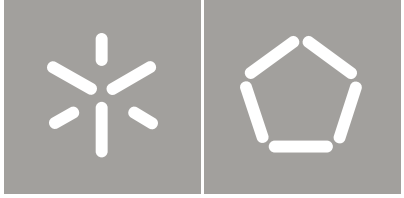


Universidade do Minho
Escola de Engenharia

Tiago Emanuel Quintas Malheiro

Object Transportation Task by a Human and a Mobile Manipulator: a non-linear attractor dynamics approach

Master Dissertation
Master in Industrial Electronics and Computers Engineering

Dissertation work done under the scientifc orientation of
Professor Estela Guerreiro Silva Bicho Erlhagen

October 2011

DECLARAÇÃO

Nome: Tiago Emanuel Quintas Malheiro

Correio electrónico: tmalheiro@dei.uminho.pt

Tel./Tlm.: 969855241

Número do Bilhete de Identidade: 13369187

Título da dissertação:

Object Transportation Task by a Human and a Mobile Manipulator: a non-linear attractor dynamics approach
Ano de conclusão: 2010/2011

Orientador(es): Professora Doutora Estela Guerreiro Silva Bicho Erlhagen

Designação do Mestrado:

Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia Eletrónica Industrial e Computadores

Área de Especialização: Automação, Controlo e Robótica

Escola de Engenharia – Universidade do Minho

Departamento de Eletrónica Industrial

Guimarães, ___/___/_____

Assinatura: _____

Acknowledgments

I wish to leave here my gratefulness to the people who aided me in writing my dissertation.

My thanks and appreciation go to Professor Estela Bicho, who stirred up my interest in this issue and for helping me with my work with patience throughout the time it took me to complete my research and write the dissertation.

To Doctor Sérgio Monteiro, thank you for your valuable and institutional support.

I wish to thank to Toni Machado and Miguel Sousa, whose extensive technical and moral support were fundamental during the research, especially in the experiments with the robot, and in the writing of the dissertation.

I am grateful to my Mobile and Anthropomorphic Robotics Laboratory colleagues and friends Luís Louro, Rui Silva, Flora Ferreira, Eliana Silva and Emanuel Sousa for helping me in many ways, above all, creating a good working environment.

I wish to thank to my parents and family for encouragement during all my studies and for always be there for me.

Finally, a singular thank you to my friends, especially Vitor Silva, who gave me a continuous moral support, and Tânia Peixoto, whose kindness and endless support I will never forget.

Resumo

A interação humano-robô na execução de tarefas cooperativas tem uma forte aplicabilidade no nosso quotidiano. Diversas tarefas, entre as quais o transporte de objetos de grandes dimensões, podem beneficiar da associação entre a inteligência humana e a velocidade e destreza dos robôs. Estes sistemas robóticos devem ser capazes de cooperar autonomamente com humanos, apresentando um comportamento suave e tendo em consideração as ações do humano.

Nesta dissertação, pretendemos desenvolver um manipulador móvel autónomo, capaz de transportar cooperativamente um objeto longo com um humano até uma localização alvo. Assim como em muitas tarefas cooperativas entre humanos, a comunicação verbal e não-verbal constituí um aspeto fundamental para o sucesso da tarefa em questão, especialmente quando esta é executada num ambiente não estruturado, contendo vários obstáculos, tanto estáticos como dinâmicos.

Inicialmente é feita uma descrição do *hardware* e do *software* desenvolvidos que constituem o manipulador móvel. De seguida, propomos uma aproximação ao problema do transporte de objetos por um humano e um manipulador robótico tendo por base a abordagem dos Sistemas Dinâmicos à Geração de Comportamentos. A teoria de sistemas dinâmicos não lineares é utilizada para a conceção e implementação do comportamento do robô sendo que a direção e a velocidade do movimento são obtidas a partir de soluções atractoras dos sistemas dinâmicos. Adotamos uma estratégia líder-seguidor para a tarefa cooperativa de transporte de objeto, onde o humano (líder) procura o melhor caminho para a localização alvo enquanto o robô deve assistir o humano no transporte e ao mesmo tempo ser capaz de se desviar de obstáculos.

Os resultados obtidos demonstram que o robô é capaz de gerar o seu comportamento e este é suave e robusto mesmo lidando com a incerteza que caracteriza a maioria dos ambientes. Toda a informação é recolhida pelo seu sistema sen-

sorial. Quando o robô não consegue acompanhar um determinado movimento do humano (como entrar em passagens estreitas ou velocidades elevadas), este é capaz de alertar verbalmente o humano para a situação.

Abstract

Human-Robot interaction in cooperative tasks has a large application in our everyday life. Several tasks, such as transporting a long object, may benefit from the association of human intelligence and robot speed and dexterity. Such robot systems must autonomously cooperate with the human, exhibiting a smooth behavior and taking into account the human's actions.

In this dissertation, we aim to develop an autonomous mobile manipulator able to cooperatively transport a long object with a human partner to a goal location. As in most human-human cooperative tasks, verbal and non-verbal communication is a key feature to the success of the task in hand, specially when it is performed in an unstructured and cluttered environment, with static and dynamic obstacles.

We begin with a description of the hardware and software developed that constitute the mobile manipulator. Later, we propose an approach to Human-Mobile manipulator object transportation based on the Dynamical Systems approach to Behavior Generation. Non-linear dynamical systems theory is used to design and implement the robot's behavior. The time course of robot's heading direction and path velocity are obtained from attractor solutions of the dynamical systems. We adopt a leader-follower strategy to the object transportation task, where the goal location is given to the human (leader) and the robot should assist the human while, at the same time, being able to avoid obstacles.

The results show the ability of the robot to generate its own behavior, based on the information from the environment, which is gathered by the robot's own sensory system while the task is executed. The robot's navigation is very smooth and robust when dealing with uncertainties of the environments. Furthermore, the robot is able to verbally express itself when it can not perform certain movements required by the human (such as a narrow passages or high velocities).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Problem statement

Human-Robot cooperation is one of the key technologies to broaden the application field of robots. Using the strengths of both human and robot the system will be effective in more elaborated tasks where robots used to be inapplicable. More explicitly, this system may benefit of human intelligence and problem solving skills and robot speed and dexterity to perform autonomously a task where only a human-human partnership or a specific designed robot system would be able to perform (Green et al., 2008).

The combination of this opportunity and the recent development of robot technologies is taking several researchers to work towards robots that exhibit a smooth behavior and that take into account the human actions. Leading robots to acquire a place in humans environment and cooperate autonomously with them accomplishing many different tasks.

One of the problems addressed in Human-Robot cooperation is object transportation (Lawitzky et al., 2010; Bicho et al., 2003; Takubo et al., 2002). Large or long objects are usually difficult to transport by grasping only a single point. The task may be enhanced if a human holds one end of the object and a robot the other one. The two partners may this way easily transport the object from one point to the destination point without the frustration felt by a human when such large, long or irregular objects are transported by grasping near the object's center of mass.

Figure 1.1: Human-Robot object transportation

In the problem addressed in this dissertation, see Figure 1.1, the human brings intelligence and experience to the task, global task knowledge, as goal and path planning, while the robot must help the human to transport the object avoiding static and dynamic obstacles. At same time, the robot should interact with the human to express its intentions, and alerting the human partner when some proposed movement is not feasible to be performed by the robot.

In order to perform such cooperative transportation task, a mobile manipulator, instead of a simple mobile robot as in Bicho et al. (2003), may bring some enhancements to the task. First, combining a mobile platform and a multi-link manipulator creates redundancy, allowing that a particular movement of the human partner may be followed by the robot moving the manipulator, moving the mobile platform or a combined motion of both. Second, because of slow dynamic response of the mobile platform, the manipulator introduces more dynamic interaction between robot and human in cooperative transportation. Third, it enables the robot to fetch the object to transport, and lower it down in the destination without the need of a third partner (human or robot) to do it.

Despite mobile manipulators offer a tremendous potential for object transportation and other tasks, they bring about a number of challenging issues rather than simply increasing the structural complex of it. From the point of view of the cooperating robot (i.e. mobile manipulator) the environment, which consists of the manipulated object, the human and world scenario (static or dynamic), exhibits complex dynamic behavior. The problem is exacerbated when the environment is unknown and changing, since decisions must be made on-line and according to those changes.

Several, and different, approaches have been proposed to solve these problems. Some of the most relevant reported results in the literature include the use of passive robotics concepts (Goswami et al., 1990; Hirata and Kosuge, 2000; Fukaya et al., 2006), compliant motion (Hogan, 1985; Yamamoto et al., 1996), virtual nonholonomic constraints (Takubo et al., 2002), load sharing (Choi et al., 1992; Lawitzky et al., 2010), Human intention recognition and motion estimation (De Carli et al., 2009; Maeda et al., 2001) and behavioral-based approaches (Bicho et al., 2003).

It seemed a reasonable requirement that autonomous mobile manipulators be reactive to dynamic aspects of the environment and be able to generate robust behavior in the face of uncertain sensors, unpredictable environment, and a changing world. Making human-robot collaboration natural and efficient is crucial and is the major issue concerning the control of the robot under such conditions. The movements have to be smooth and the robot must exhibit a behavior that is aware that its movement may influence its partner movement, since humans have to interact with the robot transporting the object without letting it to fall. Behavior-based approaches have become the dominant methodologies for designing control schemes for robot interaction with the environment. One of them is the so called Dynamic Approach to Behavior Generation (Schöner and Dose, 1992; Schöner et al., 1995; Bicho and Schöner, 1997a; Steinhage, 1997) which is based on the mathematical theory of non-linear dynamics. It provides a theoretical framework and tools that allow designing a control architecture that generates the behavior of the cooperating robot.

This approach has been extensively and successfully implemented in mobile robot navigation (Bicho, 2000; Large et al., 1999; Althaus, 2003), multiple robots coordination for object transportation and formations (Soares and Bicho, 2002; Soares et al., 2007; Soares, 2007; Monteiro and Bicho, 2010), mobile manipulator navigation (Ellekilde and Christensen, 2009), and Human-Mobile robot object transportation (Bicho et al., 2003).

Results have shown that robot navigation is very smooth and robust when dealing with uncertainties of the environment. Furthermore, a mobile robot was able to successfully transport an object with a human. These results lead us to believe that the dynamical systems approach is especially well suited for mobile manipulator control in the context of Human-Robot cooperative transportation task.

1.2 Previous Work

Particularly focused on human safety, passive robotics concepts have been proposed by Goswami et al. (1990), in which systems move passively according to an external force without using any joint actuator. Using the concepts of passive robotics, Fukaya et al. (2006) introduced a concept of passive robotics for realizing a physical interaction between Human and Robot safely on an object transportation system. Despite the continuous work on that concept (Hirata et al., 2006, 2009, 2011), there are some situations were such passive robots can not generate enough force for supporting human's motion. The necessary force to keep the object on track may have a certain direction and magnitude such that the passive robot does not have enough brake units to generate the desired force. Wannasuphoprasit et al. (1997) and Lynch and Liu (2000) studied the design of passive guide constraint, which assists the human in manipulating a load. In these approaches, the guide confines the load to a one degree-of-freedom curve in its configuration space.

Hirata and Kosuge (2000) proposed distributed robot helpers, referred to as DR Helpers, and a decentralized control algorithm for them to transport a single object in cooperation with a human/humans.

The distributed robot helpers are multiple autonomous mobile robots, which were developed for enabling a human to move a heavy or large object without any human assistant. In the algorithm proposed, each robot is controlled as if it has caster-like dynamics and interacts with the human through a intentional force/moment that the human applies to the object. Based on this intentional force/moment, the robot transports the object in cooperation with the human. This algorithm is designed based on the passivity-based control system to guarantee the stable realization (Hirata and Kosuge, 2000) of human-robot interaction through a manipulated object.

Nevertheless, in a system using the proposed approach by Hirata and Kosuge (2000) a human could not easily manipulate a large object if each robot is controlled passively along all directions (Hirata et al., 2001). The authors pointed out that the human may not be able to apply a moment precisely to a grasping point of the human to adjust the orientation of the object. Furthermore the human would also have to apply intentional force/moments to the object precisely to transport it to the destination avoiding obstacles (Hirata et al., 2001; Arai et al., 2000).

To overcome this problem, Hirata et al. (2001) suggested a system where each robot has a map information of the environment where it will be deployed. Under the assumption that this system is used in a known environment such as an office, a hospital and a home, etc., each robot could generate a path on the known environment and move along with a path velocity based on the intentional force applied by a human. In this system the human could transport an object easily together with multiple robots avoiding static (known) obstacles. In addition, the human would not have to apply a moment to the object to change the orientation of the motion.

