$.

4 If the task is completed terminate the algorithm.

5 If $\exists\, w \leq n_c : a^p(q^p)\big|_k \in B^w(q^w)$ then

- 5.1 let $p = w$,
- 5.2 if not already in pseudo-state $a^p(q^p)$ switch to $a^p(q^p)$, where $a^p = F(q^w)$.

6 Otherwise,

- 6.1 let $p = p + 1$,
- 6.2 let $q^p = a^{p-1}(q^{p-1})\big|_k$,
- 6.3 let $F(q^p) = a^p = a$, where $a(q^p)$ is the new state, defined by the *Supervision* block, where the agent must switch to,
- 6.4 define a new region $B^p(q^p)$.

7 Let $n_c = \max\{n_c, p\}$.

8 Go to step 4.

Table 3.3: The covering build algorithm.

The algorithm incrementally generates a sequence of pseudo-states, $a^p(q^p),\ p = 0, 1, \ldots,$ such that the trajectories leading to the task execution are generated by the composition of the pseudo-states $a^r(q^r) \circ a^{r-1}(q^{r-1}) \circ \cdots \circ a^1(q^1) \circ a^0(q^0)$, and bounded by the cover set $\cup_{p=0}^{r} B^p(q^p)$. Furthermore, the algorithm introduces a mapping, denoted by F, that records the a^p actions taken at each enlargement of the covering, so that the agent can use the past experience therein registered in subsequent executions of the task. Figure 3.8 shows the detail of the *behaviours* architecture with the inclusion of the elements of Table 3.3. The dashed lines represent the control lines from the supervisor, through which the states to be composed and expanded are chosen. The state expansion presented in Figure 3.7b is considered.

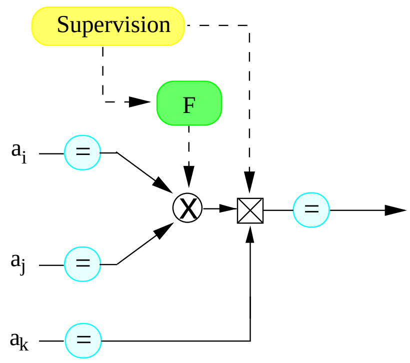


Figure 3.8: Detail of the *behaviours* architecture, following Table 3.3.

The algorithm of Table 3.3 implicitly assumes that the supervisor generates an adequate sequence of states to be followed by the agent. However, given the constraints on the $B^p(q^p)$ subsets expressed by (3.20), the user is allowed to define, the covering created by the algorithm in Table 3.3, may not result in a valid state bounding set, in the sense that step 5 may occasionally cause the agent to leave the covering. If the agent operates in a cluttered workspace the subsequent extension of the covering may not be appropriate. Therefore, once a solution to the execution of a task is found it may be advantageous to constrain the a^p actions such that any attempt to move out of the covering already established is avoided in future executions of the task. In a sense, these constraints transform the covering constructed by the algorithm of Table 3.3 into a valid state bounding region. The following proposition defines sufficient constraints for the a^p to generate trajectories contained inside the referred covering. It must be emphasized that constraining the a^p actions prevents the algorithm of Table 3.3 of generating alternative coverings for the execution of a given task.

Proposition 3.4.1 (Convergence of the trajectories generated by the $a^p(q)$) *Assume that, for an arbitrary task, the* Supervision *block in the* behavioural *control architecture defines a sequence of states such that the composition $a^p(q^p) \circ a^{p-1}(q^{p-1}) \circ \cdots a^1(q^1) \circ a^0(q^0)$ leads to the execution of the task. The motion direction at $q^w \in B^w(q^w)$, $w = 0, \ldots, p$, is defined by the pseudo-state $a^w(q^w)$ and, by the above assumption, generates a portion of trajectory that leads to the execution of the assigned task. Define the distance between a configuration q and a set K, both in Q, by $d_K(q) = \inf_{z \in K} \|q - z\|$.*

If each action a^w has the following property,

$$\forall\, q \in B^w(q^w),\ \ \exists\, k \geq 0\ : d_{a^w(q^w) \cap B^w(q^w)}(a^w(q)\big|_k) = 0, \tag{3.21}$$

the algorithm of Table 3.3 iteratively defines a covering for a subset bounding a state that leads to the execution of the assigned task.

Demonstration: The state leading to the task execution, $a(q) = a^p(q^p) \circ a^{p-1}(q^{p-1}) \circ \cdots a^1(q^1) \circ a^0(q^0)$, is bounded by some subset of the covering $\cup_{w=0}^{p_i} B^w(q^w)$ containing all the portions of trajectories generated by the pseudo-states that lie inside the cover subsets, i.e., defined by $\cup_{w=0}^{p_i} a^w(q^w) \cap B^w(q^w)$. By the algorithm of Table 3.3, whenever the agent is at any configuration $q \in B^w(q^w)$, the F^w mapping automatically selects the a^w action to be executed, and thus to guarantee the task execution it suffices to constrain each of the pseudo-states $a^w(q)$ to generate trajectories approaching the subset containing the trajectories that lead to the execution of the task.

$\square$

It is worth to note that the constraint expressed by (3.21) is the counterpart of the *Boulingand contingent cone* used in Viability Theory to define the viability domain of a dynamic system (see Appendix B for a brief overview on some of the main aspects of the Viability Theory).

Instead of relying exclusively on the *Supervison* block for the choice of the actions, the covering build algorithm can be improved by searching in the space of actions. A search algorithm, applied at steps 5 and 6 of the algorithm of Table 3.3, is defined in Table 3.4.

1	Define an action adjustment mapping $M : A \times A \mapsto A$, such that $a^r = M(a^p, a^q)$ converges towards a^p in the sense that a^r generates trajectories approaching those generated by a^p.
2	Define a metric in $S \times T$ to compute a distance between states.
3	Using the metric defined in step 2, apply M to each of the k-nearest neighbours of $a^p(q^p)$, for each p, such that their actions are adjusted towards a^p.

Table 3.4: The alternative learning stage of the state space navigation algorithm.

The mapping M searches the *a priori* unexplored regions of Q, which correspond to the subspaces denoted by U in (3.19). Given a value for the index p, M only modifies that part of an action generating trajectories contained inside $B^p(q^p)$.

The introduction of the environment, S, does not weaken Proposition 3.4.1, as the actions modified by the mapping M are still forced to generate trajectories lying in the region $\cup_{k=0}^{p} B^k(q^k)$.

A version of the algorithm in Tables 3.3 and 3.4 appeared in [Millan, 1997] under the name of *Incremental Preserving Topology Map* (ITPM).

The correspondence between the concepts defined in the previous sections and the basis building blocks of the *behaviour* based architecture represented in Figure 2.16 is given in Table 3.5, along with the implementation guidelines.

Concept		Block		Symbol		Implementation
Action	$\leftrightarrow$	Action	$\leftrightarrow$	$\boxed{\text{Action}_i}$	$\leftrightarrow$	Q subspaces
Group operation	$\leftrightarrow$	State composition	$\leftrightarrow$	$\otimes$	$\leftrightarrow$	Robot dependent
Supervisor	$\leftrightarrow$	Supervision	$\leftrightarrow$	$\boxed{\text{Supervision}}$	$\leftrightarrow$	Finite State Machine
Equality relation	$\leftrightarrow$	State equality	$\leftrightarrow$	$\ominus$	$\leftrightarrow$	Tables 3.3 and 3.4
State expansion	$\leftrightarrow$	State expansion	$\leftrightarrow$	$\boxtimes$	$\leftrightarrow$	Table 3.3 and supervisor

Table 3.5: Correspondences between concepts and basis building blocks.

3.5 A decentralized cooperation model

This section discusses the use of the *behavioural* architecture in the decentralized control of multiple agent systems. A Finite State Machine supervisor structure is discussed to illustrate the relationships between the controllability of a multiple agent system and the topology of the interconnections between agents.

The intuitive notion of cooperation is supported on the idea that no centralized supervising scheme is necessary to have multiple agents jointly executing a task. This notion of cooperation requires that each supervisor be prepared to interpret incoming information from other agents, taking adequate decisions, and that proper connections between agents exist so that the necessary information can be exchanged.

Consider a single agent, using the cart robot, with the assigned task being specified by: reach a given configuration and follow a straight line afterwards. Moreover, assume that, while reaching this task, the agent must avoid a single obstacle in the middle of a large room. With this assumption, the obstacle can be avoided with the agent moving either by the left or right sides of the obstacle, this eventually leading to significant differences in the performance. The ability to go either by the left or right sides of the obstacle is expressed by the controllability properties of the agent. The choice of which side the agent goes around is expressed by the supervisor's ability to define adequate tasks.

The two aforementioned ideas of controllability and task assignment are extendible to multiple agent systems. The controllability for multiple agent systems is defined similarly to the single agent case. The point-to-point controllability of each single agent in a multiple agent system is a necessary condition for the point-to-point controllability of the whole system. However, it is not a sufficient condition as it does not express any constraints due to eventual couplings between agents. For instance, consider the transportation of a bar by two mobile manipulators, each of them being point-to-point controllable. The bar introduces a physical coupling between the two mobile manipulators that constrains the motion of each of them relative to the motion of the other. Therefore, for some tasks, the point-to-point controllability of each agent may not be sufficient for the task feasibility. The point-to-point controllability of each agent in a multiple agent system is thus a necessary condition (and not sufficient) for the task feasibility of the whole system. The task assignment problem in multiple agent systems is related with the subdivision of the global task among the agents.

The classical technique to treat multiple agent systems is to jointly consider their state spaces. For a n-agent system the state set is then defined by the n-tuple $\tilde{t} = (t_1, \ldots, t_n)$ where t_i is the state of agent i. The task assigned to the system, called the global task, is represented by an element of this joint space, $\tilde{t}_t = (a_{t_1}(q_{t_1}), \ldots, a_{t_n}(q_{t_n}))$, with $a_{t_i}(q_{t_i})$ representing the task assigned to the i-th agent. For the sake of readability, the set of all global tasks is denoted by $\mathcal{T} = \{\tilde{t}_t\}$. The *Supervision* block of the i-th agent receives the state information from all the agents with whom the i-th agent is connected. Representing by T_i the state space of the i-th agent, the *Supervision* block implements a mapping $S_i : T_1 \times \ldots \times T_n \times \mathcal{T} \mapsto T_i$, i.e., it determines the state to be followed by the *Execution* block, given its own state, the states of the other agents and the global task assigned to the system.

Assuming an adequate global task $\tilde{t}_t$ subdivision among the agents and the adequate controllability properties such that there is no need of information exchange between the agents, each of the S_i generates an output converging to its own assigned task, $\tilde{t}_{t_i}$, independently of the connections c_{ij}, i.e., the whole system executes the global task $\tilde{t}_t$ without exchange of information between the agents. Moreover, assuming no explicit central supervision scheme other than the global task subdivision, no information exchange between agents, and using the C-space extension technique, a decentralized model defining how the multiple agent system evolves along the event axis is represented by

$$\begin{bmatrix} t_1 \\ \vdots \\ t_n \end{bmatrix}_{k+1} = \begin{bmatrix} S_1(t_1, \tilde{t}_{t_1}) \\ \vdots \\ S_n(t_n, \tilde{t}_{t_n}) \end{bmatrix}_k \tag{3.22}$$

where k represents the event index and $\tilde{t}_{t_i}$ is the task assigned to the i-th agent.

In model (3.22), the task assigned to the i-th agent, $\tilde{t}_{t_i}$, contains the sufficient knowledge to be executed in such a way that the whole set of agents exhibits the desired behaviour without any information exchange. It is implicitly assumed that no non modelled disturbances exist.

In general, the ideal, or even adequate, task assignment assumption is not valid, [Brocket, 1993]. To overcome this problem, model (3.22) is extended with the knowledge about the global task, $\tilde{t}_t$

(and not just a particular task $\tilde{t}_{t_i}$) and about the state of other agents so that it can adjust its behaviour, i.e.,

$$\begin{bmatrix} t_1 \\ \vdots \\ t_n \end{bmatrix}_{k+1} = \begin{bmatrix} S_1(c_1, \tilde{t}, \tilde{t}_t)) \\ \vdots \\ S_n(c_n, \tilde{t}, \tilde{t}_t)) \end{bmatrix}_k \tag{3.23}$$

where $c_i = (c_{i1}, \ldots, c_{in})$ defines the interdependence of agent i from the remaining agents. Figure 3.9 shows a system with three agents fully interconnected.

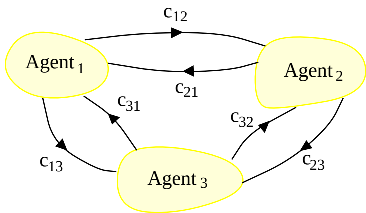


Figure 3.9: Multiple agents interconnection.

Model (3.23) enables the adjustment of each agent's performance as a function of the performance of the other agents. Thus, it may account for poor task assignments that, although the global task is executed, lead some of the agents to a final state different from their respective assigned tasks. This is a characteristic of some cooperation problems, where, in general, not all the agents need to complete their tasks for the system to have its global task executed. Equivalently, instead of assigning a single task to each single agent, a set of tasks is assigned to each of them and the agent chooses one task in this set to execute. Representing by $V_i(\tilde{t}_i)$ the subset of states the i-th agent can be in at the end of the global task execution, a global task can be represented by

$$\tilde{t} = (\tilde{t}_1, \ldots, \tilde{t}_m, V_{m+1}(\tilde{t}_{m+1}), \ldots, V_n(\tilde{t}_n))$$

where the first m agents are required to execute a specific task and the remaining $n-m$ agents need only to reach some state in their respective $V_i(\cdot)$ subsets. This form of task assignment removes the necessity of point-to-point controllability of each single agent. In what concerns the execution of the global task, each of the first m agents has to detect all of the necessary and sufficient events, whereas the remaining $n-m$ agents may fail to detect some events and still reach a task in their $V_m(\cdot)$ subsets.

Consider a system with n agents and $A^p = \{a_1, \ldots, a_r\}$ and assume that each agent is able to detect r events related with each external agent to whom it is connected. A skeleton structure for the Finite State Machine implementation of the *Supervision* block is defined by (assuming that the current state is $a_i(q_i)$)

$$
\begin{array}{l}
\text{do } \{ \\
\ \text{if } E^1_{i1}(t_1) == 1 \text{ then } \{ \\
\quad \text{switch to state } a_1(q_{11}) \text{ with } a_1(q_{11}) \circ a_i(q_i), \\
\quad \text{if necessary perform } a_{\mathrm{x}1}(q_{\mathrm{x}1}) \boxtimes a_{1\circ i}(q_i), \ \text{for some } a_{\mathrm{x}1}(q_{\mathrm{x}1}) \ \} \\
\qquad \vdots \\
\ \text{if } E^r_{i1}(t_1) == 1 \text{ then } \{ \\
\quad \text{switch to state } a_r(q_{1r}), \dots \ \} \\
\qquad \vdots \\
\ \text{if } E^r_{in}(t_n) == 1 \text{ then } \{ \\
\quad \text{switch to state } a_r(q_{rr}), \dots \ \} \\
\ \text{else keep the present state} \\
\} \\
\text{while } 1,
\end{array}
\tag{3.24}
$$

where $E^k_{ij} : T_j \mapsto \{0, 1\}$ is the mapping that defines the pre-conditions for the detection of the k-th event, given the state information from the j-th agent, and where $\{0, 1\}$ indicates that the event was, respectively, undetected or detected. For the sake of simplicity, it is assumed that each event is determined by information from a single agent.

Assuming that the tasks assigned to each of the first m agents can be reached using some sequence of states, then each of the agents is required to detect the corresponding events, so that the global task can be executed. Using supervisors of the type defined by (3.24), the connections for the first m agents are implicitly defined. In general, multiple solutions for the connections to the remaining $n-m$ agents are possible. To every sequence of states driving the agent towards one of states in the respective $V_i(\cdot)$ subset corresponds a sequence of events that has associated a set of connections, defined in (3.24).

Assuming that all the relevant events are considered, the skeleton (3.24) defines the interconnections between the i-th agent and the rest of the system. The controllability properties the i-th agent are defined by the mapping $S_i : T_1 \times \dots T_n \times \mathcal{T} \mapsto T_i$. If S_i is able to generate the whole T_i then the i-th agent is controllable (point-to-point or action controllable) and hence the relationships between controllability and the topology of the interconnections between agents are established.

3.6 Summary of the *behavioural* framework

This section summarizes in a list of topics the main aspects of the *behavioural* framework proposed in this thesis.

1. Following the ideas expressed in the introduction of the thesis, the framework is defined to support a user's perspective in the control of single agents.

2. The narrow level of the user's perspective is introduced in the framework through a number of *a priori* user defined entities, called *actions*. Each of these actions is able to define a set of trajectories, in the configuration space, for the robot to follow.

3. The framework is suported in a state space where each element contains information about the agent's current motion and the corresponding initial conditions.

4. To support the *behaviour* concept, specific equality relations in the space of the actions and in the state space are introduced. Two states are equal if and only if they present a similar evolution in time and space. Two actions are equal if and only if all of the trajectories they generate under identical initial conditions have similar spatial and temporal evolution.

5. A group structure is introduced in the space of actions as the elementar structure that enables the switching between states. The corresponding group operation (called state composition operation and represented by the symbol $\circ$) defines the conditions under which the switching between any two states occurs.

6. The switching between different states, performed by the state composition operation, handles any motion constraints eventually imposed by the kinematics while switching between states. While inside a state, the management of the motion constraints is shown to be also representable using the state composition operation and the group structure. Therefore, the motion management, either inside a state or during a transition between states is encapsulated under a single binary operation, denoted by $\otimes$.

7. A *behavioural* architecture for robot control is defined using the mathematical objects defined in the previous points. This architecture establishes two main building blocks, called the *Supervision* and the *Execution* blocks. The *Supervision* block commands the switching between states using a Finite State Machine and, together with the *a priori* defined set of actions, constitutes the narrow level of the user's perspective. The wide level of the user's perspective defines the topological aspects of the architecture that are problem independent and it is totally contained inside the *Execution* block.

8. The homomorphic relation between the linear combination of actions and the general framework is used to illustrate the tradeoff in the distribution of the implementation complexity among the building blocks of the architecture.

9. The analysis of the maneuvering in the state space leads to the extension of the group structure with a state expansion operation, denoted by $\boxtimes$. This additional operation enables the expansion of the state bounding regions to account for further motion constraints.

The group structure with the $\boxtimes$ operation is a *group with operators in A*, (see [Bourbaki, 1989]) and a non-proper semiring (see [Almeida Costa, 1974, Hazewinkel, 1996]).

10. An iterative algorithm defines a covering for the regions bounding the trajectories generated by the states that lead to the execution of a task. The covering is formed by user defined elements, each of which is bounded by a closed subset of Q. Each element of the covering is associated with an action, proposed by the *Supervision* block, in the sense that only that action is executed inside the covering element. This association is made the first time the agent enters a region of the C-space that was not yet covered.

To adapt the task execution to specific environmental conditions, the actions taken in a neighbourhood of the covering element the agent is in are adjusted towards the current action.
Multiple executions of the same task use the experience accumulated during the past executions by the associations between covering elements and actions. This may result in the construction of alternative coverings for different states that lead to the task execution. The actions executed inside each of the covering elements can be constrained to avoid the search for alternative coverings.

11. A decentralized model for a multiple agent system, supported in the framework developed, is considered. The relationships between controllability and the topology of the interconnections between agents are discussed.

12. Any aggregate of trajectories lying inside the bounding set generated during an execution of a task is an element of the equivalence class, defined by the relation of Definition 2.3.2, composed by all of the possible aggregates of trajectories that are contained inside the same bounding set and verify the temporal similarity.
This encapsulation of a number of mathematical objects, in this case the aggregates of trajectories, generated during the execution of a task, coherently represents the robotics meaning for the term *behaviour* considered in Chapter 1. As a consequence, the following definition of *behaviour* is proposed:

Definition 3.6.1 (*Behaviour*) *A* behaviour *is a set of equivalence classes, in the sense of Definition 2.3.2, of aggregates of trajectories in Q, all of them originated in a region $B_{a_e}(q)$ of any configuration q.*

By Definition 3.6.1 a *behaviour* expresses the possibility of a task execution thorugh different states. This situation may arise as a result of multiple executions of the same task, for example when a mobile platform avoids an obstacle by moving by the left or by the right side, and is a consequence of the diversity of environment conditions that cause the local actions a^p, used by the algorithm of Section 3.4, to violate the conditions in Proposition 3.4.1. Therefore, Proposition 3.4.1 must be considered only as indicative of a convergence property, as it may avoid some potential solutions for the task execution.

Chapter 4

The experimental setups and the design of the actions

4.1 Introduction

This chapter describes the experimental setup used to test the framework proposed in this thesis, along with a set of actions able to lead the considered agents to the execution of a variety of tasks. Section 4.2 describes the physical characteristics of the robots used and their integration in the experimental setups. The specification of the agents and the class of tasks they are able to execute are considered in Section 4.3. These involve single mobile manipulators and two mobile manipulators carrying a bar. The task assignment strategy is detailed in Section 4.4. This section also introduces the concept of *agent dominance*, which significantly simplifies the execution of some tasks. Section 4.5 details the actions used by the agents in the execution of the classes of tasks considered in the thesis. Section 4.6 presents the algorithms performing the state composition operations for the considered agents. A brief discussion on the need to account for emergency situations in *behavioural* approaches is presented in Section 4.7. Section 4.8 describes the implementation of the $\ominus$ block and the graphical notation used in the presentation of the experimental results.

4.2 The experimental setups

The experiments described in this thesis were conducted using both real and simulated robots organized in two setups, hereafter called *real setup* and *simulated setup*. The simulated setup considers exclusively simulated robots whereas the real setup considers either real or a mixture of real and simulated robots. The robots are a cart mobile platform and a 6-dof manipulator mounted on top of the mobile platform. The real system is shown in Figure 4.1.

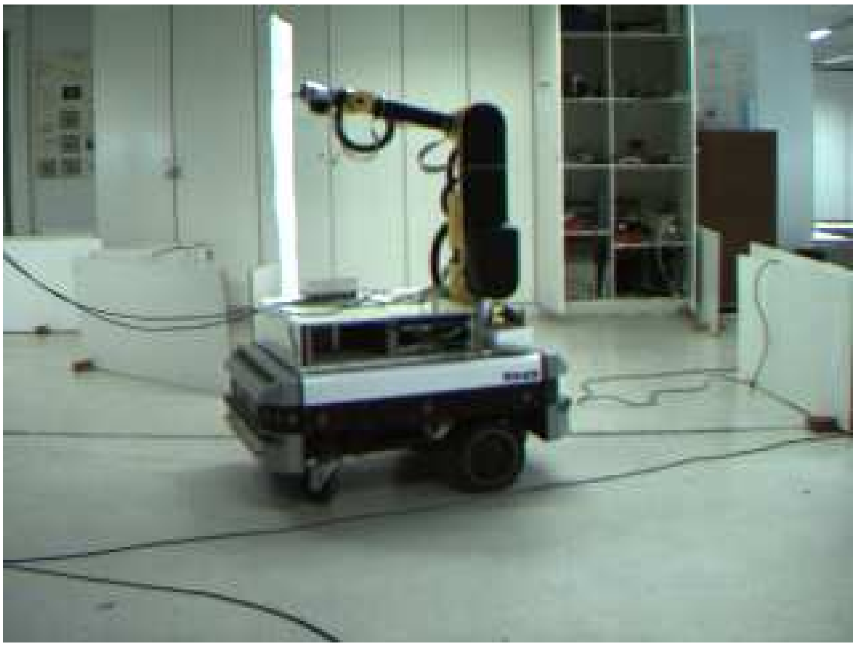


Figure 4.1: Manipulator mounted over a mobile platform.

The mobile platform, a Robuter from Robosoft, has a rectangular shape (0.73 m wide and 1.04 m long), with two drive wheels aligned with the y axis of the platform reference frame, and two free wheels located at the front. A ring of 24 ultrasonic sensors is located around the vehicle, giving reliable measures between 0.15 m and 6 m, with 1 cm accuracy[1]. Figure 4.2 illustrates the distribution of the ultrasonic sensors and the location of the reference frame in the body of the mobile platform.

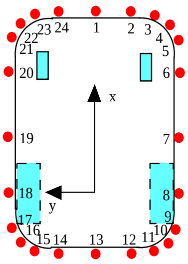


Figure 4.2: Mobile platform ultrasonic sensor distribution.

The manipulator, a GT 6A from GT Productique SA, has six revolution joints, of which the first three are used for x, y, z positioning and the remaining are used for orientation, α, β, γ. The dimensions are shown in Figure 4.3.

The software environment, represented in Figure 4.4a, is composed of three modules that implement:

Module 1: the control architectures (Figures 2.15 and 2.16),

[1]Data from the manufacturer, Robosoft Local Area Ultrasonic Network Users Manual, ref. RSL-UM-111-1.0.

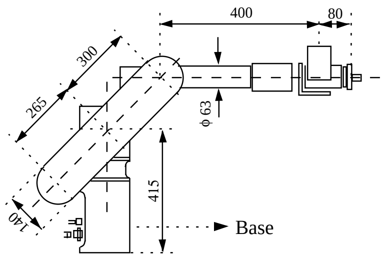


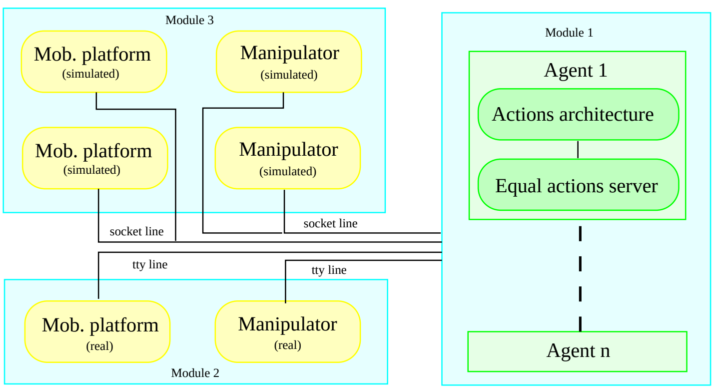
Figure 4.3: Manipulator schematics (dimensions in mm).

Module 2: command interpreters for one manipulator and one mobile platform, communicating with Module 1 by two serial lines (9600 baud),

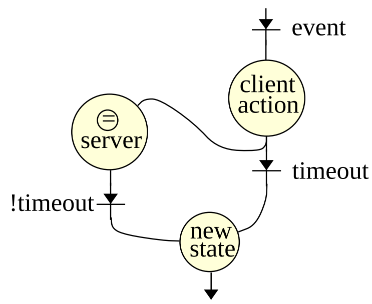
Module 3: kinematics' simulators for two manipulators and two mobile platforms, communicating with Module 1 by Unix datagram sockets.

This environment supports the test of the proposed control architectures either with real or simulated mobile manipulators and of experiments on cooperation using one real and one simulated or two simulated mobile manipulators. New actions can be tested or tuned in the simulated environment, searching for adequate *behaviours*, prior to their use by the real setup. Each module has the capability to detect some communication errors. However, no error correction protocol over the tty lines is implemented. Information that arrives corrupted or out of time is ignored. If the error is detected within a specified time interval the corresponding information is requested again.

The simultaneous use of one real and one simulated mobile manipulator requires the use of the second simulated mobile manipulator to mirror the configuration of the real device, obtained through the odometry readings and from the position sensors at each joint of the manipulator. This mirror configuration is used to compute the corresponding ultrasound measurements in the simulation environment which are then transmitted to the real mobile manipulator, overriding any measurements from the real ultrasound sensors.



(a) Software environment.



(b) Detail of the software architecture for an agent.

Figure 4.4: Software architectures for the environment and the agent.

The software implementing the *behaviours* architecture for each agent is composed of two distinct modules: the *actions* architecture of Figure 2.14 and the ⊜ server to generate the extended architecture of Figure 2.16. The two components are connected using the protocol illustrated by the Petri net in Figure 4.4b. At each event, the *actions* architecture defines the new action to be executed by the agent. A request is sent to the ⊜ server, indicating whether or not an action has been already registered for the present sensory conditions. In the affirmative case, the action indicated by the ⊜ server is chosen to be executed. Otherwise, the action defined by the *actions* architecture is chosen, being simultaneously registered by the ⊜ server. The protocol uses a timer to measure the interval between the request and the answer. Whenever this interval exceeds a specified value (10 ms in all experiments presented in the thesis) the request is considered outdated and is ignored and the action defined by the *actions* architecture is executed. This client-server philosophy avoids the increase in the sampling period of the architecture's feedback loop, generated by an eventual delay introduced by the ⊜ server while searching for an alternative action (for instance due to computational load caused by foreign applications running in the same workstation that supports the system).

Both the simulators of the mobile platform and of the manipulator rely on a simple integrator dynamics, with the controls being, respectively, linear and angular velocities for the mobile platform and the angular velocities of each joint for the manipulator. This simplifying assumption on the dynamics of the real robots involved is supported on the large weight and low operation velocities considered in all the experiments. The real mobile manipulator weights approximately 195 Kg (of which approximately 45 Kg are due to the manipulator) and, to avoid wheel slippage, operates at a maximum linear and angular velocities of 0.15 m/s and 0.2 rad/s. As for the manipulator, although the arm joints are able to operate up to 1.4 rad/s (2 rad/s for the gripper joints), the low data exchange rate available supports the integrator dynamics simplifying assumption. Moreover, to avoid mechanical fatigue the maximum angular velocity is set to 0.2 rad/s for each joint. Each of the simulated ultrasonic sensors emulates the emission cone, with 20° aperture, of the ultrasonic sensors in the real setup. The simulation does not account neither for eventual multi-reflections of the ultrasound beams in the surrounding environment, nor for the non-principal lobes on the radiation diagram. The results presented in Chapter 5 support the aforementioned simplifying assumptions on the robots' simulators, i.e., regardless the differences between the results obtained with the two setups, the simple simulation setup considered is a powerful tool in to the design/tuning of the actions.

4.3 Agent specification

Given a set of robots, the specification of the agents to be considered is directly related with the class of tasks to be assigned to the whole system. In general, each robotic device is associated with a single agent. However, the specification of the actions may be considerably simplified by a division of a robot into multiple smaller robots (in terms of the number of joints), for instance due to the smaller dimension of the C-spaces of each of the small robots. The physical interdependencies between the small robots are taken into account in their respective kinematic models.

A practical class of tasks that can be assigned to mobile manipulation systems include the positioning of the manipulator end-effector in a specified point of the x, y, z workspace, with a specified orientation, α, β, γ[2]. Another practical class of interest is obtained by extending the previous class to consider particular motions after the specified configuration is reached, e.g., pressing a button after having reached its vicinity. Both these classes of tasks are considered in this thesis.

For the manipulator considered in the experimental setup, the mechanical design of the last three joints of the manipulator reduces the interferences with the motion of the first three joints. Moreover, given their location in the setup, for a vast class of tasks they do not interfere with the motion of the mobile platform. For example, this is the case when a single mobile manipulator is moving towards a task. The last three joints of the manipulator can be left out in the control procedure of the first three, this reducing the number of degrees-of-freedom to account for during part of the task execution and hence simplifying the specification of actions for the mobile platform and manipulator.

In what concerns the aforementioned classes of tasks, the division into small robots is considered only for the manipulator, with the first three joints defining one small robot and the last three joints defining the other small robot (see Figure 4.5). Hereafter, and for the sake of simplicity of exposition, the first three joints of the manipulator will be referred as *the arm* while the last three will be named as *the gripper*.

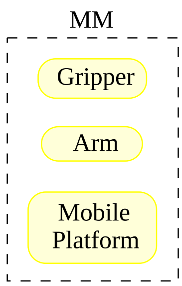


Figure 4.5: Agents on a mobile manipulator (MM).

4.4 Task assignment

At the core of the cooperation paradigm is the idea that some agents may influence the performance of the system more than others. This corresponds to a concept of dominance of one agent over the others. For example, if a mobile manipulator has to traverse a region cluttered with obstacles, it

[2]Representing Euler angles.

may be advantageous to define an intermediate task, referred to the mobile platform. Since this intermediate task is of no significance to the gripper and the arm, the mobile platform can ignore (totally or partially) the state information from the other agents. Potential disturbances induced by the state information of the arm and the gripper can thus be avoided.

During the execution of a task, a dominant agent may be influenced by the other agents, in the sense that it continues to receive their state information. Whether or not that information is used depends on the specific actions involved. This concept is expressed by the following definition.

Definition 4.4.1 (Agent dominance) *Consider a multiple agent system with a task assigned to each agent. The dominant agent is the one that, after finishing the execution of its task, leaves the whole system in a state such that the global task can be considered as finished.*

By Definition 4.4.1, the dominant agent sets the criteria that define the end of the mission. Therefore, the dominant agent implicitly defines the information exchanged with the other agents that is relevant to the task execution.

Using the terminology defined in the previous chapters, a global task can be represented by

$$a(q_t) = a^i([x_t, y_t, z_t, \alpha_t, \beta_t, \gamma_t]') \tag{4.1}$$

where the superscript i stands for g, a, or v, respectively, indicating the dominant agent as being the *gripper*, the *arm* or the *vehicle*. The tasks in all the missions considered in the thesis have either the gripper or the mobile platform as dominant agents.

The first class of tasks considered in this thesis is a particular case of (4.1) and is represented by $a_e^i(q_t^i)$, where the subscript e stands for the null action of the i-th agent[3]. Within this class, a task is considered executed if the agent reaches a neighbourhood of the task location specified in q_t and stays inside it.

The second class of tasks considered for the mobile manipulators is represented by a pair of states $(a(q_{t_1}), a_e(q_{t_2}))$, meaning that the mobile manipulator must reach the neighbourhood of q_{t_1} and then execute the action a until a neighbourhood of q_{t_2} is reached. The null action is then executed ending the execution of the task. A practical situation is the pushing of a button. The first task, $a(q_{t_1})$, is to place the gripper in an appropriate configuration in a neighbourhood of q_{t_1} (the button location) and use the action a to move the gripper in a straight line so that the button is pushed. Whenever a neighbourhood of q_{t_2} is reached, the second task is executed and the gripper stops as the null action is executed. This class of tasks can also be seen as being defined through a trajectory in the state space, instead of a single element of the state space, as in the first class. When specifying tasks in this class it will be assumed that q_{t_1} is specified such that the trajectory

[3]Recall that the null action is the one that, when executed, does not affect the future behaviour of the agent.

generated by the state $a(q_{t_1})$ reaches the region $B_{a_e}(q_{t_2})$, meaning that no maneuvering, other than the one specified by the action a, will be needed for the agent to complete its task once it reaches the state $a(q_{t_1})$.

The subdivision of a global task $a(q_t)$ into the tasks assigned to each of the agents is made according to Table 4.1, independently of the dominant agent i. The superscript notation defined above for the agent dominance is used to represent the individual agents. For the mobile platform the task location in the xy plane, (x_t, y_t), refers to the origin of the platform reference frame (see Figure 4.2). As for the arm, (x_t, y_t, z_t) is referred to the end-effector, i.e., the gripper tip.

$$
\begin{array}{ccc}
 & a_e^i([x_t, y_t, z_t, \alpha_t, \beta_t, \gamma_t]) & \\
\swarrow & \downarrow & \searrow \\
a_e^v([x_t, y_t]') & a_e^a([x_t, y_t, z_t]') & a_e^g([\alpha_t, \beta_t, \gamma_t]')
\end{array}
$$

Table 4.1: Global task assignment for the mobile manipulator agents.

4.5 Specification of the actions

In general, due to the complexity of the interactions between the agents and the environment, the user does not *a priori* know the characteristics of the *behaviours* that will emerge during a task execution. Users define the B^p regions, used by the algorithm described in Table 3.3, that bound each pseudo-state $a^p(q^p)$, and hence indirectly influence the set that bounds the region defining each equivalence class in T, according to Definition 3.6.1. Besides their importance to the design of adequate basis actions, the outcome of the interactions agent-environment is also used by the supervisors to detect the events that trigger the actions in A^p. The aforementioned complexity often leads to frequent interventions by the *Supervision* block, which results in the short duration of each state and, in some sense, represents a local characteristic of the actions. The use of vector fields as generators of trajectories valid for short periods of time is the underlying argument supporting the identification with actions (as used in Section 3.2) and provides a rough methodology for the *action* synthesis that is used in this section.

In the sequel, a set of actions for each agent is presented. The mobile platform actions are defined using the projection principle of Section 3.4, i.e., the trajectories generated by each action are defined by a vector field on the x, y space, instead of the x, y, θ space. As for the arm and gripper, the actions are specified using vector fields in the x, y, z space, without using the projection principle.

In general, the vector fields defining the actions depend on a number of parameters. For instance, a *move in a straight line* action may be tagged with parameters specifying the velocity, acceleration and/or, for the sake of safety, the maximum time period for the execution of an action. Often, these parameters reflect additional knowledge about the specific environment where the agent is operating, leaving the general knowledge about a class of environments to the definition of the vector fields.

4.5.1 Mobile platform's actions*

The following classes of actions are defined. The relevant parameters for each of these classes will be described either together with the action definition or with the presentation of the experimental results.

Null action (Null) This action stops the motion of the mobile platform. For example, whenever information from the arm indicates that the global task can be executed without any intervention from the mobile platform, it should stop.

Move Forward a specified distance (MoveFwd) This action is triggered whenever the arm has not enough maneuverability to execute its task. The mobile platform moves a fixed distance (pre-specified) in its forward direction.

Move Backward a specified distance (MoveBwd) Similar to the previous action but with the motion in the backward direction.

Reconfigure Linear Velocity (ReconfLV) The physical design of the mobile manipulator rends the execution of most tasks easier if the mobile platform approaches the task moving backwards. This action is triggered whenever there is enough space for the mobile platform to change its heading by 180 degrees, so that it can approach the task moving backwards.

Escape from Horizontal Gripper Singularity (EscHGS) The division of the manipulator into small robots, though simplifying the design of the actions, leads to the necessity of considering specific actions for the mobile platform, as the current one. While moving towards the task location, it may happen that the gripper can not reach the orientation specified in the task. This is of no relevance whenever the mobile manipulator is far from the task location but requires a special action for some trajectories approaching the task location. This situation, called in the thesis a *gripper singularity*, is solved by applying the algorithm illustrated in Figure 4.6.

The action introduces a new task, for the mobile platform, in a location such that, while the mobile platform reaches this task, the arm continues pursuing its own task and automatically rotates the base joint. The net effect is that the joint angles defining the specified orientation can now be reached. This strategy solves only a particular class of singularities, arising when the gripper orientation specified by the task forces a horizontal attitude by the gripper. This

*The work related with obstacle avoidance was developed jointly with the NATO-SFS PO-ROBOT project.