Despite the good performance in human-robot object transportation reported when robots are controlled based in such approaches, the applicability of that systems is very limited. First, forcing a robot (or team of robots) to follow a pre-specified path does not allow changes in the specified task such as a new destination point. More critical, since these systems are intended to "live" in human environments, it are not able to avoid dynamic or unknown static obstacles, which may imply the failure of the task execution. Second, changes to the environment would have to be represented in each robot accordingly. Third, such approach requires some mechanism for position estimation as dead reckoning, which may accumulate positioning errors and lead robot out of desired path.

Compliant motion has been addressed by several researchers (Hogan, 1985; Al-Jarrah and Zheng, 1997a; Ikeura and Inooka, 1995) as an approach for robot-environment and human-robot interaction where both the dynamic behavior and the position of the manipulator are controlled based on the concept of mechanical impedance (Hogan, 1985).

Ikeura et al. (1994) investigated the human characteristics in a task in which two humans cooperated with each other in carrying an object. It was shown that the damping factor is important for cooperation when the human characteristics are approximated by impedance control.

It was shown that in human-robot cooperation based on impedance control, the human can not operate as quickly as in human-human when the damping factor is high. However, when the damping factor is low, the human can perform the task quickly but the positioning operation is not stable. This problem arises from the fact that conventional control uses constant impedance. Ikeura and Inooka (1995) applied these results to a human-robot cooperation task. The impedance characteristics were changed according to the speed of the motion and it was verified that

the human was able to perform quick actions and with the same controller having a positioning action without oscillations. From this, better cooperation characteristics are given to the robot when the control is based on variable impedance control instead of conventional control based on constant impedance.

Al-Jarrah and Zheng (1997a) proposed a reflexive motion control as a coordination mechanism for a manipulator to share the load with the human arm. Reflexive motion is an inspiration of biological systems where muscular activities are regulated via reflexes. The authors applied this approach to a manipulator to reduce the strain on the human arm and to increase the speed of the manipulation on the object.

Takubo et al. (2002) proposed a virtual nonholonomic constraint approach in order to solve some problems identified in impedance control. As stated before, Kosuge et al. (1993); Ikeura and Inooka (1995); Al-Jarrah and Zheng (1997a) proposed impedance control based approaches. However, it is difficult to apply a large moment at the end of a long object and the force the operator exerts is mainly translational. Though axial translation is easy, the rotation and normal translation may cause side-slip of the object and complicate the manipulation (Takubo et al., 2002; Arai et al., 2000). To solve this problem, Takubo et al. (2002) proposed a method that suppresses the side-slip of the object by assigning a virtual constraint at the robot hand (the translational velocity at the robot hand is restricted only in the axial direction), which is equivalent to a wheel being attached to the object in the axial direction. With this approach the human may intuitively understand the object's behavior because the direction of mobility of the object is restricted by the nonholonomic constraint, which implies the human to steer the manipulation as a cart or wheelbarrow.

Yamamoto et al. (1996) proposed an approach which actively utilizes the locomotion and manipulation of a mobile manipulator to transport a large object cooperatively with a human. Here, no a priori planning was assumed by the robot and the human takes the lead while the mobile manipulator supports the object jointly and following the human. The approach preferred operating region was adopted by the authors as a coordination mechanism between the platform and the manipulator. The control of the manipulator is achieved by two parts: force control in inertial z-axis direction to sustain the object, actively compensate the dynamics of the manipulator in xy-plane and make the manipulator free-floating inside the plane. This way, the manipulator will be dragged to move in xy-plane

by the friction force between the end effector and the object. They have used the manipulability measure of the manipulator as a criterion for optimal coordination of both platform and manipulator motions. However, the necessary force to move a joint of the manipulator may be to high and supersedes the friction force between the end-effector and the object not allowing that the manipulator even moves. If the manipulability of the manipulator does not change, the task may not be executable since the platform coordinates the movement based on that measure. Furthermore, manipulator joints may exert different forces when it moves in one direction and then in the opposite, which may lead the manipulator to a constant low manipulability state.

Pereira et al. (2010) proposed an approach to transport an object by a mobile nonholonomic robot in coordination with a human using a non-linear controller. Instead of a conventional force/moment sensor at the robot end-effector, they used two infra-red sensors to measure the displacement on the support of the object to be carried. In this approach, the object size is assumed to be known a priori, which may not be always available or not practical to be acquired when the task may be executed several times with different object sizes.

Another important issue in human-robot cooperation is the load sharing problem. Several researchers studied the load sharing problem in the dual manipulator coordination (Choi et al., 1992; Kim and Zheng, 1991). However, to the success of the task, the motions of the manipulators are assumed to be known. This is impractical in human-robot cooperation since no predefined path is given, therefore object motion may be unknown to the robot.

To overcome this problem, Al-Jarrah and Zheng (1997b) proposed a load sharing method using variable compliance control. Using information of the force/moment from the end-effector as a measure to change the impedance of the compliant control, the authors verified that the load sharing between a manipulator and a human may be improved. Increasing the impedance of the manipulator will result in increasing the force being exerted by the operator. The desired behavior is then to decrease impedance, which will result in more force being exerted by the manipulator.

Lawitzky et al. (2010) proposed a systematic derivation of the effort sharing policies from a system-theoretic analysis of the joint object manipulation task under environmental constraints. They have considered three different policies: Balanced-effort behavior; Maximum-robot-effort behavior and Minimum-robot-effort behav-

ior. This parametrization of load sharing policies with respect of their degree of assistance allows the tuning of the pro-activity of the robot. Besides the overall performance of the suggested approach, a commonly known trajectory of the object configuration was assumed, which is impractical in real-world applications since most of robot tasks are subjected to uncertainty.

Different approaches for human-robot cooperation have also been proposed based on Human intent measuring and recognition (De Carli et al., 2009; Fernandez et al., 2001) and motion estimation (Maeda et al., 2001).

De Carli et al. (2009) proposed an impedance-based approach with user intent measuring feature. The user can induce the robot to adopt a new path by pushing the load sufficiently far off the current path such that the magnitude of force that should be applied by the robot to keep the load on path is higher than a preselected threshold. This makes the robot to "store" the new path direction and change manipulator movement accordingly to human intent. When this occurs, a new path is computed and, regardless of the particular path adopted, at the time of adoption, the new path will be precisely the one intentioned by the user which makes the robot applied force to decrease. This sudden change will be apparent to the user. They propose to take advantage of this response (measuring it by physiological biometric sensors) in order to confirm the user's recognition that the path has changed.

Fernandez et al. (2001) introduced an intention recognition approach for active human-mobile manipulator cooperation. Like others, the human is the leader and the robot the follower. The interaction between human and robot is based on force/moment sensor. When using impedance control based approaches, the human has to continuously exert a certain force to maintain the movement. Through intention recognition the robot cooperative behavior can be active, reducing the human effort and makes transportation faster. The authors proposed an intention recognition mechanism based on the search for spectral patterns in the force signal measured at the manipulator gripper. They have shown that a system designed based on intention recognition demonstrates satisfactory performance. Despite the success, the system needs an improvement in robustness (Fernandez et al., 2001) and robot movement does not integrate obstacles avoidance capabilities.

Maeda et al. (2001) have also proposed an approach for active human-robot cooperation based on motion estimation. Using virtual compliance control as basis of manipulation movement, the authors estimate the position of the human hand

and treat it as the desired position of virtual compliance control. The motion of the human partner is estimated with the minimum jerk model which gives human characteristics to motion estimation and therefore to the overall movement of the manipulator. They have reported that using this approach the human could manipulate the object has intended and the manipulation was not as "heavy" as when they turned off motion estimation. In this approach, no a priori path was given to the robot and the velocity of the movement was left to human intent. However, for simplicity, the task was carried under limited movement of horizontal one dimensional transportation of the object.

Another approach considered in the literature to human-robot collaboration in transportation tasks is the Behavior-based approach. Bicho et al. (2003) proposed a dynamic control architecture, as a Behavior-based approach, based on non-linear dynamical systems, that controls the behavior of the autonomous robot (without a manipulator) that must transport a large size object in cooperation with a human. In the proposed approach, the robot has no a priori knowledge of the environment and no absolute position mechanism is embedded on the robot. The robot navigation is based on information of target orientation relatively to robot heading direction and obstacles orientation and distance. As the sensed world changes, the robot's behavior is changed accordingly. It was shown that the system is robust against perturbations, stable and the trajectories are smooth, while the robot helps the human to carry a long object in a unstructured indoor environment.

In this dissertation we aim to extend the previous work into the domain of Human-Mobile manipulator cooperation, which must integrate verbal and non-verbal communication.

1.3 Scope and Outline of the Dissertation

We first aimed to setup a mobile manipulator,which involves both hardware and software, and later propose an approach to Human-Robot (mobile manipulator) object transportation based on the Dynamical approach Systems Behavior Generation. More specifically, we use a leader-follower approach to control the mobile manipulator behavior so that it cooperatively transports a large object with a human, in unstructured and dynamic environments. The motor control of the platform and manipulator relies on the attractor dynamics approach to behavior-based robotics (Bicho, 2000) where a smooth and stable trajectory should be generated

even in the presence of environment constraints such as avoidance with as static or dynamic obstacles.

This means that non-linear dynamical systems theory is used to design and implement the robot's behavior. Specifically, the time course of the control variables are obtained from solutions of dynamical systems. The attractor solutions (asymptotically stable states) dominate these solutions by design. The benefit is that overt behavior of the robot is generated as a time course of asymptotically stable states, that, therefore, contribute to the overall asymptotic stability of the complete control system and makes it robust against perturbations.

In this work only real robot implementations and experiments were taken into account. One of the important results stressed here is the ability of the robot to generate its own behavior based on the information locally gathered from the environment while the task is executed. Opposing to most of the proposed approaches in literature, the robot does not have a map of the environment, neither a pre-planned path is given to the robot and it is able anyway to successfully avoid unknown static and dynamic obstacles. Another important feature is the attributed capability of self expressing verbally to the human when an intended movement is not feasible to be performed by the robot itself and to give advice on possible movement alternatives.

The remainder of this dissertation is organized as follows:

In Chapter 2, basic principles of non-linear dynamical systems are covered as a basis for behavior generation for the robot's dynamical control architecture presented in Chapter 4.

Chapter 3 provides a description of the mobile manipulator that has been built in the scope of this dissertation. Hardware features of the mobile manipulator, referred here as Dumbo, are presented followed by a exposure of developed software to interface with hardware. Also auxiliary software developed to monitor the robot is briefly presented, and robot kinematics is given as a point to behavioral variables.

Based on principles given in Chapter 2, a Dynamical Systems Architecture is designed in Chapter 4. Target acquisition and obstacles avoidance behaviors are studied. Furthermore, dynamics of path velocity are introduced.

We continue to Chapter 5 where experimental results are reported in a variety of scenarios prepared to demonstrate the robot's overt behavior and support the

proposed Dynamical Architecture developed in Chapter 4. The chapter ends with a summary and discussion of achieved results.

Finally, conclusion and future work are drawn in Chapter 6.

Chapter 2

Theoretical Framework: Non-linear Dynamical Systems

Non-Linear Dynamical Systems approach was introduced as a framework to behavior generation by Schöner and Dose (1992) and Schöner et al. (1995). In these works, a number of concepts were provided as a theoretical language to design autonomous robotic architectures. These concepts are based on the mathematical theory of dynamical systems and two main basic ideas are described here: (1) The concept of *behavioral variables*, which consists of variables that can describe a particular behavior and define behavioral dimensions along which behavior can change. Specific values of these variables correspond to task constraints. (2) The concept of *behavioral dynamics* according to which behaviors are generated as attractor solutions of dynamical systems.

This chapter follows the presentation provided by the chapter entitled *The dynamical approach to behavior generation* in the PhD work *"Dynamic Approach to Behavior-Based Robotics: Design, Specification, Analysis, Simulation and Implementation"* (Bicho, 2000) with the proper authorization of the author Estela Bicho.

2.1 Basic Principles

2.1.1 Behavioral Variables

To design a behavior in the context of the dynamic approach, the first step is to find variables that can describe, parameterize and internally represent the behavior (state of the system). These variables are called behavioral variables. They define behavioral dimensions along which behavior can change. A specific instance of the behavior corresponds to a point in the space of the behavioral dimensions. Behavioral variables must be chosen such that the following requirements are fulfilled:

a) At any time a behavior must be associated with particular values of its corresponding behavioral variables and task requirements must be expressed as values or set of values of these variables.

b) The specified values for a behavioral variable, that express the task, must be independent of its current value.

c) It must be possible to specify the values by the on-board sensors or by another behavioral model (for example a representation system).

d) Finally, and very important, the behavioral variables must enable the design of control systems that impose their values on an effector system.