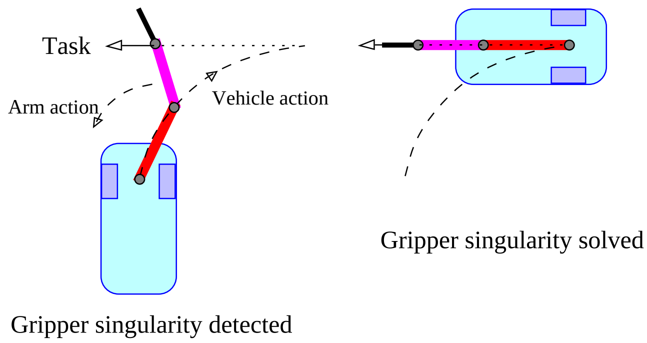


Figure 4.6: Escaping from a Horizontal Gripper Singularity.

is the reason why this action is referred as "horizontal singularity". In the horizontal plane of the workspace, joint 6 controls the rolling motions not interfering with the gripper attitude.

Go Straight to Goal (GoStG) This action computes the initial configuration for the next state by analyzing the error, in the xy plane, between the task location and the actual configuration, $e_t^v = q_t^v - q^v$. The motion direction that defines the trajectory of the next state, is defined as

$$\mathbf{v}^v = \gamma \cdot < e_t^a, e_t^v > \cdot e_t^v / \|e_t^v\| \tag{4.2}$$

where e_t^a is the arm task error vector in the xy plane, $<,>$ stands for the scalar product in $\mathbb{R}^2$ and γ is a weighting scalar constant. Whenever the projection of the arm task error over the platform task error goes to zero this **GoStG** action tends to the null action (see Figure 4.7[4]).

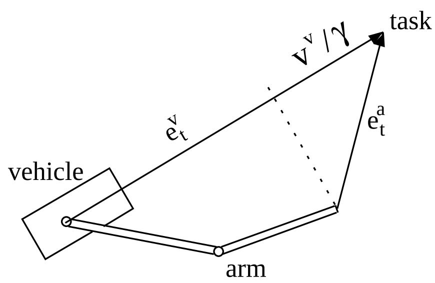


Figure 4.7: Generating the motion direction for the *go straight to goal* action.

The practical effect of this action is to move the mobile platform slowly if the arm is not headed to the task location. If the arm completes its own task while the mobile platform is moving, the supervisor changes the assigned task, forcing a null action to be executed by the mobile platform, i.e., stopping its motion.

As $\|\mathbf{v}^v\| \to 0$ the x, y reference position sent to the mobile platform becomes close to its actual position, generating small oscillations in the platform orientation. This problem aggravates with any increase of the delay in the control architecture feedback loop. Also, its effects become more visible as the arm extends itself. The implemented solution stops the mobile

[4]For the sake of readability the dimensions of the arm are scaled.

platform whenever $\|\mathbf{v}^v\|$ goes bellow a fixed threshold, giving some time to the arm to take any corrective action.

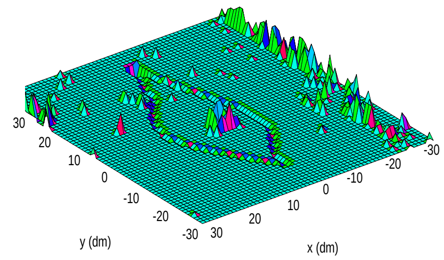
Safe Obstacle Avoidance (SaOA) This action is triggered whenever at least one ultrasound sensor returns a measurement bellow a pre-specified threshold, called *obstacle detection threshold* and meaning that an obstacle was detected. Unless otherwise noted, this threshold is set to 0.35 m.

When far from the task location and in an uncluttered environment, the mobile platform attempts to go around any detected obstacle using a set of two *a priori* defined lookup tables (LUTs). Each of these LUTs defines a motion direction according to the ultrasonic measured data pattern. Figure 4.8 illustrates the definition of the two LUTs used by this action. The motion direction generated by this action is the linear combination of the motion directions defined by the LUT that matches the ultrasound data. Each of the small lines superimposed at each ultrasonic sensor location represents a vector (with origin at the sensor location) specifying a motion direction. Once specified (before starting the task execution) these vectors remain unchanged. If an ultrasonic sensor measures a distance bellow the obstacle detection threshold the corresponding vector contributes to the final motion direction. Otherwise it is ignored. The final motion direction to be followed by the mobile platform is the sum of all the vectors contributed by the ultrasonic sensors.

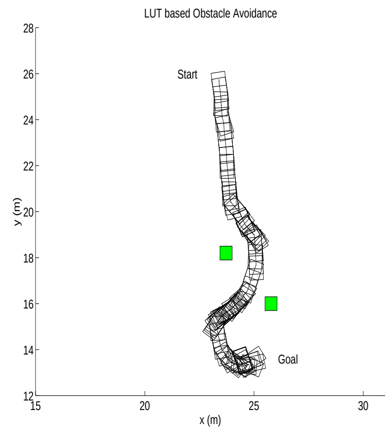
Forward motion generation LUT. Backward motion generation LUT.

Figure 4.8: Safe obstacle avoidance LUTs definition.

Regardless the lack of sophistication of the LUT-based approach to the obstacle avoidance problem, the results are considered good enough for the purpose of assessing the framework presented in the thesis. Figure 4.9 shows a comparison using the Vector Field Histogram (VFH), [Borenstein and Koren, 1991], and the proposed LUT method. The VFH method plot shows a grid map with the obstacle being represented by the central "mountains" defined from the processed data acquired by the ultrasound sensors. The elevations in the right side of the figure represent a wall. Note also the existence of scattered outliers (erroneous measures), represented by the small elevations. The trajectories obtained in two runs are represented using "depressions" in the same grid map. Clearly, both trajectories go around the obstacle (example from [Freitas et al., 1997]). The example using the LUT based method shows an environment slightly more complex, with two obstacles, represented by the filled rectangles. The border of the mobile platform is shown in the plot. Regardless of the differences in



Obstacle avoidance using the Vector Field Histogram method.



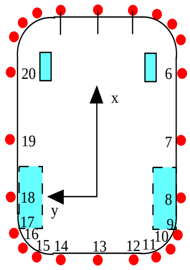
Obstacle avoidance using the LUT method.

Figure 4.9: Comparing two obstacle avoidance strategies.

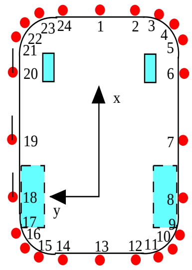
the test environments, the performances are clearly comparable, at least for low complexity situations such as the ones used in the examples.

Unsafe Obstacle Avoidance (UnOA) This action is triggered whenever the mobile platform detects an obstacle at a distance inferior to a pre-specified threshold, called *wall detection threshold*. Typical values vary from 0.15 m to 0.35 m. In general, this threshold is lower than the obstacle detection threshold to have the mobile platform scanning the proximity of the obstacle in greater detail than when using the **SaOA** action. The number of LUTs used in this case reflects this higher scanning requirement than in a safe obstacle avoidance situation. A total of 11 classes of LUTs are defined in the mobile platform agent, with each class containing two LUTs, for forward and backward motion respectively. The classes are named upon the ultrasound data pattern they detect: corridor, left/right walls, front/back wall, front/back U-shaped wall, and front/back right/left L-shaped wall. Figure 4.10 illustrates four of them, defined for forward motion (the interpretation is similar to Figure 4.9). Similar LUTs are also defined for backward motion. The detection of each pattern occurs whenever any of the measurements of the corresponding ultrasound sensors lie bellow the wall detection threshold. Moreover, if the ultrasound measurements do not match any of the LUT patterns specific of this action, the LUTs of the **SaOA** action are used.

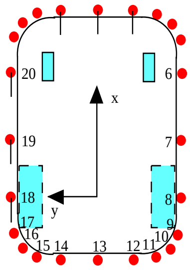
Figure 4.11 illustrates the performance of the LUT method in real environments. The walls of these environments are made of wooden plates with a plain texture. No special concern was taken at the junction of the multiple plates used. The obstacles are represented in the figure by the shadowed regions. The rectangles superimposed in the trajectory indicate the mobile platform attitude at some points. The initial and final positions are shown by the symbol $\circ$. Figure 4.12 shows two perspectives of the experiment with the real L-shape environment.



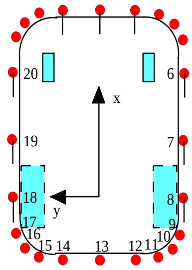
Front walls LUT example.



Left sidewalls LUT example.



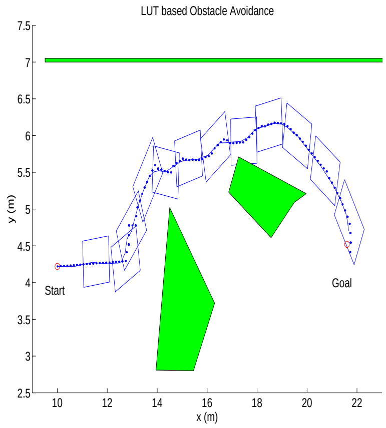
L-shaped walls LUT example.



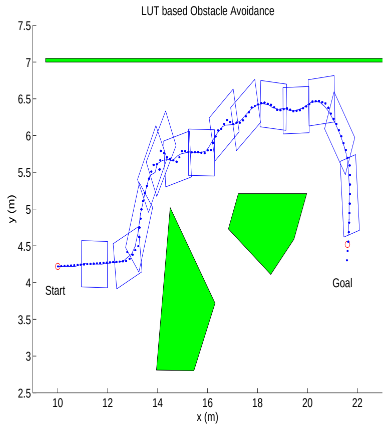
U-shaped walls LUT example.

Figure 4.10: Unsafe obstacle avoidance LUT definition.

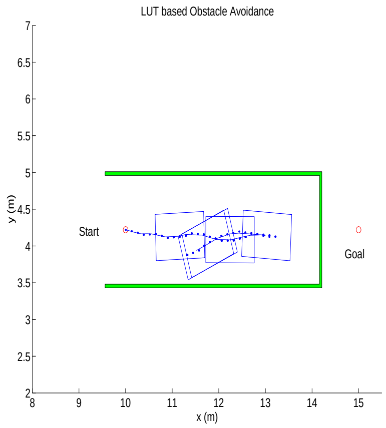
The first experiment (Figure 4.11a in open space navigation clearly shows a collision. The amovable wooden walls used to build the test environment allowed the mobile platform to continue its motion. The coherence of the complete trajectory suggests that the collision was a consequence of the ultrasound reflections that caused the sensors to give erroneous measurements. The result of Experiment 2 (Figure 4.11b, with the position of the obstacle that caused the collision slightly modified, shows a correct detection and avoidance of the obstacles, thus agreeing with the previous interpretation. As the LUT based method relies on unprocessed sonar measurements this kind of phenomena may arise. Nonetheless, for the purpose of demonstrating the control architecture in cooperation problems the method is adequate, as will be demonstrated in Chapter 5. The small changes of heading in the corridor experiment (Figure 4.11e illustrate the influence of the motions directions associated with the lateral ultrasound sensors. The effect of the ultrasound reflections is visible at the configuration where the border of the robot almost touches the right wall of the corridor. In the wall experiment (Figure 4.11d these changes of heading disappear once the robot is away from the wall. In the U experiment (Figure 4.11c, the backward motion, observed after the robot reached a point near the front wall, is the result of the front wall LUT shown in Figure 4.10. In the L-turn experiment one intermediate point (shown in Figure 4.11f was used to avoid the potential problems caused by reflections of the ultrasound beams at sharp corners that induce erroneous LUT matches. A typical example is the existence of a sharp corner, near the vehicle, in the middle of the space between two consecutive ultrasound sensors. The sharp corner lies in a blind zone (i.e., a region not covered by any ultrasound sensor) and its sharpness causes the reflection of the beams. The LUT based scheme is thus deceived by the misreadings and may move the mobile platform in such a way that it collides with the obstacle. The via point pulls the mobile platform to a region were this problem is minimized.



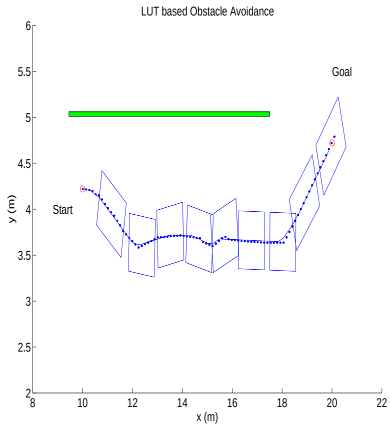
a) Open space, experiment 1.



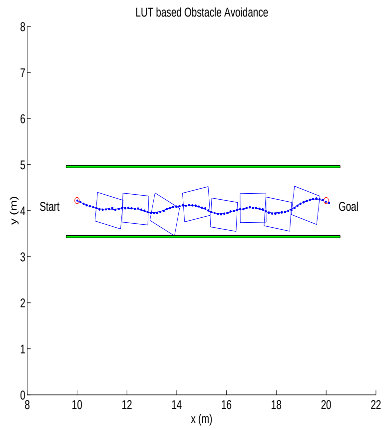
b) Open space, experiment 2.



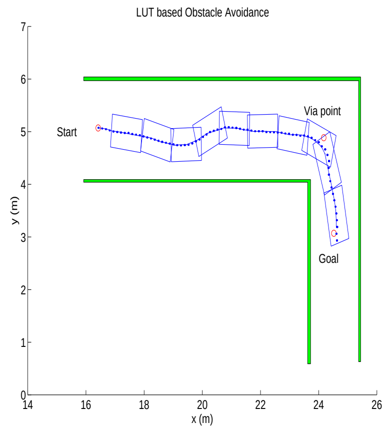
c) U-shape environment.



d) Left wall.



e) Corridor.



f) L-shape environment.

Figure 4.11: LUT based obstacle avoidance examples.

Figure 4.12: Two real perspectives of the L-shape environment experiment.

4.5.2 Arm's actions

The kinematics model for an arm with three joints with the mechanical characteristics illustrated in Figure 4.3 is given, in a velocity formulation, by[5],

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} s_0(-l_1c_1 + l_2s_{12}) & -c_0(l_1s_2 + l_2c_{23}) & -c_0c_{12}l_2 \\ c_0(l_1c_1 - l_2s_{12}) & -s_0(l_1s_1 + l_2c_{12}) & -s_0c_{12}l_2 \\ 0 & -l_1c_1 + l_2s_{12} & l_2s_{12} \end{bmatrix} \cdot \begin{bmatrix} \dot{\theta}_0 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{4.3}$$

where l_1 and l_2 represent the lengths of the two arm links (see [Craig, 1986]). The three joints of the arm have physical limits $|\theta_0| \leq 135°$, $|\theta_1| \leq 123°$ and $|\theta_2| \leq 135°$, with a maximum angular velocity of 0.2 rad/s for all the joints.

The singular configurations of the mechanism and the workspace regions in their neighbourhoods tend not to be useful in what concerns the task execution since the motion in some directions is of restricted amplitude or not possible at all. Therefore, two regions were defined around the arm singularities. The first region, named S_1, is a 0.05 m thick spheric crown, centered at the origin of a reference frame placed at the base of the manipulator and having an interior radius of 0.65 m. The second region, named S_2, is a cylinder having symmetry axis coincident with the z axis of the arm's base reference frame. Whenever the end-effector enters S_1 (respectively S_2), the arm is said to be in a *singularity region of type 1* (respectively in a *singularity region of type 2*). Both S_1 and S_2 correspond to low manipulability regions in the arm workspace.

A third singularity region, named S_3 and defined by the intersection of singularity regions of type 1 and 2, proves useful for the actual experimental setup. In practical terms, the arm is in this region when it is in the upward position. This singularity region is called *double singularity region*.

The following classes of actions are defined.

[5]The abbreviations $s_i = \sin(\theta_i)$, $c_i = \cos(\theta_i)$, $s_{ij} = \sin(\theta_i + \theta_j)$ and $c_{ij} = \cos(\theta_i + \theta_j)$ are used, with θ_i the i-th joint variable.

Null action (Null) This action stops the motion of the arm.

Go Straight to Goal (GoStG) This action has a function similar to the one described for the mobile platform, i.e., it drives the agent towards the task. The motion direction is defined by

$$e_t^a = \begin{cases} [x_t, y_t, z_t]' - q^a & \text{if } \|q^a - q^a_{\text{base}}\| > \nu \\ [x_t, y_t, 0]' - [x, y, 0]' & \text{Otherwise} \end{cases}$$

where ν is a decision threshold related with the manipulability of the arm. High values of ν allow the arm to enter the low manipulability region close to S_1. In the experiments considered ν was set to 0.65 m. While out of low manipulability region, the end-effector tends to move in a straight line towards the task location. This is accomplished by the first condition. If the end-effector enters the low manipulability region only the rotation component of the error vector is used, aligning the arm with the direction defined by the task location and the base of the manipulator, this being the second condition. Note that, in this case, the error in the z component is forced to zero to generate the rotational motion that aligns the arm in the direction of the task location.

Approach to Goal (ApG) This action is similar to the *go straight to goal*, except that the threshold ν is extended to enable the arm to strecht itself into the S_1 region. This action is used only whenever the arm can execute its own task without any help from the mobile platform.

Get Out of Type 1 Singularity region (GetOutT1S) With this action the arm jumps out of S_1. Whenever the end-effector enters S_1 the arm tends to recoil, while rotating to align with the task location. The motion direction is defined by

$$\alpha_1([x^a, y^a, z^a]' - [x^a_{\text{base}}, y^a_{\text{base}}, z^a_{\text{base}}]') + \alpha_2([x^a_t, y^a_t, 0]' - [x^a, y^a, 0]') + \alpha_3([0, 0, z^a_t]' - [0, 0, z^a]')$$

where the α_i are weighting constants. By weighting differently the x, y components of the error vector (the second term) from the z component (the third term) it is possible to emphasize the rotational component of the error vector.

Get Out of Type 2 Singularity region (GetOutT2S) With this action the arm jumps out of the S_2 region. The motion direction is defined by

$$\beta_1 [-y^a, x^a, 0]' + \begin{cases} \beta_2 [x^a_t - x^a, y^a_t - y^a, 0] & \text{if } [x^a, y^a] \cdot [x^a_t - x^a, y^a_t - y^a]' > 0 \\ [0, 0, 0] & \text{Otherwise} \end{cases}$$

where the β_i are weighting constants. Whenever the arm enters the S_2 region the first component forces it to rotate until it points towards the task location. At that point, the translation component, introduced by the second term, forces the arm to move away from the singularity region.

Get Out of Double Singularity region (GetOutDS) In what concerns the evaluation of the *behavioural* framework, the specification of tasks requiring the arm to enter the S_3 region is of little interest. Tasks requiring the elevation of the arm up to the S_3 region can be executed by arms with similar kinematics and longer links without entering their own double singularity regions. Therefore, this action specifies a motion direction that drives the arm to a region of high manipulability, after which it is assumed that some other action drives it towards the task.

Collision Avoidance while in Cooperation (ColAvCoop) The manipulator controller box (white box shown in Figure 4.1 on top of the mobile platform) limits the natural workspace of the manipulator, i.e., there is a set of physically valid configurations that lead to collisions between the first link of the arm and the controller box. This problem is overcame by a judicious specification of the mobile platform tasks, for instance making the mobile platform stopping at some distance of the location of the task assigned to the arm such that the actions previously defined ensure the necessary displacement of the arm.

In general, while cooperating with another mobile manipulator it is also necessary to account for collisions between the mobile platforms and the bar and between the manipulators and the bar. For the missions considered in this thesis, the actions previously defined for both the arms and the mobile platforms tend to drive the mobile manipulators a certain distance away from each other. Large values (in the order of the bar length) tend to minimize the possibility of collisions between the bar and the mobile platforms. This possibility can be further reduced by an adequate selection of the tasks assigned to each mobile manipulator. Supported in these arguments, the examples presented in this thesis were chosen to minimize the possibility of such occurrences and therefore this action is presented uniquely for the sake of completeness, as it is never used by the arms.

The possibility of a collision between the bar and the manipulator occurs more frequently than between the bar and the mobile platform, e.g., inadequate gripper motion easily generates collision between the bar and any of the arm links. Although, as in the previous situations, the occurrence of collisions can be minimized or even eliminated by an appropriate definition of the tasks, the arm must be able to solve potential collisions induced by dynamic characteristics of the environments. The strategy used by this action is shown in Figure 4.13. The simplified schematics shows the base reference frame, the two links and the gripper. The bar is represented by the shadowed region.

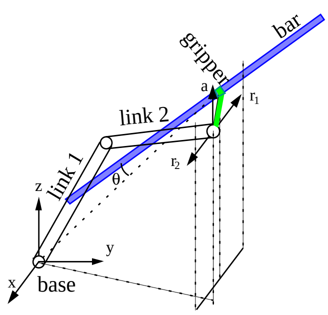


Figure 4.13: Manipulator-bar collision avoidance strategy.

In the situation of the figure the bar intersects both links of the arm. The motion direction to avoid the collision is defined by the vectors shown. The vectors r_1 and r_2 represent two alternative motion directions used to repel the bar from the arm links. The vector a tends to move up the arm, eliminating the colision and enabling the arm to move towards its goal.

4.5.3 Gripper's actions

The gripper kinematics is defined by

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{cases} \begin{cases} \arctan_2(r_{10}, r_{00}) & \text{if } \beta \neq \pm\pi/2 \\ 0 & \text{Otherwise} \end{cases} \\ \\ \arctan_2(-r_{20}, \sqrt{r_{00}^2 + r_{10}^2}) \\ \\ \begin{cases} \arctan_2(r_{21}, r_{22}) & \text{if } \beta \neq \pm\pi/2 \\ \pm\arctan_2(r_{01}, r_{11}) & \text{if } \beta = \pm\pi/2 \end{cases} \end{cases}, \tag{4.4}$$

where the r_{ij} are the elements of a 3-dimensional rotation matrix defined by the Euler angles α, β, γ. See [Craig, 1986] for details.

This agent has **Go Straight to Goal** and **Null** as its actions. The agent is always trying to reach the orientation specified by the task, generating a warning to the other two agents whenever the whole system is in a pre-specified neighbourhood of the task location, (x_t, y_t, z_t) and it is not able to reach the specified orientation. This singularity corresponds to the reaching of a physical limit in the gripper's first or second joint. The radius of the neighbourhood depends on the experiment. Typical values vary between 5 cm and 15 cm. These are detailed together with the task specifications for the experiments presented in Chapter 5.

4.6 State composition operations

4.6.1 Mobile platform state composition

The control of the mobile platform is made using a technique that transforms the nonholonomic problem into a holonomic one. The underlying idea is to control a point, $\mathbf{p}$, other than the origin of the reference frame (to which the cart model is referred). The details are presented in Appendix C.

The link path between the configurations $q = (x, y, \theta)$ and $q_i = (x_i, y_i, \theta_i)$ is the trajectory generated by the control law

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta + \phi)\|\mathbf{p}\| \\ \sin(\theta) & \cos(\theta + \phi)\|\mathbf{p}\| \end{bmatrix}^{-1} \begin{bmatrix} \dot{x}_d \\ \dot{y}_d \end{bmatrix}, \tag{4.5}$$

where $\mathbf{p} = (1,0)$ is a vector defined in the mobile platform reference frame and ϕ is the orientation relative to the same reference frame. The desired velocity vector $[\dot{x}_d, \dot{y}_d]$ is expressed by

$$\begin{bmatrix} \dot{x}_d \\ \dot{y}_d \end{bmatrix} = \left(\begin{bmatrix} x_i - x \\ y_i - y \end{bmatrix} - \lambda \begin{bmatrix} -(y_i - y) \\ x_i - x \end{bmatrix} \right) \epsilon, \tag{4.6}$$

where the parameters λ and ϵ weight, respectively, the convergence rate to the reference point, (x_i, y_i), and the control effort. The first term on the right side of (4.6) "pulls" the mobile platform orientation towards the orientation of the line joining (x,y) and (x_i, y_i). The second term in (4.6) "pulls" the mobile platform position to the desired (x_i, y_i).

It is interesting to interpret the expression (4.5) within the proposed framework. Representing by $U = \{[v, \omega]'\}$ the space of controls and by $\mathbb{T}$ the time, the control law in (4.5) is a mapping $Q \times Q \times \mathbb{T} \mapsto U$, where the first term Q stands for the space of initial configurations and $Q \times \mathbb{T}$ represents the trajectory generated by (4.6). The cart model maps the space of controls and the current state (present configuration together with present velocity) into the next state future (configuration together with future velocity), i.e., $U \times Q \times Q \times \mathbb{T} \mapsto Q \times Q \times \mathbb{T}$. The composed mapping yields $Q \times Q \times \mathbb{T} \times Q \times Q \times \mathbb{T} \mapsto Q \times Q \times \mathbb{T}$, which is coeherent with the definition of the state composition as a mapping $T \times T \mapsto T$.

The state composition operation defined by (4.5) and (4.6) always generates controls, v and ω, that drive the mobile platform to a neighbourhood of the reference configuration $q_i = (x_i, y_i, \theta_i)$. Using the definition of the null action for the mobile platform, it is guaranteed that the link path in the state composition $a_i(q_i) \circ a_j(q_j)$ ends in a neighbourhood of q_i defined by $a_e(q_i)$.

Replacing (4.6) in (4.5) and using the cart model yields,

$$\dot{q} = \begin{bmatrix} \cos(\theta)\cos(\theta+\phi) & -\cos(\theta)\sin(\theta+\phi) \\ \sin(\theta)\cos(\theta+\phi) & -\sin(\theta)\sin(\theta+\phi) \\ -sin(\theta) & \cos(\theta) \end{bmatrix} \frac{\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \right)(q_i - q)\epsilon}{\cos(\theta)\cos(\theta+\phi) - \sin(\theta)\sin(\theta+\phi)} \tag{4.7}$$

which simplifies to

$$\dot{q} = \begin{bmatrix} \cos(\theta)\cos(\theta+\phi) - \lambda\cos(\theta)\sin(\theta+\phi) & \cos(\theta)\cos(\theta+\phi)\lambda - \cos(\theta)\sin(\theta+\phi) & 0 \\ \sin(\theta)\cos(\theta+\phi) - \lambda\sin(\theta)\sin(\theta+\phi) & \sin(\theta)\cos(\theta+\phi)\lambda - \sin(\theta)\sin(\theta+\phi) & 0 \\ -\sin(\theta) + \lambda\cos(\theta) & -\sin(\theta)\lambda + \cos(\theta) & 0 \end{bmatrix} (q_i - q)\epsilon. \tag{4.8}$$

A straightforward calculation indicates that the null space of the matrix in (4.8) is the subspace spanned by $(0,0,\theta)$. Since the matrix in (4.8) is bounded, the system is stable, which means that either it has limit cycles or converges to one of the equilibrium points. Since the subspace of equilibrium points coincides with the the null space of the matrix, $q_i - q \rightarrow 0$ which means that q converges to a point in the subspace spanned by $q = [x_i, y_i, \theta]'$. In practice, whenever the mobile platform position lies in a 5cm neighbourhood of the point (x_i, y_i) the null action is triggered.

The mobile platform can not perform trajectories with curvature radius bellow 0.4 m, even though it can perform pure rotations. This means that, prior to the application of the control signals given by (4.5), it is necessary to verify the turning radius of the trajectory and, if it is lower than the 0.4 m limit, perform a pure rotation until this constraint vanishes.

4.6.2 Arm state composition

Since the arm is holonomic, a path between any two configurations in the reachable workspace can be generated using expression (4.3). The user specifies the trajectories to be generated by the arm actions in the cartesian space. Using (4.3) the corresponding linear velocities are transformed into the joint angular velocities. Clearly, the joint angular velocities generated tend to infinity when the arm approaches a singular configuration. This problem is overcame by limiting the angular velocity of the joint with the highest velocity absolute value to a pre-specified limit (0.2 rad/s) and scaling the angular velocities of the remaining joints in order to maintain the motion direction specified by the linear velocity. The situation of having the determinant of the Jacobian matrix in (4.3) equal to zero never occurs, being avoided by the actions that account for the singularities.

The “crossing” of a joint limit represents a composition between two states that is handled by a separate procedure. Once the trajectory being generated by the actual state reaches a configuration near a joint limit, the arm's motion is stoped and a pre-specified link path is generated. This link path drives the arm towards a configuration away from the joint limit, making the arm enter a new state and continue its motion.

4.7 Emergency situations

The use of vector fields to generate the trajectories emphasizes a local characteristic, i.e., the action generates a motion that will often need to be adjusted due to unpredicted environment conditions. These unpredicted conditions often cause emergency situations that were not *a priori* accounted for. For example, the LUT method for obstacle avoidance proposed in Section 4.5.1 fails to generate a safe trajectory in some situations were the ultrasonic sensor beams are subject to reflections (see Figure 4.11). Even though more sophisticated obstacle detection and avoidance algorithms can be used, alternatively to the LUT based method, emergency situations can always be created, e.g., by placing an unexpected obstacle into the mobile platform trajectory. The adopted strategy to deal with emergency situations is to completely stop the motion of all the robots and wait for the intervention of the user. Two types of emergencies are considered: distance to a detected obstacle lower than 15 cm and communication errors.

4.8 Implementation details of the $\ominus$ block

This section details the implementation of the $\ominus$ block of the *behaviours* architecture (see Figure 2.16), along with its configuration for the experiments considered in this thesis. The $\ominus$ blocks are used only in the control of the mobile platforms.

At every time tick of the control loop, each of the mobile platforms accesses its $\ominus$ server by sending a message containing a header specifying the request, the present configuration and the reference configuration generated by the *actions* architecture. The answer either indicates an action already registered at some past event, or, in the absence of any registered action, the answer returns the same reference configuration received from the *actions* architecture (see Figure 4.4b.

If the mobile platform configuration lies in the region $B^p(q^p)$, all the k-nearest neighbour regions update their actions using an action adjustment mapping M, referred in Table 3.4 ($k = 2$ in all the experiments using the $\ominus$ block). The distance between any two regions, $B^p(q^p)$ and $B^r(q^r)$ is defined by

$$d(p, r) = \|(x^p - x^r, y^p - y^r)\| \tag{4.9}$$

where $(x^p, y^p, \theta^p) = q^p$ and $(x^r, y^r, \theta^r) = q^r$.

The inclusion of alternative sensor data in (4.9), e.g., ultrasound measurements, can be accounted, for example with the measure

$$d(p, r) = \|(x^p - x^r, y^p - y^r)\| + \beta \sum_{i=0}^{n} (s_i^p - s_i^r)^2, \tag{4.10}$$

where the s_i^p and the s_i^r represent the ultrasound measurements from the i-th sonar registered in the p and r regions and β is a scalar weighting the contribution of the ultrasounds. See [Millan, 1997] for an example using this kind of metric. The use of the alternative metric in (4.10) with unprocessed ultrasound data leads to distance values that do not correctly express the environment reality. Therefore, the experiments with the $\ominus$ blocks presented in this thesis consider exclusively the distance given by (4.9).

The algorithm M for the adjustment of the action a^p (see Table 3.4) rotates the vectors defining the motion directions of the neighbour actions a^r towards the one of a^p. Representing by $\theta^p\big|_k$ and $\theta^r\big|_k$ the orientation of the vectors defining, at instant k, the trajectories generated by the actions a^p and a^r, respectively, the adjustment is defined by

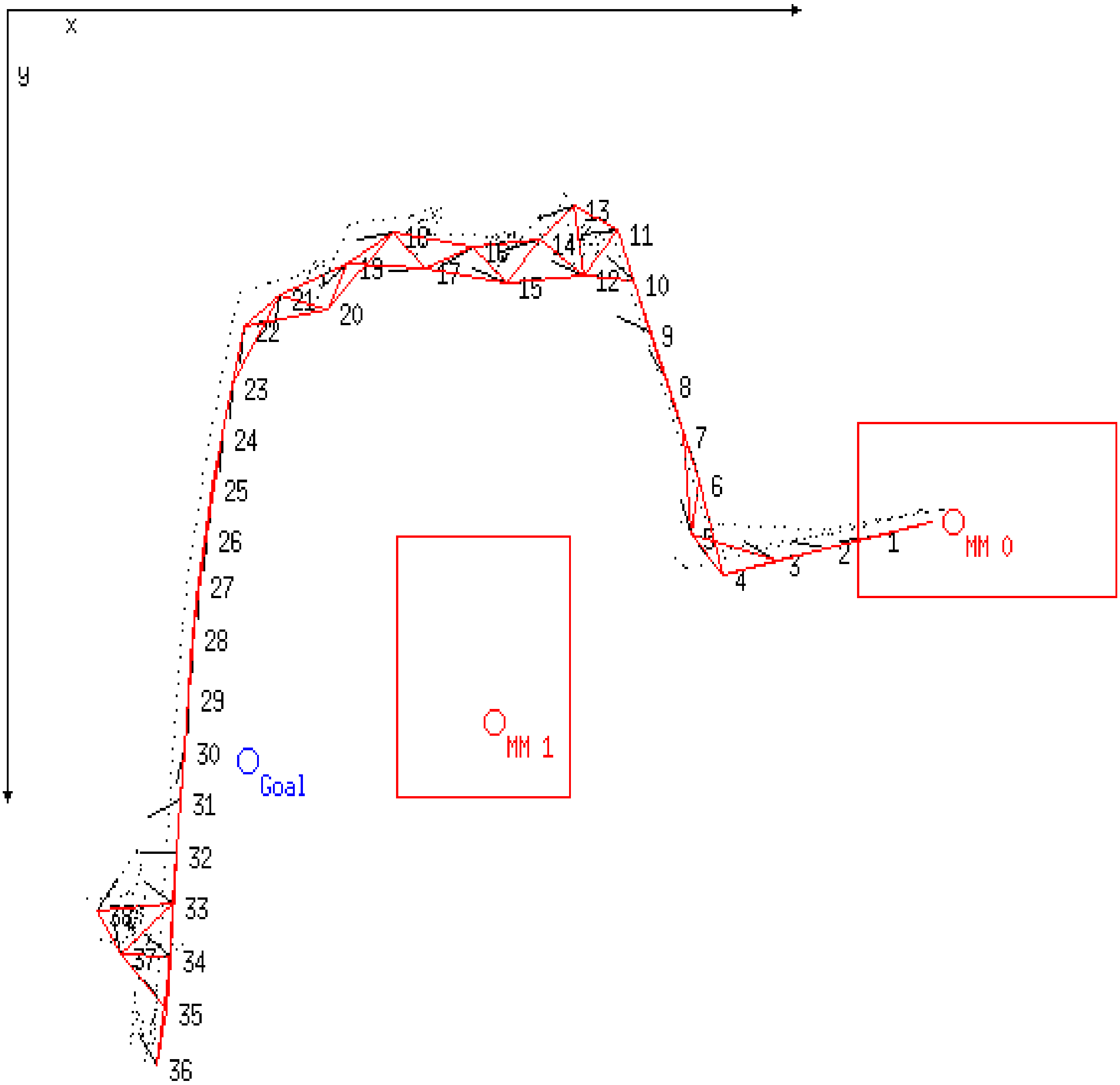
$$\begin{aligned}\theta^p\big|_{k+1} &= \theta^p\big|_k + \epsilon_p(\theta\big|_k - \theta^p\big|_k)\\ \theta^r\big|_{k+1} &= \theta^r\big|_k + \epsilon_r(\theta^p\big|_k - \theta^r\big|_k)\end{aligned} \tag{4.11}$$

where ϵ_p and ϵ_r are scalars respectively set to 0.003 and 0.01 in all the experiments and $\theta\big|_k$ represents direction of motion of the mobile platform at instant k, defined by $\tan(\theta)\big|_k = \frac{\dot{y}}{\dot{x}}\big|_k$.

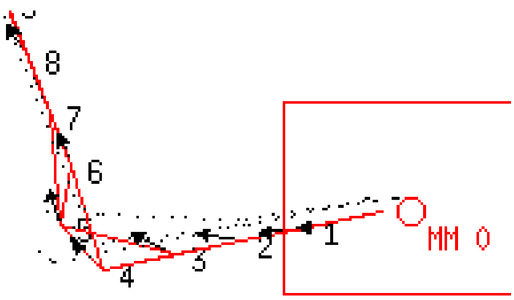
Figure 4.14 graphically illustrates the results of the operation of the $\ominus$ block for a mobile platform during the execution of a task involving the avoidance of a static obstacle (this example is further detailed in Chapter 5). MM_0 represents a mobile platform that must reach the location represented by the small circle, denoted by *Goal*. In the meanwhile, MM_0 must avoid the static obstacle represented by the rectangle MM_1. The notation MM_i is extensively used in Chapter 5 to denote the i-th *mobile manipulator* present in an experiment.

In all the experiments in the thesis, the $B^p(q^p)$ regions are cylinders in the x, y, θ space, with simmetry axis θ, located at (x^p, y^p), and diameter 0.2 m (thus being represented by circles, with diameter 0.2 m, in the xy plane centered at (x^p, y^p)). Hereafter, to avoid cumbersome statements, the $B^p(q^p)$ regions are also denoted by B^p. The numbers represent the values of the p index define the creation sequence of the $B^p(q^p)$ regions. Whenever a new region is created, the location q^p is linked to the two nearest neighbours by a solid line[6]. The mapping M follows these links to update, using (4.11), the actions in the neighbour regions of the one the mobile platform is in. The small solid lines with origin at q^p (represented by arrows in the detail plot shown in Figure 4.14b indicate the motion direction being proposed, at the time the plot was generated, by the action a^p.

[6]Grey in black and white printing, red in a colour printing.



(a) Full plot.



(b) Detail of the region around the starting point.

Figure 4.14: ⊜ block graphical output.

Chapter 5

Experiments on *behaviour* based cooperation and control

5.1 Introduction

This chapter presents three series of experiments using the architectures supported in the framework defined in Chapters 2 and 3 and the set of actions described in Chapter 4. Some of the experimental results were first presented in [Sequeira and Ribeiro, 1998a] and [Sequeira and Ribeiro, 1998b]. Section 5.2 presents a general discussion on the performance of robots and reviews known performance assessment criteria in a cooperation perspective. Sections 5.3 to 5.8 present the experiments which are summarized in Table 5.1.

The first series of experiments, included in Section 5.3, focuses on the difference of performance between the *actions* based (Figure 2.15) and the *behaviours* based (Figure 2.16) architectures for single mobile manipulators. The second series, presented in Sections 5.4, 5.5 and 5.6, focuses on cooperation among robotic devices tightly coupled, i.e., when the motion of an agent is constrained by the motion of another agent, independently of the assigned task. This is the case of the single mobile manipulator described in Chapter 4, where the manipulator is rigidly linked to the mobile platform. The third series, presented in Sections 5.7 and 5.8, is motivated by material handling operations in a warehouse and emphasizes the cooperation among robotic devices loosely coupled, i.e., the motion of an agent may/may not be constrained by the motion of another agent, depending on the particular task assigned and/or on the environment. All of these experiments consider two identical mobile manipulators of the type described in Chapter 4.