Now we give an example that aimes to clarify these requirements. In the task of autonomous robot navigation in the plane the movement must be controlled such that locations of obstacles are avoided while a particular target position is reached. To express the behavior that this movement represents the heading direction, ϕ, in the world (i.e. relative to some arbitrary but fixed world axis), is an adequate variable since (see Figure 2.1):

First, task requirements of moving toward the target while avoiding collisions with obstacles can be expressed as particular independent values of the heading direction. The direction Ψ_{tar} represents the orientation at which the target lies from the current view point relative to the world axis, while Ψ_{obs} represents the direction at which the obstacle is seen. Moving toward the target, which is a desired behavioral state, is associated with $\phi = \Psi_{tar}$. Conversely, moving toward an obstacle is associated with $\phi = \Psi_{obs}$, and is of course an undesirable behavioral state.

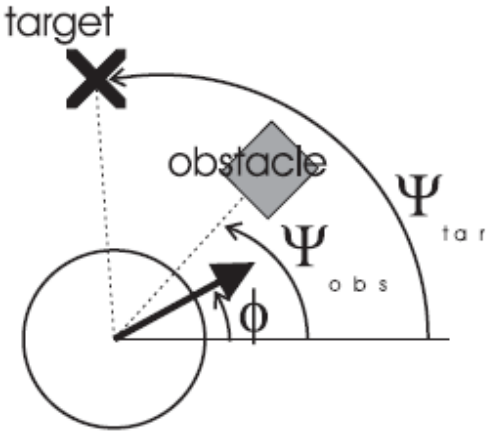


Figure 2.1: An adequate behavioral variable for the task of moving in the plain toward a target location while avoiding to run into obstacles

Second, if the robot turns on the spot the specified values Ψ_{tar} and Ψ_{obs}, either desired or to be avoided, for the heading direction are kept invariant, i.e. the specified values Ψ_{tar} and Ψ_{obs} do not depend on the current value of the behavioral variable (i.e. ϕ). Since the values expressing the task to be performed ($\phi = \Psi_{tar}$ expressing target acquisition and $\phi \neq \Psi_{obs}$ expressing obstacle avoidance) are independent from the current value for the heading direction the individual behaviors can be designed independently from each other.

Third, on-board sensors may specify the values Ψ_{obs} and Ψ_{tar} as long as an estimate of the current orientation of the robot in the world is maintained (in reality, the correct calibration of this value is not fundamental as long as the calibration drift is slow).

Finally, the heading direction can be easily controlled by providing incremental commands to the vehicle's motors or to a steering module.

Path velocity and angular velocity are also appropriate behavioral variables for the example given above (Neven and Schöner, 1996; Bicho and Schöner, 1997a,b; Bicho et al., 2000, 1998). This will become clear as we present the material in this dissertation.

2.1.2 Behavioral Dynamics

The next step is to generate values for the behavioral variables in time, which control the robot's action. For this purpose a dynamical system for the behavioral variables is designed. Mathematically such dynamical system is time-continuous and is defined by a differential equation in which the dynamical state variables are the behavioral variables. For example, for the heading direction, ϕ, a dynamical system defines the temporal rate of change of the heading direction as a function of the current value, i.e.

$$\frac{d\phi(t)}{dt} = f(\phi(t), \text{parameters}) \tag{2.1}$$

The function $f(\cdot)$ defines a *vector field*; i.e. to each point in the state space it assigns a vector $f(\phi)$. Each of these vectors determine the direction in which and the rate with which the system will move from the point where the vector is anchored.

Fixed points: Attractors and Repellers

As a design principle, we are interested in a particular type of solutions of dynamical systems called *fixed points* or *equilibrium solutions*. These are the points at which the vector field is null,

$$\frac{d\phi(t)}{dt} \,|\phi = \phi_{\text{fixed point}} = f(\phi_{\text{fixed point}}) = 0 \tag{2.2}$$

Fixed points are, in other words, constant solutions of the dynamical system: The system does not change state in time. But the system being "stuck" in a state does not mean that it is stable. This is depicted in Figure 2.2.

For a dynamical system in one variable the stability of the fixed points can easily be investigated graphically. This is illustrated in Figure 2.3. Panels **A** and **B** in Figure 2.3 depict a linear and a non-linear dynamical system, respectively. These two dynamical systems exhibit a fixed point at $\phi = \phi_A$. Because the slope at this fixed point is negative, the fixed point is an asymptotically stable state. In this case the fixed point is also called an *attractor* because it attracts the behavioral variable to the value specified by the fixed point. To see this consider a value slightly to the right of the fixed point ϕ_A, i.e. ϕ_1. At this point because the rate of change of the behavioral variable is negative the system is driven toward decreasing

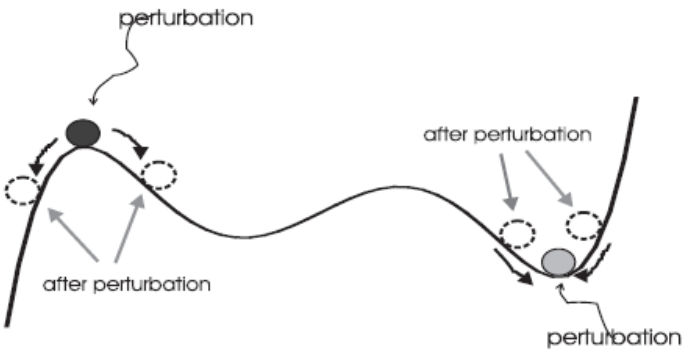


Figure 2.2: Unstable and stable equilibrium points. Initially the two balls are in rest. The black ball is at an equilibrium point at the top of a hill. This is an unstable equilibrium point since a very small perturbation will send it down. By contrast the gray ball is at stable equilibrium point. Once the perturbation ceases to act the ball returns to its initial point

values of the behavioral variable, i.e. toward the fixed point. Analogously, at points starting to the left of this fixed point, for example ϕ_2, the rate of growth is positive thus driving the system toward increasing values, i.e toward the fixed point again. When the system arrives at this fixed point it stays there. The vectors fields of both these dynamical systems behave as attractive forces that drive the system to the state specified by $\phi = \phi_A$. Thus, for example, in the target acquisition behavior in which the direction at which the target is seen is a desired value for the heading direction of the robot, we can make that direction an attractor by erecting a *attractive force-let* (vector-field) with a zero at that direction and a negative slope. The range of the behavioral variables over which a force-let exerts its attractive influence can be unbounded (Panel A) or limited (Panel B). Thus, an important concept related to the idea of attractors is the *basin of attraction*. For a given attractor this refers to the region in the state space in which all initial conditions will converge to the attractor.

Conversely, when the slope at a fixed point is positive, (Panels C and D in Figure 2.3), the fixed point is an unstable state and is called a *repeller* because it repels the system from its value. This can be read on the phase plots of the dynamics: consider a value slightly to the left of the fixed point ϕ_B, i.e. ϕ_3. At this point negative rate of growth for the behavioral variable drives the system toward decreasing values, thus driving the system away from the fixed point. Analogously

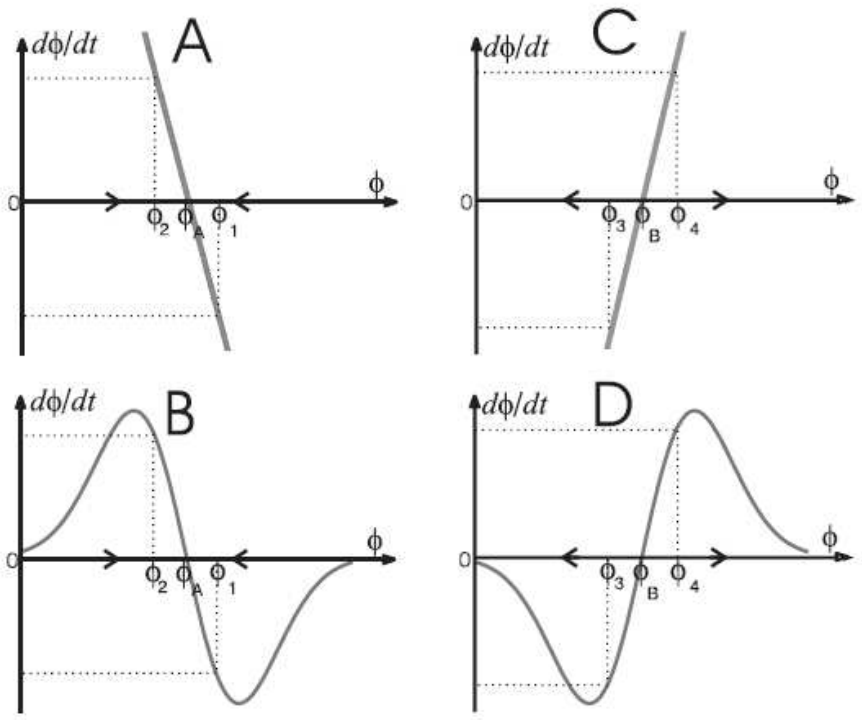


Figure 2.3: Four *phase plots* of one dimensional dynamical systems: the rate of change $d\phi/dt$ is plotted as a function of ϕ. The points at which $d\phi/dt$ is zero (ϕ_A and ϕ_B) are the fixed points of the dynamics and the slope of $d\phi/dt$ there indicates their nature. Panel **A**: The system is linear with a fixed point at $\phi = \phi_A$. The slope at this fixed point is negative. This makes that fixed point an attractor. The system converges in time to the state defined by the fixed point. Panel **B**: A non-linear system with a fixed point attractor also at $\phi = \phi_A$. As for the linear case, the system converges in time to ϕ_A. Panel **C**: The system is linear with a fixed point at $\phi = \phi_B$. The slope at this fixed point is positive thus making this fixed point a repeller. The system diverges away in time from the state specified by the fixed point. Panel **D**: A non-linear system with a fixed point repeller also at $\phi = \phi_B$. As for the linear case, the system diverges from ϕ_B as time increases. The direction of the arrows indicate the evolution of the behavioral variable as time increases.

for a value of the behavioral variable starting to the right of the fixed point, ϕ_4, the system is driven toward increasing values because the rate of change is positive. Once again the system is driven away from the fixed point. In such a case the vector field is called a *repulsive force*. At the value of the fixed point the rate of change is zero but an arbitrary small perturbation immediately causes the system to diverge from the state (unstable) defined by the repeller. Now for example, an obstacle can be modeled by erecting a repulsive *force-let* at the direction at which the obstacle lies, because one has to prevent the heading from taking that direction.

The range of repulsion, i.e. the region of points over which a repulsive force-let wields its influence, can be unbounded (Panel C) or limited (Panel D).

Stability measures or strength of fixed points

For a linear dynamical system (Panels A and C in Figure 2.3) the slope determines how strongly attractive or repulsive a fixed point is. When the fixed point is an attractor the steeper this slope, the stronger the restoring force and the faster the system relaxes to the attractor after a perturbation. For a repeller the steeper the slope the faster the system relaxes away from the repeller after a perturbation. Thus the slope represents the stability of the system in the fixed points.

Since relaxation is exponential it can be characterized by a *time scale*. For instance, if an initial perturbation puts the linear system depicted in Panel A at point ϕ_1, then the system evolves in time according with the solution

$$\phi(t) = \phi_A + (\phi_1 - \phi_A) \exp\left[-\frac{t}{\tau}\right] \tag{2.3}$$

where τ determines the *relaxation time* with which the systems approaches the fixed point ϕ_A. When shifted to a distance $|\phi_1 - \phi_A|$ from the attractor, the system reduces this distance by a factor of e (e is the natural number) in a time interval of τ. Relaxation is faster, the smaller the time scale τ is. Therefore, τ can be used to characterize quantitatively the stability of the system in the fixed points. τ can be obtained from the reciprocal of the slope of the linear vector field at the fixed point

$$\tau = -\left[\frac{df(\phi)}{d\phi} \,|\phi = \phi_{\text{fixed point}}\right]^{-1} \tag{2.4}$$

Negative values for τ indicate that the corresponding fixed point is a repeller.

$\tau = 0$ denotes that the fixed point is *semi-stable*.

For non-linear dynamical systems (Panels B and D in Figure 2.3) with hyperbolic fixed points[1] we can use the linearization method to characterize the stability of the fixed points (Perko, 1991; Crawford, 1991). For example, a stability measure of the attractor ϕ_A of the non-linear dynamical system depicted in Panel B may be obtained by approximating this system to a linear system near ϕ_A. This approximation represents in essence the behavior of the non-linear system in the neighborhood of the attractor. Expanding the vector field $f(\phi)$ in a Taylor series around the fixed point ϕ_A and keeping only the terms up to first order yields a dynamics with a linear vector field,

$$\frac{d\phi}{dt} = f(\phi) \approx \left(\frac{df(\phi)}{d\phi} \,|\phi = \phi_A \right) (\phi - \phi_A) \tag{2.5}$$

the solution of which has the form of Equation (2.3) and where the time scale, τ, is again given by the inverse of the slope of the (non-linear) vector field at the fixed point as for the linear case. Thus, τ as given by Equation (2.4) can also be used to characterize quantitatively the local stability of the fixed points of a non-linear dynamical system. Strong behavioral states have very short local relaxation times.