In the second series of experiments, Section 5.4 considers an isolated mobile manipulator in an obstacle free environment. The results presented refer to both the real and the simulated setups aiming at validating the simulated setup as a tool to aid in the design/tunning of the actions to be used by the real setup. Several missions are presented, with slight differences between them, to

Single agent experiments

Experiment	Section	Real	Sim.	Summary
Actions vs. *behaviours* architecture	5.3		×	The *actions* and *behaviours* architectures are compared in the control of a mobile platform.

Single mobile manipulator experiments

Experiment	Section	Real	Sim.	Summary
Real vs. simulated setups	5.4.1	×	×	Compares the two setups on an identical mission designed to avoid interactions with external environment. The *actions* architecture is used.
Pushing a button	5.4.2	×	×	Compares the two setups on the same push button mission.
Corridor traversal	5.5.1	×		Test of the corridor traversal with reduced obstacle avoidance interactions. MM_0 starts aligned with the corridor middle line, near the entrance.
Corridor approach and traversal	5.5.2	×		MM_0 starts unaligned with the corridor middle line, far from the entrance. Two separate phases, approach to the corridor and traversal, are considered.
Door traversal	5.5.3	×	×	The experiments on corridor approach and traversal are merged with a set of movements to push the handle and open the door.
Grasping an object over a table	5.6	×		Multiple missions consider situations with and without unexpected obstacles to be avoided by the mobile platform.

Experiments with a bar and two mobile manipulators

Experiment	Section	Real	Sim.	Summary
Robustness test 1	5.7.1		×	MM_1 is forced not to move (as in a malfunction), while MM_0 tries to execute its mission without letting the bar fall down. The *actions* architecture is used.
Robustness test 2.	5.7.2		×	The *actions* and the *behaviours* architectures are compared in a non-cooperative mission, where the distance between the task locations is either too small or too large when compared with the dimension of the bar.
Active task changing strategy by MM_0.	5.7.3	×		An alternative strategy for the supervisors in MM_0 to react to excessive or insuficient distance between the bar grasping points is tested.
Warehouse 1.	5.7.4	×	×	Transportation and deposit of a bar onto a shelf in a warehouse. The *actions* architecture is used.
Warehouse 2.	5.7.4	×	×	A slight variation is introduced in the previous MM_0's mission.
Warehouse 3.	5.7.4	×		Further variations are introduced in the previous mission assigned to MM_0. MM_1's mission is kept unchanged.
Warehouse 4.	5.7.4	×	×	Influence of the dynamics of each of the setups in the task execution.
Warehouse 5.	5.7.4	×	×	Mission requiring the relative exchange of positions by the two mobile manipulators. An intermediate task is used aiming at simplifying the overall mission.
Warehouse 6.	5.7.4	×	×	The intermediate task of the previous mission is removed.
Bar transportation through a corridor.	5.8		×	The starting configurations of MM_0 and MM_1 force to some obstacle avoidance interaction before entering the corridor, with the LUT method steering the mobile platforms in an almost straight line along the corridor.

Table 5.1: Overview of the experiments presented in this chapter.

provide a better understanding of the related phenomena. Section 5.5 considers a door traversal experiment by a single mobile manipulator. The mission is divided into several phases, each of them illustrating relevant aspects of the performance of the system. Results using both real and simulated setups are presented. In Section 5.6 a grasping experiment in a real, unstructured, environment is considered.

For the third series of experiments, Section 5.7 presents a set of experiments using two mobile manipulators. Each of the considered experiments illustrates potential situations arising inside an automated warehouse, where large bars must be handled and stored by mobile manipulators. The robustness of the system, i.e., the ability to react to adverse operating conditions, namely the failure of one of the mobile manipulators, is verified in two of the experiments. Finally, Section 5.8 considers the transportation of a bar, by two mobile manipulators, along a corridor.

With the exception of the mission in Sub-section 5.4.2, all tasks are assumed to be of the first class defined in Section 4.4, i.e., at the task location a **Null** action is executed.

The missions are specified by the x, y, z location and α, β, γ[1] orientation of the tasks in the case of gripper dominance. Whenever the mobile platform is the dominant agent, its orientation, θ, is also specified. A tolerance value indicates the maximum position error (in the xyz space) at the end of the execution of each task. Units are meters and radians. For the sake of simplifying the implementation of the 3D display software, the missions are specified using an anti-direct reference frame for positions and a direct reference frame for orientations (shown in the figures whenever relevant).

The display of the results is supported in plots of the trajectories in the xy and xz planes and in the xyz workspace. Whenever relevant, plots of the orientation of the mobile platforms are also presented. If not otherwise stated, the starting and the task locations are represented, respectively, by the symbols $\circ$ and $+$. The trajectories of the mobile platform are represented in the xy plot using dashed lines. It should be noted that the difficulty of having a clear 3D representation limits the readability of the figures and rends difficult the assessment of the system performance, which is clearly displayed with a video. A list of the sources of relevant MPEG2 video files is included in Appendix F and available through the World Wide Web.

The software implementing the control architecture, which is identical both for the simulated and the real setups, has an average cycle time of 0.3 seconds. For the low level control software, the simulated setup has a minimum value of 0.01 seconds[2], while for the real setup the low-level control software has an average cycle time of 0.12 seconds[3]. Additionally, the data exchange (using

[1] Euler angles are used.

[2] Note that the whole software system runs in a multi-user workstation, this introducing a dependency of the sampling period on the computational load.

[3] Both the Robuter mobile platform and the GT 6 manipulator control boards use a clock period of 0.01 s. The respective control software requires an average of 0.08 s per cycle for the manipulator and 0.12 s for the mobile

a 9600 baud serial line) with the real setup introduces a minimum delay of about 0.2 seconds (in the absence of communication errors). The delay introduced by the data exchange (using Unix socket based communications) with the simulated setup can be neglected. Thus, for performance assessment purposes, the real and the simulated setups are considered, respectively, as having an average cycle time of 0.62 and 0.31 seconds. Whenever the arm is stretched, the practical effect of this delay, affecting both the inmost and the outmost feedback loops of the control architecture[4], is "amplified" by the arm, i.e., a small error in the angular position in the base joint generates a larger error in the position of the end-effector. To reduce the consequences of this problem a time horizon is associated to each arm action. This time horizon represents the time interval after which a null action is automatically executed. Unless otherwise noted, the time horizon considered in the presented experiments is set to 0.5 seconds. The mobile platform actions do not use time horizons to avoid the frequent accelerations and decelerations that may lead to wheel slippage and the subsequent odometry errors.

Most of the experiments in Section 5.5 and those in Section 5.6 were conducted with the control architecture running in a multi-user Sun Sparcstation 10 workstation. In this case the average cycle time degraded to 2 seconds. All the other experiments were conducted with the control architecture running in a multi-user, dual processor, Sun Sparcstation 20 workstation.

5.2 Criteria for the performance assessment

This section presents a performance measure defined in the context of the proposed *behavioural* framework. This measure is motivated by a brief discussion on maneuvering, as currently defined in literature, and on a performance measure proposed by Laumond.

Laumond, in [Laumond, 1993], used the sum of sign changes of each control variable, u_i, to measure the performance of a mobile platform trajectory,

$$\sum_i \#(\text{sign changes of } u_i). \tag{5.1}$$

Measure (5.1) may give arbitrary high values for a trajectory of the arm with $x, y = 0$, e.g., a straight line, upward, trajectory along the z axis, with the base joint alternatively rotating in the positive and negative directions. Although expressing the complexity of the control, (5.1) fails to express the influence of the straight line trajectory on the task execution (hypothetically, the motion in straight line). This simple example shows the dependency of (5.1) on the particular type of robot considered.

For mobile platforms moving in the $\mathbb{R}^2$ workspace, Latombe, in [Latombe, 1992], pp. 424, defines

platform.

[4]The exact contribution of each of the feedback loops to the delay was not investigated. The presented value represents an optimistic estimate for an upper bound value.

maneuver as any change in the sign of the linear velocity of the platform. These sign changes correspond, for example, to the existence of cusps in the trajectory but may also lie hidden in a straight line trajectory. Latombe's definition of maneuver can be applied to the trajectories generated by the manipulator considered in the thesis and thus a robot independent performance measure can be defined as the total number of "break" points in the trajectory. As it will be apparent from the experiments presented in this chapter, the assessment of the performance of the system during the execution of a task or mission using the number of maneuvers exhibited by the trajectories of one or more agents is highly dependent on engineering aspects not directly related with the framework proposed in the thesis, namely the delays in the feedback loops. Moreover, non-smooth trajectories generated by a manipulator may be less harmful, from a task execution perspective, than non-smooth trajectories generated by a mobile platform.

An interpretation for the term maneuver, broader than Latombe's definition, was presented in Section 3.3, in association with state transitions. The underlying rationale is that, using Definition 2.3.2, the trajectories generated by two identical states may differ by an arbitrary number of maneuvers (in Latombe's sense) and still provide identical contributions for the execution of a task. Therefore, even though the number of maneuvers (in Latombe's sense) expresses the complexity of a single trajectory, it may fail to capture the complexity of the task. On the contrary, the number of state transitions during the execution of a task expresses this complexity and will be considered as the performance measure in this thesis.

The above discussion motivates the philosophy followed in the definition of the experiments presented in this chapter. Sections 5.5 and 5.6 emphasize, for a set of missions representing relevant situations, the performance measure defined above through the presentation of the temporal evolution of the switching between actions.

5.3 Actions vs. *behaviours*

This section presents a set of experiments on the ability of the considered mobile platforms to avoid obstacles under both the *actions* and the *behaviours* architectures. The mission starts at configuration $(x, y, \theta) = (3.9, 2.07, 0)$, which is represented by the symbol $\circ$. MM_1 remains static at configuration $(x, y, \theta) = (2, 2.9, \pi/2)$ and is used as an *a priori* unknown obstacle to be avoided by MM_0's mobile platform. The task is located at $(0.97, 3.06)$ in the xy plane, and in all diagrams of Figure 5.1 is represented by the symbol $\circ$.

Figure 5.1 shows a chronological sequence of four diagrams illustrating the trajectories and the iterative creation of the B^p regions. Each B^p region is a cylinder with a 0.2 m radius base in the xy plane and symmetry axis θ. The diagrams show the projection in the xy plane of the center of each of these regions. The numbers (representing the value of the p index) indicate the chronological sequence of region creation. The network of solid lines illustrates the connections established between neighbouring regions and used by the learning stage in Table 3.4, i.e., each B^p region is connected to the two nearest neighbour regions (as defined in Section 4.8, the metric is the Euclidean distance between the respective (x^p, y^p) center coordinates. The motion directions generated by the actions proposed at each region are represented by the small segments. The trajectories of the mobile platform are represented by dotted lines. The first plot shows the trajectories for the first three runs superimposed. This notation was already detailed in Section 4.8, were diagram 1 of Figure 5.1 was carefully detailed and parts of it zoomed for the sake of a clear understanding.

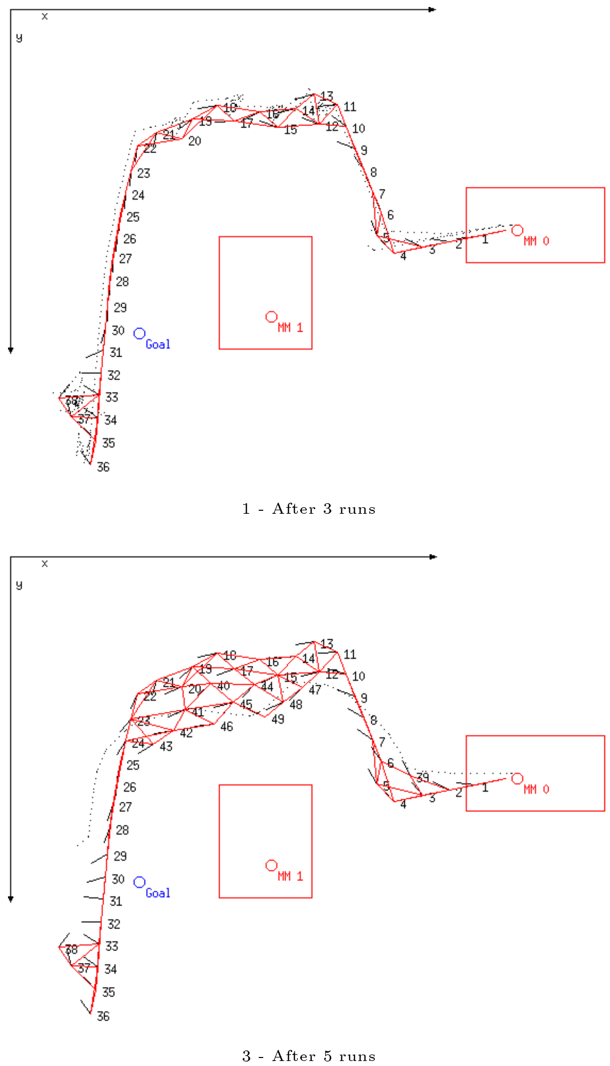


Figure 5.1: Four samples using the ⊜ block in a mission with obstacle avoidance.

In each iteration of the algorithm, the mapping M, defined in Section 3.4, in Table 3.4 is applied to each of the two nearest neighbours of the current pseudo-state (denoted by $a^p(q^p)$ in Section 3.4), to update their actions, making them closer[5] to the current pseudo-state action.

In the first run the *actions* architecture is used. The LUT based obstacle avoidance scheme does

[5]In the sense that, locally, the trajectories generated by two actions are close to each other.

not handle correctly the collision avoidance leading to an emergency stop. This can be observed in diagram 1, near the region B^4, at the point were the dotted line stops. Figure 5.2 shows a detail of this region.

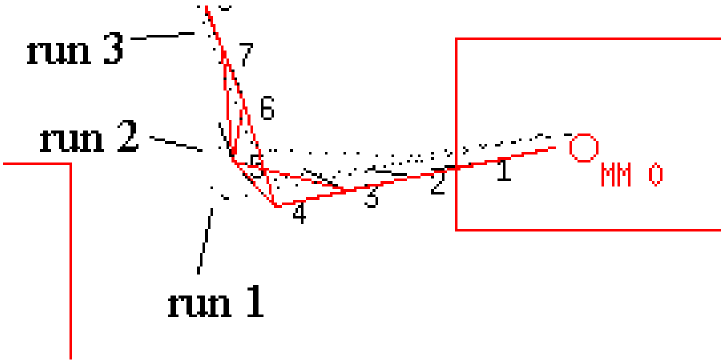


Figure 5.2: Detail of diagram 1 showing the broken trajectories corresponding to emergency stops in runs 1 and 2 near the beginning of the mission.

During the second run, the learning algorithm of Table 3.4 changes the proposed actions deviating the trajectory and leading to the creation of B^5. Nevertheless, the change was not yet enough to avoid the collision. During the third run the proposed actions continue to be modified leading to a greater deviation from the initial pattern and to the creation of B^6. At this stage no collision occurs and the LUT based scheme steers the mobile platform away from the obstacle.

The region starting at B^{10} and extending to B^{19} is characterized by an intense maneuvering, with the actions alternating between **GoStG** and **UnOA**. Starting in the vicinity of B^{22}, the mobile platform heads directly to the goal, with either the **GoStG** or the **UnOA** actions generating the same straight line motion. Near the goal, **UnOA** prevails over **GoStG**, detecting a left wall (in this case the stopped MM_1) so the mobile platform continues its motion until region B^{36}, where the obstacle ceases to be detected. The **GoStG** action generates the backward motion, leading the mobile platform to B^{37} and B^{38}. At this point the arm is able to finish the execution of the task.

The remaining three diagrams illustrate the creation of alternative regions by the learning algorithm. Region B^{39} expands the influence of the obstacle avoidance action leading to the start of the avoidance motion from the beginning of the mission, instead of waiting until the obstacle is detected. Alternative maneuvers found by the mobile platform to move around the obstacle are represented by the additional branches of the network shown in the upper area of the diagrams. This experiment shows how the use of the learning strategy may improve the basis actions.

5.4 Single mobile manipulator in an obstacle free environment

This section presents two sets of experiments, testing the two classes of tasks referred in Section 4.4. The first set (Sub-section 5.4.1) contains five runs of the same mission considering both the real and the simulated setups. The mission in the second set of experiments (Sub-section 5.4.2) is composed by a single task. Instead of having a null action executed at the task location, an action generating a short straight line to be followed by the gripper (as if the manipulator had to push a button) is considered.

In both sets of experiments the environment is free of obstacles. Thus, only the odometer data is used during the mission.

5.4.1 Real vs. simulated setups

This experiment assesses the relative performance of both setups in a situation where the external environment does not force any interactions with the agents other than the ones due to physical couplings between them.

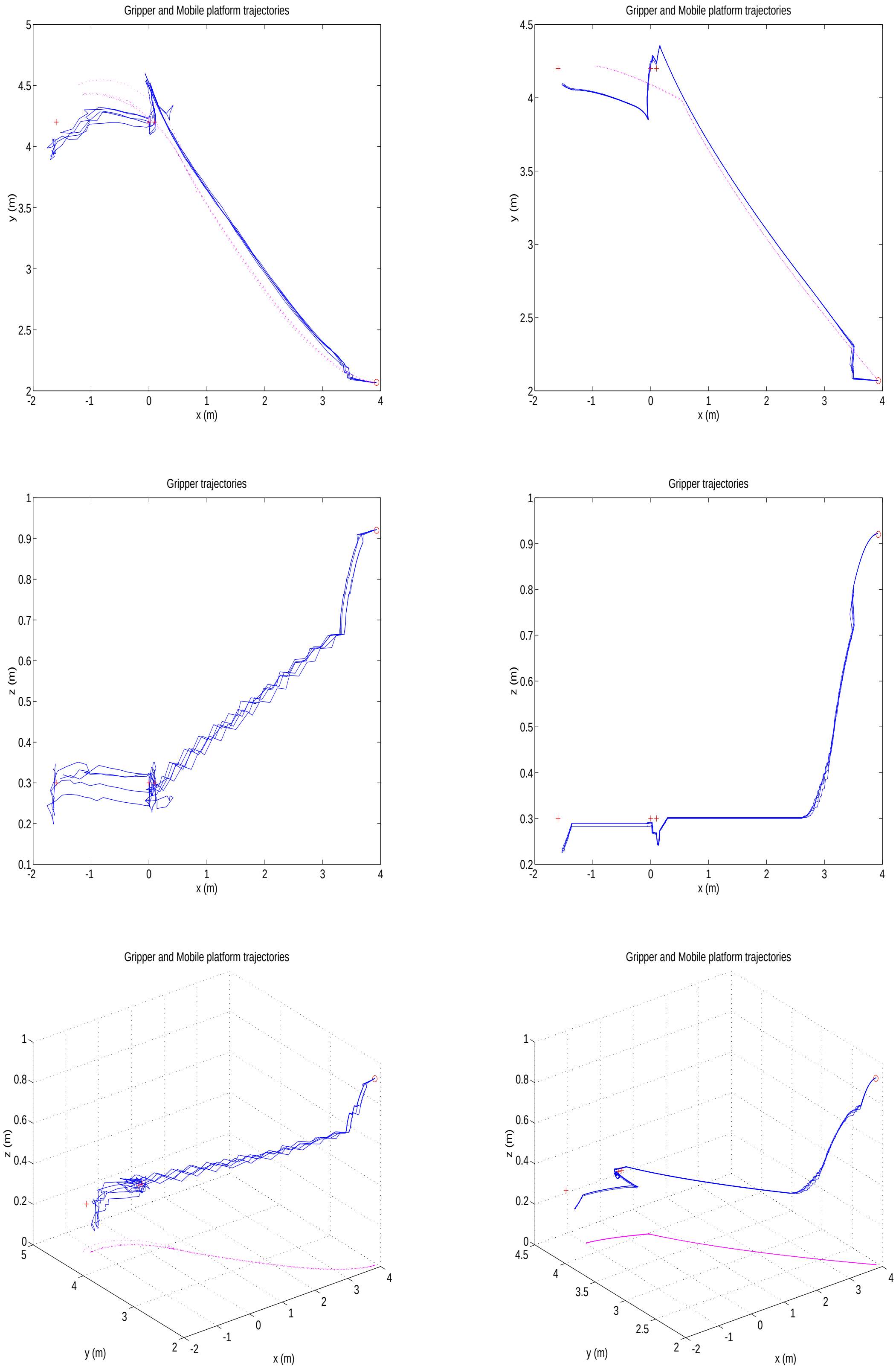
The mission, specified in Table 5.2, is composed by three tasks, all of them referred to the gripper considered as the dominant agent (see Definition 4.4.1) during the entire mission. Figure 5.3 shows the results obtained for 5 superimposed runs.

	x	y	z	θ	γ	β	α	tolerance
Start	3.93	2.07	0.92	0	0	0	0	-
Task 1	0.1	4.2	0.3	-	0	-1.57	0	0.05
Task 2	0	4.2	0.3	-	0	-1.57	0	0.05
Task 3	-1.6	4.2	0.3	-	0	-1.57	0	0.15

Table 5.2: Mission definition.

The trajectories generated by both setups are qualitatively similar. For each of the setups, the "breaking" points near Task 1, observed in all the mobile platform trajectories in the bundle, indicates the region where the mobile platform waited for the arm and gripper to complete their tasks.

The relative differences observed between the trajectories generated by the two setups are due to the delay in the exchange of data between the robots and the computer implementing the control architecture. This problem, which causes the slight oscillation in the xz plot of the real setup, is particularly relevant for the manipulator as it has physical limits constraining its workspace. When far from the limits, as in this case, the trajectories are qualitatively similar to those of the simulation setup, thus supporting the claim on the use of the simulation environment as a tool to design/tune the initial actions.



Real setup. Simulated setup.

Figure 5.3: Performance of the real vs. simulated setups (5 superimposed runs).

5.4.2 Pushing a button

In this section a mission of the second class (see Section 4.4) is presented. The mission has a single task with the particularity that the action to be executed whenever the arm reaches the task location is not the **Null** action. Instead, the arm is compelled to move along a straight line for 0.1 m in the direction pointed by the gripper at that time, stopping afterwards. The resulting movement is similar to the act of pushing a button. Table 5.3 details the mission with the single task being referred to the gripper. Figure 5.4 presents the trajectories obtained using both the real and the simulation setups, for a single run of the mission. The left column in this Figure represents the trajectories followed up to the task completion, whereas the diagrams in the right hand side detail some parts of the trajectories.

	x	y	z	θ	γ	β	α	tolerance
Start	3.93	2.07	0.92	0	0	0	0	-
Task 1	0	4.2	0.3	-	0	-1.57	0	0.05

Table 5.3: Task assignment for the push button experiment.

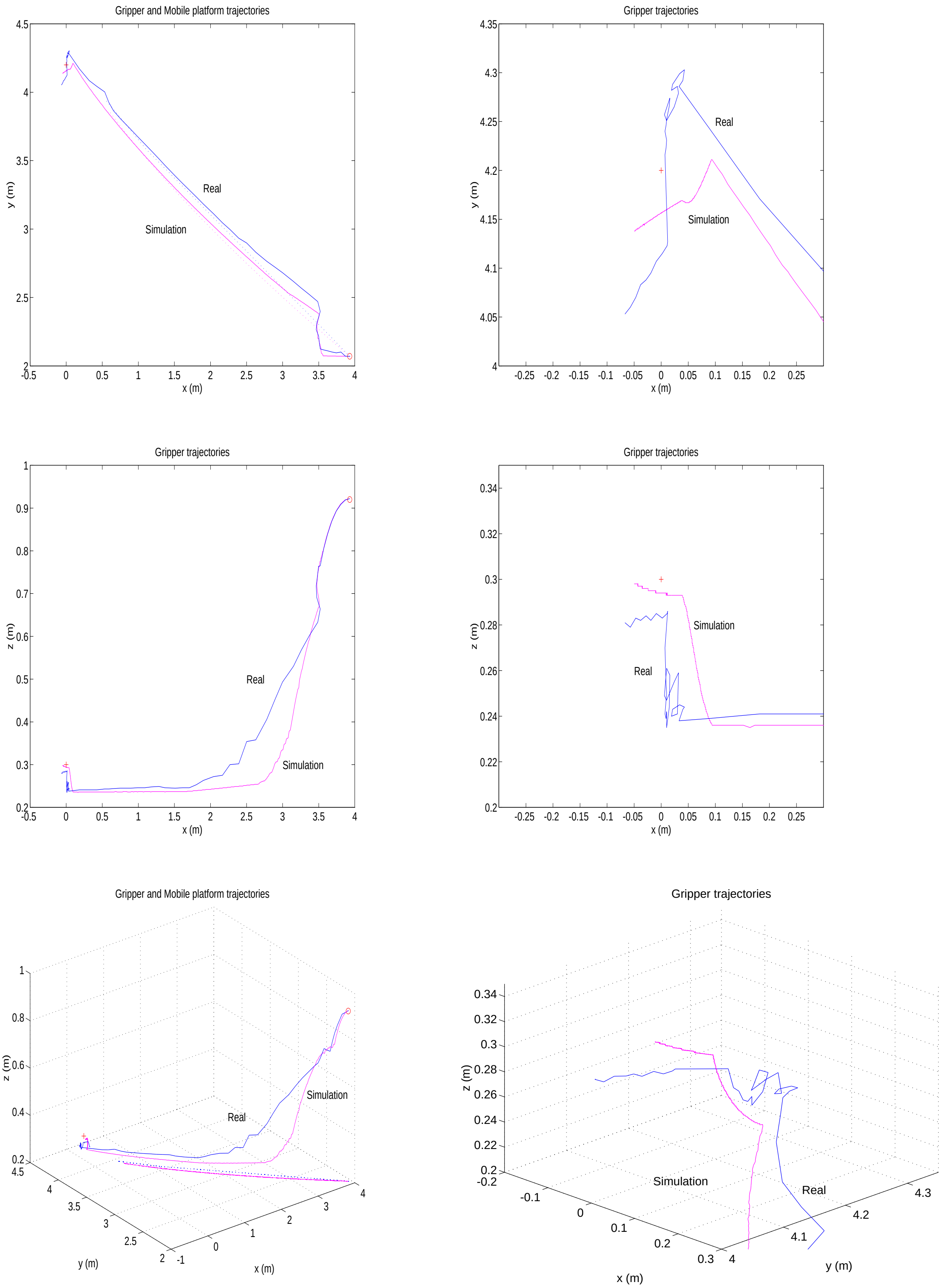
The mission does not require any special maneuvering by the mobile platform, as it can be perceived from the trajectories presented (dashed lines in the xy plot) for both the setups. In both cases the mobile platform stops at approximately 0.5 m of its task and waits for the arm and the gripper to complete their own tasks. While far from the task location, the gripper trajectories are qualitatively similar. Even though the mobile platform does not execute its task, the cooperation with the manipulator rends possible the existence of the solution to the global task (that is referred to the gripper). Once the mobile platform reaches a position from which both the arm and the gripper can execute their tasks the mobile platform stops.

The plots in the right hand side of Figure 5.4 show a detail of the gripper trajectories during this final phase of the mission. At the approach to the task location, the delay in the feedback loops of the control architecture is reflected in the higher number of "maneuvers" made by the real gripper when compared with the simulated one. In both cases, the action executed at the task location generates trajectories of rectilinear type, with some slight deviations caused by small variations in the orientation of the gripper. The difference between the orientation of the grippers in the two setups is reflected in the relative orientation of the two trajectories.

5.5 Door traversal by a mobile manipulator†

The experiments in this section consider the problem of approaching a door, opening it and moving through. The multiple missions in this section consider both the real and the simulated mobile manipulator.

†All the experiments in this section were performed at the Joint Research Centre of the European Commission, Ispra, Italy, with the exception of those in the last sub-section.



Complete trajectories. Trajectory details.

Figure 5.4: Button pushing experiment.

The environment contains static obstacles (the walls and the door) that generate multiple interactions with the agents. The observation of a human traversing a door suggests the three phase approach specified in Table 5.4, composed by *approaching*, *opening* and *traversing*, as a guideline to the mission specification. The distinction between the three phases seems a consequence of a planning activity that is beyond the scope of this thesis.

Phase 1: **approaching**	The human tries to reach an adequate positioning, facing the door, so that the remaining tasks are simplified. Obstacle avoidance is a major concern in this phase. This phase includes the approach to the door corridor.
Phase 2: **opening**	The human stops and moves the arm to open the door. Eventually, the traversal of part of the door corridor is required before being able to open the door.
Phase 3: **traversing**	The door is opened and the human just moves through, in a straight line, using the arm to push the door. The traversal of the door corridor is completed during this phase.

Table 5.4: The three phases in door traversal.

Figure 5.5 shows a typical environment with a set of tasks specified according to the aforementioned three phases. The tasks are sequentially numbered after the corresponding execution sequence. The execution of Task 1 (the approach) leads the mobile manipulator to keep a safe distance from the obstacles surrounding the door area, thus avoiding enter any obstacle avoidance procedures. With Task 2 the mobile manipulator approaches the door and stops at a distance such that the arm is able to touch the handle. In this phase obstacle avoidance interactions may arise. At Task 3 the manipulator touches the door, as a preparation to the descent motion required to reach Task 4, at which the corresponding action operates the handle. In a practical application, the touch of the door handle must be made using either force control or some compliant device. To reach Task 5 the manipulator pushes the door causing the whole system to traverse the corridor while trying to reach Task 6. As suggested by Figure 5.5, it will be considered that the door approaching phase will always be done through a corridor, even though it may be as small as the door support structure. The experiments in this series assume that the length of the corridor is not neglectable, aiming at testing the system in more difficult door traversal conditions than when the corridor is of reduced length. The task assignment in Figure 5.5 assumes that it is not necessary to keep pushing the door while traversing the door corridor.

During each of the three phases, one of the agents composing the mobile manipulator has its role naturally emphasized. In Phase 1, (*approach*), the mobile platform has a dominant role, while in the subsequent phases (*opening* and *traversing*) the importance of the arm and gripper prevails over the mobile platform. This corresponds to the application of the agent dominance concept of Definition 4.4.1. Also supporting the idea of dominance is the fact that, in general, different sensorial capabilities, other than the ones required for low level control, are used by each agent.

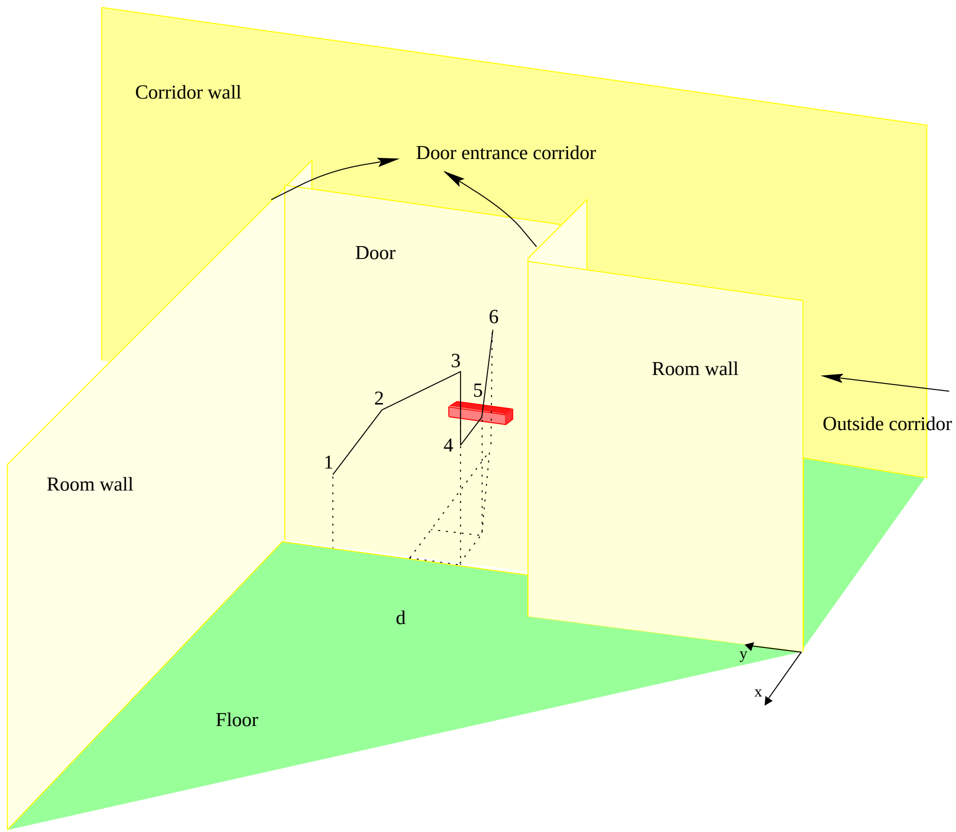


Figure 5.5: Possible task assignment (relative to the gripper) for the door traversal mission.

In the setups considered, the mobile platform is limited to the use of an ultrasound ring while the manipulator has no sensors. The mobile platform dominates in what concerns obstacle avoidance. In fact, the lower limit of 15 cm for the ultrasound measurement system in the real setup determines that blind navigation is used in corridors narrower than 1.03 m, as the mobile platform is 0.73 m wide. While approaching the door, the agent dominance concept leads to a task assignment such that the mobile manipulator attitude simplifies the execution of the subsequent tasks.

For the sake of highlighting the relevant aspects in the operation of the system, the experiments are divided into three groups: corridor traversal, corridor approaching and door traversal. As the experiments show, the definition of adequate tasks in the approach phase depends on how the mobile platform handles the corridor entrance and hence justifies this particular presentation sequence. In the first group, the system is assessed assuming that the mobile manipulator already passed Phase 1 and reached an adequate configuration to initiate either Phase 2, if a partial traversal of the door corridor is required, or Phase 3, if no corridor traversal is required in Phase 2. Instead of having the system executing the specialized tasks of opening the door, this set of experiments will assume that it is only necessary to push the door and traverse the corridor. The second group evaluates the system in Phase 1 (the approach of the door). The experiments in the third group illustrate the performance when the mission encompasses all the three phases.

5.5.1 Corridor traversal

This section presents the results of four experiments with the corridor traversal mission specified in Table 5.5. The values in the *Start* entry indicate the starting position of the mobile platform. The manipulator starts in a random configuration, with the z coordinate near the corresponding task value in all but one of the experiments. The gripper is considered the dominant agent.

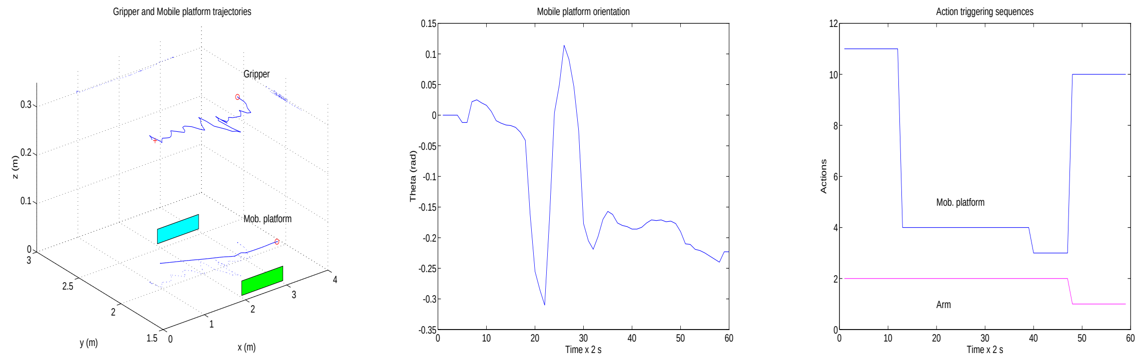
	Dominance	x	y	z	θ	α	β	γ	tolerance
Start	-	3.93	2.07	-	0	-	-	-	-
Corridor end	Gripper	0.97	2.07	0.3	-	0	-1.57	0	0.05

Table 5.5: Task assignment for the corridor traversal experiment.

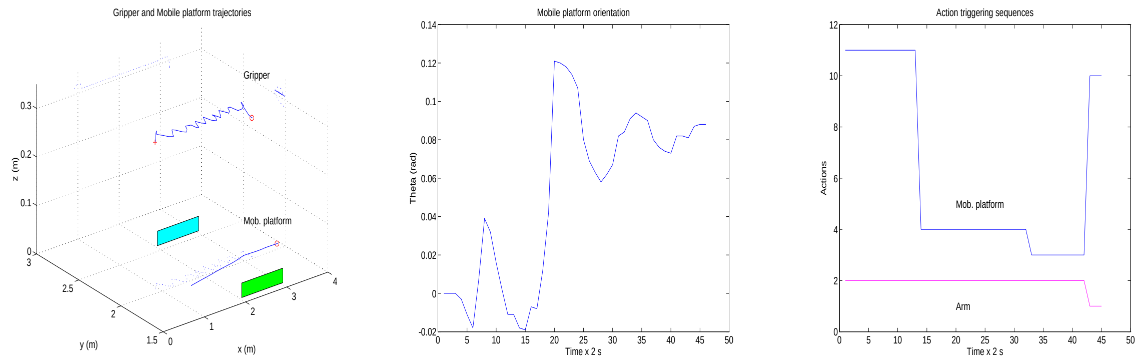
The mobile platform always starts aligned with the middle line of the corridor, as if it were at the location of the first task shown in Figure 5.5. The corridor has dimensions 1×1 meters (width $\times$ length). Giving the dimensions of the mobile platform, these values force the lateral ultrasound sensors to always detect the minimum distance, thus forcing the robot to blindly navigate while inside the corridor. The wall detection threshold is set to 0.35 m. In larger corridors, if the mobile platform starts closer to one of the corridor walls, the lateral wall LUT in the obstacle avoidance action, will force the robot to deviate from the wall. Eventually, it will approach the other wall and the same behaviour will be repeated. Repeated occurrences of these patterns will lead the mobile platform to move along the corridor middle line.

Figure 5.6 shows the results obtained in a first set of two runs, with the solid lines representing the trajectories of the gripper and of the mobile platform. The corridor walls are represented by the shadowed regions. The manipulator starts in a random configuration, with the value of the z coordinate near 0.3 m in both runs. For the sake of readability, the gripper trajectories projections on the xy, xz and yz planes are shown using dashed lines. The rightmost plot presents the actions' trigger sequences for the mobile platform and the arm. The gripper **GoStG** action is always triggered. Each numerical value in the y coordinate of these plots represents one specific action for each agent (different lines with the same value can represent different actions belonging to different agents). This plot allows, primarily, the verification of the actions trigger rate as a function of the environmental conditions.

The diagonal trajectories made by the mobile platform indicate the blind navigation phase. Although the actions' trigger sequence is the same in both runs, the global behaviour varies. In the first part of the trajectories of both missions the mobile platform is heading directly to the door, using the **GoStG** behaviour, (11). When the walls start to be detected by the ultrasonic sensors, the mobile platform switches to the **SaOA** action, (4). The detected obstacle, corresponding to the corridor entrance, is also perceived in the abrupt changes of orientation. When one of the walls becomes too close the **UnOA** action, (3), is activated. Near the task completion the arm sends a message to the mobile manipulator, indicating that it has enough manipulability to complete the



First run.



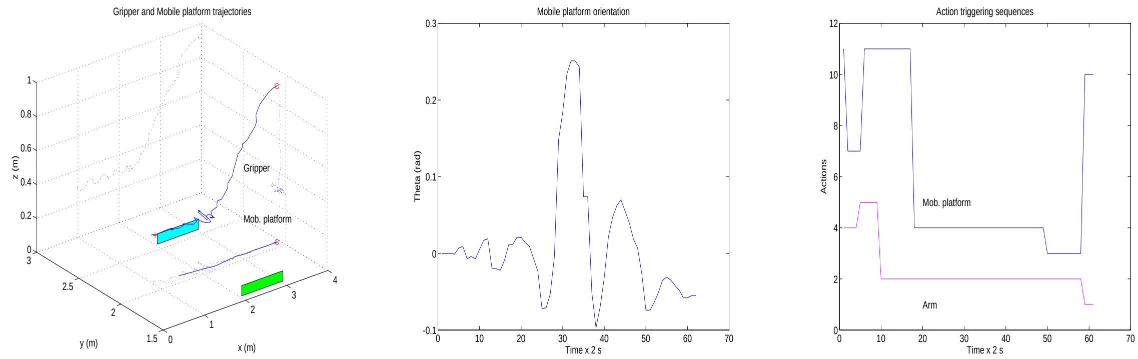
Second run.

Figure 5.6: Corridor traversal phase. Wall detection threshold set to 0.35 m. Two runs.

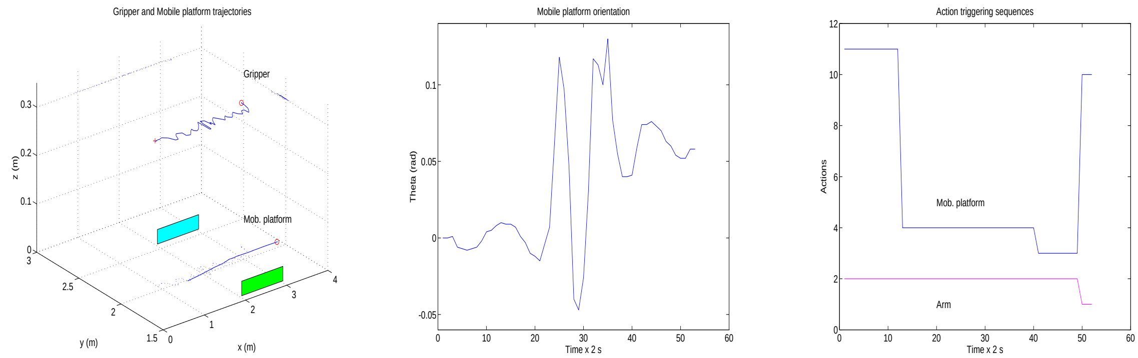
task by itself, and therefore the mobile platform stops using the **Null** action, (10). In what concerns the arm, it starts by applying the **GoStG**, (2), action. In the vicinity of the task it switches to the **ApG**, (1). The gripper keeps always triggered its **GoStG** action.

The second set of two runs was performed using a wall detection threshold of 0.15 m. Figure 5.7 shows the results for two runs of the mission. In the first run the manipulator starts in the upward position. In the second run the manipulator starts at the final configuration of the first run.

The actions triggered by the agents during these tests were the same as in the first set of experiments. However, due to the new wall detection threshold, the general obstacle avoidance LUT of the **SaOA** action (shown in Figure 4.8) was used by a larger period of time than in the first set of experiments. This alternative to the left/right wall LUTs of the **UnOA** action represents a slight increase in the space for free maneuvering (and hence trajectory adjustments) in the proximity of the corridor entrance. The adjustments of the mobile platform trajectory while inside the corridor are visible in the xyz plots. As in the first set of experiments, the abrupt changes in the orientation, corresponding to the obstacle detection, are visible in the middle plots.



First run.



Second run.

Figure 5.7: Corridor traversal phase. Wall detection threshold set to 0.15 m. Two runs.

5.5.2 Corridor approach and traversal

The next step in the evaluation of the system towards the complete door traversal mission is the merging of the previous experiments (Sub-section 5.5.1) with the approach phase (Phase 1 in Table 5.4) and the evaluation of the extent to which the area at the door corridor entrance must be structured, given the simplified obstacle avoidance scheme used. This section presents two experiments illustrating, separately, the approach phase (Phase 1 in Table 5.4) and the approach and traversal phases joined together (Phases 1 and 3 in Table 5.4).

As already observed in Figures 5.6 and 5.7, the LUT based obstacle avoidance method, using unprocessed ultrasound data, tends to make the mobile platform bouncing at the entrance of the corridor. To reduce this effect, the task representing the approach phase should force the mobile platform to approach the door corridor aligned with its middle line.

The mission specifying the approach and traversal tasks is defined in Table 5.6, with the *Start* entry referred to the mobile platform starting position. The manipulator starts in a random configuration in the two experiments. The obstacle and wall detection thresholds are set, respectively, to 0.35 m and 0.15 m and the gripper is the dominating agent in all the tasks. Figure 5.8 shows the trajectories and action sequences in the experiment testing the approach phase. Figure 5.9 presents the results of the experiment testing the approach and traversal. In both figures, the shadowed areas indicate the corridor. For the sake of understanding, a simplified door is also shown.

	Dominance	x	y	z	θ	α	β	γ	tolerance
Start	-	3.93	2.07	-	0	-	-	-	-
Approach	Gripper	3.08	3.25	0.3	-	0	-1.57	0	0.05
Traversal	Gripper	0.97	3.25	0.3	-	0	-1.57	0	0.05

Table 5.6: Task assignment for the corridor approach and traversal experiment.

To simplify the aforementioned alignment of the mobile platform with the corridor middle line, the approach task is not aligned with the corridor middle line. The projections of the gripper trajectories in the xy, xz and yz planes are shown as dashed lines.

The specification of the mission is simplified by the use of the dominance concept. The definition of the approach task relative to the gripper, as in Table 5.6, leads to an uncertain positioning of the mobile platform at the corridor entrance. As the width of the corridor is 1 m, the mobile platform is about 0.7 m wide and the ultrasonic sensors do not give measurements bellow 0.15 m, there is a null tolerance for trajectory adjustments. Therefore, the uncertainty in the position of the mobile platform at the corridor entrance may lead to additional obstacle avoidance interactions. These can be minimized by setting the mobile platform as the dominant agent for the approach task. Table 5.7 shows the corresponding mission specification also with the *Start* entry referring to the mobile platform starting position and with the manipulator starting in a random configuration at each run.

	Dominance	x	y	z	θ	α	β	γ	tolerance
Start	-	3.93	2.07	-	0	-	-	-	-
Approach	Vehicle	3.08	3.07	0.3	-	0	-1.57	0	0.05
Traversal	Gripper	0.97	3.07	0.3	-	0	-1.57	0	0.05

Table 5.7: Task assignment for the corridor approach and traversal experiment with multiple dominating agents.

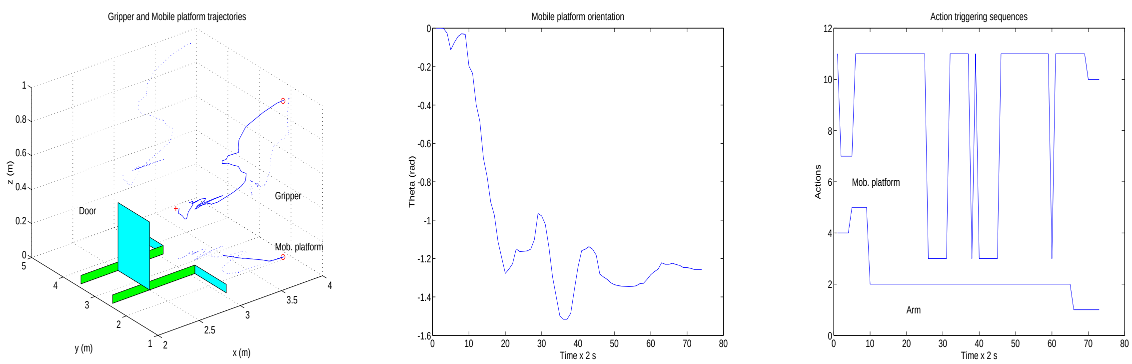


Figure 5.8: Experiment assessing the approach phase (gripper dominates).

The results of this mission are presented in Figures 5.10 and 5.11 . The runs in the Figure 5.10

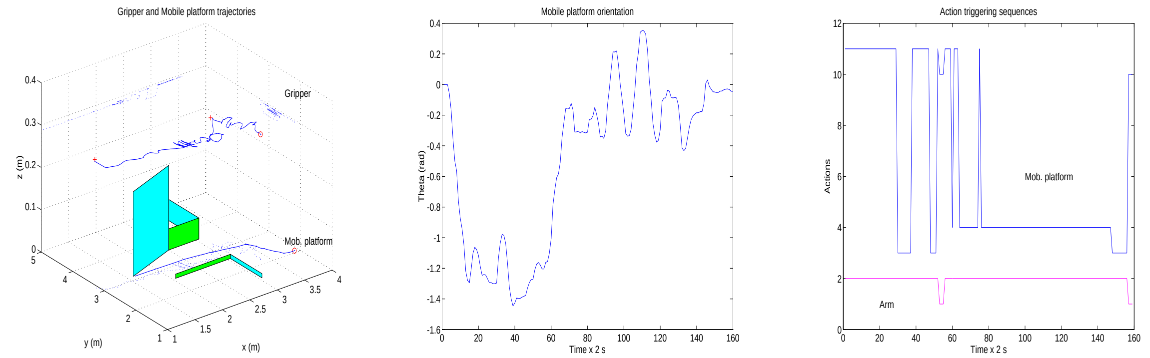


Figure 5.9: Experiment assessing the approach and traversal phases (gripper dominates).

(first and second runs) were made with the mobile platform "on blocks[6]" and without any external perturbation (e.g., walls). They serve to compare with the real experiments. The results in Figure 5.11 (third and fourth runs) correspond to runs in normal operating conditions, with the mobile platform running normally on the floor. The combined influence of the mobile platform dynamics and of the obstacle avoidance is clearly visible from a comparison between the orientation plots in Figures 5.10 and those in Figure 5.11. During the second run, the time horizon for the arm actions was set to 1 second. The delay in the feedback loops of the control architecture is clearly reflected in the gripper trajectory while traversing the corridor. All the other runs use the usual 0.5 seconds time horizon.

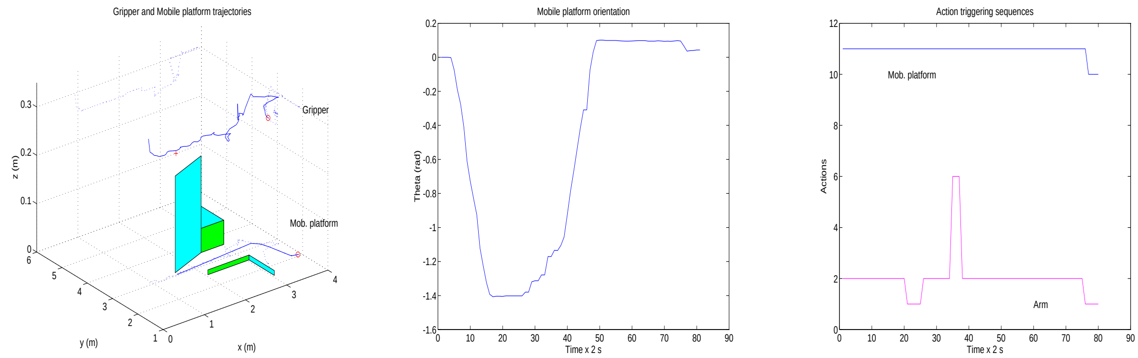
Figure 5.12 shows the views of the real setup at the beginning (upper left image), in the vicinity of the approach task (upper right and lower left images) and at the final of the mission (lower right image).

The results obtained when using the mobile platform as the dominant agent in the approach task show a switching rate between the mobile platform actions lower than that obtained when the gripper dominates all the tasks. When the gripper dominates, the orientation plots also show that the bouncing of the mobile platform at the corridor entry starts earlier than when the mobile platform dominates the first task. Ahead in this chapter, the same technique of changing the dominant agent will be used in other experiments.

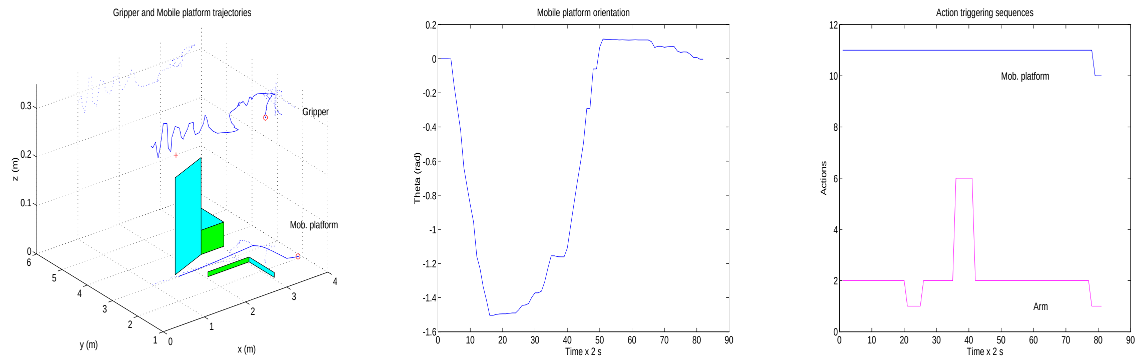
5.5.3 Door traversal

This section presents the door traversal experiment, with the six tasks represented in Figure 5.5, using both the real and the simulated setups. The corresponding mission, containing the six tasks outlined in Figure 5.5, is specified in Table 5.8 with the manipulator starting in the upward configuration and the gripper dominating all the tasks. Two environments are considered: environment

[6]This corresponds to a mobile platform placed on a structure such that the wheels are not in contact with the floor.



First run.



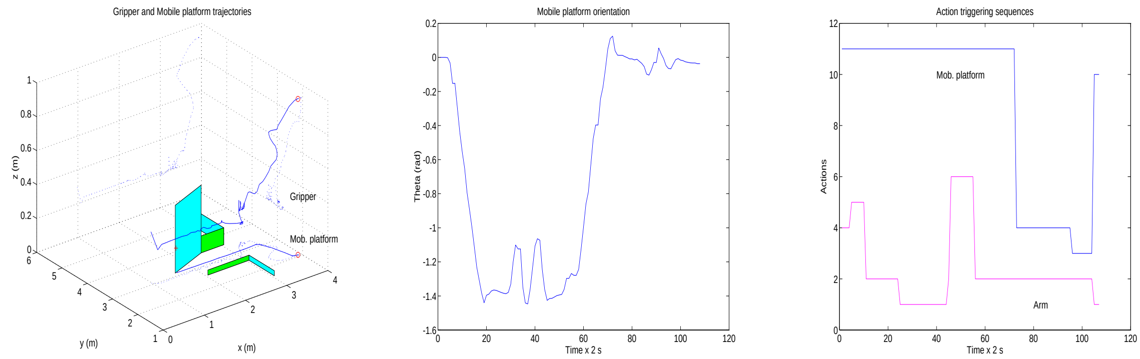
Second run.

Figure 5.10: Corridor approach and traversal (mobile platform dominates the 1st task). Mobile platform “on blocks”.

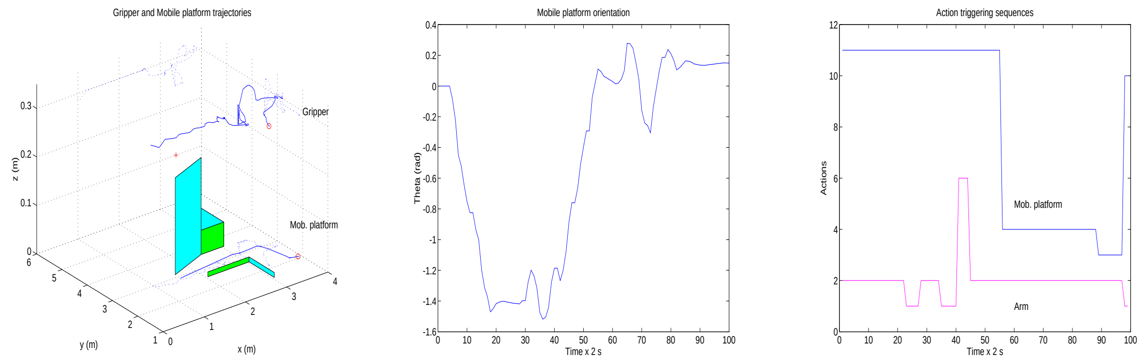
without corridor, used by both setups to establish performance references, and environment with corridor. In Table 5.8, the starting position and the location of the last task entries indicate two alternative values considered in the experiments. For the simulated setup the obstacle and wall detection thresholds are set, respectively, to 0.42 m and 0.35 m. For the real setup an obstacle detection threshold of 0.62 m is used, whereas the wall detection threshold remains unchanged.

In the absence of obstacle avoidance interactions between the mobile platform and the corridor entrance, the distance between the initial position and the location of Task 2 is large enough to have the mobile platform aligned with the corridor middle line before the terminus of the approach phase and thus it is not necessary to have the mobile platform dominating in the approach phase. For the real setup operating in the real corridor, this distance is further increased to ensure that the mobile platform reaches the corridor entrance in a configuration that simplifies the subsequent phases. Furthermore, the location of the last task is moved away from the corridor to avoid the potential maneuvering due to the obstacle avoidance interactions arising when the mobile platform moves away from the corridor.

The experimental results are presented in Figure 5.13. The left hand side plots considers both setups with no obstacles present (either corridor or door). The xy plot shows the trajectories of



Third run.



Fourth run.

Figure 5.11: Corridor approach and traversal (mobile platform dominates the 1st task). Mobile platform running normally on the floor.

the mobile platforms represented by their border lines. The shadowed areas in the bottom left plot represent the corridor and serve only to improve the readability of the figure. The right hand side plots consider both setups in an environment with obstacles. The corridor is 1.4 m wide and 1 m long, with wooden walls (see Figure 5.14). The corresponding xy plot represents the trajectories of the mobile platforms by their border lines. The shadowed volumes in the xyz plot represent the corridor.

The results obtained with the environment without obstacles show a similar behaviour by the two mobile manipulators, real and simulated, except for the final part of the mission, while reaching Task 4.

5.6 Grasping an object on a table‡

This section considers the problem of moving a mobile manipulator to grasp a static object placed on a plain surface, e.g., a table. For the sake of simplicity, it is assumed that no other objects that

‡Experiments performed at the Joint Research Centre of the European Commission, Ispra, Italy.

1 - Starting position.

2 - Approaching.

3 - Approaching.

4 - Traversal.

Figure 5.12: Four views of the corridor traversal experiment.

may disturb the grasping action are present on the surface. In all the experiments the obstacle and wall detection thresholds are set, respectively, to 0.35 m and 0.25 m.

As in the door traversal experiment, a number of fundamental phases are identified through the observation of human behaviour, aiming at providing a guideline for the task assignment. Table 5.9 presents the three phases considered for the grasping mission which are also schematically illustrated in Figure 5.15.

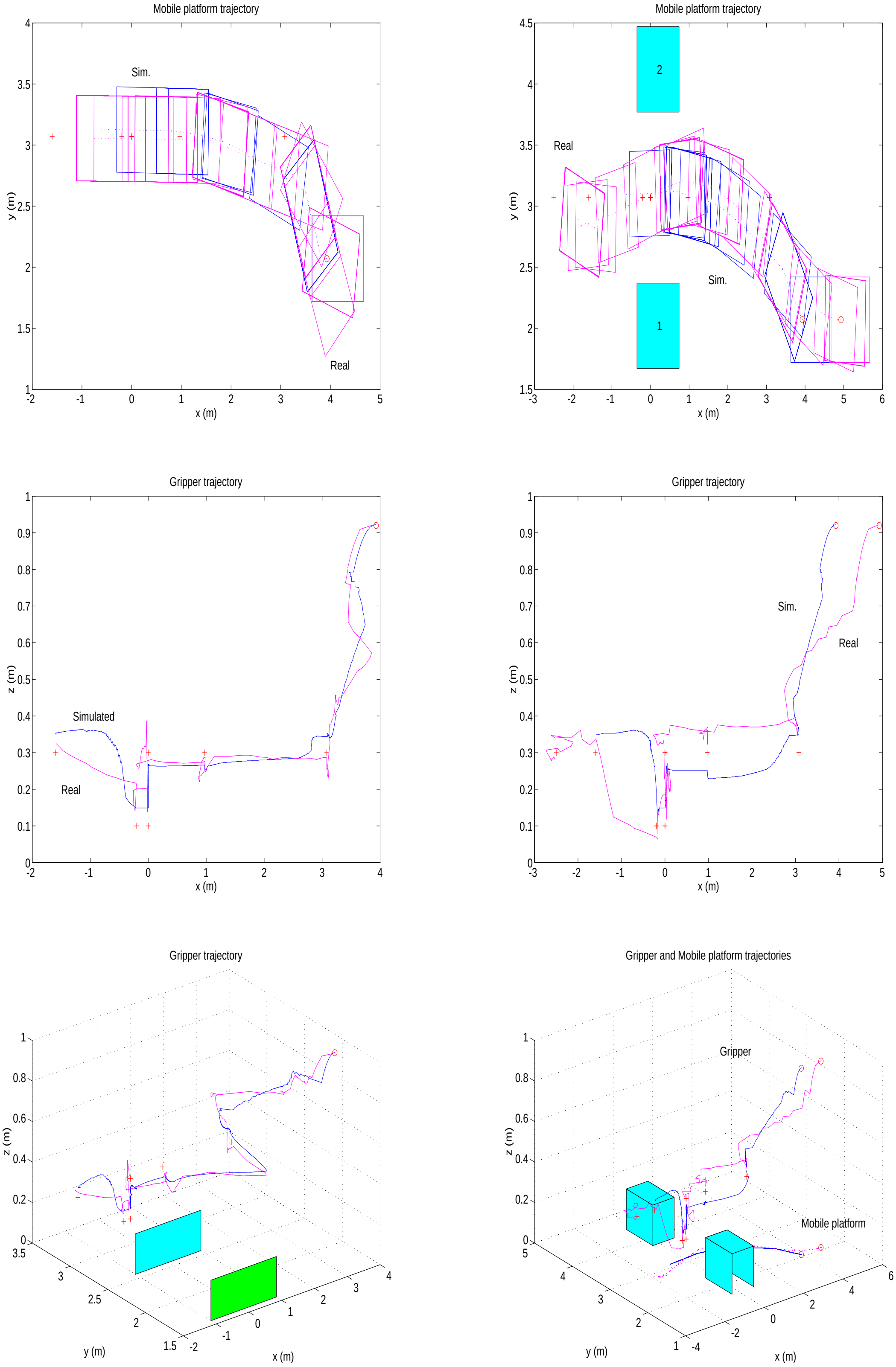
The mission specification is presented in Table 5.10, with the gripper dominating all the tasks. As the area surrounding the table is free of obstacles, Phases 1 and 2 are merged together. Three situations of the grasping mission were tested:

1. without obstacle detection,

2. with obstacle detection and uncluttered environment (see Figure 5.16), and

3. with a set obstacles placed in the path between the initial configuration and the first task (see Figure 5.17).

Figure 5.18 summarizes the results obtained for each of the three grasping situations considered.



No obstacles. With obstacles.

Figure 5.13: Complete door traversal experiment.

			x	y	z	θ	α	β	γ	tolerance
	Start		3.93 (4.93)[a]	2.07	0.92	0	0	0	0	-
Phase 1 {	Task 1	(far alignment)	3.08	3.07	-	-	-	-	-	0.05
	Task 2	(near alignment)	0.97	3.07	0.3	-	0	-1.57	0	0.05
Phase 2 {	Task 3	(touch door)	0	3.07	0.3	-	0	-1.57	0	0.05
	Task 4	(push handle)	0	3.07	0.1	-	0	-1.57	0	0.05
Phase 3 {	Task 5	(push door)	-0.2	3.07	0.1	-	0	-1.57	0	0.05
	Task 6	(traverse)	-1.6 (-2.5)[a]	3.07	0.3	-	0	-1.57	0	0.05

Table 5.8: Task assignment for the door traversal experiment (following Figure 5.5).

[a]Used by the real setup when operating in the environment with the corridor.

Figure 5.14: Final configuration, after the traversal of the hypothetical door.

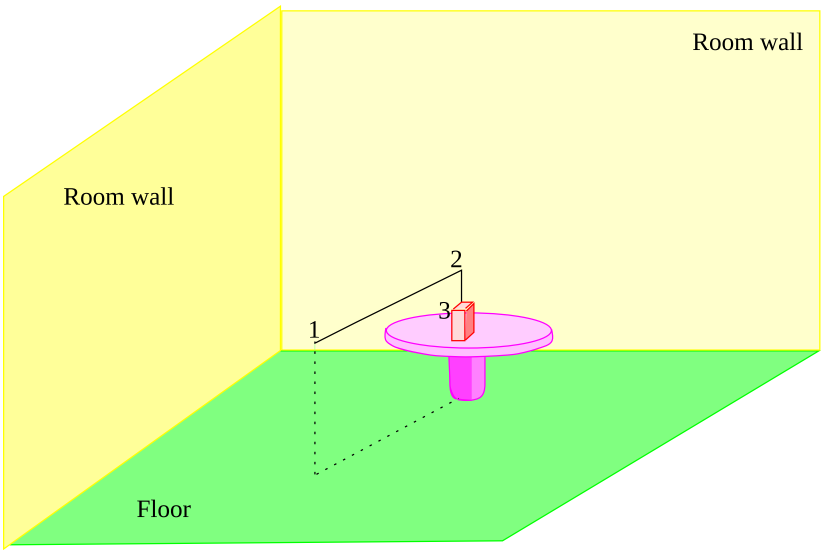


Figure 5.15: Grasping mission.

Phase 1:	Approaching the table	During this phase the mobile manipulator reaches a configuration such that the execution of the subsequent tasks is simplified. This phase requires open space navigation capabilities.
Phase 2:	Positioning of the manipulator	During this phase the end-effector of the manipulator reaches the vicinity of the object to be grasped. More specialized obstacle avoidance capabilities by the mobile platform are expected to be required.
Phase 3:	Grasp the object	This phase represents the actual grasping of the object.

Table 5.9: The *a priori* defined phases for the grasping mission.

	x	y	z	θ	α	β	γ	tolerance
Start	3.93	2.07	0.92	0	0	0	0	-
Approach and Positioning	2.61	4.15	0.3	-	0	0	1.57	0.05
Grasping	2.61	4.15	0.2	-	0	0	1.57	0.05

Table 5.10: Task assignment for the grasping experiment.

The gripper and the mobile platform trajectories are shown respectively in solid and dashed lines while the table is represented by the shadowed polygon.

In the first run the mobile platform starts with the **GoStG** action, (11). The arm starts the mission in the upward position, which causes the mobile platform to perceive it, during a short time period, in a double singularity position and causing the triggering of the **MoveBack**, (7), action. When the arm reaches a position away from the singularity region the mobile platform returns to the **GoStG**, that is kept until a neighbourhood of the location of the approach and positioning task is reached. At this time the mobile platform switches to the **Null** action, (10), waiting for the completion of the first task by the manipulator, this finishing the table approaching and positioning phases. The grasping task requires only a vertical movement of the arm. However, due to the delay in the feedback loops the arm wanders in the vicinity of the grasping task location causing the mobile platform to switch again to the **GoStG** action. This wandering is shown by the blurred region near the grasping task. After a short period of time the **Null** action is re-activated and the mobile platform holds its configuration, waiting for the arm and gripper to complete their tasks. The arm starts by using the **GetOutDS** action, (4), to move out of the S_3 region, followed by the **GetOutT1S**, (5), to move out of the S_1. It then heads to the approach and positioning task with the **GoStG**, (2), action until a neighbourhood of the task location is reached, where it changes to the **ApG**, (1), action. This action is kept active while reaching for the grasping task.

In the second run the arm starts at the final configuration of the first run by using the **GoStG**, (2), action and changing to the **ApG**, (1), action when the neighbourhood of the approach and positioning task is reached. The mobile platform starts heading to the first task using the **GoStG**, (11), action. The detection of the table triggers the **UnOA**, (3), action. A short oscillation

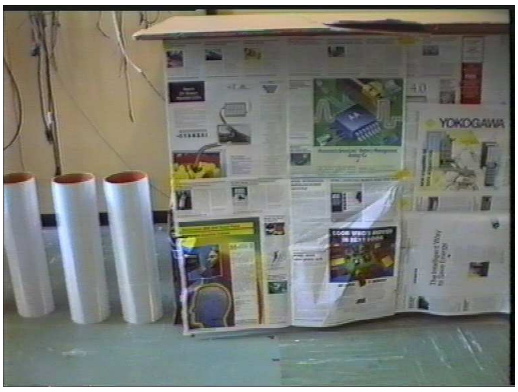


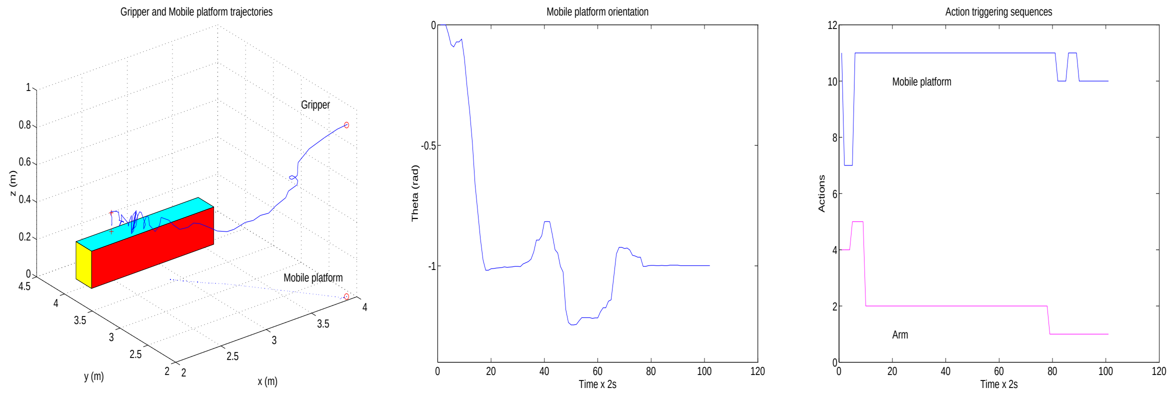
Figure 5.16: A general view of the proximity of the table. The paper covering the table support frame improves its detection by the ultrasound sensors. Three laser reflectors are shown on the left side of the image.

Figure 5.17: Unexpected obstacle situation at the beginning of the mission. A waste basket and two laser reflectors are considered.

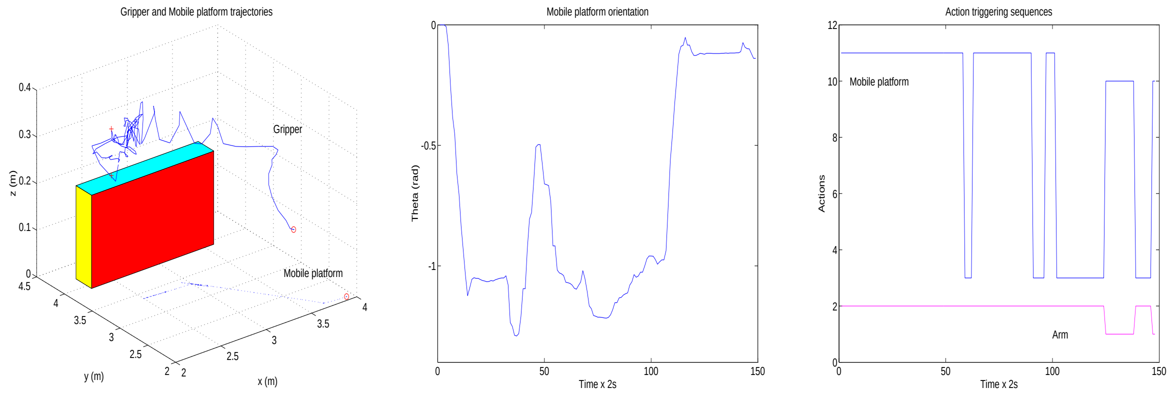
between backward and forward motion is observed at this point, due to the oscillation between the **UnOA** and **GoStG** actions. During the unsafe approach motion (see Chapter 4) the arm enters a neighbourhood of the first task, causing the mobile platform to trigger a **Null** action, (10), waiting for the arm to reach the first task. The arm finishes the first task using the **ApG** action, after which it wanders for a while due to the delay in the control loop. As in the first run, this wandering is illustrated in the plots by the blurred trajectory region over the table. The wander motion pulls the arm out of the neighbourhood of the grasping task inducing the mobile platform to adjust its position. This adjustment is performed by the **UnOA**, (3), action as obstacles are in the proximity. After the adjustment motion a **Null** action stops the mobile platform and the arm uses again the **ApG**, (1), action to finish the mission.

In the third run the unexpected obstacles (waste basket and laser reflectors) shown in Figure 5.17 are detected immediately at the beginning of the run, causing the switch to **SaOA**, (4). However, after avoiding these obstacles, the mobile platform continues to detect them, now with the ultrasound sensors located backwards, generating the oscillation between the **SaOA**, (4), and the **GoStG**, (11), actions. This oscillation does not cause any relevant perturbations on the mobile platform trajectory (recall that the LUT used in this situation simply instructs the mobile platform to move away from the obstacle in a straight line). As in the second run, the oscillation between the **Null**, (11) and the **UnOA**, (3), actions characterizes the final part of the mission, with the mobile platform aligning side by side with the table. To reach each of the tasks, the arm uses the **GoStG** action followed by the **ApG** action. As in the first two runs, the effect of the delay in the feedback loops is illustrated by the almost erratic gripper trajectory.

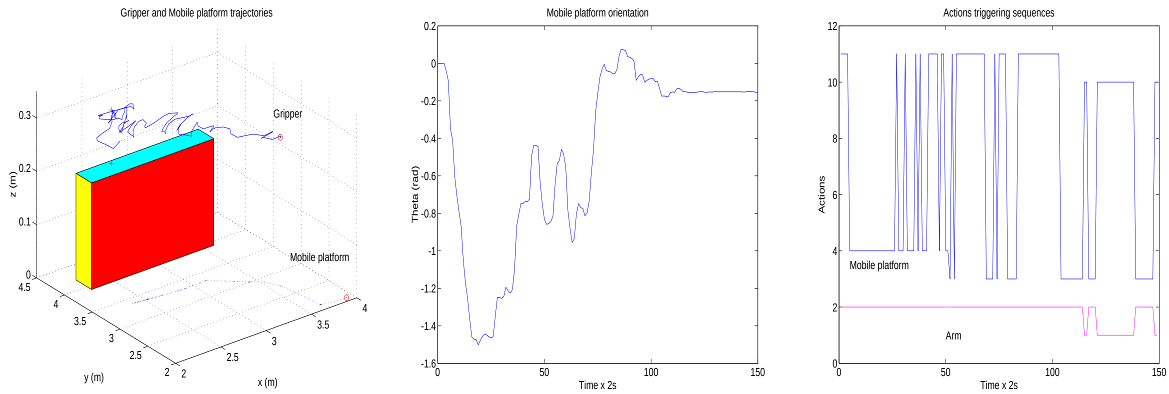
Figure 5.19 illustrates a typical final configuration for the mobile manipulator.



Run 1.



Run 2.



Run 3.

Figure 5.18: The grasping experiment.

Figure 5.19: Typical system configuration at mission completion.

5.7 Transportation of a large bar by two mobile manipulators

This section presents a set of experiments consisting in the transportation of a bar (3 m long in all the experiments) by two identical mobile manipulators. A loose coupling between the two mobile manipulators is assumed. Both experimental setups are considered.

This problem has a practical connection with the automated warehousing of heavy materials where, together with the handling of the materials, multiple interactions between the agents and the external environment must be considered, namely the mobile platforms avoiding collisions between themselves and with other obstacles.

Figure 5.20 shows a conceptual warehouse with a potential assignment of tasks for the mobile manipulators operating inside. The three phases identified on the figure have a counterpart in the missions presented in this section.

Figure 5.21 shows a 3D view of the setup in an obstacle free room (the room has dimensions 6.70 by 4.75 meters). The two mobile manipulators are named MM_0 and MM_1. The grasping of the bar is assumed to be made using compliant devices. The grasping point of MM_0 is assumed to be fixed relative to bar, whereas the grasping point of MM_1 is allowed to slide along the bar. This determines the loosely coupled characteristic of the system.

In some of the experiments the environment has no obstacles. However, since one of the grasping points is allowed to slide along the bar, each mobile platform must have detection and collision avoidance capabilities. If both mobile manipulators grasp the bar near the extremities no collision between the mobile platforms will occur since the length of the bar (3 m) is greater than the sum of the maximum lengths spanned by each mobile manipulator (0.92 m). The experimental results suggest that an adequate task assignment avoids the generation of collisions between the manipulators. Thus, no obstacle detection sensors were used. Moreover, the selected gripper orientations minimize the potential collisions between the bar and the arms.

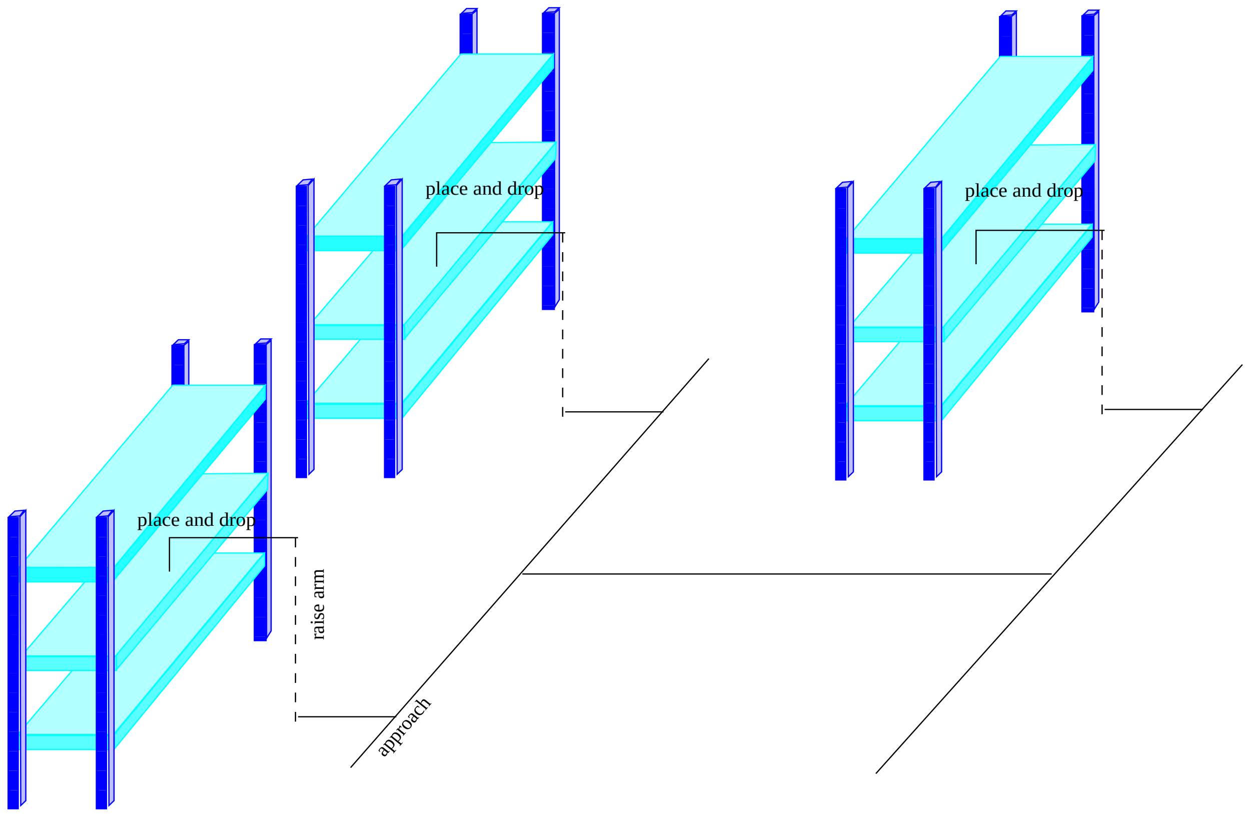


Figure 5.20: Possible task assignment in a conceptual warehouse.