Integration of elementary behaviors

The complete behavioral dynamics is build up from individual contributions (i.e. force-lets), which are added to shape the complete vector field. Each force-let represents a constraint on the behavior that we are designing. Because the range of the force-lets are limited the resulting dynamical system in non-linear. By design one makes the system to be at all times in, or very near, an attractor so that the overt behavior is really generated by attractor solutions of the dynamical system. This way powerful theoretical tools from the qualitative theory of dynamical systems (Perko, 1991; Crawford, 1991; Scheinerman, 1996), such as local bifurcations analysis, can be used to design autonomous robot architectures and quantitatively evaluate their compliance with specifications.

Each force-let models an elementary behavior. The time scale of each elementary behavior determines how strongly that behavior contributes to the vector field of the behavioral variables. Thus the hierarchy of time scales also determines the

[1]Fixed points are called hyperbolic when they have no eigenvalues on the imaginary axis.

hierarchy of behaviors. Prior behaviors have smaller time scales.

Moving attractors

When attractors are static the requirement that the system must be in or near an attractor at all times is trivially fulfilled. For a robot moving around in the environment the specified values either desired (e.g. Ψ_{tar}) or to be avoided (e.g. Ψ_{obs}) as well as of their strength of attraction or repulsion vary. Thus the individual contributions to the vector field change in time and as a consequence the attractors from the resulting dynamical system move. Since by design the system must be in or near a stable state (attractor) at all times, the rate with which the attractors move must be controlled so that the system is able to track the moving attractors. This is accomplished by making the relaxation time of the dynamics much faster than the time associated with the moving attractors.

Bifurcations

The shape of the vector field in Equation (2.1) is dependent on the parameters. Thus changing the parameter values in time may lead to *bifurcations* in the underlying behavioral dynamics. Bifurcations correspond to qualitative changes in the number, nature or stability of fixed points. Local bifurcation theory helps to make design decisions around points at which the system must switch from one type of behavioral state to another. By driving the system through bifurcations the robot is able to flexibly "decide" the appropriate behavior at any given time.

Chapter 3

The Mobile Manipulator: Dumbo

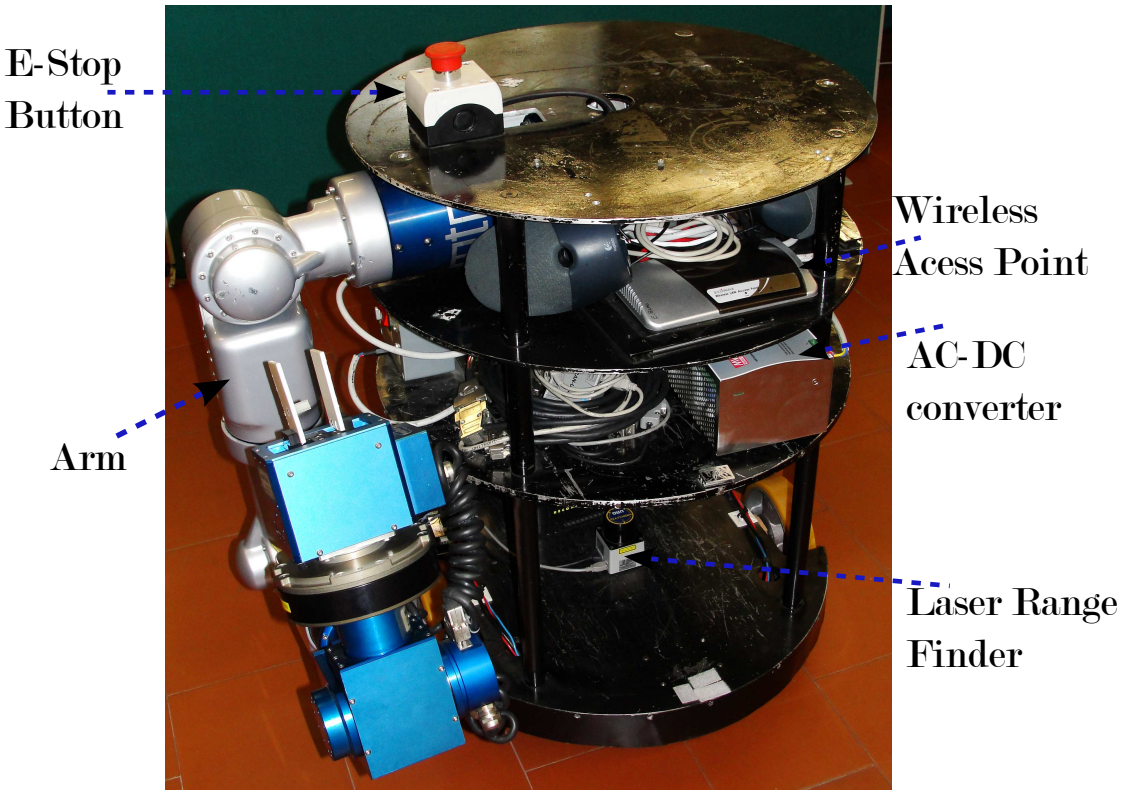
Before the Dynamic Architecture in Chapter 4 is introduced, we present in this chapter the mobile manipulator, called Dumbo, that was built in the scope of this dissertation work. The robot platform and manipulator actuators and sensors are described. An overview of the developed software is given: Hardware abstraction layer, monitor, control, and viewer. Finally, robot kinematics is given as a point to behavioral variables.

3.1 Hardware

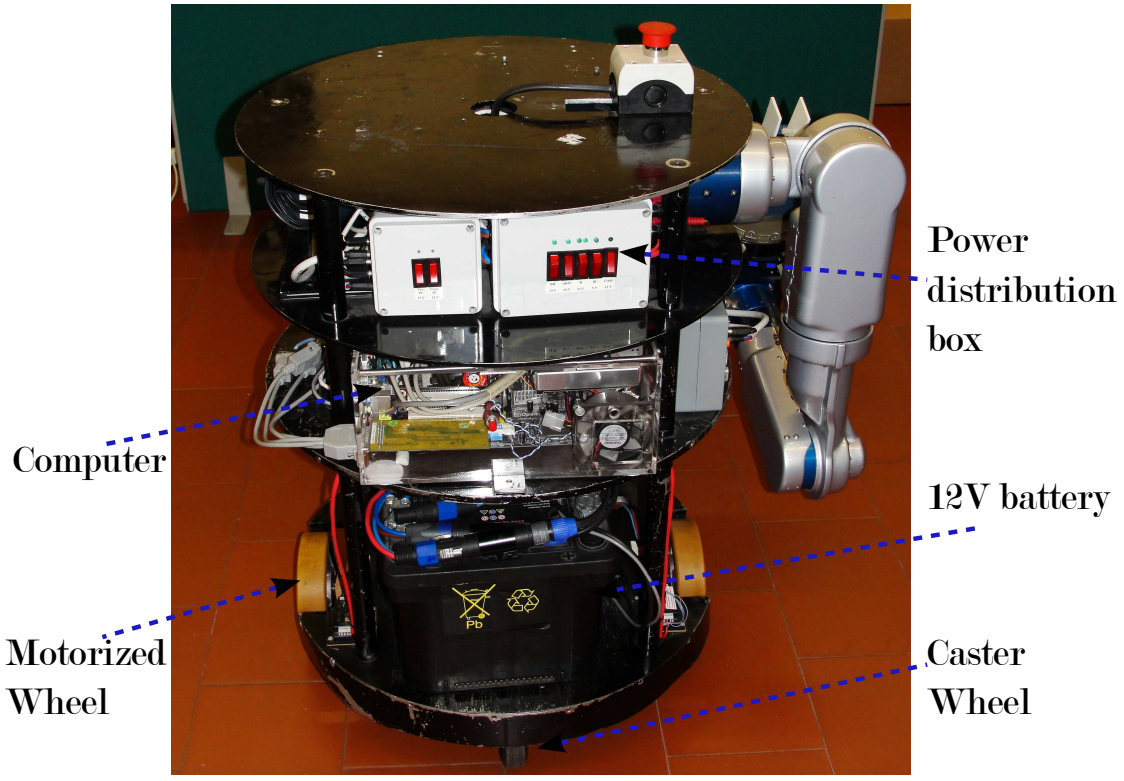
Dumbo, which is pictured in Figure 3.1, has 10 degrees of freedom (DOF), 3 axis force/moment sensor, a laser range finder and a speaker.

There is a 7 DOF arm with an one DOF gripper attached at the tool point. The mobile platform makes use of 2 DOF for steering and path velocity. The *sensorimotor* system is managed by a network of processes attached to a Windows Embedded Standard 7 operating system running on a PC. The processes intercommunicate with high control levels via interprocess communication (IPC) infrastructure.

All hardware modules presented are connected to a computer. This computer is the control unit which completes all computational tasks. It has a Centrino M 1.7GHz CPU, 2GB of RAM and 40GB of hard disk drive. To suppress all connection requirements, a 4 port RS232 - PCI expansion card and a USB hub was added.



(a)



(b)

Figure 3.1: Robot Dumbo: (a) Front and (b) Back view

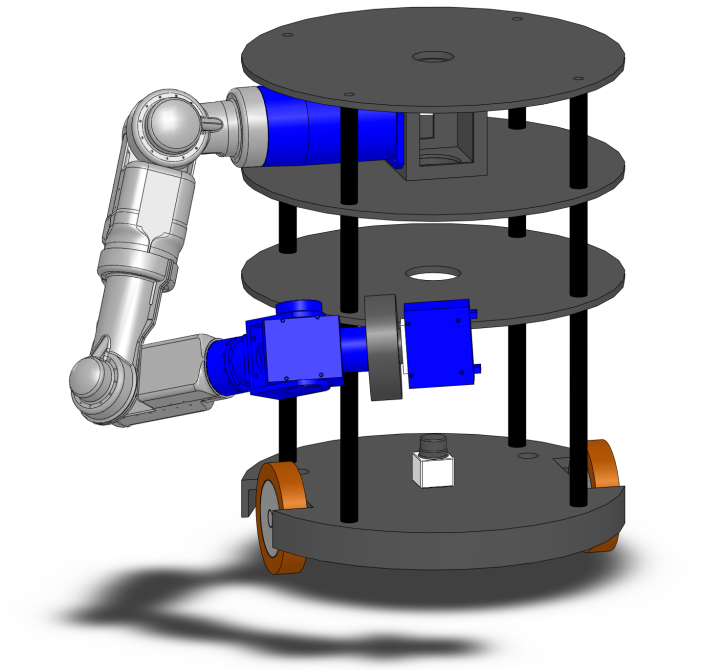


Figure 3.2: Robot Dumbo Layers

Dumbo structure and most of the hardware components used were already available in our lab. However, the robot had to be reassembled from scratch and some changes were implemented: (1) Laser range finder was moved from the front of the base plate to the center; (2) Power system was redesigned. Instead of only battery powered, Dumbo can now be connected to external AC and all the hardware components are powered from a 24V source, using proper DC-DC converters to the specific requirement of each component. Emergency stop button now only powers off motor system, allowing the remainder hardware to be powered in an emergency situation.

3.1.1 Mobile Platform

The structure of the platform has a cylindrical shape with a diameter of 55cm and 76cm height. It is organized in several layers (see Figure 3.2) to accommodate the different hardware components.

Below the base plate, 2 DC (Direct Current) motors and 2 caster wheels are responsible for locomotion of the mobile platform. Each DC motor has a gear and a quadrature encoder attached to its shaft. The motors are set on the side of the robot, while caster wheels are in front and back of robot. This configuration allows

the robot to move at a speed up to 0.2 m/s in straight line or at 0.73 rad/s when the robot rotates about itself (i.e. about the vertical axis that goes through the geometric center of the robot).

On the top of the base plate (first layer) we can find 2 motion controllers for the locomotion motors. The controllers are powered at 24 VDC and are connected to a computer through a RS232 connection. The PID (Proportional-Integrative-Derivative) controller embedded in the motion controllers have input for quadrature encoders and a PWM (Pulse Width Modulated) output for motor power connection. At the geometric center of this plate there is a laser range finder that is used by the robot control to get angle and distance to obstacles in the robot navigation path. The laser range finder has a detection range of 240° with a resolution of 0.35°, which gives a set of 682 steps. The refresh rate of the laser range finder is 10Hz when used in continuous mode, or lower if used in single scan mode. This module is connected to the computer through a USB connection which has a CDC (Communications Device Class) interface with a serial emulator. The top of the base plate accommodates also two serial connected 12 VDC sealed lead acid batteries. When external grid AC (Alternate Current) power is not available (e.g. during transportation task) they are the power source for the whole system (manipulator, platform locomotion, computer and auxiliary hardware).

The second layer accommodates the computer and an AC-DC power supply. The computer runs Windows Embedded Standard 7 operating system and is responsible for suppressing all computational requirements of Dumbo. Both hardware-software interface processes and control processes share this computer. When power grid is available, required energy to power Dumbo flows from the power grid through the AC-DC power supply to the hardware components of Dumbo.