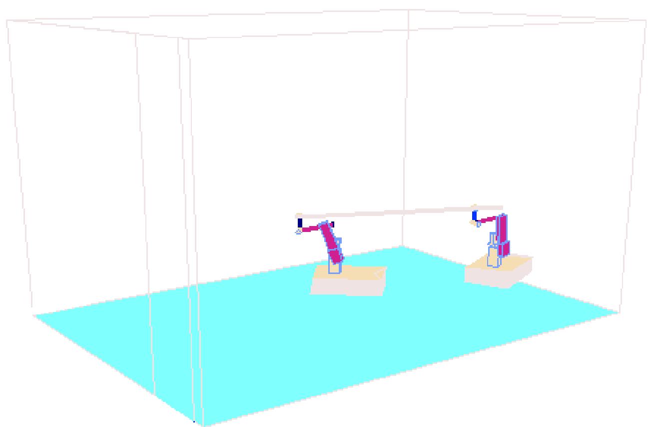


Figure 5.21: A 3D perspective of the simulation environment.

In the experiments using both real and simulated mobile manipulators, MM_0 denotes the real mobile manipulator. The strategy to merge the sensor data between the simulated MM_0 and the real MM_0 is the following: the real MM_0 uses the ultrasound data from the simulated MM_0 and the simulated MM_0 uses the odometer data from the real MM_0.

The exchange of information between MM_0 and MM_1 is limited to the grasping positions. Whenever the distance between the grasping points exceeds the limits of the interval $[1.4, 2.5]$ meters, each of the supervision blocks modifies the task assigned to its respective agent. Two strategies are considered in the thesis to handle this situation:

Passive - To compensate for a large distance value, MM_0 stops and waits for the supervisors in MM_1 to modify the tasks assigned to their respective agents. This strategy is uniquely used by MM_0 and only the upper limit of the interval is considered.

Active - Both mobile manipulators, through their supervisors, compensate the value of the distance between the grasping points. This is the unique strategy used by MM_1. In some experiments is also used by MM_0.

In the missions considered, the tasks assigned to the agents in the system belong to the first class considered in Section 4.4, i.e., they require the application of the null action at the task location. Both manipulators always start in the upward configuration. In the plots, solid lines represent the trajectories of the grippers and dashed lines represent the trajectories generated by the mobile platforms, whenever plotted jointly with the grippers data.

5.7.1 Robustness test 1 - Agent malfunctions

In this section the system is tested against a malfunction that prevents MM_1 to move. This experiment aims at assess the robustness of the chosen set of actions in a situation where MM_1 exhibits no form of intentional cooperative behaviour. The *actions* architecture (Figure 2.15) and the simulated setup are used. The mission assigns a regular task to MM_0 and keeps MM_1 in a fixed configuration.

The mission is described in Table 5.11, that also specifies the fixed location of MM_1. The gripper dominates in all the tasks.

	x	y	z	θ	α	β	γ
Start	3.93	2.07	0.92	0	0	0	0
Task	3	1	0.62	-	0	0	0

MM_0.

x	y	z	θ	α	β	γ	tolerance
6	2.9	0.92	0	0	0	0	-
-	-	-	-	-	-	-	-

MM_1.

Table 5.11: Task assignment for the robustness experiment 1.

Figure 5.22 presents, separately, the results for two runs of the mission. In both of them MM_0 uses the active task changing strategy. The dashed horizontal line in the lowermost plots indicates the bar length.

In the first run the manipulator starts the mission in the upward configuration. As this configuration lies in the double singularity region, the supervisor chooses **GetOutDS** as the first action to be executed. The resulting motion is easily distinguishable as the isolated part of the trajectories of the gripper. The mobile platform starts its motion using the **GoStG** action which, along with the initial motion of the arm, forces the increase of the distance between the bar grasping points. When this distance exceeds the specified limit, MM_0's supervisors change the assigned tasks, avoiding the bar to fall down. The supervisors change again the assigned tasks back to the original ones, at the point where the distance lies in the specified interval. The corresponding motion can be perceived by the forward and backward motion generated by the mobile platform. Meanwhile, the arm actions oscillate between **GoStG** and **GetOutT2S**, generating trajectories in the blurred region shown in the plots.

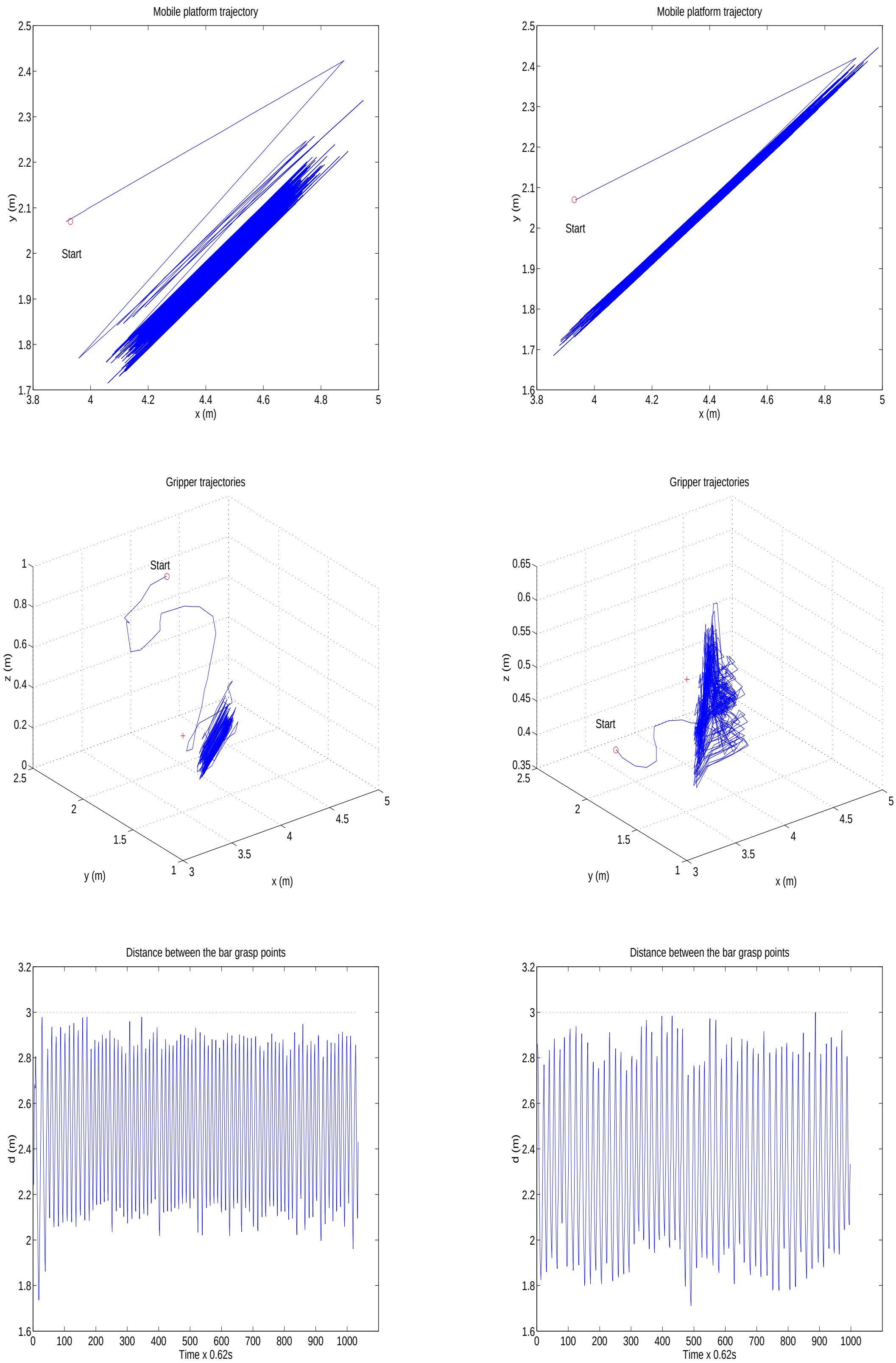
In the second run, the arm starts in a configuration outside any singularity region. Immediately after starting, its supervisor chooses the **GoStG** action, generating the initial portion of the trajectory away from the blurred region. Similarly to the first run, the changes in the assigned task forced by the supervisor induce an oscillation between the **GOsTG** and the **GetOutT2S** actions. As in the first run, the motion of the mobile platform is due to the **GoStG** action, with the changes between forward and backward motions caused by the switching between the tasks imposed by the task active strategy.

The quasi-periodic variation of the distance between the bar grasping points indicates that a stationary condition in the system operation has been reached. Some peaks of the graph are extremely close to the 3 m limit. However, the duration of approximately 620 seconds of each of these experiments, in which the bar did not fall down, indicates the robustness of the proposed sets of actions.

As there is no formal proof on the adequacy, to this particular mission, of the used set of actions, it would not be wise, in a real application, to let the MM_1 grasping point approach the tip of the bar. Keeping the actual sets of actions, a solution to this problem is to decrease the upper limit of the $[1.4, 2.5]$ meters interval, thus increasing the initial 0.5 m safety margin.

5.7.2 Robustness test 2 - *Actions* vs. *behaviours* architecture

This section presents two sets of experiments on a non-cooperative task assignment. The term non-cooperative means that the distance between some of the tasks is such that rends difficult the execution of the mission. Table 5.12 specifies the mission, with the gripper dominating in all the



First run of the mission. Second run of the mission.

Figure 5.22: MM_0's trajectories (MM_1 is kept at a fixed configuration).

tasks. Table 5.13 shows the distances between each of the task locations.

	x	y	z	θ	α	β	γ	tolerance
Start	3.9	2.07	0.92	0	0	0	0	-
Task 1	3	2	0.4	-	0	-0.5	0	0.15
Task 2	2.5	1	0.5	-	0	-0.5	0	0.15
Task 3	2	1	0.4	-	0	-0.5	0	0.05
Task 4	1	2	0.4	-	0	-0.5	0	0.05

MM_0.

	x	y	z	θ	α	β	γ	tolerance
Start	6	2.9	0.92	$\pi/2$	0	0	0	-
Task 1	1	3	0.5	-	0	-0.5	0	0.15
Task 2	1	3.5	0.3	-	0	-0.5	0	0.15
Task 3	-1	2	0.3	-	0	-0.5	0	0.05

MM_1.

Table 5.12: Mission with a non-cooperative task assignment.

		MM_1 tasks			
		Start	1	2	3
	Start	2.25	3.07	3.29	4.93
MM_0 tasks	1	3.17	2.24	2.50	4.00
	2	4.00	2.5	2.92	3.65
	3	4.45	2.23	2.69	3.16
	4	5.11	1	1.5	2

Table 5.13: Distances (in meters) between the tasks assigned to both mobile manipulators.

The first set of experiments, containing two runs of the mission, was performed using the *actions* architecture. The results are superimposed in Figure 5.23. The xy plot represents the trajectories of the grippers using solid lines and those of the mobile platforms using dashed lines. MM_0 executes its entire mission in both runs. While MM_0 is executing Tasks 1 and 2, the distance between the grasping points exceeds the 2.5 m upper limit and leads to intense maneuvering by MM_1. When MM_0 approaches Task 3 the distance between tasks tends to 2.23 m (see Table 5.13) thus simplifying the execution of MM_1's Task 1. Once MM_0 starts moving towards Task 4, the distance between tasks tends to 1.5 m, again forcing MM_1 (now trying to reach Task 2) to an intense maneuvering that rends difficult the execution of any subsequent task.

The second set of experiments is composed by five runs, all using the $\ominus$ blocks. The superimposed trajectories for the five runs are presented in Figure 5.24. During the first and second runs the global behaviour is similar to the one in the first set, MM_0 executes its entire mission whereas MM_1 executes only Task 1. In the third run, MM_0 executes the entire mission while MM_1 explores an alternative path without any results as it was not able to execute none of its tasks. However, the adjustments made during the third run in the a^p actions are enough to have MM_1 executing its two first tasks in the two subsequent runs. Even though the distance between MM_0's Task 4 and MM_1's Task 3 is smaller than the length of the bar, the natural trajectory between MM_1's Tasks 2 and 3 forces the distance between the bar grasping points to values bellow the 1.4 m limit. The subsequent constant switching between tasks, caused by the low value of the distance between grasping points, limits the progress of MM_1.

Figure 5.25 shows the evolution of the B^p regions along the sequence of the first four runs (the graphical representation is the one explained in Section 4.8). The locations of the tasks are represented by the symbols $\circ$. The trajectories made by the mobile platforms in the four runs of the

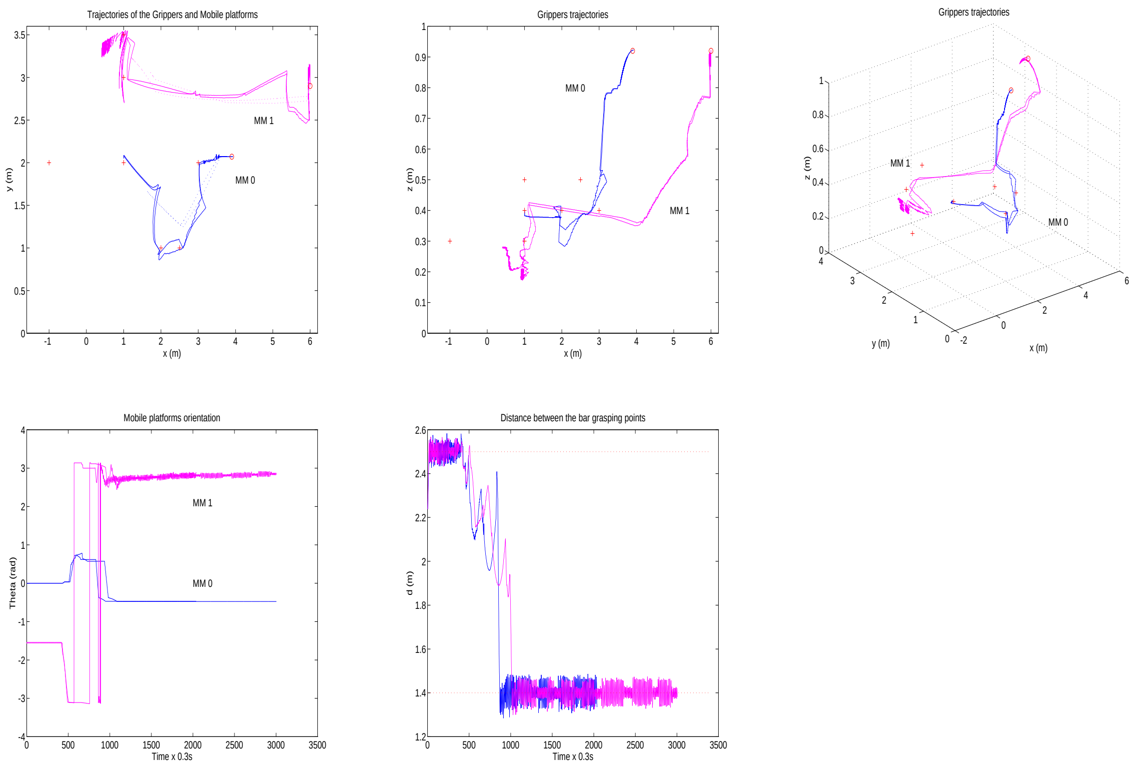


Figure 5.23: Two runs without the ⊜ block (*Actions* architecture).

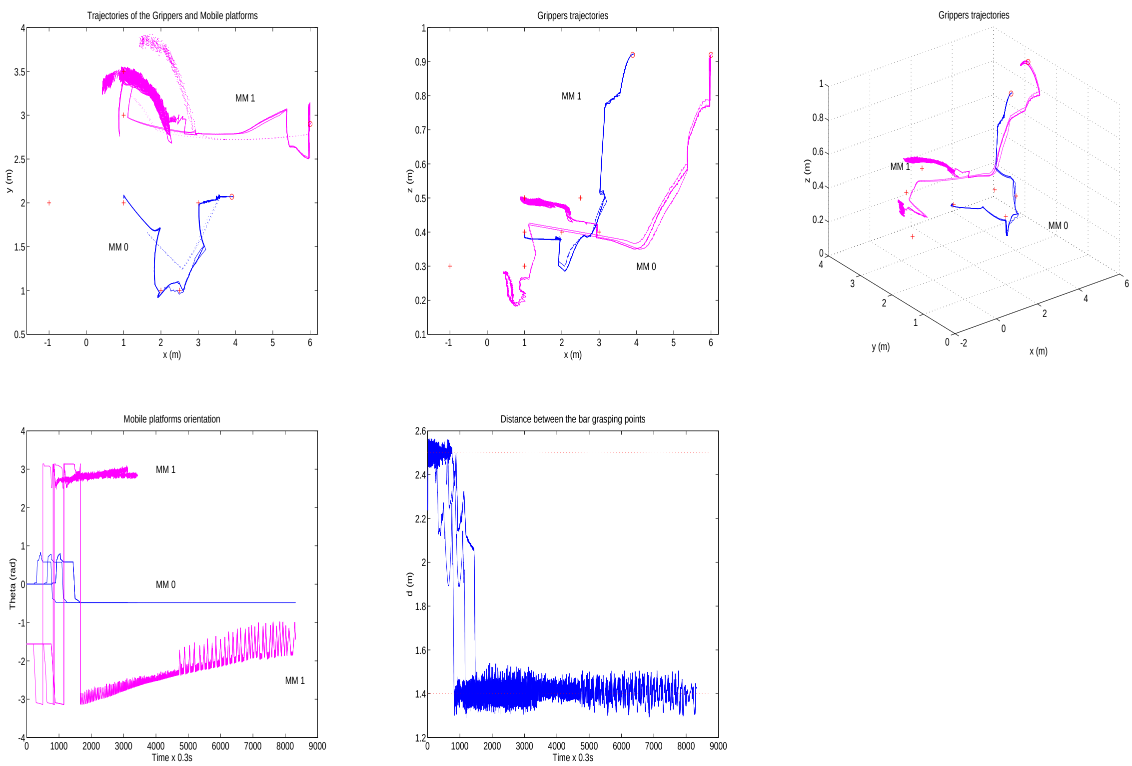


Figure 5.24: Five runs with the ⊜ block (*Behaviours* architecture).

mission are represented as dotted lines.

In the second run the action proposed at region B^8 tends to drive MM_0 away from the Task 4 currently assigned (note the small line segment pointing in the direction opposite to the location of Task 4). This action results from the convergence, during the period when MM_0 was stopped (outside B^8) waiting for the arm to complete a task, of the motion direction defined by a^8 towards a direction that avoids the obstacle avoidance interaction with MM_1. Similar situations arise also at B^8 and B^9 during the third and fourth runs. In what concerns MM_1, the actions proposed at B^{16} to B^{18} and B^{24} show the influence of the aforementioned obstacle avoidance situation that lead to the alternative path displayed in the third and fourth diagrams.

The algorithm of Table 3.3 imposes that at each configuration in B^p a single action, a^p, is executed. During the periods when the tasks assigned to the mobile manipulators are defined either by the active or the passive strategies the adjustments by the algorithm of Table 3.4 are inhibited. The inhibition prevents the frequent changes of motion direction that disturb the convergence of the a^p to a constant motion direction.

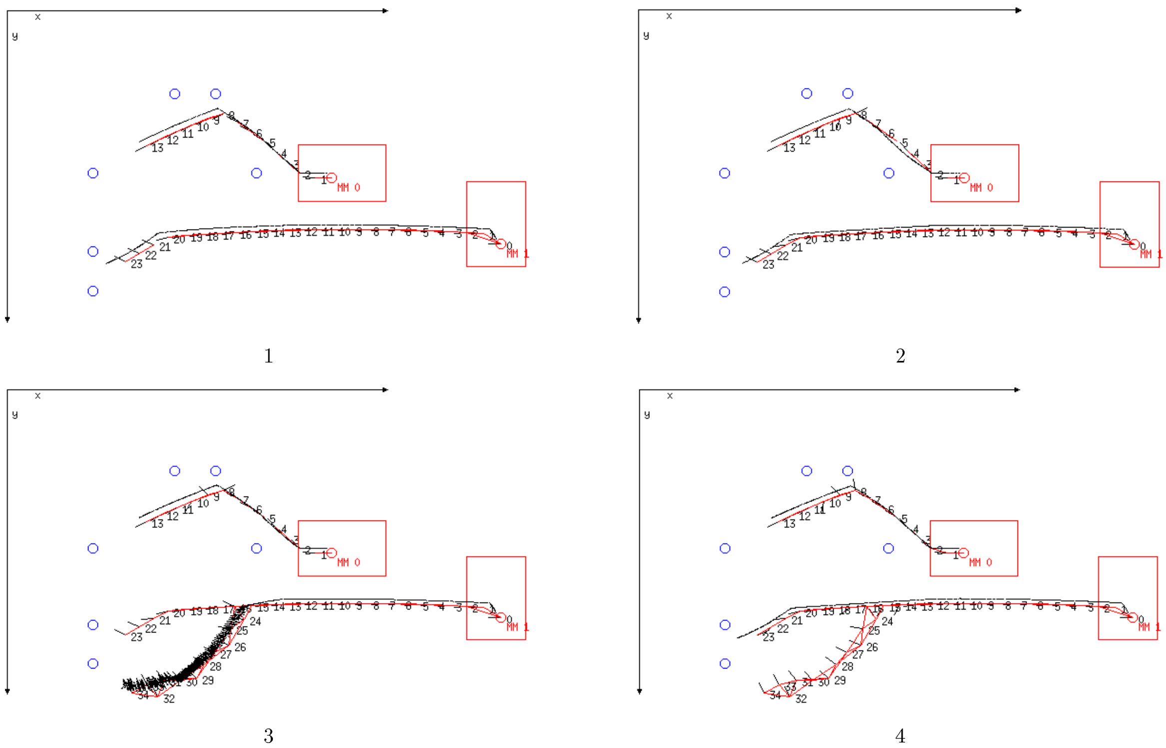


Figure 5.25: Typical *behaviours* of the mobile platforms after 4 runs.

The results obtained with each of the architectures are comparable as the trajectories obtained are qualitatively similar. However, the *actions* architecture does not have the possibility of generating alternative *behaviours* that, as shown by the experiments in this sub-section, may prove useful. This search for alternative solutions while preserving the robustness of the whole system is thus a useful characteristic of the *behaviours* architecture.

5.7.3 Using the active task changing strategy in MM_0

This experiment aims at assess the influence of the use of the active task changing strategy, defined in the introductory discussion of Section 5.7, in the performance of the real setup. The corresponding mission, defined in Table 5.14, has the tasks locations specified to force an initial maneuvering from both mobile platforms that does not generate significant obstacle avoidance interactions. Thus, the resulting trajectories display mainly the contribution of the active task changing strategy.

	x	y	z	θ	α	β	γ	tol.
Start	3.93	2.07	0.92	0	0	0	0	-
Task 1	2.5	1	0.4	-	0	0	0	0.05
Task 2	2.5	1	0.5	-	0	0	0	0.05
Task 3	2.5	0.9	0.5	-	0	$-\pi/2$	0	0.05

MM_0.

	x	y	z	θ	α	β	γ	tol.
Start	6	2.9	0.92	$\pi/2$	0	0	0	-
Task 1	4.5	1	0.4	-	0	0	0	0.05
Task 2	4.5	1	0.5	-	0	0	0	0.05
Task 3	4.5	0.9	0.5	-	0	0	0	0.05

MM_1.

Table 5.14: Task assignment for the experiment on the Active task changing strategy.

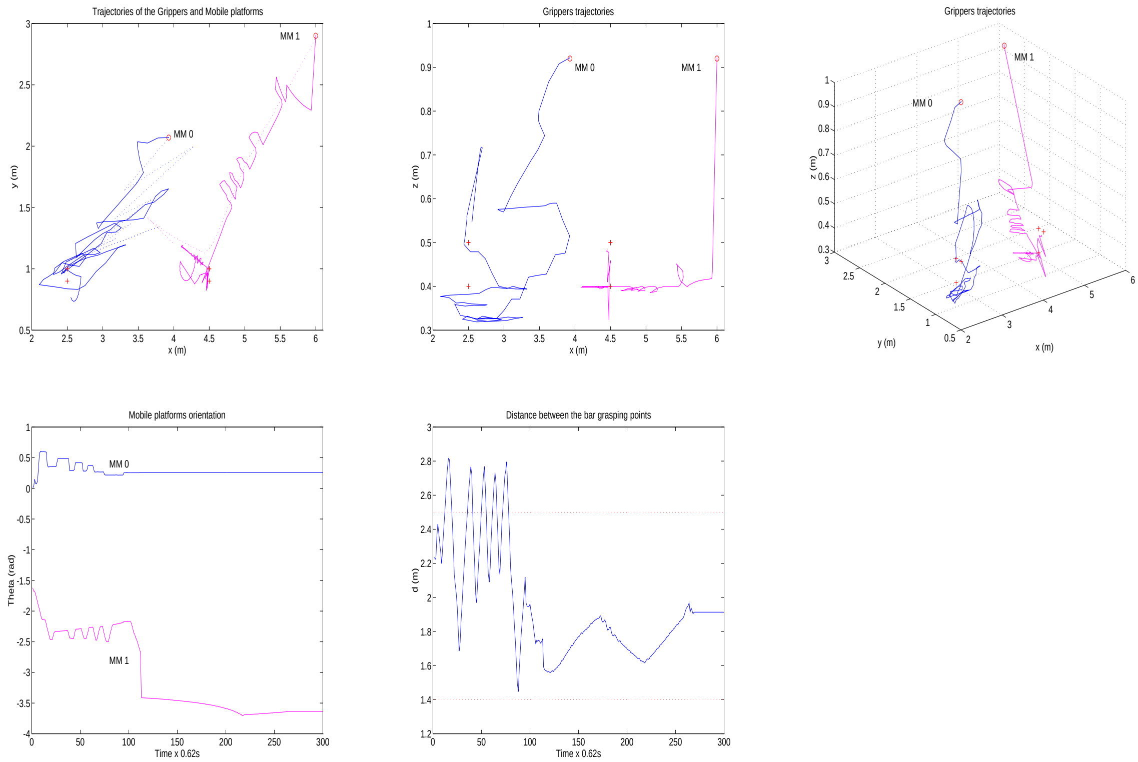
Two runs of the mission are presented in Figure 5.26, respectively taken with the mobile platform "on blocks" and operating normally on the floor. The trajectories exhibit a higher number of maneuvers than those obtained in any of the previous experiments. Moreover, this characteristic depends mostly on the task changing strategy and not on the dynamics of the mobile manipulators as the maneuvering exists in both runs. The behaviour exhibited by MM_0 does not seem compatible with a practical application and thus, hereafter the active task changing strategy will be used uniquely by MM_1.

5.7.4 Warehousing

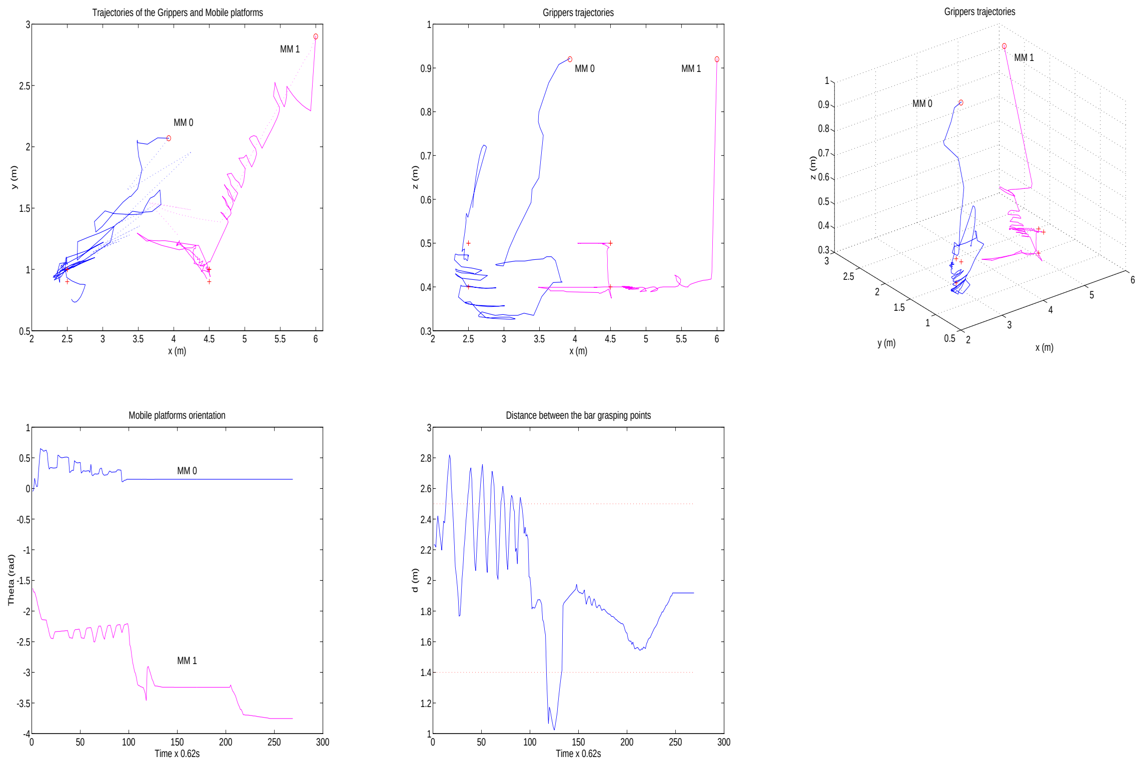
This section presents a number of missions involving the transportation of a bar inside an automated warehouse and its deposition onto a shelf. Figure 5.20 shows the conceptual phases that are used to define the tasks considered in the experiments. These experiments aim at assess the performance of the system under multiple task assignments, with slight variations between them. As the results show, slight differences between the missions, i.e., small differences between the locations of the tasks, may generate complex interactions between the two mobile manipulators.

Warehouse experiment 1

The simulated setup is used in the mission specified in Table 5.15. The tasks are defined aiming at maintaining the two mobile manipulators operating in areas such that the distance between the bar grasping points is always inside the interval $[1.4, 2.5]$ meters, thus not requiring any task changing during the execution of the mission. Task 1 represents the approach phase, Task 2 the raise of the bar to the shelf level, and in Task 3 the bar is deposited on the shelf.



Real mobile platform over blocks.



Real mobile platform on the floor.

Figure 5.26: Assessing the influence of the active task changing strategy.

	x	y	z	θ	α	β	γ	tol.
Origin	3.9	2.07	0.92	0	0	0	0	-
Task 1	2.5	1	0.4	-	0	-0.7	0	0.05
Task 2	2.5	1	0.5	-	0	-1	0	0.05
Task 3	2.5	0.9	0.4	-	0	-1.57	0	0.05

MM_0.

	x	y	z	θ	α	β	γ	tol.
Origin	6	2.9	0.92	$\pi/2$	0	0	0	-
Task 1	4.5	1	0.4	-	0	-0.7	0	0.05
Task 2	4.5	1	0.5	-	0	-1	0	0.05
Task 3	4.5	0.9	0.4	-	0	-1.57	0	0.05

MM_1.

Table 5.15: Task assignment for the Warehouse experiment 1.

Figure 5.27 shows the obtained trajectories. The leftmost plot shows the trajectories in the xy plane of the two grippers (solid lines) and the two mobile platforms (dashed lines). The middle and the rightmost plots use solid lines to represent the trajectories of the grippers of both MM_0 and MM_1.

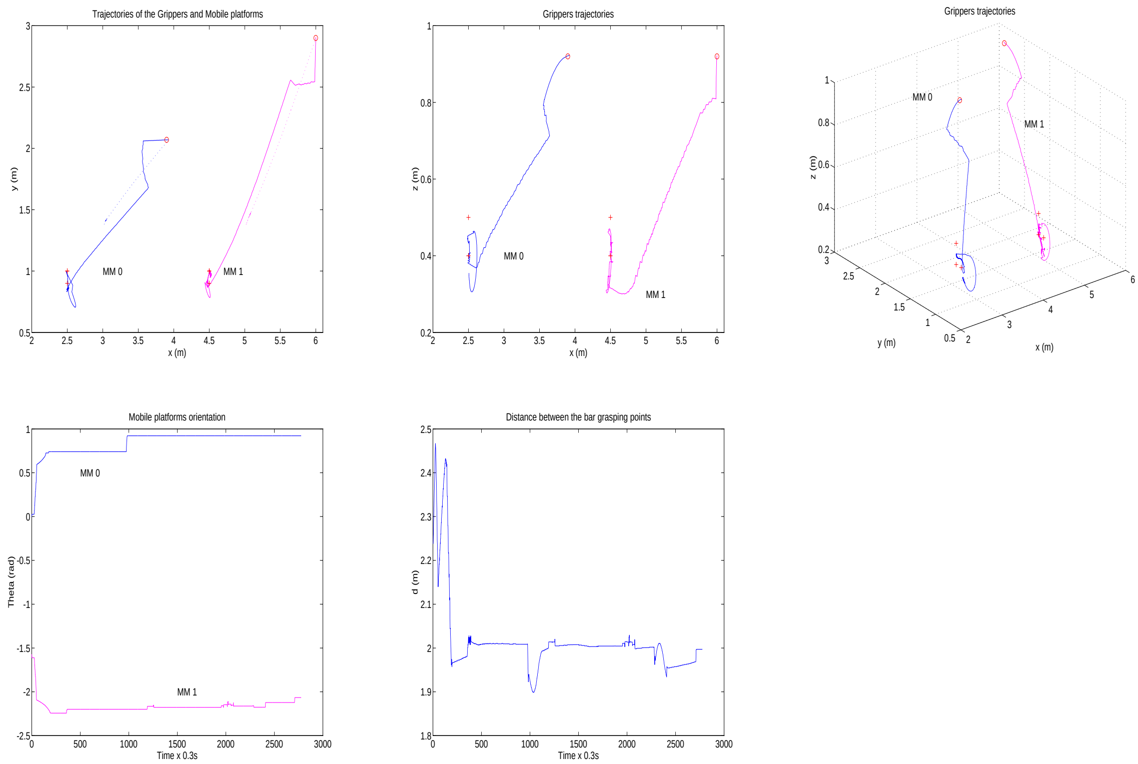


Figure 5.27: Warehouse experiment 1 (Simulated setup).

As a result of the established task assignment, no obstacle avoidance interactions between the two mobile platforms are observed. The consequence is that the trajectories made by the mobile platforms are almost parallel and without any complex maneuvering. In the neighbourhood of the task locations, the maneuvering of the gripper is due to the overstretching of the arm, which causes an oscillation between the **GetOutT1S** and the **GoStG** actions. Note that, as seen in the diagram representing the distance between the bar grasping point, the task changes are inexistent as the graphic lies always in the interval $[1.4, 2.5]$ meters.

The results of this mission suggest the importance of the task assignment in minimizing the interactions between the mobile platforms. This prevents the maneuvering that is to be expected should environment constraints arise. In a real application it is plausible that an external supervisor simplifies the mission design by specifying easy tasks, yielding low probable interactions between the mobile platforms.

Warehouse experiment 2

The mission of this experiment, specified in Table 5.16, differs from mission in Experiment 1 in the value of the z component of MM_0's Task 3 location. MM_1's mission remains unchanged.

Figure 5.28 shows the experimental results for the simulated setup. As a consequence of the change in the last task, at the final phase of the mission, the arm enters the S_1 singularity region leading to some maneuvering by the arm. At this phase, the mobile platform **GoStG** action generates small length trajectories (as it is near the task completion), to compensate the adjustments of the manipulator, that are reflected in the small variations in the orientation. The narrow blurred trajectories in the region between Tasks 2 and 3 illustrate also this phenomenon that was not present in Figure 5.27.

	x	y	z	θ	α	β	γ	tol.
Origin	3.9	2.07	0.92	0	0	0	0	-
Task 1	2.5	1	0.4	-	0	-0.7	0	0.05
Task 2	2.5	1	0.5	-	0	-1	0	0.05
Task 3	2.5	0.9	0.5	-	0	-1.57	0	0.05

MM_0.

x	y	z	θ	α	β	γ	tol.
6	2.9	0.92	$\pi/2$	0	0	0	-
4.5	1	0.4	-	0	-0.7	0	0.05
4.5	1	0.5	-	0	-1	0	0.05
4.5	0.9	0.5	-	0	-1.57	0	0.05

MM_1.

Table 5.16: Task assignment for the Warehouse experiment 2.

Figure 5.29 shows the results obtained with the real setup. The initial part of the trajectories are qualitatively similar to those obtained during Experiment 1. The delay in the feedback loop, along with the changes in the assigned tasks caused by the distance between the bar grasping points going outside the interval $[1.4, 2.5]$ meters, show their influence in the initial maneuvers. Note that the bar almost felt down (the maximal distance between the grasping points reached 2.995 m) precisely at this initial part of the trajectory. After this transient phase the distance between the grasping points is approximately constant. Also in the initial region, the mobile platforms show large variations in the orientation which are a consequence of the maneuvering. A gripper singularity by MM_0, together with some obstacle avoidance interaction between the two mobile platforms, generated the maneuvering displayed at the final phase of the mission. The motion resulting from the obstacle avoidance interaction helped to clear the gripper singularity. The long time of the mission execution is due to MM_1's gripper that moves slowly towards the task location. Both mobile platforms were already stopped a long time before the mission completion, as can be perceived from the plot representing their orientations.

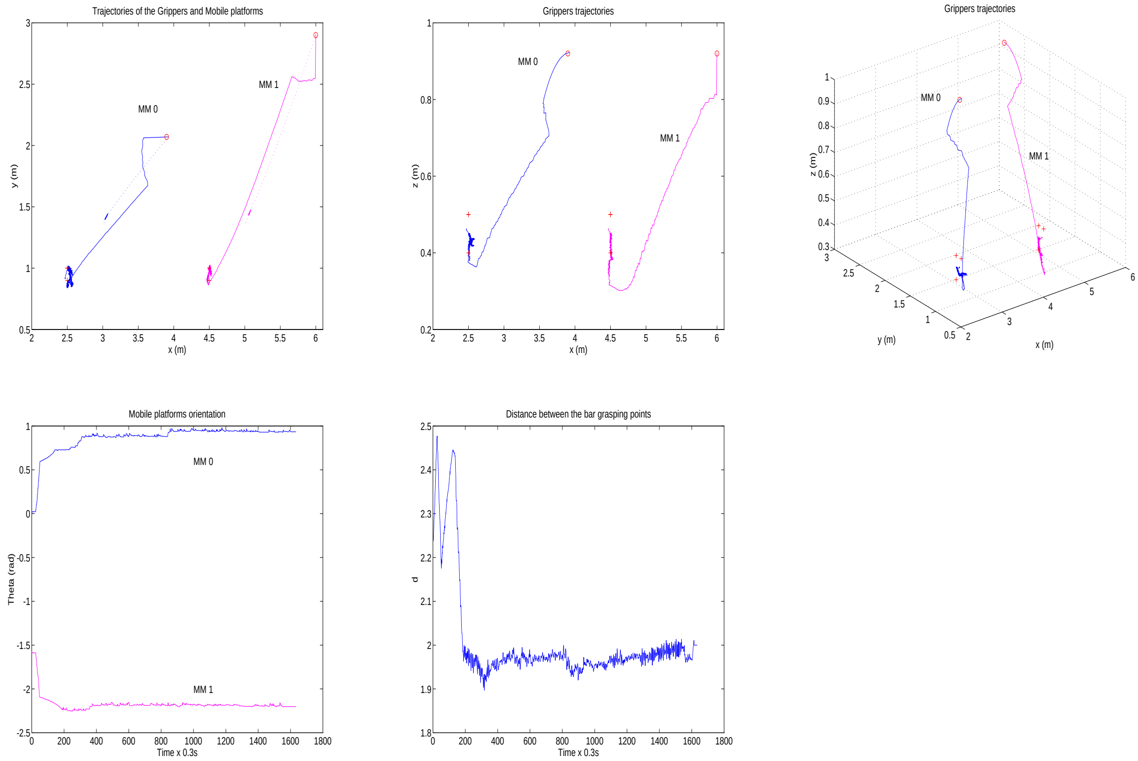


Figure 5.28: Warehouse experiment 2 (Simulated setup).

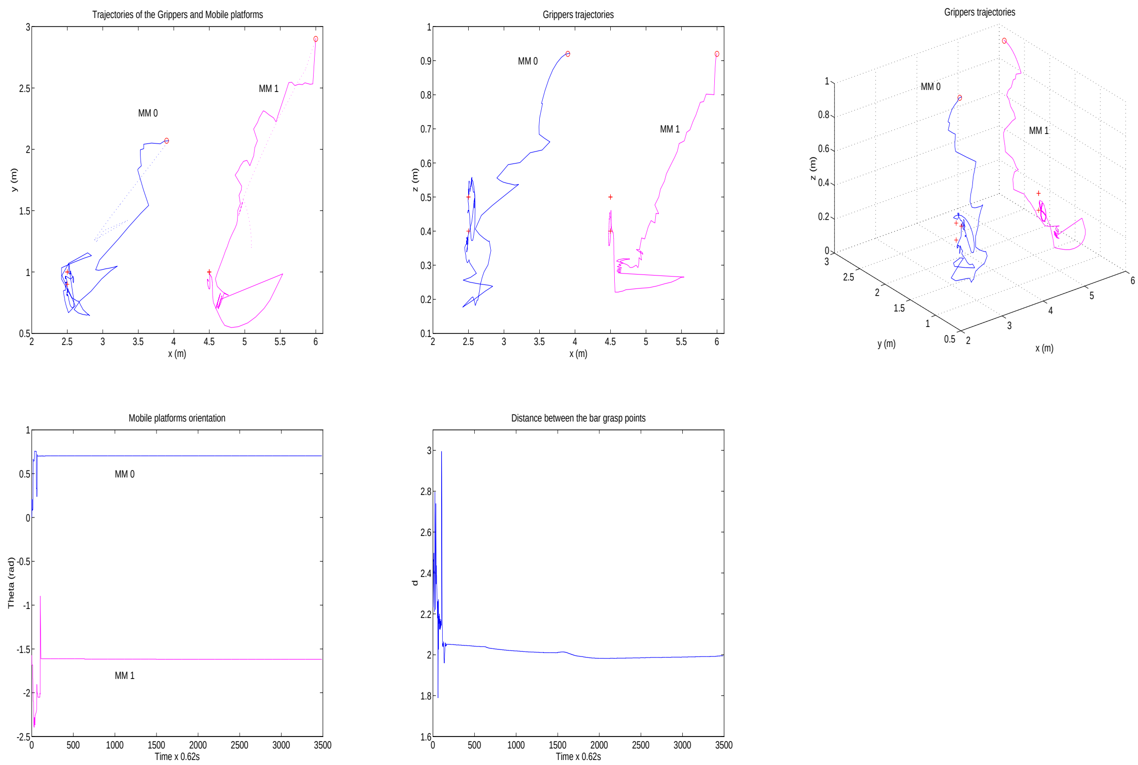


Figure 5.29: Warehouse experiment 2 (Real setup).

Warehouse experiment 3

The mission specified by Table 5.17, slightly differs from the mission in Experiment 2. The locations of the tasks assigned to MM_1 (the simulated mobile manipulator) are unchanged. As for MM_0, the z coordinate value of all the tasks was increased by 0.1 m. Also, the y coordinate value of the MM_0's Task 3 was decreased by 0.1 m to avoid the accidental task execution by the erratic motion in the neighbourhood of Task 2, caused by the delay in the architecture feedback loops. In addition, the tolerances in MM_0's mission were increased to simplify the task execution.

	x	y	z	θ	α	β	γ	tol.
Start	3.93	2.07	0.92	0	0	0	0	-
Task 1	2.5	1	0.5	-	0	0	0	0.06
Task 2	2.5	1	0.6	-	0	0	0	0.05
Task 3	2.5	0.8	0.6	-	0	$-\pi/2$	0	0.08

MM_0.

x	y	z	θ	α	β	γ	tol.
5	2.9	0.92	$\pi/2$	0	0	0	-
4.5	1	0.4	-	0	0	0	0.05
4.5	1	0.5	-	0	0	0	0.05
4.5	0.9	0.5	-	0	0	0	0.05

MM_1.

Table 5.17: Task assignment for the Warehouse experiment 3.

The trajectories obtained, shown in Figure 5.30, clearly exhibit a smaller number of maneuvers than those obtained in Experiment 2 (Figures 5.28 and 5.29). These results emphasize the importance of a good task assignment in missions with similar goals as it is the case with these warehouse experiments.

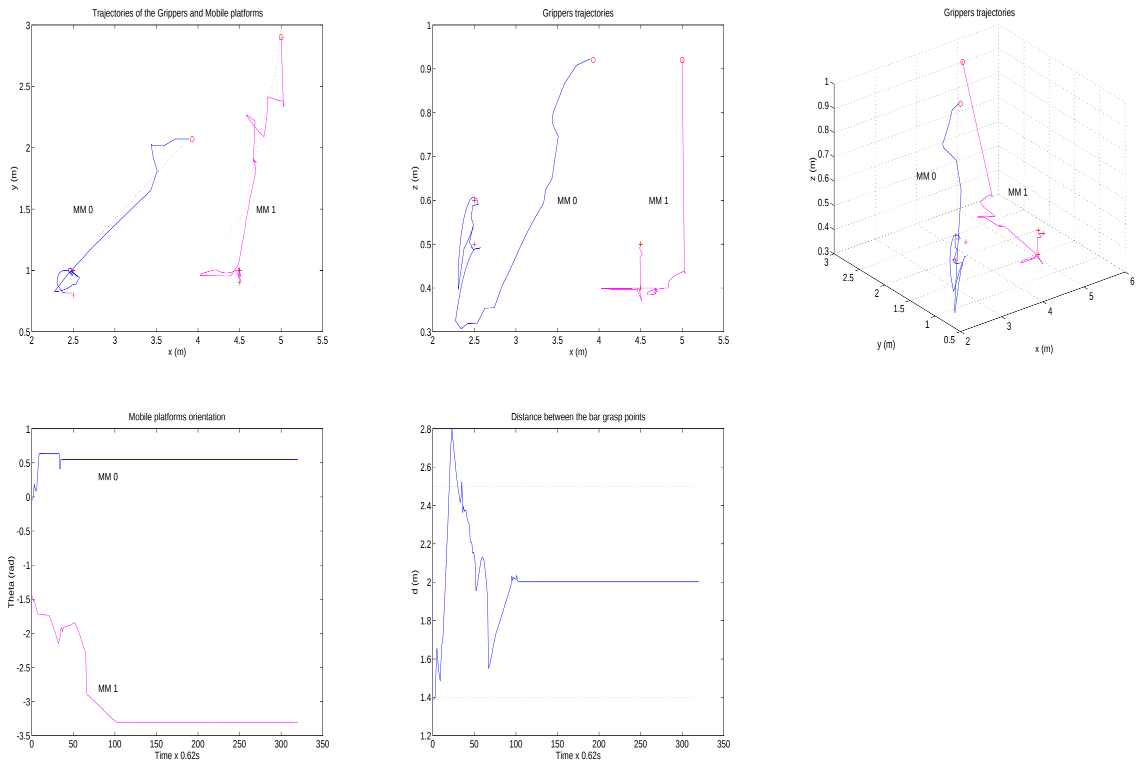


Figure 5.30: Warehouse experiment 3 (Real setup).

Warehouse experiment 4

This experiment further illustrates how the difference in the dynamics of the two setups affects the global performance. The mission, specified in Table 5.18, is defined to have the mobile manipulators carrying the bar towards the shelf without experiencing any obstacle avoidance interactions between the vehicles. Regardless the different dynamical characteristics of the two mobile manipulators, the *actions* architecture is used in this mission, as no difficult interactions between the two mobile manipulators are expected. Figure 5.31 presents the results for the simulated setup (uppermost plots) and for the real setup (lowermost plots).

	x	y	z	θ	α	β	γ	tolerance
Start	2.5	2.5	0.92	0	-	-	-	-
Task 1	2.5	1	0.4	-	0	0	-0.7	0.05
Task 2	2.5	1	0.5	-	0	0	-1	0.05
Task 3	2.5	0.9	0.5	-	0	0	-1	0.05

MM_0.

	x	y	z	θ	α	β	γ	tolerance
Start	4.5	2.5	0.92	0	-	-	-	-
Task 1	4.5	1	0.4	-	0	0	-0.7	0.05
Task 2	4.5	1	0.5	-	0	0	-1	0.05
Task 3	4.5	0.9	0.5	-	0	0	-1	0.05

MM_1.

Table 5.18: Task assignment for the Warehouse experiment 4.

The difference in the velocities with which the agents composing each mobile manipulator execute their tasks causes the arm to enter a workspace region close to a mechanical singularity. This leads to the maneuvering observed at the middle of the xy trajectories in both setups. Regardless the difference in the dynamical characteristics between the two setups considered in this experiment, the observed trajectories exhibit qualitatively similar characteristics, thus supporting the claim of Section 5.1 on the usefulness of the simulation setup.

Warehouse experiment 5

This experiments aims at assess the performance of the system in a mission that forces the interaction between the mobile platforms. The mission, defined in Table 5.19, forces the "crossing" of the mobile platforms trajectories. The mission assigned to MM_1 has an intermediate task (Task 1) to simplify the resolution of the obstacle avoidance interactions between the mobile platforms. The simulation setup, with the *behaviours* architecture, is used.

	x	y	z	θ	α	β	γ	tolerance
Start	4.5	3.5	0.92	0	-	-	-	-
Task 1	2.5	1	-	-	-	-	-	0.2
Task 2	2.5	0.3	0.5	-	0	0	-1	0.05
Task 3	2.5	0	0.3	-	0	0	-1.4	0.08
Task 4	-	-	-	-	-	-	-	-

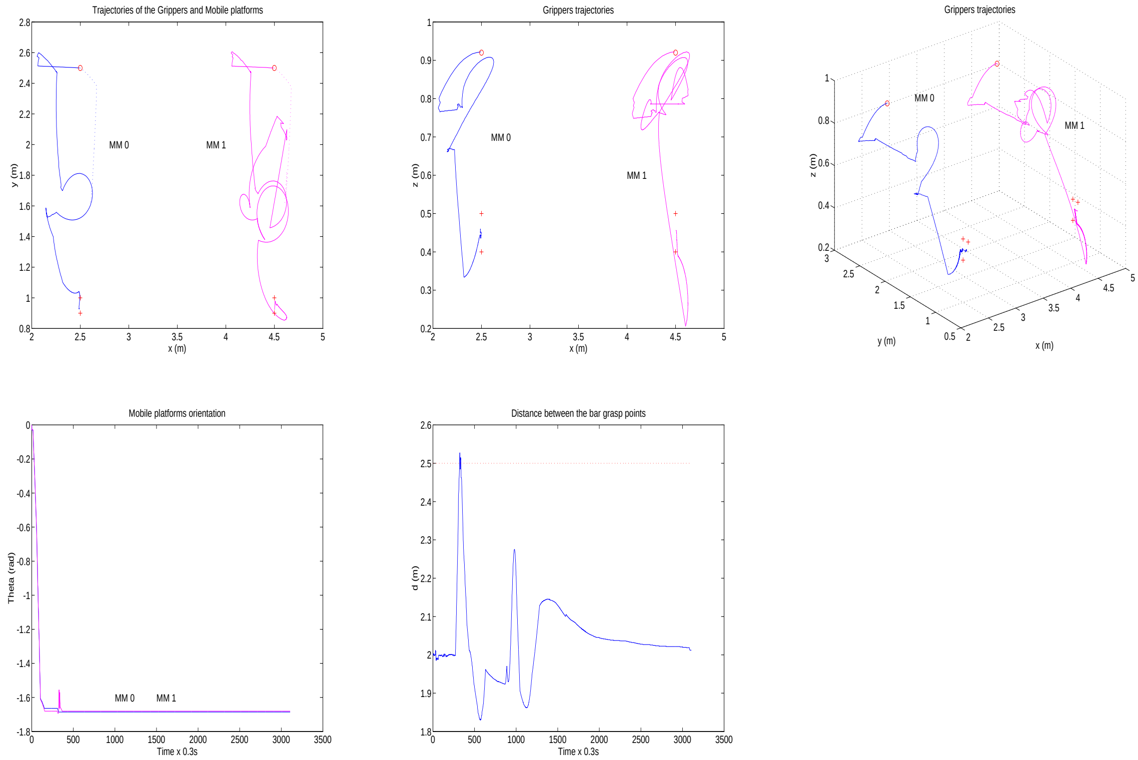
MM_0.

	x	y	z	θ	α	β	γ	tolerance
Start	2.5	3.5	0.92	0	-	-	-	-
Task 1	3.5	2.5	-	-	-	-	-	0.2
Task 2	4.5	1	-	-	-	-	-	0.06
Task 3	4.5	0.3	0.5	-	0	0	-1	0.05
Task 4	4.5	0	0.3	-	0	0	-1.4	0.08

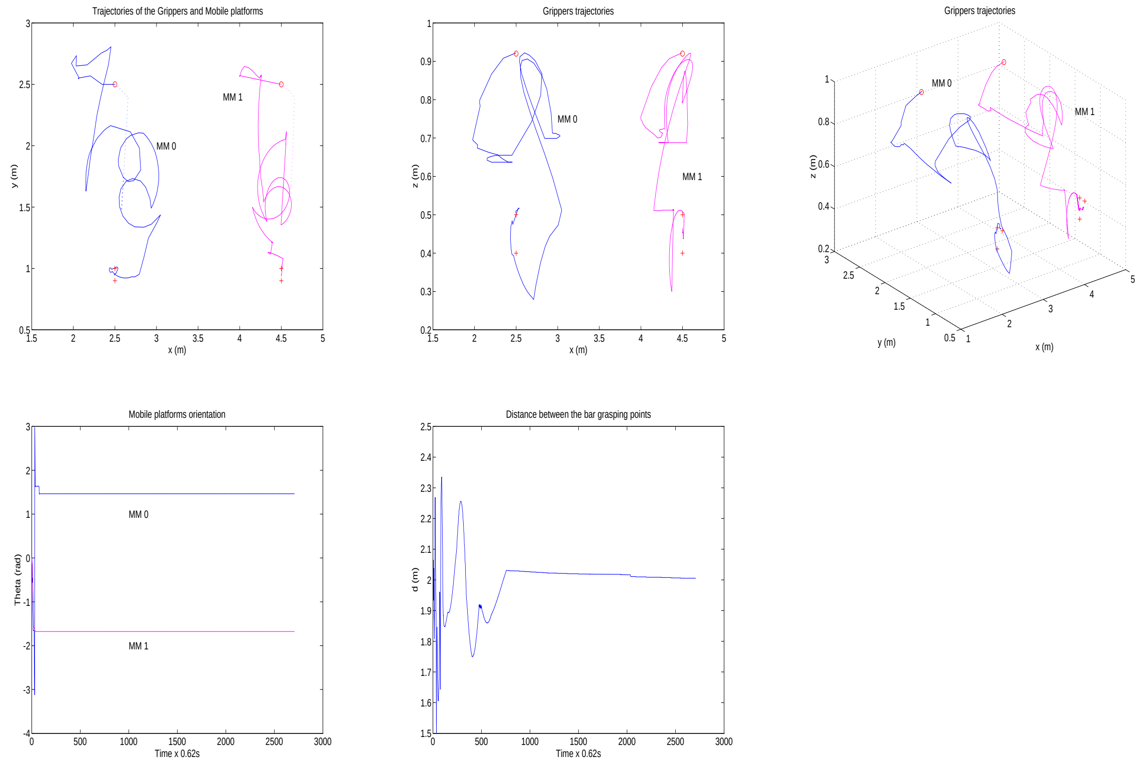
MM_1.

Table 5.19: Task assignment for the Warehouse experiment 5.

Figure 5.32 shows the results of 5 superimposed runs. The passive task changing strategy used by MM_0 induces the generation of trajectories containing less maneuvers than those generated by



MM_0 and MM_1 simulated.



MM_0 real and MM_1 simulated.

Figure 5.31: Warehouse experiment 4 (Real and simulated setups).

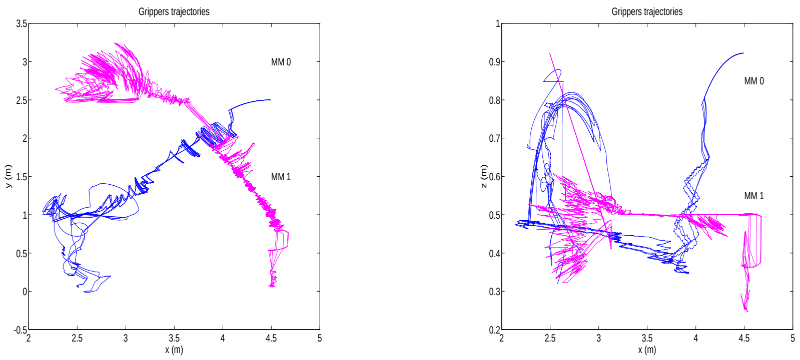


Figure 5.32: Warehouse experiment 5 (Simulation setup). Five superimposed runs

MM_1. Figure 5.33 shows a sequence of four views taken during the second run of the mission.

Typical *behaviours* for both mobile platforms, emerging after four runs of this mission, are shown in Figure 5.34. The small circles indicate the location of the tasks in the xy plane.

During the first trial run MM_0 creates B^1 to B^5 and MM_1 creates B^0 and B^1. In the next three runs the pattern shown in Figure 5.34 is created. The multiple interconnections between the B^i indicate the regions of collision avoidance interaction between the two mobile platforms. For MM_0 this region is composed by B^1 to B^6, whereas for MM_1 is composed by B^1 to B^5.

Warehouse experiment 6

The mission in this experiment, specified in Table 5.20, is a variant of the Warehouse experiment 5 with the intermediate task previously assigned to MM_1 removed. The resulting trajectories by the mobile platforms (the straight lines between their starting configurations and the first tasks) generate additional collision avoidance interactions. The real setup, with the *behaviours* architecture, is used. The first tasks assigned to the mobile manipulators are dominated by the mobile platforms. The grippers dominate the remaining tasks.

	x	y	z	θ	α	β	γ	tolerance
Start	4.5	2.5	0.92	0	-	-	-	-
Task 1	2.5	1	0.5	-	-	-	-	0.2
Task 2	2.5	0.3	0.5	-	0	0	-1	0.05
Task 3	2.5	0	0.3	-	0	0	-1.4	0.08

MM_0.

x	y	z	θ	α	β	γ	tolerance
2.5	2.5	0.92	0	-	-	-	-
4.5	1	0.5	-	0	0	0	0.06
4.5	0.3	0.5	-	0	0	-1	0.05
4.5	0	0.3	-	0	0	-1.4	0.08

MM_1.

Table 5.20: Task assignment for the Warehouse experiment 6.

Figure 5.35 shows the trajectories of both grippers in the xy and xz planes and in the xyz space.

The maneuvering in the region close to the starting configuration of MM_1 results from the collision avoidance interaction between the mobile platforms and also from the task changes necessary to keep

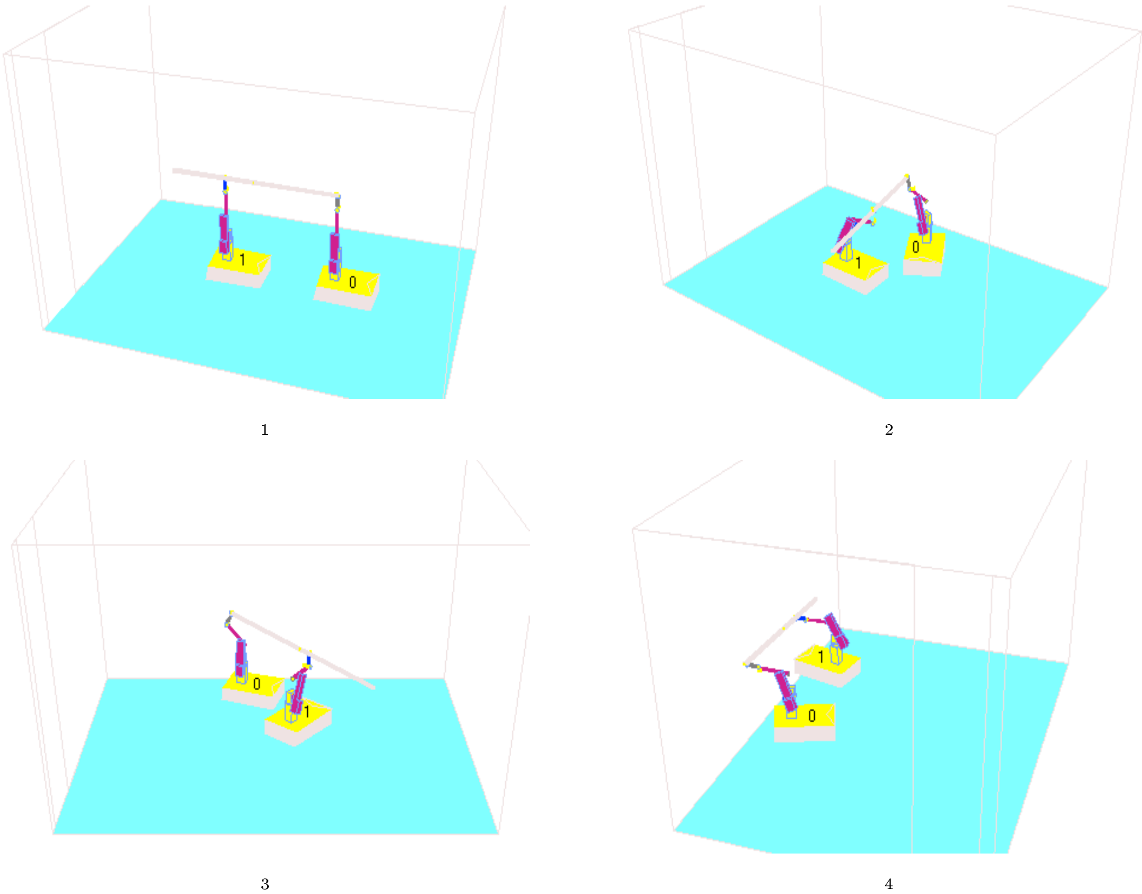
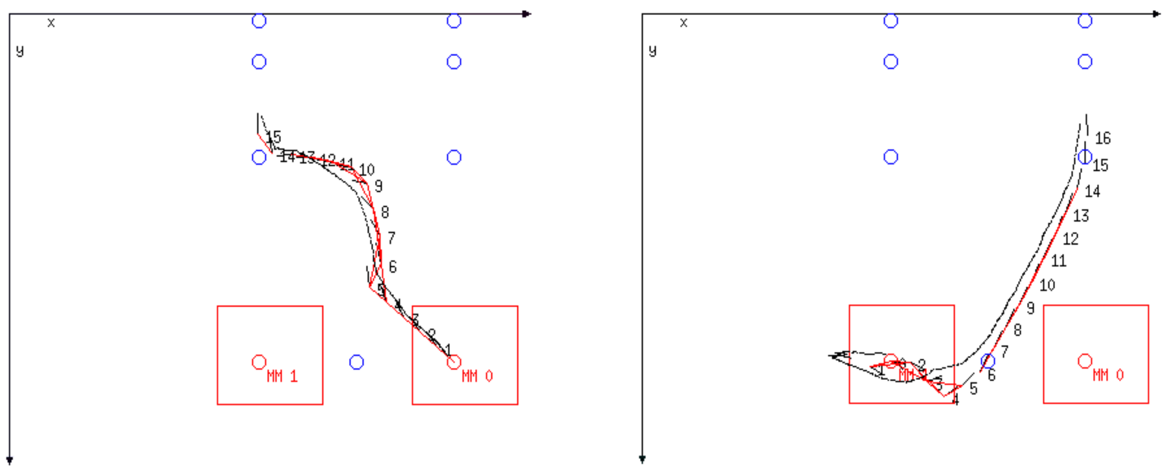


Figure 5.33: Four views of the Warehouse experiment 5.

the distance between the bar grasping points within the specified interval. The trajectory generated by MM_0 does not exhibit significant maneuvering at the beginning of the mission. The sudden changes in the direction of motion of MM_0, near the middle of the trajectory, are a consequence of a collision avoidance action taken by its mobile platform during a short time period. The lowering of the gripper to increase the distance between the bar grasping points is illustrated in the xz plots by the cross of workspace regions by low z coordinate value.

Figure 5.36 shows the typical mobile platform *behaviours* that emerged in a single run of the mission, with the two setups exhibiting different qualitative *behaviours*. The small circles indicate the task locations in the xy plane.

In the fully simulated setup (uppermost plots), both MM_0 and MM_1 generate trajectories with almost no visible influence of the collision avoidance action. The use by MM_1 of the active strategy hides the collision avoidance effects on the trajectory under the maneuvering to keep the bar in position, at the region close to the starting configuration. In the setup using a real MM_0 (lowermost plots) the collision avoidance effects are clearly visible in the trajectories of both mobile platforms. The diagrams for this setup result from two separate runs of the mission. In the first run, MM_1 created the units B^0 to B^5, leading to a *go away behaviour* that caused the failure of the mission



MM$_0$. MM$_1$.

Figure 5.34: Warehouse experiment 5. Typical *behaviours* of the mobile platforms. The second run is shown.

execution. During the first run, MM$_0$ created units B^0 to B^{12}, being able to reach the location of its first task. The adjustment of the a^p actions by the algorithm of Table 3.4 enables MM$_1$ to take the appropriate direction in the second run of the mission. During this second run MM$_0$ created the additional units B^{13} to B^{18} as a result of the collision avoidance with MM$_1$ at the approach of its first task.

Figure 5.37 shows a sequence of six views taken during one run of the mission

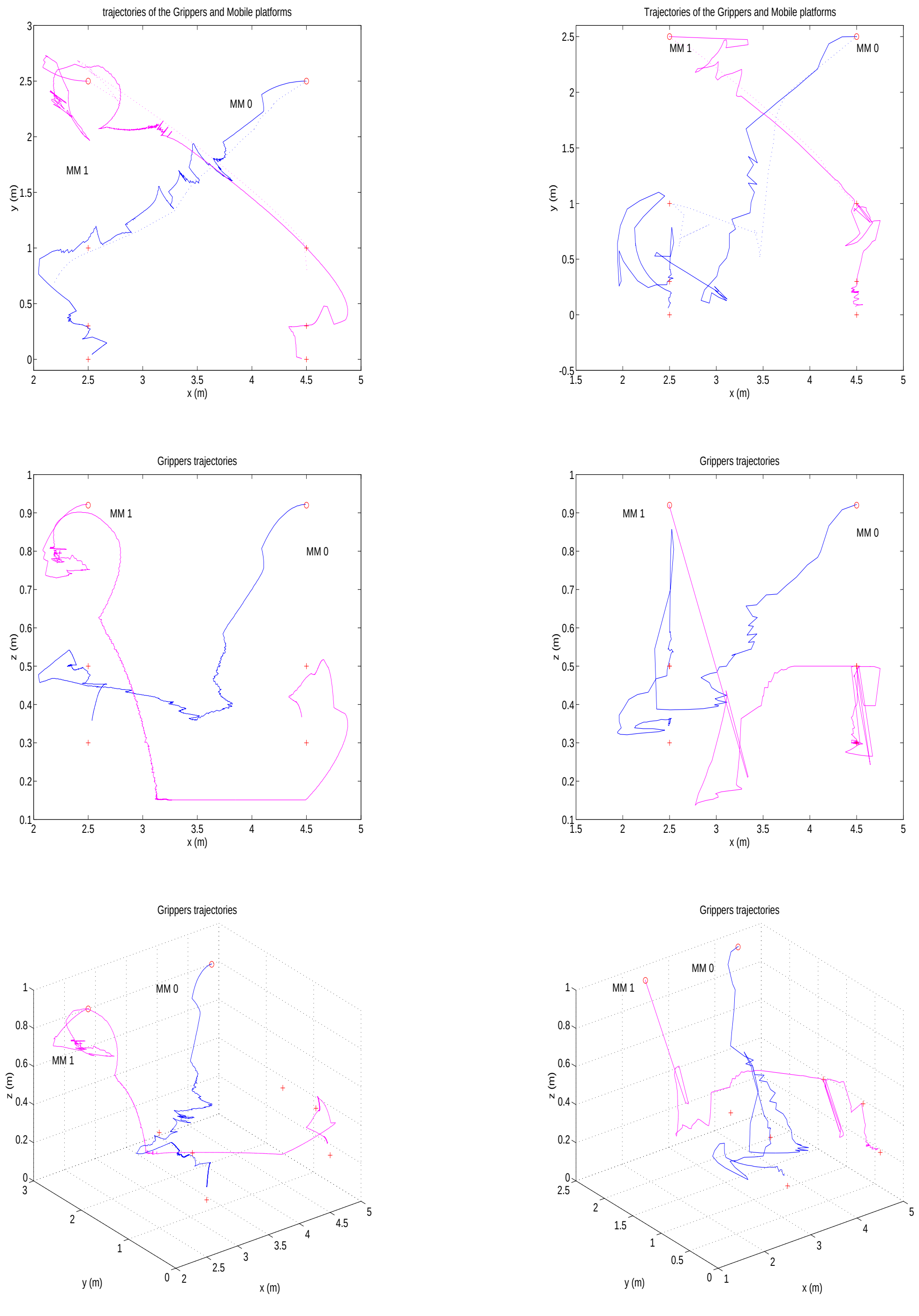
5.8 Carrying a bar through a corridor

This section merges the previous experiments on corridor traversal and bar transportation aiming at carrying a long bar through a narrow passage. Two experiments, using the simulation setup, are presented. The first experiment assesses the interactions between the mobile manipulators in an environment without obstacles. The mission corresponding to this experiment is specified in Table 5.21. In the second experiment a corridor 2.18 m long and 1.6 m wide is considered[7]. The mission corresponding to this experiment is specified in Table 5.22. The *actions* architecture is used in both missions.

In both missions four different phases are identified: approaching to the corridor, lining up the two mobile manipulators in front of the corridor, traversing it, and moving away. The gripper dominates in all the tasks.

Figure 5.38 shows the trajectories for the obstacle free mission. The xy plot shows the motion of the mobile platforms. The xyz plot shows the trajectories of both grippers and their projections

[7]The bar is 3 m long.



Simulated MM_0.   Real MM_0.

Figure 5.35: Warehouse experiment 6.

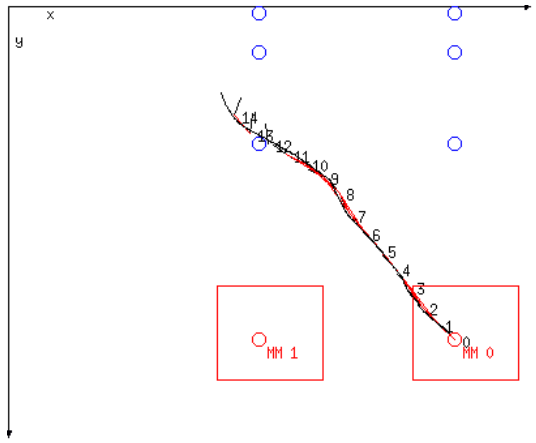
	x	y	z	θ	α	β	γ	tol.
Start	5	3.1	0,92	0	0	0	0	-
Approach	2	4.1	0.5	-	0	-0.7	0	0.05
Line up	0.5	4.1	0.5	-	0	-0.7	0	0.05
Traverse	-1	4.1	0.5	-	0	-0.7	0	0.05
Move away	-1	1.7	0.5	-	0	-0.7	-0.7	0.05

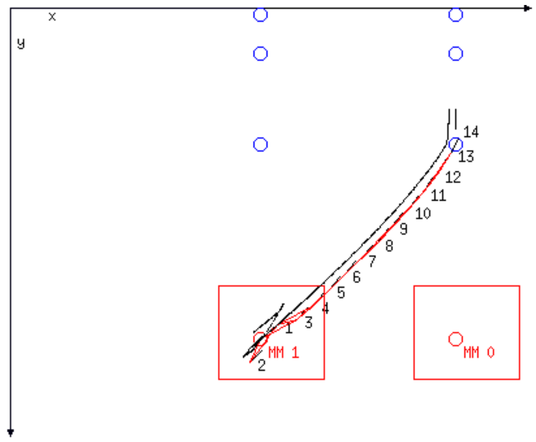
MM_0.

x	y	z	θ	α	β	γ	tol.
7	4.1	0,92	0	0	0	0	-
2.5	4.1	0.5	-	0	-0.7	0	0.05
1	4.1	0.5	-	0	-0.7	0	0.05
-1	4.1	0.5	-	0	-0.7	0	0.05

MM_1.

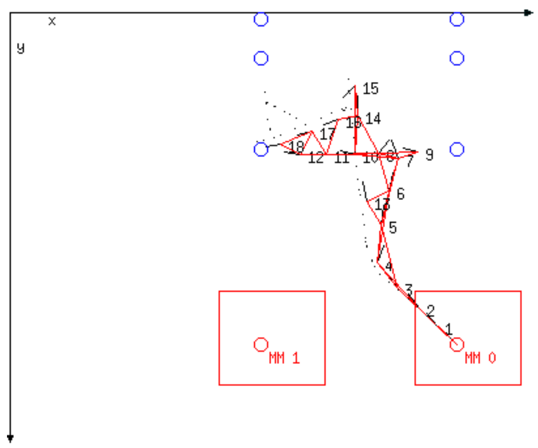
Table 5.21: Task assignment for the bar transportation experiment 1.

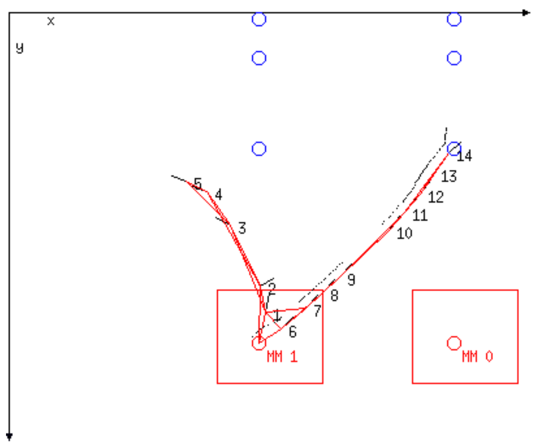




MM_0 MM_1

Simulated setup.

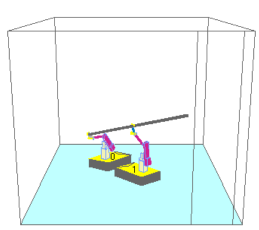




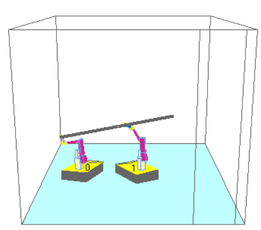
MM_0 MM_1

Real setup.

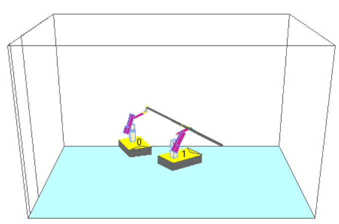
Figure 5.36: Warehouse experiment 6. Emergent *behaviours* for the mobile platforms.



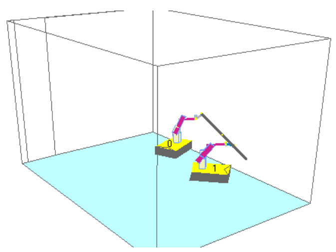
1



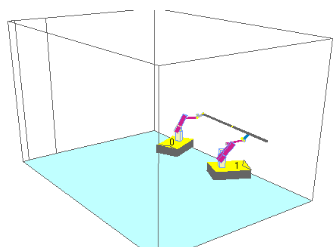
2



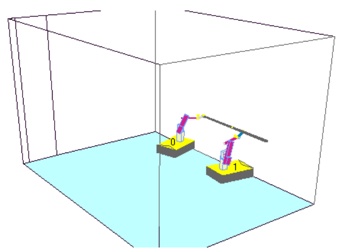
3



4



5



6

Figure 5.37: Six views of the Warehouse experiment 6.

in the xy, xz and yz planes. The bar projections plot shows the motion of the bar in the xy plane. The grasping points at the starting configurations are marked with the symbol $\circ$ (the MM_0's being the one always at one of the extremities of the bar). To improve the readability, the position of the grasping point of MM_0 is shown during all the mission. Obstacle avoidance interaction between the platforms is shown during the approach and line up phases. The trajectories of the grippers reflect also these interactions, together with the active strategy used by MM_1 to move away from MM_0. This experiment shows that, in the absence of interactions with obstacles other than those represented by the mobile platforms present in the experiment, the behaviour of the whole system is adequate to the transportation of a bar through a corridor.