The third layer has the manipulator attach base, power distribution box, one Ethernet switch and a wireless access point. The power distribution box has several on-off switches and DC-DC converters to fulfill all different power requirements of hardware. Outputs are of 24V, 12V, 9V and 7.5V and they are all fused.

An Ethernet switch allows a connection of others computers for debugging or monitoring task. Wireless access point is intended to give a wireless connection way to communicate with the control system that runs on the on-board computer for monitoring the task when experiments are being performed.

On the top of this plat there is a digital compass and an emergency-stop button (e-

Table 3.1: Joints limits of the arm

Joint No.	Min Pos $^\circ$	Max Pos $^\circ$	Min Enc $^\circ$	Max Velo $^\circ/s$	Max Acc $^\circ/s^2$
1	−165	165	0.6	52.2	208.8
2	−15	181	0.6	52.2	208.8
3	−165	165	0.6	52.2	208.8
4	−25	196	0.5	41.2	164.8
5	−165	165	0.5	41.2	164.8
6	−120	120	2	240	960
7	−165	165	3	360	1140

stop). When experiments are being performed, better documentation is achieved if we know the orientation of the robot heading direction relatively to an inertial frame of reference. This digital compass gives the heading angle of the robot relative to earth magnetic north pole. Emergency-stop button switches power off of both manipulator and platform locomotion motors. The computer and the rest of hardware are still powered up.

3.1.2 Manipulator (Arm)

A 7 DOF anthropomorphic manipulator is coupled to Dumbo in its right side. All joints are rotational and all required electronics for each joint is embedded in the arm. Each joint has an harmonic drive gear coupled to a brushless DC motor. Low level control of each joint is also embedded on the corresponding module. A CAN-Bus connects all modules of the manipulator to the computer through a CAN-USB adapter. Furthermore, all joints, except the last one, has a magnetic brake which is activated when no power is supplied to the arm(e.g. when system is turned off or e-stop button is pressed on behalf of an eminent collision which may damage the arm) or when a position control movement is finished.

The Manipulator modules have different physical characteristics, which does not allow the same dexterity among them. Table 3.1 summarizes the maximum and minimum angular positions, minimum increment, maximum angular velocity and acceleration for each joint.

Each joint has an incremental encoder for velocity and position control. Figure 3.3 shows the manipulator with the force/moment sensor attached to the last joint and a gripper fixed to such sensor.

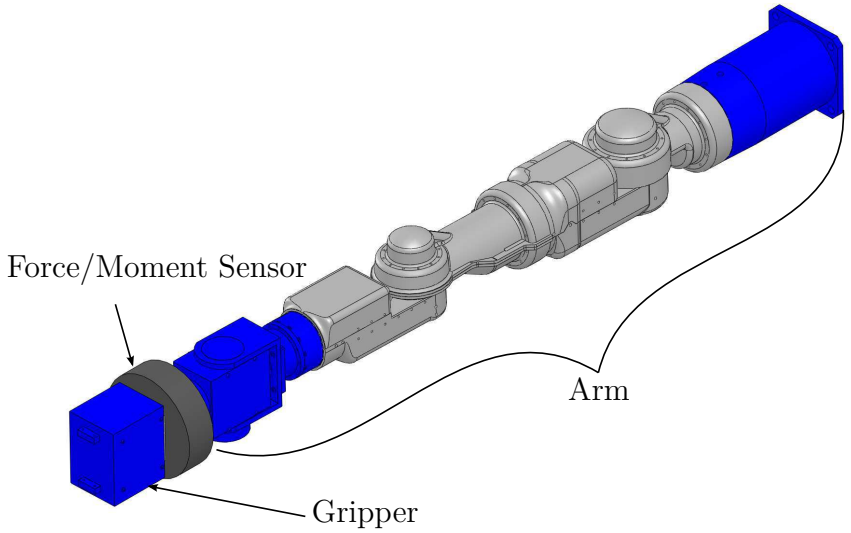


Figure 3.3: Manipulator: Arm, Force/Moment sensor and gripper

Table 3.2: Force/Moment Sensor Limits

	Units	Measurable	Overload (without damage)
F_x	N	± 150	± 200
F_y	N	± 150	± 200
F_z	N	± 150	± 180
M_x	Nm	± 4	± 8
M_y	Nm	± 4	± 8
M_z	Nm	± 4	± 13

The force/moment sensor shown if Figure 3.4 is able to measure force and moment in 3 orthogonal axis. It has an update rate of measurements of 1ms and Table 3.2 summarizes the maximum forces and moments measurable.

The gripper shown in Figure 3.3 has a prismatic joint with one degree of freedom. Table 3.3 shows technical characteristics of the gripper.

Besides physically connected to the arm, both force/moment sensor and gripper are powered by the arm and share the same CAN bus for communications.

Table 3.3: Gripper Prismatic joint characteristics

Min Pos mm	Max Pos mm	Min Enc mm	Max Velo mm/s	Max Acc mm/s^2
-10.75	78.5	0.15	80	320

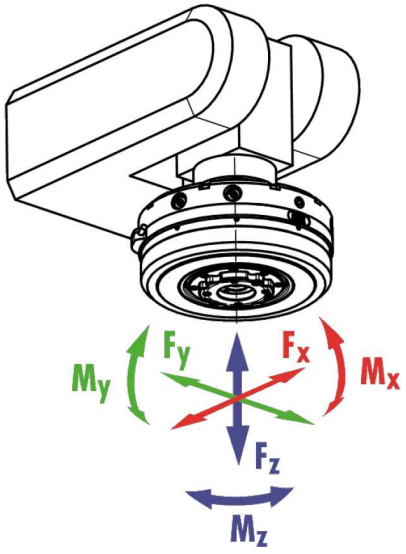


Figure 3.4: Force/Moment sensor. The arm wrist is not the same as the one used in this project, see Figure 3.3

3.2 Software

The software architecture developed in this project is based on a set of guiding principles adopted in our lab as a conceptual framework. These principles have the propose to develop software as a reusable set of libraries or processes that may be useful over different projects.

In order to accomplish that, our software architecture is divided into *devices*, control and auxiliary software as can be seen in Figure 3.5. All these different processes provide interprocess communication mechanism through YARP, Yet Another Robot Platform (Metta et al., 2006). YARP, is a platform for long-term software development for applications that underly a robot. YARP provides a useful set of libraries and tools enabling message based IPC distributed across multiple processes that may be running in different machines.

Regarding the distributed software architecture presented in Figure 3.5, not all processes (applications) presented are required to run during the execution of the transportation task considered in this project. The control application, named Cooperative Transportation, and the devices (Obstacles, Locomotion, Speech and Arm) do not require the presence of the rest of developed software modules (Monitor, Task Controller, Matlab Viewer and Compass). These are auxiliary software modules developed to enhance the monitoring and documentation of the task. The robot performs its operations autonomously with its on-board resources, without

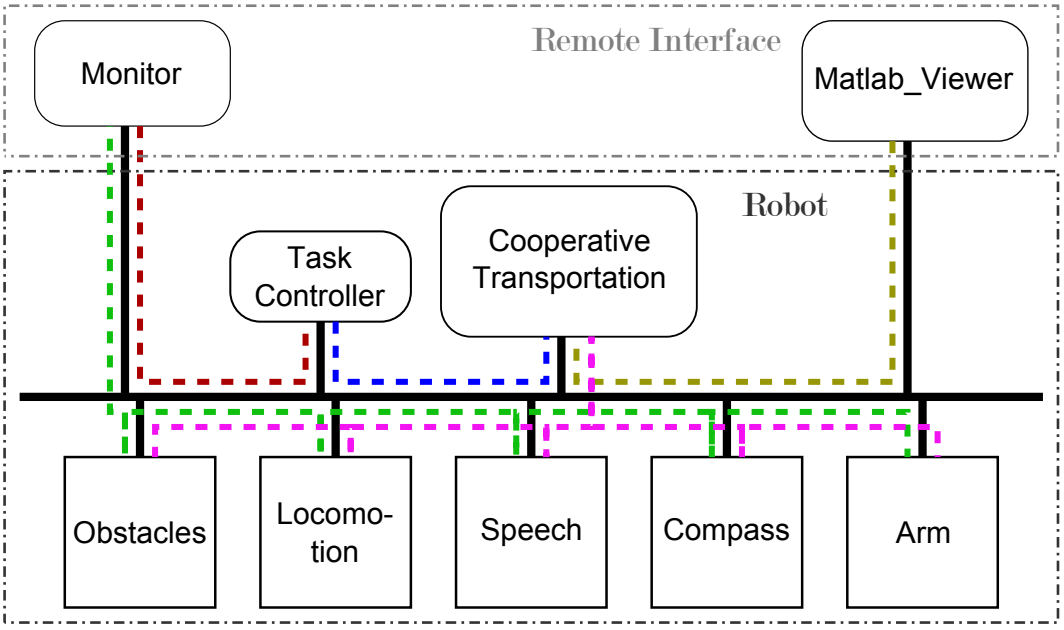


Figure 3.5: Developed Software Architecture. To the successful completion of the Transportation task, only Cooperative Transportation, Obstacles, Locomotion, Speech and Arm processes are necessary. Required resources to such task are on-board of robot. In the figure, dashed pink line shows the connection between these processes. All other lines illustrate the auxiliary connections created between the different applications.

the use of any kind of external support.

The distributed software architecture enables a better control of all *devices*, since each one is performing a simple task. During the development of the Dynamical Architecture, described in Chapter 4, the control process was required to restart several times. If all the hardware components were controlled directly by this process, every time it restarts it would perform tests to the hardware, which would consume too much time. Also, this enables that the developed software (*network_wrappers*, *modules* and others) may be used in other projects with similar requirements. Finally, applications with high computational cost may be allocated to different computers (when available) for better distribution of computational load.

All the software described in this section was developed in a host computer, being afterwards deployed in the target computer (Dumbo robot computer), which has low computational power. As previously mentioned, Dumbo computer runs Windows Embedded Standard 7 and it was configured with the minimum resources required to run the developed software and support drivers for peripheral hardware.

With this reduced installations, the operating system requires less computational power for management and no unnecessary processes are consuming processing time to run.

Despite most hardware modules were already available, only the speech synthesis *device* and the arm *device* had already software developed. Nevertheless, some performance improvements were implemented to speech synthesis. Most of error verification to arm *device* (both to hardware and software interfaces, which were developed by the Engineer Rui Silva) was implemented during the development of this project. Also velocity control was added to arm *device*, since only position control was available. Moreover, Gripper operation and force/moment sensor commands were implemented.

Furthermore, it must be mentioned that the communication protocol between the *network_wrappers* and connected clients was developed by the Engineer Toni Machado during his PhD work. However, some performance improvements and features as binary data sending were implemented.

3.2.1 Hardware Abstraction Layer

Each *device* is composed of a *module* and a *network_wrapper*, see Figure 3.6. As illustrated in the Figure 3.7, the set of all *modules* available to a specific robot is considered as the Hardware Abstraction Layer of that robot.

Each *module* provides a set of functions that allows an easy and abstract way to work with the hardware component where it is attached. In the case of the motion controllers of the platform motorized wheels, at startup, the module is responsible for checking whether it is able to communicate successfully with both motion controllers. More importantly, given a set of linear and angular velocities, the module has the task of converting it into angular velocity of the right and left wheel, send them as a proper command through serial connection to both motion controllers and check whether errors occurred or not. To perform these tasks, this module makes use of some information about the robot structure (e.g. distance between wheels, wheels radius, gear ratio, among others.) This required information may be read from a configuration file or provided by the *network_wrapper* as described below.