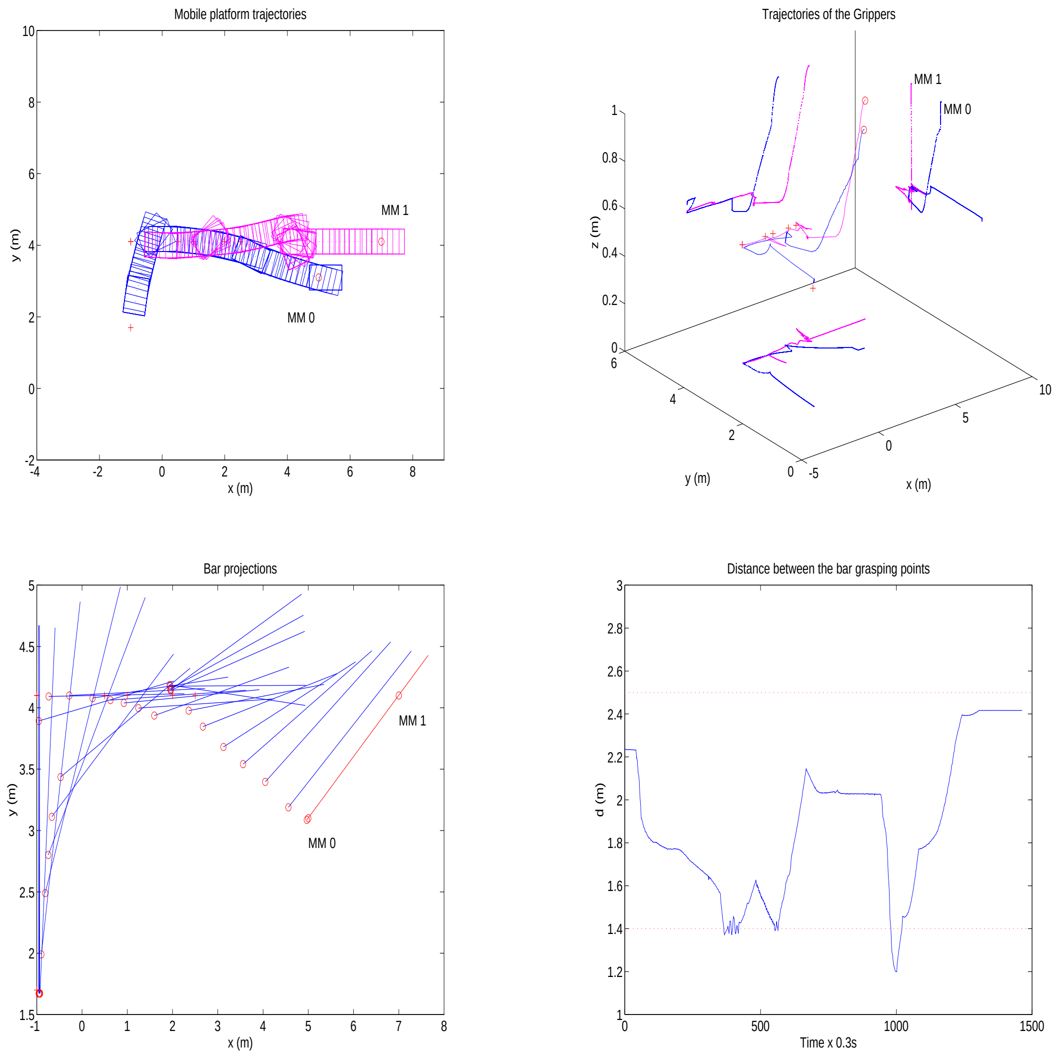


Figure 5.38: Bar transportation experiment 1.

The locations of the tasks in the second mission, specified in Table 5.22, are slightly different from those of the first experiment considered, aiming at simplifying the experiment. In fact, by the time MM_0 reaches the *Move away* task, difficult obstacle avoidance interactions between MM_1 and the final of the corridor lead MM_1 to frequent emergency stops. This phenomenon happens as close as 15 cm from the location of MM_1's final task.

	x	y	z	θ	α	β	γ	tol.
Start	6	3.1	0,92	0	0	0	0	-
Approach	4	4.1	0.5	-	0	-0.7	0	0.05
Line up	3.18	4.1	0.5	-	0	-0.7	0	0.05
Traverse	-3	4.1	0.5	-	0	-0.7	0	0.05
Move away	-2.5	2	0.5	-	0	-0.7	0	0.05

MM_0.

x	y	z	θ	α	β	γ	tol.
8	4.1	0,92	0	0	0	0	-
3.68	4.1	0.5	-	0	-0.7	0	0.05
-1.2	4.1	0.5	-	0	-0.7	-0.7	0.05

MM_1.

Table 5.22: Task assignment for the bar transportation experiment 2.

Figure 5.39 shows the trajectories obtained during the second mission. The representation conventions are identical to the ones used in the previous experiment. The xy plot shows the obstacles forming the corridor as the shadowed areas numbered from 1 to 4. The xyz plot shows the obstacles as shadowed volumes. The obstacle and wall detection thresholds are set, respectively, to 0.42 m and 0.35 m.

As in the previous mission, the trajectory of the mobile platform in MM_1 shows the obstacle avoidance interaction with the MM_0's mobile platform during the two initial phases. At the time this interaction occurred, the MM_0's mobile platform was stopped, waiting for the task completion by the arm and gripper and thus its trajectory does not reflect any perturbation. The entrance in the corridor by MM_0 is made side by side with obstacles 3 and 4, with the obstacles 1 and 2 always lying beyond the 0.42 m obstacle detection threshold. In this mission the ultrasound measurements by MM_0 never went bellow 0.35 m, thus only the general purpose LUTs were triggered. MM_1's mobile platform enters the corridor near the middle of the corridor and approaches obstacles 1 and 2 during the traversal of the corridor. The light perturbation in the MM_1's mobile platform trajectory, at the output of the corridor, is of no consequence as the arm has enough manipulability to complete the mission without needing any maneuver by the mobile platform. Similarly to MM_0, only the general purpose LUTs were used by MM_1.

5.9 Summary

The experiments presented in this chapter illustrate the following relevant issues.

Simulation setup vs. Real setup: Throughout the chapter the same set of actions was used by both the real and the simulated mobile manipulators without any parameter adjustment between experiments. The complex interactions generated during the missions among the agents were correctly handled and hence support the claim on the utility of the simulation setup to design/tune the actions.

***Actions* vs. *behaviours* architecture:** The experiments highlight the advantages in the use of learning. The performance exhibited by the experiments not using the *behaviours* architecture shows that the *a priori* knowledge incorporated in the actions may be sufficient to ensure the

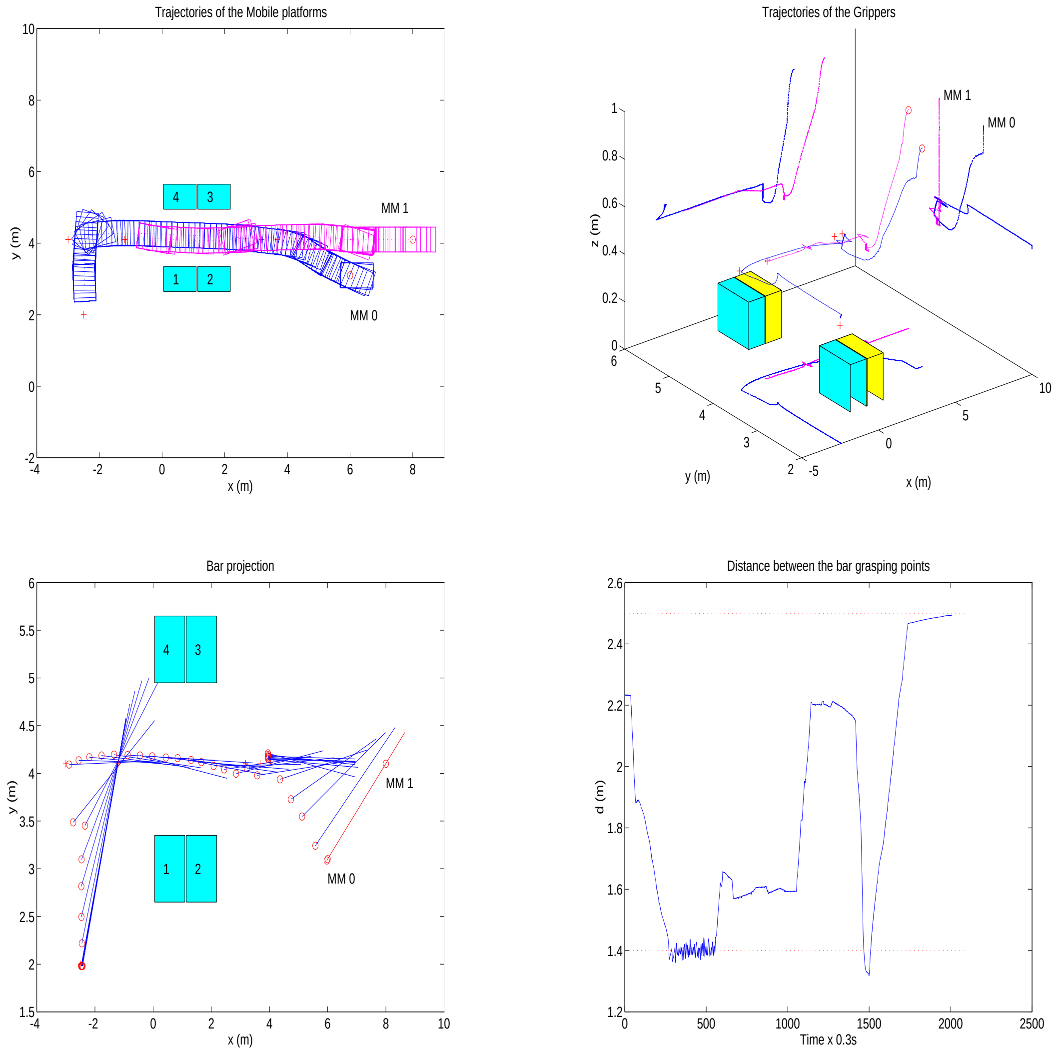


Figure 5.39: Bar transportation experiment 2.

execution of a particular mission.

The experiments in Section 5.3 illustrate the emergence of a *behaviour* in an obstacle avoidance situation that the actions failed to handle by themselves. In Section 5.7.2 the two architectures were compared in a mission with a non-cooperative task assignment. The distance between the majority of the tasks lies outside the $[1.4, 2.5]$ meters interval that sets the limits for the distance between the bar grasping points outside of which the mobile manipulators redefine their assigned tasks. The results show that the *behaviours* architecture is able to find alternative solutions relative to the *actions* architecture. Sections 5.7.4 and 5.7.4 illustrate the use of the *behaviours* architecture in complex obstacle avoidance situations between two mobile platforms (both moving) and when real and simulated agents operate together.

The use of the learning stage reflects a tradeoff between the amount of knowledge included *a priori* in the actions and the complexity of the expected interactions between the agents. The experiments presented, all related with obstacle avoidance interactions and with the learning stage being used only by the mobile platforms, were tailored to minimize the importance of the particular learning algorithm used (intended to be computationally simple). The experiments not using the *behaviours* architecture show that missions with some *a priori* expected complexity, like the transportation of the bar along a corridor, may be executed without the learning stage. As the experiments presented show, assessing the *a priori* complexity of a mission in order to determine whether or not learning should be used is a non-trivial task, as the performance of the actions depends on the particular task assignment and slight variations between missions may generate complex interactions between the agents and the environment.

Cooperation or Master-Slave relationship: The definition of the *active* and *passive* task changing strategies for the mission where the two mobile manipulators operate together does not imply any lack of generality. To support this argument consider an hypothetical problem with the two mobile platforms using the same set of actions and being perfectly synchronized (this is a highly improbable situation in a real system). If placed in front of each other in an obstacle avoidance situation, the arbitration question immediately arises, i.e., which one moves first and to which side so that the other can move to the opposite side (or simply stay still in the same place) avoiding the collision. Without some sort of *priori* unbalancing scheme, i.e., without an arbitration scheme, the deadlock situation will last indefinitely.

Task assignment: Although not directly related with the design of control architectures, with which this thesis dealt with, the task assignment problem is a major issue in the study of cooperation. The diversity of the presented experiments illustrates how the variations on the task locations affect the behaviour of the whole system (refer, for example, to the experiments in Section 5.8). A tradeoff between the task assignment and the design of the actions is implicit in the *behavioural* approach proposed in this thesis: the more knowledge is incorporated in the actions the less difficult becomes the task assignment.

Performance assessment: The results presented show the dependence of the performance measure defined in Section 5.2 on:

- the agent dominance concept, as in the door approach and traversal experiments with the gripper dominating all the task and with the mobile platform dominating the first task (Sub-section 5.5.2), and
- the interactions between the environment and the agents, as in the second and third runs of the grasping mission (Section 5.6).

Chapter 6

Conclusions

6.1 Summary

This thesis presented a theoretical framework to describe a *behavioural* approach to the control of robotic devices, along with the corresponding experimental results. Real and simulated robots were considered.

The observation of human behaviour in daily life shows the success of human beings in the execution of a large variety of tasks. The anthropomorphic characteristics, some of which are nowadays available in robots, namely in mobile manipulators, contribute to this success. The control paradigm underlying this success has been called, in the Engineering sciences, *behaviour based control* and a vast amount of work in this area of the robotics field has been presented in the last decade. Even though the obviously different nature of humans and robots, there remains a large potential in the observation of humans to be explored, if not mimicked. Humans accomplish a multitude of tasks, bypassing the complexity in the control of the large number of degrees of freedom generally required for joint locomotion and manipulation.

The framework presented in this thesis is supported in five basis principles:

1 - purposeful robots have the ability to move, which leads to consider a state space and generators of motion in the state space, named *actions* throughout the thesis;

2 - robots are given the "motion freedom" necessary to simplify their control, which directly leads to the use of a specific equivalence relation in the state space that constitutes the first mechanism to generalize knowledge obtained from the sensor data;

3 - robots have to be controlled, leading to the use of a group structure to provide the basis of motion generation in the state space;

4 - robots are equipped with a set of basis motion strategies (or *a priori* actions), accounting for the existing *a priori* knowledge about them and the environment;

5 - robot's users do not have perfect knowledge when defining the *a priori* actions, leading to the expansion of the basis group structure to a non proper semiring structure, where the second mechanism to generalize knowledge is embedded.

The term *agent* has been used to identify a single robot controlled according to these five principles. An additional principle extends the framework to multiple robot systems:

6 - cooperation is a form of multiple robot control, based on the knowledge generalization ability by each single agent and on the *a priori* assumption on its controllability.

Principles 1 to 5 lead to the identification of a set of basis building blocks for a control architecture with the distinctive characteristic of having two forms of knowledge generalization. *Behaviours* emerge from the operation of such an architecture. Principle 6 identifies the cooperation in a multiple robot systems with some form of controllability of each isolated robot.

The two knowledge generalization mechanisms lead to two specialized forms of the control architecture for single agents, named,

- ***actions* architecture**, encompassing only the first knowledge generalization mechanism, and
- ***behaviours* architecture**, encompassing both knowledge generalization mechanisms.

A *feasible task* assigned to an agent is identified with a controllability property of that agent. Two intermediate controllability concepts, *action controllability* and *point-to-point controllability*, were proposed. The first concept refers to the controllability inside the space bounding a state whereas the second concept refers to the ability of a system to move between any two points in the C-space using a set of *a priori* actions.

In the multiple agent case, the assumption on the adequate form of controllability of each individual agent reduces the problem of determining the feasibility of a task to that of finding appropriate interconnections between the agents so that sufficient information for the task execution is exchanged between them. The specification of a mission assumes, at least, the point-to-point controllability of the whole system of agents.

The experimental results show that

- a variety of relevant tasks can be executed using only the first knowledge generalization mechanism, specially when the set of *a priori* actions is well adapted to the environmental operating conditions,

- a learning stage, identified with the second knowledge generalization mechanism, searches the actions space for alternative actions, being useful in situations when the set of *a priori* actions is not enough for the agent to execute an assigned task.

6.2 General comments

The missions considered in the experimental results focused on a set of situations potentially relevant to real world applications. Even though the simplifying assumptions taken in the design of the simulation setup and the limitations of the real setup, the results obtained show coherency (between the two setups) and consistency (with the expected final behaviour), thus clearly reflecting the soundness of the proposed framework.

Three series of experiments were presented:

- **Single agent** experiments, illustrating the control of single agents;
- **Multiple agents (single mobile manipulator)** experiments, illustrating the control of multiple robots tightly coupled;
- **Multiple agents (two mobile manipulators)** experiments, illustrating the control of multiple robots loosely coupled.

The analysis of the experimental results leads to the following main conclusions:

Task assignment - Even though the use of *a priori* knowledge about the preferential configurations of an agent while executing a task, the assignment of appropriate tasks (in the sense that the mission is feasible) remains, in general, a difficult job. Following the discussion in Section 3.5, the task assignment tends to be difficult in point-to-point controllable agents. Regardless the use of knowledge generalization, that reduces the influence of inappropriate task assignments, a correct *a priori* assessment of the influence of each action on the global behaviour of the agent (or agents) may be difficult, even when using a small set of actions such as the one considered in this thesis.

In general, different agents operate at different velocities, this being caused, for example, by technological constraints. The experimental results presented exhibit several of such situations where the mobile platform waits for the arm to reach the task location. These situations are characterized by the absence of an explicit temporal synchronization between agents, while executing their assigned missions, and are potentially undesirable if not accounted for in the specification of the *a priori* actions. Consider, for example, a push button mission, similar to the one defined in Sub-section 5.4.2. If the mobile platform moves too fast, it reaches the location of the first task while the arm is still in the upward configuration, trying to move away from the singularity region S_3. The arm may not have enough manipulability to complete the task, leading to additional reconfiguration maneuvers by the mobile platform. Temporal synchronization of multiple agent systems during the mission execution can be indirectly provided by additional tasks that force an increase in the time an agent needs to reach a given task. However, the definition of adequate additional tasks results, again, in the aforementioned task assignment problem.

Real vs. Simulated robots - The mixing of real and simulated agents, with different dynamics, enhances the differences between the two setups considered. In general, the difference in the dynamics is reflected in the specification of tasks, as synchronization between agents may be necessary in order to ensure an appropriate mission execution. Regardless these differences, the results presented reinforce the validity of the simulation setup as a tool to design/tune the *a priori* actions, as the same set of actions was used by the two setups in most of the experiments.

***Actions* vs. *Behaviours* -** In general, the execution of tasks not generating complex interactions among the agents does not requires the use of the additional learning mechanism provided by the $\ominus$ blocks of the *behaviours* architecture. The transportation of a bar through a corridor is an example of a problem where the *actions* architecture appropriately handles the interactions among the agents and the environment.

Programming by *Actions* - This expression reflects the approach to the agent control within the proposed framework. In fact, given the control architectures, the user defines

- the preconditions for the detection of the events arising during the execution of a task or class of tasks,
- an appropriate set of motion strategies, named *a priori* actions, that steer the agent through the sequence of states, defined by the *Supervision* block, and lead to the fulfillment of the assigned tasks,
- a Finite State Machine to implement the *Supervision* block, mapping the detected events into the state space.

The architecture is highly modular and enables, if necessary, a fast replacement of any components between missions, e.g., different classes of missions may require different *Supervision* blocks and/or different sets of *a priori* actions.

The sets of *a priori* actions considered are not intended to yield action controllable agents and point-to-point controllable multiple agent systems. The experiments considered are characterized by the simplicity of their task assignments and hence the point-to-point controllability of each of the agents suffices to ensure the execution of the tasks. An example of a typical situation not currently handled arises when a mobile platform stops and waits for the arm to complete its task in a configuration such that the arm would violate a physical joint limit to get the task executed.

The experimental results presented illustrate a particular implementation of the proposed framework. The missions considered can be adapted to real problem applications, e.g., warehousing and indoor office transportation problems, eventually considering additional supervisors, external to the agents, able to handle unexpected situations, namely emergencies other than the two types described in Chapter 4. From a practical perspective, this places the *behavioural* approach presented as an intermediate control methodology between the pure automatic operation of a system and the pure teleoperation.

6.3 Future work

Future work includes the following topics:

- definition and test, within the present computational implementation, of alternative schemes to update the a^p actions whenever the agent is trapped at some point of the workspace, e.g., as in mission of Sub-section 5.7.2 where MM_1 is trapped near MM_0 and was not able to finish its mission;
- study of alternative computational implementations for the basis structure;
- study of additional structures to superimpose on the non proper semiring structure, aiming at simplifying the computational implementation and/or improving the agent's functionalities.
- definition of explicit temporal synchronization mechanisms within the theoretical framework;
- study of the influence of the single agent observability (associated in Chapter 2 with the detection of events) conditions on the performance of a multiple agent system;
- the control of other robots, namely car-like mobile platforms, redundant manipulators and free-flying robots, operating in a diversity of environments imposing specific locomotion and manipulation characteristics;
- development of a general methodology, encompassing the definition of the tasks and the tunning of the actions, for the application of the framework to a wide range of real problems.

Bibliography

[Ahmad and Luo, 1989] Ahmad, S. and Luo, S. (1989). Coordinated Motion Control of Multiple Robotic Devices for Welding and Redundancy Coordination through Constrained Optimization in Cartesian Space. *IEEE Transactions on Robotics and Automation*, 5(4).

[Alami et al., 1998] Alami, R., Fleury, S., Herrb, S., Ingrand, F., and Robert, F. (1998). Multi-Robot Cooperation in the MARTHA Project. *IEEE Robotics & Automation Magazine*, 5(1). Special issue on european projects.

[Albus, 1996] Albus, J. S. (1996). The Engineering of Mind. In *4th International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 4*. Cape Code, MA, USA.

[Albus and Meystel, 1996] Albus, J. S. and Meystel, A. M. (1996). A Reference Model Architecture for Design and Implementation of Intelligent Control in Large and Complex Systems. *International Journal of Intelligent Control and Systems*, 1(1):15–30.

[Almeida Costa, 1974] Almeida Costa, A. (1974). *Cours d'Algèbre Générale*, volume III. Fundação Calouste Gulbenkian — Lisboa.

[Antsaklis et al., 1994] Antsaklis, P., Lemmon, M., and Stiver, J. (1994). Modeling and Design of Hybrid Control Systems. In *2nd IEEE Mediterranean Symposium on New Directions in Control and Automation*. Maleme-Chania, Crete, Greece.

[Arkin, 1995] Arkin, R. (1995). Reactive Robotic Systems. In Arbib, M., editor, *Handbook of Brain Theory and Neural Networks*, pages 793–796. MIT Press.

[Arkin and Balch, 1998] Arkin, R. and Balch, T. (1998). Cooperative Multiagent Robotic Systems. In Kortenkamp, D., Bonasso, R., and Murphy, R., editors, *AI-Based Mobile Robots:Case Studies of Successful Robot Systems*. MIT Press. Conference and Symposium Papers, in press.

[Aubin, 1991] Aubin, J. (1991). *Viability Theory*. Birkhäuser.

[Bekey, 1996] Bekey, G. (1996). Biologically inspired control of autonomous robots. *Robotics and Autonomous Systems*, 18:21–31.

[Berger and Gostiaux, 1988] Berger, M. and Gostiaux, B. (1988). *Differential Geometry: Manifolds, Curves, and Surfaces*. Springer-Verlag.

[Borenstein and Koren, 1991] Borenstein, J. and Koren, Y. (1991). The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 7(3).

[Bourbaki, 1989] Bourbaki, N. (1989). *Elements of Mathematics - Algebra*, volume I. Springer-Verlag, 2nd edition. Chapters 1-3.

[Brocket, 1993] Brocket, R. W. (1993). Hybrid Models for Motion Control Systems. In H. L. Trentelman and J. C. Willems, editors, *Essays on Control: Perspectives in the Theory and its Applications*, pages 29–53. Birkhäuser.

[Brooks, 1986a] Brooks, R. (1986a). A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, RA-2(1).

[Brooks, 1986b] Brooks, R. (1986b). Achieving Artificial Intelligence Through Building Robots. Technical report, Massachussets Institute of Technology, USA. Artificial Intelligence Laboratory Memo 899,.

[Çela and Hamam, 1992] Çela, A. and Hamam, Y. (1992). Optimal Motion Planning of a Multiple-Robot System Based on Decomposition Coordination. *IEEE Transactions on Robotics and Automation*, 8(5).

[Cohen, 1996] Cohen, W. (1996). Adaptive mapping and navigation by teams of simple robots. *Robotics and Autonomous Systems*, 18:411–434.

[Colombetti et al., 1996] Colombetti, M., Dorigo, M., and Borghi, G. (1996). Behaviour Analysis and Training: A Methodology For Behavior Engineering. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(6).

[Craig, 1986] Craig, J. (1986). *Introduction to Robotics - Mechanics and Control.* Addison-Wesley.

[Freitas et al., 1997] Freitas, L. M., Carvalho, F., Sequeira, J., and Ribeiro, M. I. (1997). Sistema Integrado de Movimentação de um Robot Móvel Baseado em Dados Sensoriais. Technical Report ISR/IST-RT-602-97, Instituto de Sistemas e Robótica / Instituto Superior Técnico, Lisbon, Portugal. In Portuguese.

[Fukuda and Ueyama, 1994] Fukuda, T. and Ueyama, T. (1994). *Cellular Robotics and Micro Robotic Systems.* World Scientific.

[Gurvitz and Li, 1993] Gurvitz, L. and Li, Z. (1993). Smooth Time-Periodic Feedback Solutions for Nonholonomic Motion Planning. In Li, Z. and Canny, J., editors, *Non-Holonomic Motion Planning*, pages 53–108. Kluwer Academic.

[Hart and Mas-Colell, 1996] Hart, S. and Mas-Colell, A., editors (1996). *Cooperation: Games Theoretic Approaches.* Springer. NATO ASI Series F, Vol. 155.

[Hazewinkel, 1996] Hazewinkel, M., editor (1996). *Handbook of Algebra*, volume I. North-Holland.

[Hermann and Krener, 1977] Hermann, R. and Krener, A. (1977). Nonlinear Controllability and Observability. *IEEE Transactions on Automatic Control*, AC-22(5).

[Hungerford, 1996] Hungerford, T. (1996). *Algebra*. Number 73 in Graduate Texts in Mathematics. Springer, 8th edition.

[Isik and Meystel, 1988] Isik, C. and Meystel, A. (1988). Pilot Level of a Hierarchical Controller for an Unmanned Mobile Robot. *IEEE Journal of Robotics and Automation*, 4(3).

[Jacobs and Canny, 1993] Jacobs, P. and Canny, J. (1993). Planning Smooth Paths for Mobile Robots. In Z. Li and J. Canny, editors, *Non-Holonomic Motion Planning*, pages 271–342. Kluwer Academic.

[Jurdjevic, 1997] Jurdjevic, V. (1997). *Geometric Control Theory*. Cambridge University Press.

[Klausmeier, 1977] Klausmeier, H. (1977). *Manual de Psicologia Educacional*. Harper & Row, Brasil. In Portuguese.

[Knapp, 1996] Knapp, D. (1996). *Behavior Synthesis: Digital System Design Using the Synopsis Behavioural Compiler*. Prentice Hall.

[Košecká et al., 1997] Košecká, J., Christensen, H., and Bajcsy, R. (1997). Experiments in behavior composition. *Robotics and Autonomous Systems*, 19:287–298.

[Kreysig, 1978] Kreysig, E. (1978). *Functional Analysis*. Wiley.

[Lafferriere and Sussmann, 1993] Lafferriere, G. and Sussmann, H. (1993). A Differential Geometric Approach to Motion Planning. In Z. Li and J. Canny, editors, *Non-Holonomic Motion Planning*, pages 235–270. Kluwer Academic.

[Latombe, 1992] Latombe, J. C. (1992). *Robot Motion Planning*. Kluwer Academic.

[Laumond, 1993] Laumond, J. P. (1993). Singularities and Topological Aspects in Nonholonomic Motion Planning. In Z. Li and J. Canny, editors, *Non-Holonomic Motion Planning*, pages 149–199. Kluwer Academic.

[Li et al., 1991] Li, Z., Tzyh-Jong, T., and Bejczy, A. (1991). Dynamic Workspace Analysis of Multiple Cooperating Robots Arms. *IEEE Transactions on Robotics and Automation*, 7(5).

[Liu and Arimoto, 1996] Liu, Y. and Arimoto, S. (1996). Distributively Controlling Two Robots Handling an Object in the Task Space Without Any Communication. *IEEE Transactions on Automatic Control*, 41(8).

[Magalhães, 1993] Magalhães, L. (1993). *Integrais em Variedades e Aplicações*. Texto Editora. In Portuguese.

[Massey, 1977] Massey, W. S. (1977). *Algebraic Topology: An Introduction*. Number 56 in Graduate Texts in Mathematics. Springer, 4th edition.

[Matarić, 1994] Matarić, M. (1994). *Interaction and Intelligent Behaviour.* PhD thesis, Massachusetts Institute of Technology, USA. Department of Electrical Engineering and Computer Science.

[Meystel, 1996] Meystel, A. (1996). Intelligent Systems: A Semiotic Perspective. *International Journal of Intelligent Control and Systems*, 1(1):31–57.

[Millan, 1996] Millan, J. (1996). Rapid, Safe and Incremental Learning of Navigation Strategies. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 26(3).

[Millan, 1997] Millan, J. (1997). Incremental Aquisition of Local Networks for the Control of Autonomous Robots. In *7th International Conference on Artificial Neural Networks*, pages 739–744. Lausanne, Switzerland.

[Narendra and Thathachar, 1989] Narendra, K. and Thathachar, M. A. L. (1989). *Learning Automata - An introduction.* Prentice-Hall.

[Olver, 1993] Olver, P. (1993). *Applications of Lie Groups to Differential Equations.* Number 107 in Graduate Texts in Mathematics. Springer-Verlag, 2nd edition.

[Parker, 1998] Parker, L. E. (1998). ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240.

[Pin et al., 1994] Pin, F., Culioli, J., and Reister, D. (1994). Using Minimax Approaches to Plan Optimal Task Commutation Configurations for Combined Mobile Platform-Manipulator Systems. *IEEE Transactions on Robotics and Automation*, 10(1).

[Ram et al., 1997] Ram, A., Arkin, R. C., Moorman, K., and Clark, R. J. (1997). Case-Based Reactive Navigation: A Method for On-Line Selection and Adaptation of Reactive Robotic Control. *IEEE Transactions on Systems, Man and Cybernetics*, 27(3).

[Ramadge and Wonham, 1987] Ramadge, P. and Wonham, W. (1987). Supervisory Control Of A Class Of Discrete Event Processes. *SIAM Journal of Control and Optimization*, 25(1).

[Reignier et al., 1995] Reignier, P., Hansen, V., and Crowley, J. (1995). Incremental Supervised Learning for a Mobile Robot Reactive Control. *Intelligent Autonomous Systems.*

[Saridis, 1996] Saridis, G. N. (1996). On the Theory of Intelligent Machines: A Comprehensive Analysis. *International Journal of Intelligent Control and Systems*, 1(1):3–14.

[Sequeira and Ribeiro, 1997] Sequeira, J. and Ribeiro, M. I. (1997). A Behaviour-Based Kernel Architecture For Robot Control. In *Proceedings of 5th IFAC Symposium on Robot Control, Syroco'97.* Cité Des Congres - Nantes, France.

[Sequeira and Ribeiro, 1998a] Sequeira, J. and Ribeiro, M. I. (1998a). Experiments on cooperative behaviour between two mobile manipulators. In *Proceedings of 6th International Symposium On Intelligent Robotic Systems'98, SIRS'98.* Edinburgh, Scotland.

[Sequeira and Ribeiro, 1998b] Sequeira, J. and Ribeiro, M. I. (1998b). Rigid Bar Transportation By Two Mobile Manipulators: A Behaviour-Based Cooperative Approach. In *Proceedings of the 3rd Portuguese Conference on Automatic Control, CONTROL'98*. Coimbra, Portugal.

[Sernesi, 1993] Sernesi, E. (1993). *Linear Algebra. A geometric approach*. Chapman & Hall.

[Sheu and Xue, 1993] Sheu, P. and Xue, Q. (1993). *Intelligent Robotic Planning Systems*. World Scientific.

[Simmons, 1963] Simmons, G. (1963). *Topology And Modern Analysis*. McGraw-Hill.

[Singer and Thorpe, 1967] Singer, I. M. and Thorpe, J. A. (1967). *Lecture Notes on Elementary Topology and Geometry*. Undergraduate Texts in Mathematics. Springer.

[Sordalen, 1993] Sordalen, O. (1993). *Feedback Control of Nonholonomic Mobile Robots*. PhD thesis, The Norwegian Institute of Technology. Department of Engineering Cybernetics, N-7034 Trondheim, Norway.

[Sussmann and Jurdjevic, 1972] Sussmann, H. and Jurdjevic, V. (1972). Controllability of Nonlinear Systems. *Journal of Differential Equations*, (12):95–116.

[Valavanis and Saridis, 1991] Valavanis, K. and Saridis, G. (1991). Probabilistic Modeling of Intelligent Robotic Systems. *IEEE Transactions on Robotics and Automation*, 7(1).

[Valavanis et al., 1997] Valavanis, K. P., Gracanin, D., Matijasevic, M., Kolluru, R., and Demetriou, G. A. (1997). Control Architectures for Autonomous Underwater Vehicles. *IEEE Control Systems*, 17(6).

[Veloso et al., 1997] Veloso, M., Stone, P., Han, K., and Achim, S. (1997). CMUnited: A Team of Robotic Soccer Agents Collaborating in an Adversarial Environment. In *Proceedings of the 1st International Workshop on RoboCup, in IJCAI'97*. Nagoya, Japan.

[Ward and Arkin, 1994] Ward, K. and Arkin, R. (1994). Reactive Control of a Mobile Manipulator using Pseudo-joint Damping. Technical report, Georgia Institute of Technology, USA.

[Willems, 1991] Willems, J. C. (1991). Paradigms and Puzzles in the Theory of Dynamical Systems. *IEEE Transactions on Automatic Control*, 36(3).

[Yamamoto and Yun, 1993] Yamamoto, Y. and Yun, X. (1993). Coordinating Locomotion and Manipulation of a Mobile Manipulator. In Yuan F, Z., editor, *Recent Trends in Mobile Robots*. World Scientific.

[Yun, 1993] Yun, X. (1993). A behaviour-based blackboard architecture for reactive and efficient task execution of an autonomous robot. *The International Journal of Robotics Research*, 12(1):99–106.

Appendix A

Basis mathematical concepts

This appendix briefly presents the mathematical concepts used throughout the thesis, aiming at rending the work self-contained as much as possible. The material presented can be found in many bibliographic sources. Some books on robotics, namely [Latombe, 1992], also provide appendices that group together some of the concepts listed in this appendix.

A.1 Connectedness and path connectedness

This section follows [Magalhães, 1993], pp. 21, and [Massey, 1977], pp. 56.

A topological space is *connected* if it is not the union of any two disjoint, open, nonempty, subsets. The space is called *path connected*, or *arcwise connected*, if any two of its points can be linked by a path entirely contained in the space.

A.2 Semigroups, groups, semirings, rings, linear spaces and algebras

This section follows [Simmons, 1963], Chapter 8, [Hungerford, 1996], Chapter 1, [Almeida Costa, 1974], Chapter 14, and [Hazewinkel, 1996], Section 1F.

A semigroup is a non-empty set, G, together with an operation (usually called *multiplication*) closed in G that associates to any two elements x, y in G the *product* xy and which verifies the following property:

Associative if x, y, z are any three elements in G, then $x(yz) = (xy)z$.

If the two following additional properties are verified then the semigroup becomes a group:

Identity there exists an element, represented by e and called the *identity element* with the property that $xe = ex = x$;

Inverse to each element $x \in G$ there corresponds another element in G, represented by x^{-1} and called the *inverse element*, with the property that $xx^{-1} = x^{-1}x = e$.

A group with the property that, for any x, y in G, $xy = yx$, is called a *commutative group* or *abelian group*. When dealing with abelian groups, it is customary to use the *additive notation* to represent the group operation, [Hungerford, 1996, Sernesi, 1993].

A non-empty set, G, which is a semigroup relative to two binary operations, denoted by $+$ and $\cdot$, such that, for any three elements x, y, z of G, the properties

Left distributive $x \cdot (y + z) = x \cdot y + x \cdot z$,

Right distributive $(y + z) \cdot x = y \cdot x + z \cdot x$,

are verified, is called a *semiring*. $(G, +, \cdot)$ is called *proper* if $(G, +)$ is not a group.

A ring is an abelian group which is closed under a second operation, also called *multiplication*, and verifying the properties:

Associative if x, y, z are any three elements of G, then $(xy)z = x(yz)$;

Distributive if x, y, z are any three elements of G, then $x(y+z) = xy+xz$ and $(x+y)z = xz+yz$.

If a ring contains a non-zero element, denoted 1, such that $x1 = 1x = x$ for every element of the ring, then 1 is called the identity element and the ring is called a *ring with identity*. In the case that, for any element x, there is a single element y such that $xy = yx = 1$, then y is called the *inverse* of x.

A ring with identity element is called a *division ring* if all its non-zero elements have inverse.

A *field* is a commutative division ring.

A linear space over a field K is an additive group, G, such that any element $x \in G$ can be multiplied by a scalar, $\alpha \in K$, yielding a new element αx, and verifying the following properties:

(1) for any x, y in G and any scalar α, $\alpha(x+y) = \alpha x + \alpha y$;

(2) for any $x \in G$ and any scalars α, β, $(\alpha + \beta)x = \alpha x + \beta x$;

(3) for any $x \in G$ and scalars α, β, $(\alpha\beta)x = \alpha(\beta x)$;

(4) there exists a unique element in G, denoted by $\mathbf{1}$, such that $\mathbf{1} \cdot x = x$.

An algebra is a linear space which is also a ring and where the *multiplication* operation defining the ring and the scalar multiplication are related by

$$\alpha(xy) = (\alpha x)y = x(\alpha y).$$

A.3 Homeomorphisms and diffeomorphims

This section follows [Simmons, 1963], pp. 93, [Singer and Thorpe, 1967], pp. 15, and [Latombe, 1992].

Let X and Y be topological spaces, f be a mapping of X into Y, G_x be an open subset of X and G_y be an open subset of Y. The mapping f is called an *open mapping* if $f(G_x)$ is open. The mapping f is called a *continuous mapping* if $f^{-1}(G_y)$ is open, i.e., it pulls back open sets to open sets.