A *network_wrapper* is a simple piece of software scaling the features (functions)

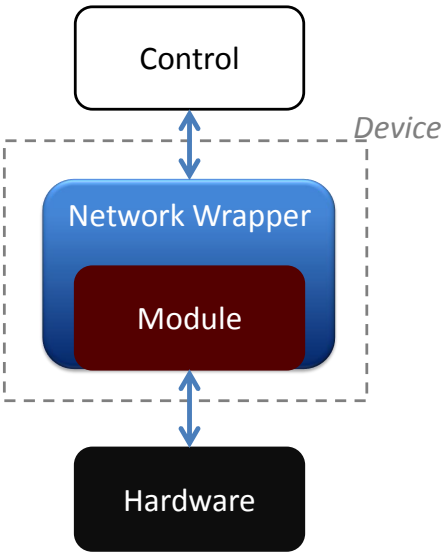


Figure 3.6: Network Wrapper and Module

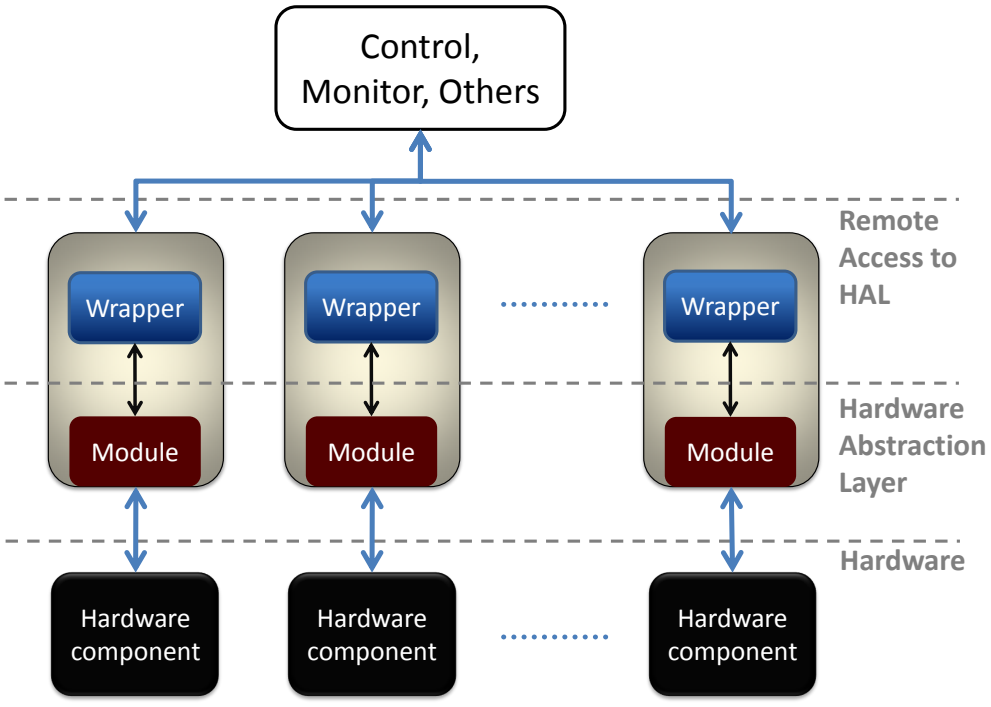


Figure 3.7: Hardware Abstraction Layer

provided by the attached *module* to an YARP network. The *network_wrapper* is designed to interface with a specific *module* (or with different *modules* providing the same features) and waits for requests on a specific *yarp port name* that the *network_wrapper* registered on the *name server*. When a request is received by the *network_wrapper* on that *port*, it checks whether the request is valid, performs the requested action on the *module* and replies with the returned results from the *module*, which may be data or just error status.

Using the example of platform locomotion as before, at startup time, the *network_wrapper* will try to start the *module*. If it can not connect to motion controllers, the startup is interrupted. Otherwise the *network_wrapper* registers a *yarp port name* called */Dumbo/Locomotion* on the *name server*. When some other process wants to work with the *module* features, it just connects to that *port* and sends requests with a *command* that specifies which feature of that module wants to access. For instance, when the control application wants to specify platform velocity, it sends the linear and angular velocity on a request with the *SET_VELOCITY command*. The *network_wrapper* interprets this message and calls the corresponding function of the *module* giving it those values. In the end, the *network_wrapper* replies to that request with error status. The sequence diagram of the Figure 3.8 illustrates this operation.

Yarp port name and others configurations (as *module* configurations) may be specified in a configuration file when the *network_wrapper* is started. When the *module* configurations are described in this file, the *network_wrapper* provides it to the *module*.

The remaining of the devices work in a similar way as described for locomotion (movements of wheeled platform).

In the case of the manipulator, the *module* is responsible for management of all direct communications with a server from the manipulator vendor and monitor the status of the manipulator. This server interfaces with low level controllers embedded in the manipulator through the USB-CAN adapter. This *module* accepts simple commands as position or velocity values for all joints (or only one joint). It performs not only the request to the manipulator server, but also all error verifications necessary to check whether the request was successful or not. This result is replied to who performed the request. The *network_wrapper* registers an *yarp port name* on the *name server*, waits for requests and replies to them until it is interrupted and closed.

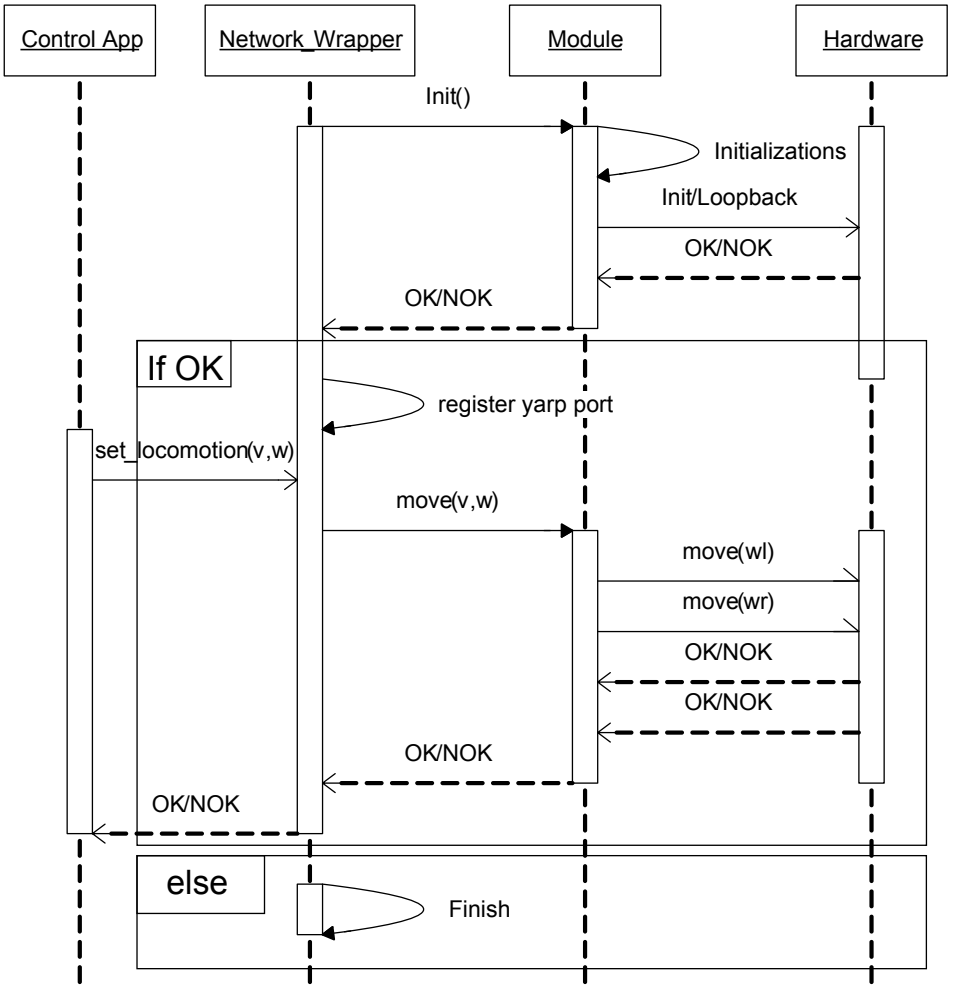


Figure 3.8: Locomotion *device* operation: the *network_wrapper* initializes the locomotion *module* and waits for requests.

For the speech synthesis *device*, the *network_wrapper* accepts requests specifying a string with a sentence to be synthesized be the *module*. This *module* uses text-to-speech engine of SAPI, Speech Application Programming Interface (Microsoft, 2011), developed by Microsoft to allow the artificial production of human speech within Windows Applications.

The digital compass *network_wrapper* accepts requests from the control application, forwarding it to the compass *module*. The physical compass replies with the current heading direction relatively to the north magnetic pole. The *network_wrapper* sends this value back to control application.

For the laser range finder *device*, or obstacles *device*, the basic usage of the *device* works just like the ones previously described. When a *GET_OBSTACLES* request arrives at the *network_wrapper*, it calls the corresponding feature of the *module*. In turn, the *module* performs a data acquisition request of a single scan to the hardware component. When the scan is finished, a vector of data and information is returned to the *module*, which decodes the data. It then gives to the *network_wrapper* a simple vector with the distance to obstacles. This is the synchronous mode of operation. This device allows also an asynchronous mode, where the *module* will deliver to an *yarp port* a vector with the obstacles distances each time a scan is performed by the laser range finder (in our case each 100ms, it is the maximum rate achievable by the hardware of the laser range finder). As the sequence diagram in Figure 3.9 suggests, an initial request configures the *module* in a continuous operation mode. No subsequent requests of *GET_OBSTACLES* needs to be performed. In this mode, the *module* registers a second *yarp port name* for the *device*. An initial request to the *network_wrapper* with a destination *yarp port name* is required. This configures the *module* in a continuous mode of operation, which in turn configures the laser range finder in the same way. The *module* waits asynchronously for data from the laser range finder. When it arrives, the data is decoded and the distances vector is sent through the *module yarp port* to the specified *yarp port name* in the configuration request. This operation mode is finished when a *STOP command* is received by the *network_wrapper*, the destination *yarp port* is unexpectedly closed, or the hardware component reports a malfunction status.

This Hardware Abstraction Layer allows then that the user (in this case, the control application) does not need to know robot structure details (e.g. distance between wheels) or specific hardware communications protocols and control. For instance,

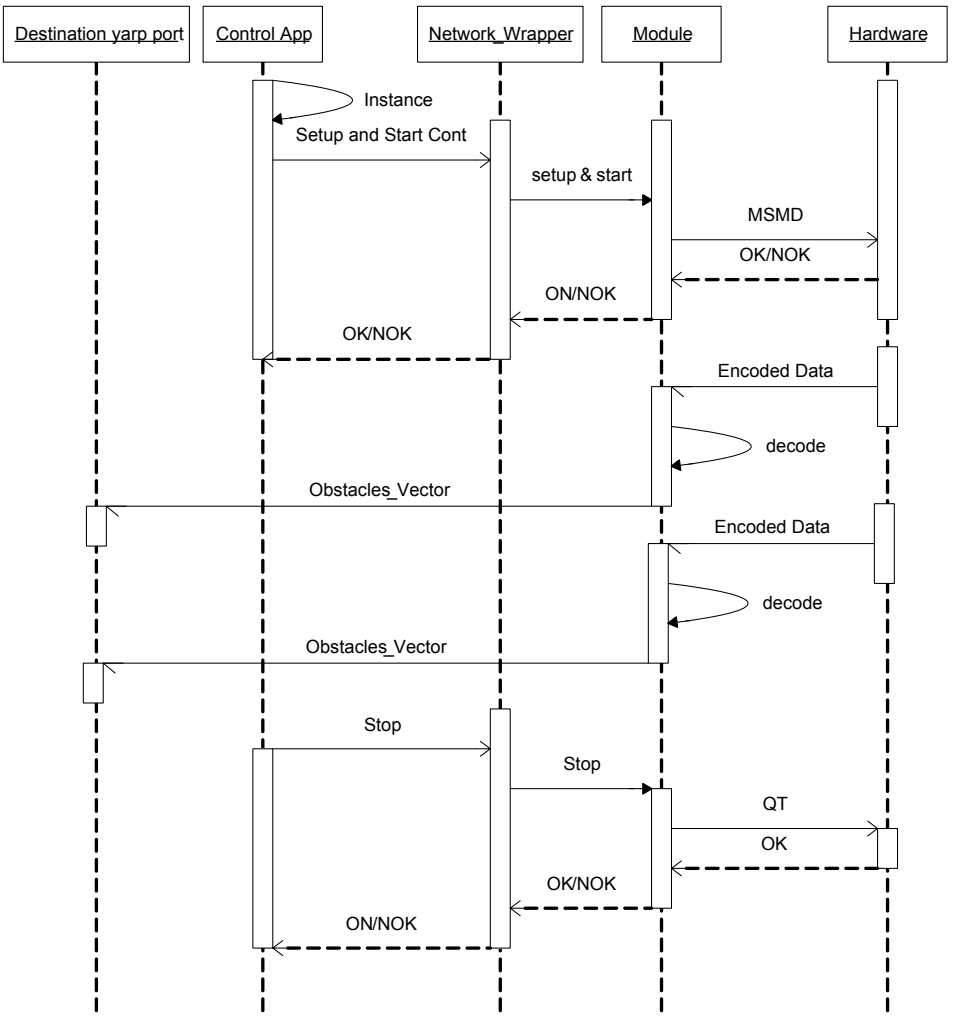


Figure 3.9: Obstacles *device* asynchronous operation

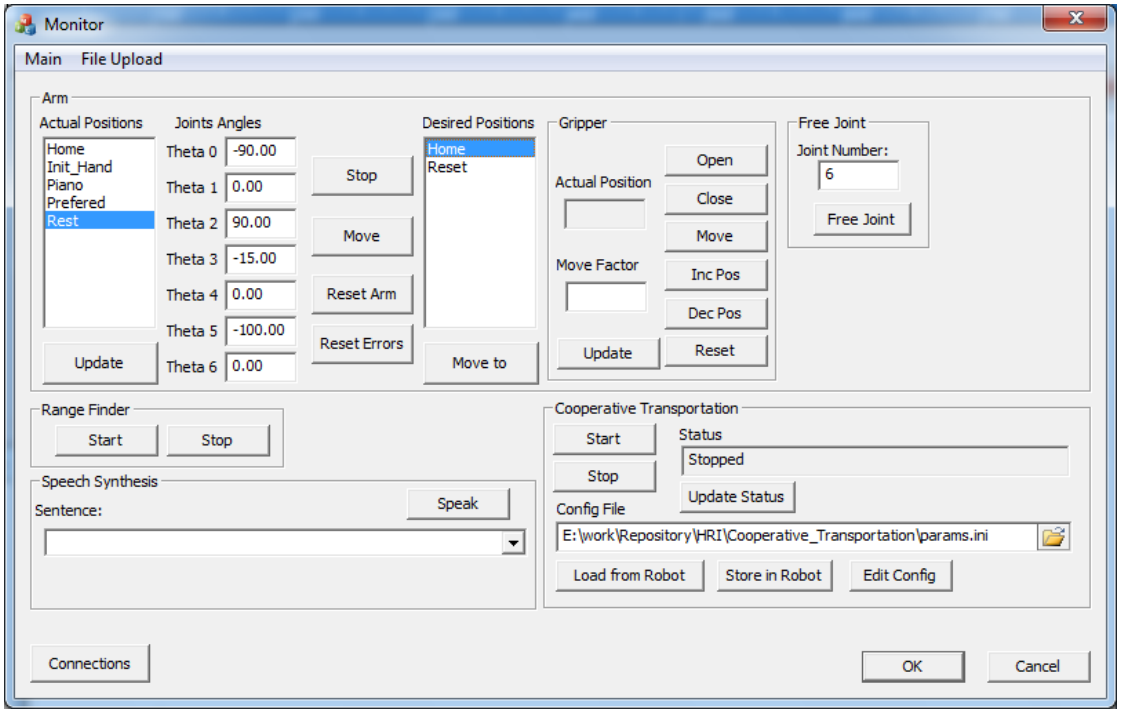


Figure 3.10: Monitor Application

in the laser range finder *device*, the user only needs to send a simple request to the *network_wrapper* and the *module* will take care of all complex message structure that must be sent to the hardware and decode the reply message from sensor containing the data. The reply message to control will be a simplified message only with a vector of distances to the obstacles. The angle formed between the robot heading direction and obstacle can be calculated by the control, see Control Application in the following subsection.

3.2.2 High Level Software

Having all sensor and motor interface been described, we present here the rest of developed software (monitor, control, viewer and auxiliary) to this project.

Monitor

The monitor application, Figure 3.10, was developed primarily as an application to perform several tasks related to test and initialize most of the software and hardware previously presented. Later it was extended to control, monitor and update all that software from a remote computer.

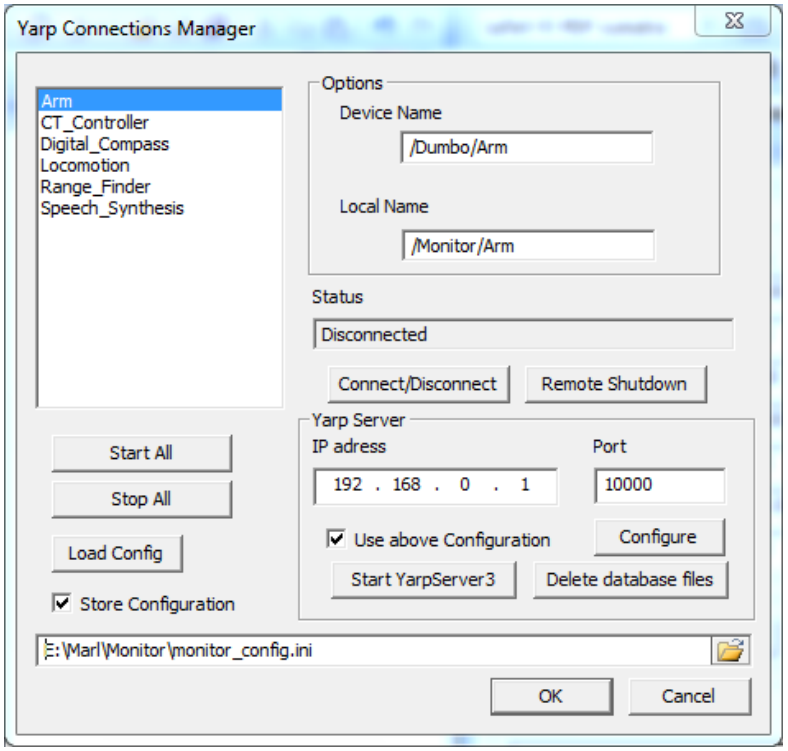


Figure 3.11: Yarp Connections Manager Dialog Window

Below we describe each functionality of this monitor application:

Connections Manager: before any operation may be performed, it is necessary to connect the monitor to the corresponding *network_wrapper* of the *device* or control application. These connections are managed on another dialog, Figure 3.11, that appears when the Connections button is pressed. On the left side are listed *devices* (*CT_Controller* will be described later) that the monitor is able to connect to. Selecting a *device*, upper right side (Options) is automatically filled if a proper configuration file was created before startup or loaded with the button Load Config. Status field describes the state of the connection. When the button Connect/Disconnect is pressed, or a *device name* is double clicked, it checks whether it is connected/disconnected and connects/disconnects accordingly. When connection is not successful, Status field shows a message with the error that cause the failure (e.g. "Yarp Name Server is not running!" when no *yarp name server* is running under the IP (Internet Protocol) address and port number specified). Each *device's network_wrapper* has an additional feature that allows any other application to remotely shut it down. When the connections manager is connected to the *network_wrapper* of a *device*, pressing the Remote Shutdown button will

send to the selected *device* a request with the *SHUTDOWN command*. The *network_wrapper* will finish pending requests and close *module* afterwards.

Yarp Server group allows the management of *yarp name server* configuration and start tasks. IP address and Port fields reflects actual configuration saved in the operating system as a configuration file to the *yarp name server*. These values are read at load time of the yarp connections manager and can be changed to reflect the actual IP address of the machine where the *yarp name server* is running and its root port number. Pressing Configure, those value are written to the configuration file and becomes accessible to all other processes trying to connect to the *name server*. Pressing the button Start YarpServer3, an instance of the *name server* is created. However, sometimes it is necessary to delete the database files generated by this tool (because e.g. the port number or IP address has changed) to run it again. Delete Database Files button does the work and warns the user when an error occurs.

At exit time of this dialog, it may be desirable to store the configurations of actual session. For that, the user may select the Store Configuration checkbox and choose the destination file. When OK button is pressed, this is verified and configurations are stored in respective file.

Arm: After the connection with the arm *network_wrapper*, the arm may be moved to desired positions, actual position may be viewed and reset may be performed, Figure 3.10. When the arm is powered up, it is necessary to perform a reset to each joint to allow the low level control of the joints calibrate itself. Using the monitor, such operation is performed by selecting an approximation to the actual arm position in Actual Positions list and pressing the Reset Arm button. A new dialog appears whether the user should confirm the requested operation. It is a safety measure to the arm, since before reset the arm cannot determine its own joints positions, so, a proper script should be selected that will take into account current position of the arm and reset the joints in a row that the reset movements of each joint will not make it collide with Dumbo structure or components.

When an error occurs in some joint of the arm (e.g. joint velocity limits achieved) it will stop working and it is necessary to correct the problem and reset the errors (without power down the arm which would be expensive both in time and actions since a physical reset would be required). The button Reset Errors clears actual errors of the arm that may be cleared without a

physical reset.

As the name specifies, Stop button sends a request to the *network_wrapper* with a *STOP command* to all joints (magnetic brakes are activated), followed by a request with a *command* to release the brake of the last joint, since it does not have a magnetic brake and is the motor who has to hold it still. The Update button refreshes the joint angles values displayed.

Arm movements may also be performed using this interface: Specifying current position from the Actual Positions list, destination position from the Desired Positions and pressing the button Move To. Monitor application will run a predefined script moving each joint at a time based on a pre-specified sequence. If a position is selected from Actual Positions list and the Move button is pressed instead, the arm will be moved to the new position specifying a velocity for each joint that will move all joints simultaneously and complete the movement at the same time. The time duration of the task is calculated based on the maximum velocity of the slowest joint. The third approach to move the arm is using the text boxes of the joints angles to specify the angle position for each joint. Pressing again Move button, the arm will move to the specified position the same way as described before (to when Move button is pressed).

In a subgroup of the arm, we find the gripper group. This allows the position control of the gripper prismatic joint. Update button refreshes current position of gripper joint. Open/Close buttons changes the value of the prismatic joint to its maximum and minimum, respectively. Specifying a value in Move Factor box and pressing the button Move, will move the joint to that absolute position (the value is specified in millimeters). On the other hand, when IncPos/DecPos is pressed, the prismatic joint will increment or decrement, respectively, the current value of its position with the value specified in Move Factor. Reset button will perform a physical reset on gripper prismatic joint and clear the errors.

To complete arm group operation description, the subgroup Free Joint allows the user to release the magnetic brake or motor brake of a specified joint.

Range Finder: The laser range finder group, Range Finder (obstacles), has a simple interface. A Start and Stop button and a result file is all this group performs. Despite the simple interface, below, the monitor application per-

```
1  Monitor: Range Finder execution log.
2  Start step: 44
3  End step: 725
4  Cluster count: 1
5  Units: mm
6
7  No. of Steps | Values ...
8  682          | 5600 3065 3061 3053 3042 3042 3042 3030 3028 3020 3008
9  682          | 5600 3066 3066 3064 3049 3040 3028 3021 3016 3008 3001
10 682          | 5600 3083 3065 3051 3047 3033 3030 3022 3012 3007 3000
11 682          | 5600 3070 3062 3053 3049 3031 3026 3020 3010 3009 2993
12 682          | 5600 3080 3068 3065 3053 3038 3037 3035 3015 3014 3004
```

Figure 3.12: Result file from Obstacles in Monitor: only an excerpt of the file is presented here. This file has 682 values in each line, corresponding to the distance to obstacles in each of the steps analyzed by the laser range finder.

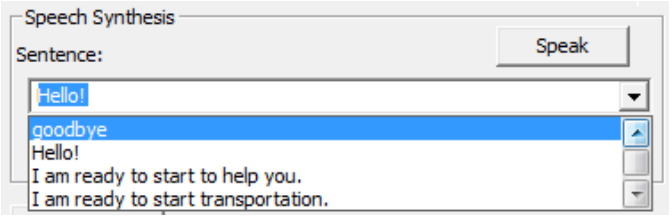


Figure 3.13: Monitor Speech Synthesis Dropdown menu. New speech sentences are presented in subsequent uses

forms the steps and actions mentioned and described by the Figure 3.9 and during operation, received values are constantly stored in a file.

An excerpt of this file can be seen in Figure 3.12, and may be analyzed after the Stop button is pressed.

Speech Synthesis: For speech synthesis group, a sentence may be selected from the dropdown menu, see Figure 3.13, or written directly in the text box. When the Speak button is pressed, a request with that sentence is sent to the *network_wrapper* of the speech synthesis *device*. New messages written in the text box are stored and presented in the dropdown menu for subsequent uses.

File Upload: Another important feature added to the Monitor application is the option of remotely sending the updated executables of installers to Dumbo computer, see Figure 3.14. During the development of the software, several modifications were performed to the different applications that run on Dumbo computer. Those applications were developed in a desktop computer and gathered together in several installation files to deploy in Dumbo (as installation files of Dumbo *devices*, Monitor, Task Controller, between others).

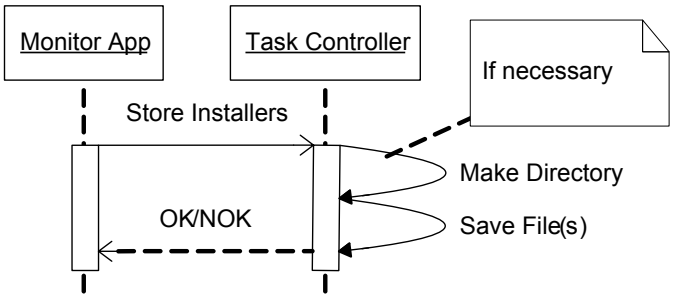


Figure 3.14: Send Installers files sequence diagram. Task Controller will create the corresponding desired destination directory when it does not exist.