A *homeomorphism* is a one-to-one continuous mapping from one topological space onto another topological space being also an open mapping.

If f and f^{-1} are C^∞ differentiable then f is called a *diffeomorphism.*

A.4 Manifolds and tangent spaces

This section follows [Latombe, 1992], Appendix A, and [Berger and Gostiaux, 1988], pp. 88.

A topological space M is a *manifold* if every point $x \in M$ has an open neighbourhood homeomorphic to an open ball in $\mathbb{R}^n$, for some n independent of x. The number n is the dimension of the manifold. The manifold is smooth if the relation with $\mathbb{R}^n$ is diffeomorphic. A submanifold of a manifold M is a subset $N \subset M$ that is also a manifold.

Let X be a submanifold of $\mathbb{R}^n$. A vector $z \in \mathbb{R}^n$ is said to be *tangent to V at x* if there exists a C^1 curve $\alpha : I \mapsto X$ (where $I \subset \mathbb{R}$ containing 0) such that $\alpha(0) = x$ and $\dot{\alpha}(0) = z$. The set of all tangent vectors to X at x is called *the tangent space to X at x* and usually denoted by TX_x. The union $TX = \cup_{x \in X} TX_x$ is called *the tangent bundle of X*.

Let X and Y be manifolds and $f : X \mapsto Y$ a differentiable mapping. The mapping f is called an *immersion* at x if the mappings between the tangent spaces of the two manifolds, denoted by $T_x X$ and $T_{f(x)} Y$, is injective.

The mapping f is called an *embedding* if f is an *immersion* and a homeomorphism onto its image $f(X)$. In general, a robot's C-space, Q, is a subset of an euclidean space $\mathbb{R}^n$, in which case $\mathbb{R}^n$ is said to be the *ambient* space of Q. The representation of the C-space is the *embedding* of Q in $\mathbb{R}^n$.

As an example consider the group G of rotations in the plane, usualy denoted by SO(2), whose elements, g, can be represented, in matrix form, by

$$g = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \tag{A.1}$$

where $\theta \in \mathbb{R}$. The mapping defined by (A.1) is a homeomorphism between $\mathbb{R}$ and G. Note that identical elements of G may have been mapped from different elements in $\mathbb{R}$.

The embedding is represented by the mapping between the group G and $\mathbb{R}^4$, which has enough space to represent the elements of G.

A.5 Homomorphisms

This section follows [Hungerford, 1996], pp. 30.

Let G and H be semigroups with the operations $\diamond$ and $\triangle$, respectively. A mapping $f : G \mapsto H$ is a *homomorphism* provided that

$$\forall a, b \in G \quad : \; f(a \diamond b) = f(a) \triangle f(b).$$

If f is injective then the homomorphism is called a *monomorphism*. If f is surjective it is called an *epimorphism*. If f is bijective it is called and *isomorphism*. An homomorphism $f : G \mapsto G$ is called an *endomorphism* of G and an isomorphism $f : G \mapsto G$ is called an *automorphism* of G.

A.6 Homotopy

This section follows [Singer and Thorpe, 1967], pp. 49.

Let X and Y be topological spaces, and let f_0 and f_1 be continuous mappings $X \mapsto Y$. f_0 is *homotopic* to f_1 if there exists a continuous mapping $F : X \times [0,1] \mapsto Y$ such that

$$\begin{array}{l} F(x,0) = f_0(x) \\ F(x,1) = f_1(x) \end{array}$$

for all $x \in X$. The mapping F is called a *homotopy* from f_0 to f_1.

A.7 Groups of transformations and Lie groups

The material in this section is based in [Sernesi, 1993], Chapter 14, [Massey, 1977], Appendix B, and [Olver, 1993], Chapter 1.

Definition A.7.1 (Group of transformations) *Let Q be a set and A be a group of operators, with multiplication $\circ$ and null element a_e. For any element $a \in A$ and $q \in Q$, consider the mapping $\Psi : (q,a) \mapsto a(q)$, that transforms the pair (q,a) into a generator of trajectories in Q, all of them starting in a neighbourhood of q and ending in another neighbourhood of some terminal configuration. Assume that Ψ verifies the following properties*

$$\begin{array}{ll} (1) & \textit{For any } q \in Q, \\ & \quad \Psi(q,a_e) = q \quad \textit{or} \quad a_e(q) = q, \\ (2) & \textit{For any } q \in Q \textit{ and } a_1, a_2 \in A, \\ & \quad \Psi(q, a_1 \circ a_2) = \Psi(\Psi(q,a_1),a_2) \quad \textit{or} \quad (a_1 \circ a_2)(q) = a_1(a_2(q)). \end{array} \tag{A.2}$$

Then, A is a group of transformations of Q.

Definition A.7.1 considers the left action of the operators a_i on Q. Similarly, it is possible to consider the right action (see [Massey, 1977] for details). A subgroup of A is called a *transformation group* of Q, [Sernesi, 1993]. Clearly, A is also a *transformation group* of Q. To simplify the identification with the *behavioural* framework, a notation similar to the one considered in Chapter 2 is used in the (A.2) properties.

In the control area it is also of interest to consider more specialized groups, namely Lie groups.

Definition A.7.2 (Lie group) *An r-parameter Lie group is a group G which is also an r-dimensional manifold and where the group operation, $m : G \times G \mapsto G$, and the inversion $i : G \mapsto G$, $\quad i(g) = g^{-1}$ are smooth mappings between manifolds.*

Whenever the group operation and group inversion are not defined everywhere, the group is said to be *local*.

The groups of transformations have a local character if A is a local group, i.e., the group operation is defined only in a neighbourhood of the null element. Moreover, the group may not act in all points of Q.

The set of all the $n \times n$ invertible matrices, usually denoted by GL_n, is a group under the usual matrix multiplication. For $n = 3$ and $n = 4$ the subsets of GL_n with elements of the form

$$\begin{bmatrix} R_{n-1} & \mathbf{t}_{n-1} \\ \mathbf{0}_{1,n-1} & 1 \end{bmatrix},$$

where R_{n-1} satisfies $R'_{n-1}R_{n-1} = \mathbf{I}_{n-1}$ and $\det(R_{n-1}) = 1$ (i.e., R_{n-1} is a rotation matrix), $\mathbf{t}_{n-1}$ is a column vector of order $n-1$ and $\mathbf{0}_{1,n-1}$ is a line vector with $n-1$ zeros, form a subgroup of special importance in robotics as they describe rigid body motions (rotations and translations). This group is usually denoted by SE_2 for 2D motions and by SE_3 for 3D motions. The subset of GL_n with elements of the form

$$\begin{bmatrix} R_{n-1} & \mathbf{0}_{n-1,1} \\ \mathbf{0}_{1,n-1} & 1 \end{bmatrix}$$

is the subgroup that describes rigid body rotations. These groups are usually denoted by SO_2, for $n = 3$, and SO_3, for $n = 4$. If $\theta \in \mathbb{R}$, M_n is a $n \times n$ arbitrary matrix, and $S_{n-1}(\theta)$ is an element of SO_{n-1}, the transformation defined by

$$\Psi_{S_{n-1}(\theta)}(M_n) = S_{n-1}(\theta) \cdot M_n$$

corresponds to a rotation, by an amount of θ radians, of each column vector of M_n around a fixed axis. Identifying, in Definition A.7.1, $S_{n-1}(\theta)$ with A (with the usual matrix product as group operation) and GL_n with Q, the set with elements of the form $\Psi_{S_{n-1}(\theta)}(\cdot)$ is the transformation group of the rotations around a fixed axis.

Another example of a transformation group is the group of translations along a fixed direction in $\mathbb{R}^n$. Let $x \in \mathbb{R}^n$, $\epsilon, \gamma \in \mathbb{R}$ and $\mathbf{v}$ be the vector in $\mathbb{R}^n$ along which the translations are to occur. Identifying $\mathbb{R}^n$ with Q, $(\mathbb{R}, +)$ with A and making $\Psi_\epsilon(\mathbf{v}, x) = x + \epsilon\mathbf{v}$ the transformation group of translations along $\mathbf{v}$ is obtained. A straightforward computation verifies the properties in Definition A.7.1

$$\begin{array}{l} \Psi_\epsilon(\mathbf{v}, \Psi_\gamma(\mathbf{v}, x)) = \Psi_{\epsilon+\gamma}(\mathbf{v}, x) \\ \Psi_0(\mathbf{v}, \Psi_\epsilon(\mathbf{v}, x) = \Psi_\epsilon(\mathbf{v}, \Psi_0(\mathbf{v}, x)) = \Psi_\epsilon(\mathbf{v}, x) \\ \Psi_{-\epsilon}(\mathbf{v}, \Psi_\epsilon(\mathbf{v}x) = \Psi_\epsilon(\mathbf{v}, \Psi_{-\epsilon}(\mathbf{v}x) = \Psi_0(\mathbf{v}, x). \end{array}$$

A.8 The Lie bracket and its geometrical interpretation

This section follows [Olver, 1993], both for the geometrical interpretation of the Lie bracket, pp. 33-35, and for the notation used for actions of vector fields on functions, pp. 30-31.

Denote the value of the scalar function f at point $x = (x^1, \dots, x^n)$ by $f(x)$ and the action of the vector field $\mathbf{v}$ on the function f at x by

$$\mathbf{v}(f)_x = \sum_{i=1}^{n} \xi^i \frac{\partial f(x)}{\partial x^i}. \tag{A.3}$$

where the ξ^i are scalars. The vector field $\mathbf{v}$ at point x is denoted by $\mathbf{v}_x$ and a mapping $\eta : M \mapsto M$ at point x is denoted by η_x.

Considering two vector fields on a smooth manifold M, expressed in local coordinates, $\mathbf{v} = \sum_{i=1}^{n} \xi^i \frac{\partial}{\partial x^i}$ and $\mathbf{w} = \sum_{i=1}^{n} \eta^i \frac{\partial}{\partial x^i}$, their Lie bracket is the vector field defined by

$$[\mathbf{v}, \mathbf{w}] = \sum_{i=1}^{n} (\frac{\partial \xi^i}{\partial x^i} \eta^i - \frac{\partial \eta^i}{\partial x^i} \xi^i) \frac{\partial}{\partial x^i}. \tag{A.4}$$

which, using the notation (A.3), yields

$$[\mathbf{v}, \mathbf{w}] = \mathbf{v}(\eta) - \mathbf{w}(\xi). \tag{A.5}$$

The geometrical interpretation of the vector field $[\mathbf{v}, \mathbf{w}]$ follows from the analysis of the trajectories of the system

$$\dot{x} = \mathbf{v}(x) u_v + \mathbf{w}(x) u_w, \tag{A.6}$$

where $x \in M$ and u_v, u_w are scalars representing the controls of the system. Assuming some initial condition, x_0, let the controls be given by the sequence

$$(u_v, u_w) = \begin{cases} (1,0) & t \in [0, \epsilon) \\ (0,1) & t \in [\epsilon, 2\epsilon) \\ (-1,0) & t \in [2\epsilon, 3\epsilon) \\ (0,-1) & t \in [3\epsilon, 4\epsilon) \end{cases} \tag{A.7}$$

where t represents the time and ϵ an arbitrarily small constant.

The geometric interpretation of the Lie bracket follows from a Taylor series expansion of which, for the sake of understanding, the four main steps are presented. After applying the sequence of controls defined by (A.7), the system passes through the states $x_1 = x(\epsilon)$ to $x_4 = x(4\epsilon)$ (see Figure A.1) given by

$$
\begin{aligned}
x_4 &= x_3 - \epsilon\eta_{x_3} + \frac{\epsilon^2}{2}\mathbf{w}_{x_3}(\eta_{x_3}) + \mathbf{O}(\epsilon^3) \\
x_3 &= x_2 - \epsilon\xi_{x_2} + \frac{\epsilon^2}{2}\mathbf{v}_{x_2}(\xi_{x_2}) + \mathbf{O}(\epsilon^3) \\
x_2 &= x_1 + \epsilon\eta_{x_1} + \frac{\epsilon^2}{2}\mathbf{w}_{x_1}(\eta_{x_1}) + \mathbf{O}(\epsilon^3) \\
x_1 &= x_0 + \epsilon\xi_{x_0} + \frac{\epsilon^2}{2}\mathbf{v}_{x_0}(\xi_{x_0}) + \mathbf{O}(\epsilon^3).
\end{aligned} \tag{A.8}
$$

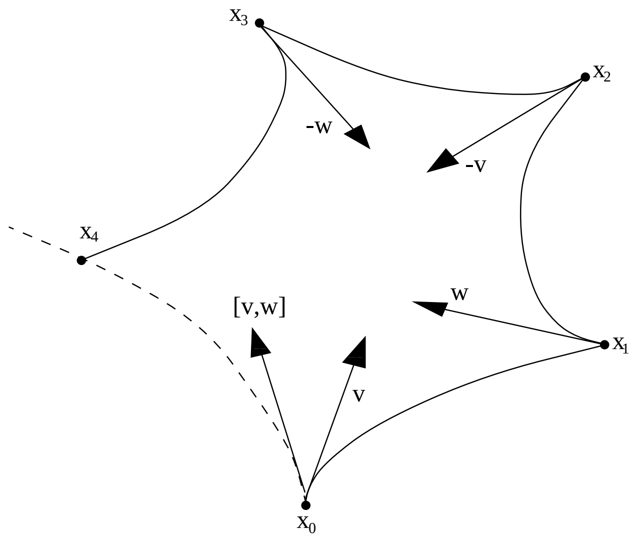


Figure A.1: Geometrical interpretation of the Lie bracket.

Note that

$$
\mathbf{w}_{x_3}(\eta_{x_3}) = \begin{bmatrix} \frac{\partial\eta^1}{\partial x^1} & \cdots & \frac{\partial\eta^1}{\partial x^n} \\ \vdots & & \vdots \\ \frac{\partial\eta^n}{\partial x^1} & \cdots & \frac{\partial\eta^n}{\partial x^n} \end{bmatrix}_{x_3} \begin{bmatrix} \dot{x}^1 \\ \vdots \\ \dot{x}^n \end{bmatrix}_{x_3} \tag{A.9}
$$

and similarly for $\mathbf{v}_{x_2}(\xi_{x_2})$, $\mathbf{w}_{x_1}(\eta_{x_1})$ and $\mathbf{v}_{x_0}(\xi_{x_0})$.

Using the Taylor series expansion, the two following sets of expressions hold:

$$
\begin{aligned}
\eta_{x_3} &= \eta_{x_2} + \epsilon\mathbf{v}_{x_2}(\eta_{x_2}) + \mathbf{O}(\epsilon^2) \\
\xi_{x_2} &= \xi_{x_1} + \epsilon\mathbf{w}_{x_1}(\xi_{x_1}) + \mathbf{O}(\epsilon^2) \\
\eta_{x_2} &= \eta_{x_1} + \epsilon\mathbf{w}_{x_1}(\eta_{x_1}) + \mathbf{O}(\epsilon^2) \\
\xi_{x_1} &= \xi_{x_0} + \epsilon\mathbf{v}_{x_0}(\xi_{x_0}) + \mathbf{O}(\epsilon^2),
\end{aligned} \tag{A.10}
$$

$$
\begin{aligned}
\mathbf{w}_{x_3}(\eta_{x_3}) &= \mathbf{w}_{x_2}(\eta_{x_2}) + \mathbf{O}(\epsilon) \\
\mathbf{v}_{x_2}(\xi_{x_2}) &= \mathbf{v}_{x_1}(\xi_{x_2}) + \mathbf{O}(\epsilon) \\
\mathbf{w}_{x_2}(\xi_{x_2}) &= \mathbf{w}_{x_1}(\xi_{x_1}) + \mathbf{O}(\epsilon) \\
\mathbf{v}_{x_1}(\eta_{x_1}) &= \mathbf{v}_{x_0}(\eta_{x_0}) + \mathbf{O}(\epsilon) \\
\mathbf{v}_{x_1}(\xi_{x_1}) &= \mathbf{v}_{x_0}(\xi_{x_0}) + \mathbf{O}(\epsilon) \\
\mathbf{w}_{x_1}(\xi_{x_1}) &= \mathbf{w}_{x_0}(\xi_{x_0}) + \mathbf{O}(\epsilon).
\end{aligned}
\tag{A.11}
$$

Replacing the second expression in (A.8) into the first and using the first expressions in (A.10) and (A.11) yields

$$x_4 = x_2 - \epsilon\xi_{x_2} + \frac{\epsilon^2}{2}\mathbf{v}_{x_2}(\xi_{x_2}) - \epsilon\eta_{x_2} + \epsilon^2\mathbf{v}_{x_2}(\eta_{x_2}) + \frac{\epsilon^2}{2}\mathbf{w}_{x_2}(\eta_{x_2}) + \mathbf{O}(\epsilon^3) \tag{A.12}$$

which, after the substitution of the third expression in (A.8), of the second and third expressions in (A.10) and (A.11) and some cancellations yields

$$x_4 = x_1 - \epsilon\xi_{x_1} + \epsilon^2(\mathbf{v}_{x_1}(\eta_{x_1}) - \mathbf{w}_{x_1}(\xi_{x_1}) + \frac{\epsilon^2}{2}\mathbf{v}_{x_1}(\eta_{x_1}) + \mathbf{O}(\epsilon^3) \tag{A.13}$$

which, after the substitution of the fourth expression in (A.8), of the third expression in (A.10) and of the last three expressions in (A.11) and some cancellations yields

$$x_4 = x_0 + \epsilon^2(\mathbf{v}_{x_0}(\eta_{x_0}) - \mathbf{w}_{x_1}(\xi_{x_0})) + \mathbf{O}(\epsilon^3). \tag{A.14}$$

Therefore,

$$\lim_{\epsilon\to 0^+} \frac{x_4 - x_0}{\epsilon^2} = \mathbf{v}_{x_0}(\eta_{x_0}) - \mathbf{w}_{x_1}(\xi_{x_0}) = [\mathbf{v}, \mathbf{w}]_{x_0}, \tag{A.15}$$

which means that the Lie bracket $[\mathbf{v}, \mathbf{w}]_{x_0}$ is the limiting direction, obtained at x_0, when the sequence of controls defined by (A.7) is applied to a system of the form (A.6).

A.9 The controllability rank condition

This section follows [Hermann and Krener, 1977].

Consider a system of the form

$$\dot{x} = f(x, u) \tag{A.16}$$

where $x \in M$ is the state variable with values in the smooth connected manifold, M, of dimension m, f is a smooth function and $u \in \Omega$ with $\Omega \subset \mathbf{R}^l$, $l \leq m$, representing the control variables.

Each constant control $u^i \in \Omega$ defines a vector field $f^i = f(x, u^i)$ in the manifold's tangent space at x, denoted by TM_x. Denote by $\mathcal{F}^0$ the subset of all these vector fields. Let $\mathcal{F}$ denote the smallest subalgebra of the Lie algebra of vector fields on M that contains $\mathcal{F}^0$. A typical element of $\mathcal{F}$, evaluated at x, is of the form

$$[f^1, [f^2, [\cdots [f^{k-1}, f^k] \cdots]]]_x.$$

The system (A.16) is said to satisfy the *controllability rank condition at x* if the dimension of the vector space spanned by the elements of $\mathcal{F}$ at x is m. If this condition is verified for any $x \in M$, then the system (A.16) is said to verify the *controllability rank condition*.

Appendix B

Brief notions on Viability Theory

The material in this appendix is from [Aubin, 1991], Chapter 1. The notation is kept close to the original.

Consider the system with state $x(t) = (x_1(t), \dots, x_n(t))$ defined by

$$\dot{x} = f(x, u), \quad x(t_0) = x_0, \tag{B.1}$$

where $u = (u_1, \dots, u_m)$ represents the controls of the system.

In control problems, instead of focusing on the control and constraints, it is sometimes advantageous to consider the set of permissible velocities or directions, that is, sets of the form

$$F(x) = \{y \in \mathbb{R}^n : y = f(x, z), z \in \Omega(x)\},$$

where $\Omega(x)$ is the subspace of $\mathbb{R}^m$ (the control space), generated by some feedback controller, such that the velocity of the system verifies

$$\dot{x} \in F(x(t)), \quad x(t_0) = x_0. \tag{B.2}$$

A mapping $\psi : [0, T] \mapsto X$ verifying $\forall t \in [0, T], \psi(t) \in K$, with K a preassigned subset of X, is said to be *viable in K on* $[0, T]$. The set K is said to be *locally viable* under F if, for any initial state $x_0 \in K$, there exists a $T > 0$ and a solution ψ of (B.2), starting at x_0 and defined in $[0, T]$, which is viable in K. The set K is called *globally viable* under F if $T = +\infty$.

The control of (B.1) can be stated as a viability problem by: given some set K, determine the mapping F such that K is *viable* under F. Alternatively, this corresponds to finding, at each point

$x \in K$, a direction $v \in F(x)$ that is "tangent to K"[1], i.e., it pushes the state to the interior of K, rending $x(t)$ viable in K. The concept of tangency used is the so called *Boulingand contingent cone*, defined by

$$T_K(x) = \left\{ v \in X : \lim_{h \to 0+} \inf \frac{d_K(x + hv)}{h} = 0 \right\} \tag{B.3}$$

where $d_K(y)$ denotes the distance of y to K, defined by

$$d_K(y) = \inf_{z \in K} \|y - z\|.$$

The control of the system (B.1) requires to the definition of a sequence of small time steps, $h_n \to 0$, and a sequence of motion directions, $v_n \to v$, such that

$$\forall n \geq 0, \;\; x + h_n v_n \in K. \tag{B.4}$$

By Nagumo's theorem, [Aubin, 1991], pp. 27, this form of convergence corresponds to K being a *viability domain of f*, expressed by

$$\forall x \in K, \;\; F(x) \in T_K(x).$$

[1]Here the original was quoted.

Appendix C

Mobile platforms

C.1 The cart mobile platform

This appendix presents the model of the cart, or unicycle, mobile platform. The kinematics is described by the model.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{C.1}$$

where x, y are the position coordinates in the plane and θ the orientation (see Figure C.1).

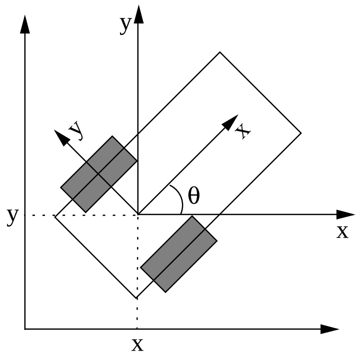


Figure C.1: Variables for the cart mobile platform.

C.2 The car-like mobile platform

This section presents the kinematics of a car-like mobile platform, derived from the kinematics of the cart robot, expressed by (C.1).

Figure C.2 details the reference frame assignment and the main variables for the car like robot.

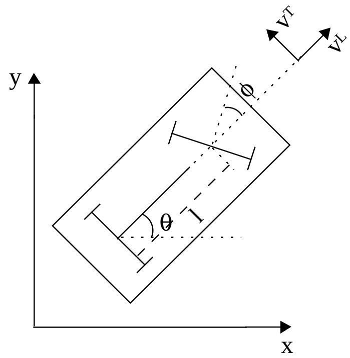


Figure C.2: Variables for the car-like mobile platform.

From the figure,

$$\begin{aligned}&\dot{\theta} = \omega\\&\omega = v^T/l\\&v^T = \tan(\phi)v^L\\&v = v^L\end{aligned}$$

this leading to the model

$$\begin{bmatrix}\dot{x}\\\dot{y}\\\dot{\theta}\\\dot{\phi}\end{bmatrix} = \begin{bmatrix}\cos(\theta) & 0\\\sin(\theta) & 0\\\tan(\phi)/l & 0\\0 & 1\end{bmatrix}\cdot\begin{bmatrix}v\\\Omega\end{bmatrix}. \tag{C.2}$$

The new input Ω has the meaning of the wheel turning rate, while v maintains the same meaning of the cart robot.

C.3 Control of a cart mobile platform

This section details the principles of the control algorithm used to implement the state composition operation described in Chapter 4.

Model (C.1) assumes that the robot's reference frame is located in the middle point between the two wheels, with the x-axis pointing in the forward direction. Using this model the control variable ω does not have a direct effect in two of the directions of the configuration space, x and y, thus rending difficult the design of a controller. One way to circumvent this problem is to define a point to be controlled other than the origin of the reference frame.

Consider the Figure C.3, representing the reference frame for the mobile platform described by (C.1).

The new point to control is $p = (p_x, p_y)$, with p_x and p_y referred to the mobile platform reference frame. Since the robot is a rigid body, the rotational velocity in the point p is identical to the one

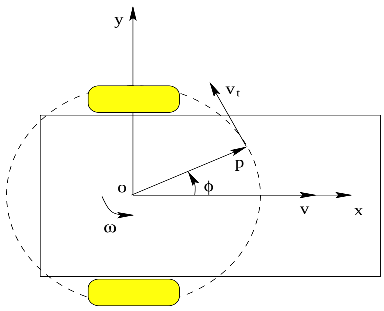


Figure C.3: Frame assignment and main variables for the cart control problem.

of the origin of the reference frame. The linear velocity of p, written in the cart reference frame and denoted by $\mathbf{v}_p$, is described by

$$\mathbf{v}_p = \mathbf{v}_o + \mathbf{v}_t = \mathbf{v}_o + \mathrm{Rot}(z, \phi) \begin{bmatrix} o \\ \omega \cdot \|p\| \end{bmatrix} \tag{C.3}$$

where $\mathbf{v}_o$ is the linear velocity of $(0,0)$, $\mathbf{v}_t$ represents the contribution of the angular velocity of the cart to the linear velocity of the point p and $\mathrm{Rot}(z, \phi)$ represents a rotation of ϕ around the z axis.

The transformation of the velocity $\mathbf{v}_p$ to the world frame is made through the rotation of an angle θ around the z axis, yielding

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}_p = \mathrm{Rot}(z, \theta) \cdot {}^o\mathbf{v}_p = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} v - \sin(\phi)\omega\|p\| \\ \cos(\phi)\omega\|p\| \end{bmatrix} \tag{C.4}$$

which, after some manipulation, and the introduction of the θ state variable becomes

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}_p = \begin{bmatrix} \cos(\theta) & \sin(\theta + \phi)\|p\| \\ \sin(\theta) & \cos(\theta + \phi)\|p\| \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{C.5}$$

For $\|p\| = 0$ the original model (C.1) is recovered. Clearly, the submatrix

$$A = \begin{bmatrix} \cos(\theta) & \sin(\theta + \phi)\|p\| \\ \sin(\theta) & \cos(\theta + \phi)\|p\| \end{bmatrix} \tag{C.6}$$

is always invertible. Therefore, it is always possible to determine a control (v, ω) that steers the point p in the mobile platform through a reference motion direction. Denoting the reference motion direction by $(\dot{x}_{\mathrm{ref}}, \dot{y}_{\mathrm{ref}})$, the control law is given by

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = A^{-1} \cdot \begin{bmatrix} \dot{x}_{\mathrm{ref}} \\ \dot{y}_{\mathrm{ref}} \end{bmatrix}. \tag{C.7}$$

The computation of the trajectory of the robot's reference frame origin is straightforward from the trajectory obtained for point p.

Appendix D

Common problems in the robots used

This appendix describes some problems arising in the real cart mobile platform and the manipulator used in the real setup considered.

In what concerns the mobile platform, the first situation illustrates the problems at low speeds. In Figure D.1 the mobile platform is commanded to make a circular trajectory, with 1m radius, linear velocity 0.1 m/s and angular velocity 0.1 rad/s. The circumference shows the trajectory obtained. The quarter of circumference, with 1 m radius, is displayed for comparison purposes. The error displayed is caused by the differences in the dynamics of each of the motor axis of the mobile platform. The same reference causes different responses from each of the two axis.

The circumference in Figure D.2 represents the trajectory obtained when the mobile platform is commanded to perform a circular trajectory with radius 0.5 m but using a higher angular velocity than in the previous situation. A quarter of circumference is also displayed for comparison purposes. Even though the smaller radius of the reference trajectory, the difference between the real and the reference trajectory diminishes with the increase in the control values.

Figure D.3 illustrates the effect of this problem, in a trajectory composed of two arcs of circumference, using the same velocity values of the first experiment. The mobile platform starts at $(x, y) = (0, 0)$ and it must reach the point $(0.8, 0.7)$, after which it must switch to another circular trajectory to reach the point $(1, 1)$. The differences in the dynamics of the axis, together with the processing time necessary to the acquisition of the odometer data and comparison with the reference position, introduce a small delay that causes the large deviation shown.

Another source of problems respects the usable range of the control space. For instance, when specifying a small velocity reference, lasting for a short time interval, the dynamics of the mobile platform induces a dead-zone effect, i.e., the mobile platform is not able to follow short duration

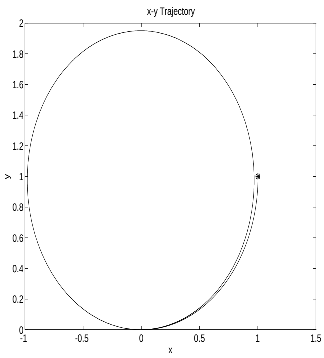


Figure D.1: Circular trajectory ($v = 0.1$ m/s, $w = 0.1$ rad/s).

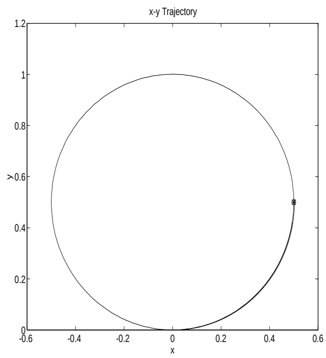


Figure D.2: Circular trajectory ($v = 0.1$ m/s, $w = 0.2$ rad/s).

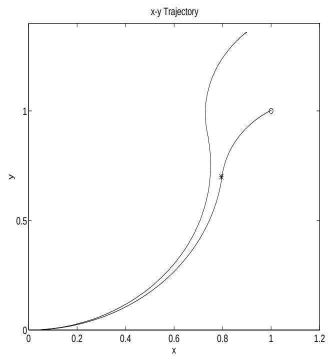


Figure D.3: 'S' trajectory ($v = 0.1$ m/s, $w = 0.1$ rad/s).

velocity profiles[1]. This error source must also be accounted for when designing the actions. For instance, the **GoStG** action defined in the Chapter 4 automatically stops the mobile platform whenever the estimated velocity for the task execution goes bellow a pre-specified threshold (the value of 8 cm/s was used in all of the experiments presented in the thesis).

In what concerns the execution of a task, the maximum of 0.2 rad/s for the angular velocity of the mobile platform, in a pure rotation, is a reasonable value, as higher values tend to generate wheel slippage. The 0.15 rad/s value for the maximum angular velocity of every joint in the manipulator is a low value when compared with the 1.4 rad/s limit indicated by the manufacturer. This value represents a tradeoff, achieved after multiple tests, accounting for the mechanical fatigue of the structure and of the actuators. For instance, long time operation at 0.15 rad/s of the gripper joints leads to the overheating of the actuators with consequent damages. However, from the point of view of a real application, the 0.15 rad/s value is clearly too low. This kind of problems is emphasized in cooperation tasks, where the agents tend to be heavily solicited, and must also be accounted for in the design the actions.

[1]For the Robuter mobile platform, the minima are: 0.05 m/s and 0.1 rad/s for the velocities and 0.001 m/s^2 and 0.001 rad/s^2 for the accelerations.

Appendix E

A trajectory generator for a cart mobile platform

This appendix presents a set of results obtained with a trajectory generator for the cart robot with limited turning radius, supported in a finite number of trajectory elements and assuming no constraints on the controls.

From the model (C.1) results that straight lines and arcs of circumference can be easily generated by using constant controls. Moreover, it can be easily shown that these two types of actions are sufficient for the point-to-point control of the cart: by using the control variable ω any reference orientation, θ, can be reached at a position such that the value of the control variable v can be computed, to lead the robot to the reference position.

Table E.1 defines the main actions used by the trajectory generator of the unicycle robot (the null action can be easily implemented by null control values, thus being omited). Table E.2 defines a non-minimal set of action sequences used to control the robot (the state composition notation is used, though no initial state configurations are indicated).

a_R	Pure rotation
a_{FT}	Forward straight line
a_{BT}	Backward straight line
a_{CT}	Arc of circle

Table E.1: Trajectory generator basis actions.

$a_{FT} \circ a_R$	$a_{BT} \circ a_R$	$a_R \circ a_{FT}$	$a_R \circ a_{BT}$	$a_{FT} \circ a_{CT} \circ a_R$
$a_{FT} \circ a_{CT}$	$a_{CT} \circ a_{FT} \circ a_R$	$a_C \circ a_{FT}$	$a_{BT} \circ a_{CT} \circ a_R$	
$a_{BT} \circ a_{CT}$	$a_{CT} \circ a_{BT} \circ a_R$	$a_{CT} \circ a_{BT}$	$a_{2CT} \circ a_R$	

Table E.2: Action sequences for the control of the cart mobile platform.

Figure E.1 shows some examples of trajectories generated with the above set of actions with the specified configurations displayed in Table E.3. Each configuration is represented by a vector defined by the line segment between the symbols o and $+$. Examples 1 and 2 use maneuvering. As can be seen in the experimental results, some of the action sequences in this predefined set generate maneuvers. The criterion defining whether or not maneuvers are acceptable depends on the available space in the environment. By comparing examples 1 and 3 it is easily seen that maneuvering may require a large area in the workspace. If smooth trajectories are required, as in Examples 4, 5 and 6, maneuvering can be avoided but also at the expense of spanning some amount of workspace.

	Example 1	Example 2	Example 3
q_0	$1.31, 1.46, 0.2)$	$(1.17, 0.88, 0.56)$	$(1.35, 1.09, 0.28)$
q_1	$(1.52, 2.63, -0.62)$	$(1.39, 1.87, -0.33)$	$(1.74, 2.13, -0.29)$
q_2		$(2.25, 2.4, \pi/2)$	
q_3		$(2.78, 3.85, 0.17)$	

	Example 4	Example 5	Example 6
q_0	$(1.48, 1.6, 0.11)$	$(1.26, 0.82, 0.02)$	$(1.33, 1.47, 0.04)$
q_1	$(2.03, 3.79, -0.21)$	$(1.89, 1.68, -0.19)$	$(1.33, 3.41, 0.01)$
q_2		$(2.75, 2.46, 1.54)$	$(2.38, 1.74, -3.11)$
q_3		$(1.9, 3.4, 2.73)$	$(2.46, 3.8, -3.13)$
q_4		$(1.66, 4.51, 0)$	
q_5		$(3.74, 4.54, -0.55)$	
q_6		$(4.51, 2.97, -1.58)$	

Table E.3: Origin, intermediate and ending configurations for the represented examples.

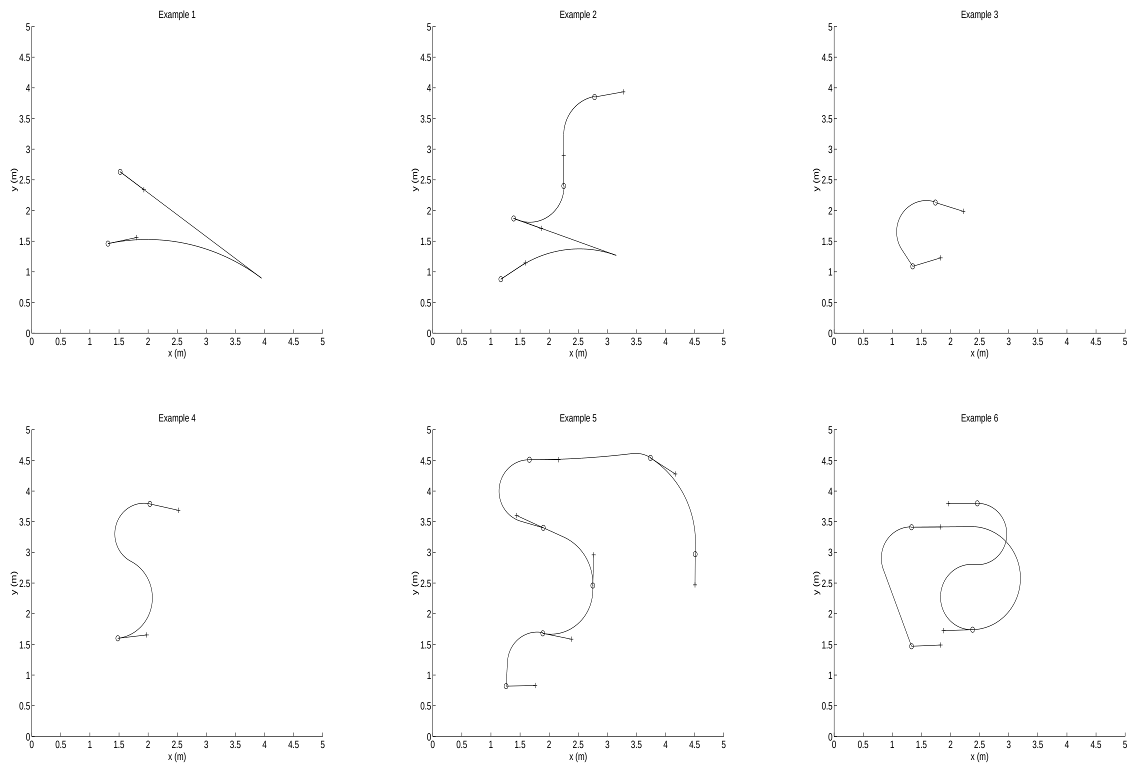


Figure E.1: Examples of trajectories for the unicycle robot.

Appendix F

Experimental results MPEG 2 video files

This appendix presents a list of available MPEG 2 video files illustrating some of the missions included in Chapter 5. The files can be requested by electronic mail to jseq@isr.ist.utl.pt or accessed through the World Wide Web site of the IST/ISR Mobile Robotics Laboratory (URL http://www.isr.ist.utl.pt/lrm), under Project COOPERA. Should any difficulty arises in the viewing of this page please contact through the referred e-mail address.

In each of the movies, the changes in the viewing perspective are made preserving the system configuration in order to minimize the visual disturbance.

Section	Comment	MPEG file name	Setup	Size (Kb)
5.5.3	Door traversal	FULL_DT	Simulated	261
5.7.2	Robustness test	ROBUSTNESS	"	222
5.7.4	Easy warehousing	EASY_WARE_1	"	202
5.7.4	Easy warehousing	EASY_WARE_2	"	200
5.4.2	Button pushing	PUSHBUTTON_sim	"	131
5.4.2	Button pushing	PUSHBUTTON_real	Real	191
5.7.4	Easy warehousing	EASY_WARE_3	Simulated	199
5.7.4	Crossing	CROSSING_1	"	237
5.7.4	Crossing	CROSSING_2	"	317
5.7.4	Crossing	CROSSING_3	"	325
5.8	Bar through a corridor	BAR_CORR	"	374

Table F.1: MPEG2 files.