Using this feature, those installation files are easily sent to Dumbo in order to install them afterwards. When File Upload|Installers is selected, an explorer dialog is presented allowing the user to select the file(s) to be sent. After, a new dialog is presented to the user so that he/she can select the destination folder in the remote computer to store these files. As before, a dropdown list is presented with the previously selected paths. If a different path is inserted, it will also be stored for future uses. The file(s) is(are) then sent to the remote computer. In the remote computer the auxiliary application CT_Task_Controller is responsible for the completion of this task. This auxiliary application is described in the following Cooperative Transportation description. In case the installers files are sent through this feature, it is necessary an additional step in the remote computer: run the installer to update current applications to the newer versions.

However, during the development of the control application for the cooperative transportation task (written in C++) creating an installer, send it and then run it in the remote computer is time expensive. To overcome this problem, a specific option was added to send only the executable of the control application without the need to install it. File Upload|Cooptrans performs the desired task. The Figure 3.15 shows the sequence of actions performed upon such request. The directory and the name of the destination file may be changed in a configuration file of task controller, see Figure 3.16.

Cooperative Transportation: As mentioned before, during the development of an application as the Cooperative Transportation task it is inevitable that the executable needs to be updated to solve some code errors and add features. Moreover, after sending it to the target computer (Dumbo computer), it is required to start and stop the application. Furthermore, when no update

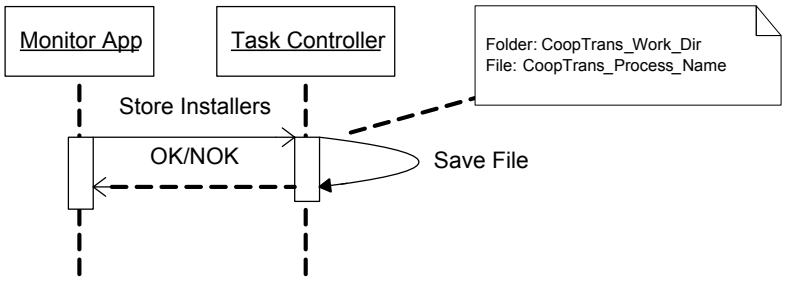


Figure 3.15: Send CoopTrans executable sequence diagram. Destination folder and file name are specified in the configuration file of the Task Controller, see Figure 3.16

```
1  App_Name = /Task_Controller
2  CoopTrans_App_Name = /CoopTrans
3  CT_Server_Name = /CT_Controller
4  TC_Server_Name = /Controller
5  TC_Client_Name = /CT_Controller_Client
6  CoopTrans_Work_Dir  = ..\\CoopTrans
7  CoopTrans_Process_Name  = Cooperative_Transportation.exe
```

Figure 3.16: Task Controller configuration file. Lines 1 to 6 define *yarp port names* for the communication channels. Lines 6 and 7 define the working directory to run CoopTrans Application and the name of the executable, respectively. Lines 1 and 2 defines prefix names for all *yarp port names* of Task Controller and Cooperative Transportation, respectively. The string specified in line 3 will be appended to the one in line 2, forming */CoopTrans/CT_Controller*, which is the *port name* where Task Controller can connect to get status information and send stop requests. Likewise, line 4 and 5 will be appended to line 1. */Task_Controller/CT_Controller_Client* will be the *yarp port name* that will connect to */CoopTrans/CT_Controller*. An *yarp port* under the name */Task_Controller/Controller* will be registered in the *name server* and where Task Controller will wait for requests.

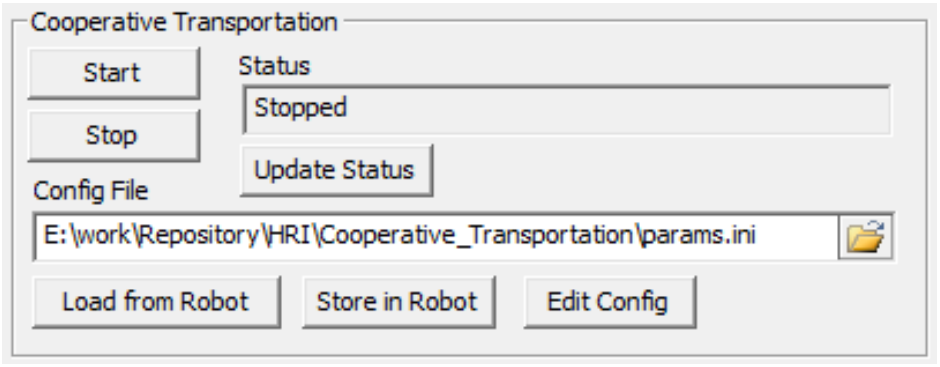


Figure 3.17: Cooperative Transportation execution and configuration file management group

to the executable is required, it may be important to be able to change the parameters to the dynamics control architecture of the robot.

The Cooperative Transportation group, Figure 3.17, was developed in order to suppress the needs described above. It allows Start/Stop and status update of the remote application and to edit, load and store the configuration file.

The local copy (in host computer) of the configuration file may be selected through the open document dialog or entering the path directly in the Config File box. Store in Robot button sends the local copy of the configuration file to the target computer overwriting the old one. Load from Robot button reads the configuration file from the target computer and saves it in the host computer. This file will not overwrite the one specified in the Config File box. A save file dialog appears prompting for a destination folder and a file name. Edit Config button opens the file selected in the Config File box for editing in a text file editor.

The exchange of files between the two computers and the remote Start/Stop of cooperative transportation task are possible because of 3 components:

- Task controller application, always running in the target computer;
- Cooperative Transportation application *network_wrapper*, running in the target computer;
- Monitor Application *network_wrapper*, running in the host computer.

Figure 3.18 illustrates the organization of the specified components.

Task Controller (CT_Task_Controller) is always running in the target com-

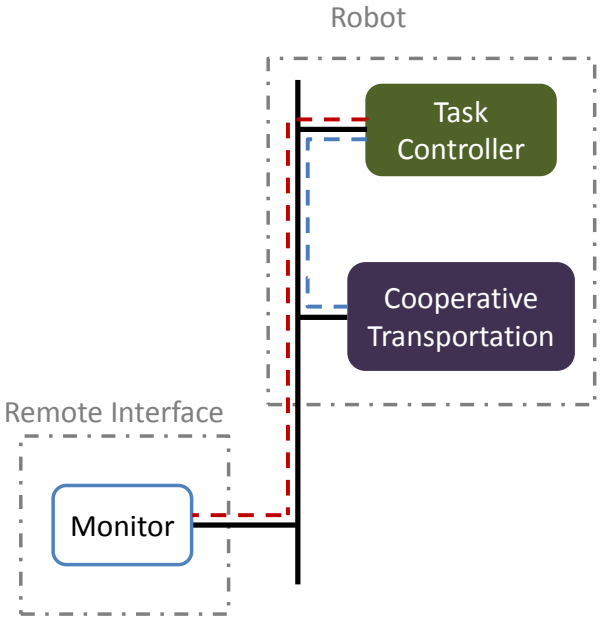


Figure 3.18: Organization of components: Dumbo Computer runs Task Controller process which, in turn, controls the execution of Cooperative Transportation and files exchange. Monitor application and others run on the host computer and connects to Task Controller via wired Ethernet or Wireless

puter. It manages the execution of Cooperative Transportation Task (CoopTrans) and interacts with monitor application for files exchange. Figure 3.19 shows a sequence diagram for Start/Stop of CoopTrans, update status and Load/Store of configuration file.

When a *Start* request is performed, the task controller creates a new process of CoopTrans and tries afterwards to connect to an *yarp port* registered by this process (which by default is named */CoopTrans/CT_Controller*) that allows the task controller to get the status of execution and send a stop request. There are times when the Cooperative Transportation application takes longer time to connect to all *devices* of the Hardware Abstraction Layer and, for that reason, the task controller will not be able to connect to such *yarp port* immediately after the process was created. If the connection fails, the task controller will try five more times timely spaced of 1 second expecting that */CoopTrans/CT_Controller* becomes ready during this time. When no success is achieved after the five tries, the task controller will report the error to the monitor application.

When a *Stop* is requested, the task controller will try to connect to */Coop-*

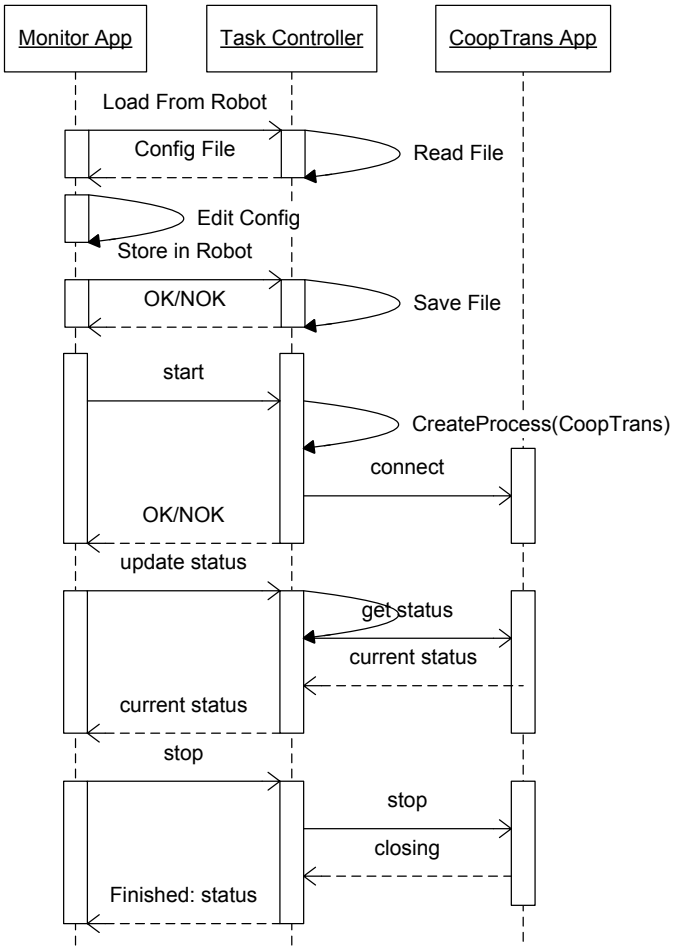


Figure 3.19: Task Controller performs all remote requests from the Monitor in the target computer. A typical session involves reading CoopTrans configuration file, modifying it in the remote computer, send it back and start the application. During execution, update of status gets running information (some error may occurred in hardware) and finally in the end execution may be stopped.

Trans/CT_Controller (if it was not successful on *Start* request) and send a stop command. If the CoopTrans application does not finish within 10 seconds, or could not connect to such *yarp port*, the application will be aborted and the error reported to monitor.

On a *Update Status* request, the task controller redirects the request also to the Cooperative Transportation application through the previous channel created between the two (*yarp port* named */CoopTrans/CT_Controller*). The task controller will reply with its operation status and Cooperative Transportation status, which may be in error state, because of some *device* failure, for instance.

The sequence diagram of the Figure 3.19 also describes the sequence of actions performed when a store or load file operation is requested. These operations only involve the monitor application and task controller. The Cooperative Transportation application will take into account the new configuration file when it is restarted.

Control Application

The development of the control application (Cooperative Transportation application) is the main reason to the development of all the software described. It is the high level control of the hardware under the Hardware Abstraction Layer. This application was developed to implement and test the robot's dynamical control architecture, which is presented in Chapter 4. This application is divided in two main tasks:

- Control loop, where all calculus related to robot's dynamical architecture are performed;

- Auxiliary thread, where interface with Hardware Abstraction Layer is performed. This thread is necessary to deal with the asynchronous characteristic of the obstacles *device*. This task is hidden inside a C++ class named `Robot`.

The algorithm of the control loop is presented in Algorithm 1. It is a high level control, since all low level part is realized by the auxiliary software, leaving to the robot's dynamical control architecture developer an abstract way to interface with the robot.

Algorithm 1: Control Loop of Cooperative Transportation

```
1  Initialize variables;
2  Connect to Hardware Abstraction Layer;
3  Configure obstacles;
4  while no stop command received do
5  |  Get input from sensors;
6  |  if no error received from input then
7  |  |  Compute of heading direction dynamics;
8  |  |  Compute of path velocity dynamics;
9  |  |  Send new values to manipulator;
10 |  |  Send new values to platform locomotion;
11 |  |  if speech sentence to synthesize then
12 |  |  |  Send speech sentence;
13 |  |  end
14 |  |  Save input from sensors;
15 |  |  Save dynamics results;
16 |  |  Provide to Matlab_Viewer;
17 |  else
18 |  |  Show error: "Device returned error";
19 |  |  Stop execution until error is fixed;
20 |  end
21 |  Update cycle time;
22 end
23 Disconnect from Hardware Abstraction Layer;
24 Finish;
```

The C++ class Robot assumes all the required tasks related to management of connections to all the *devices* that compose the Hardware Abstraction Layer. After initializations, all the work that the control have to do, to get for example the obstacles, is call `robot.get_obstacles(&obstacles_vect)`, supposing that a class `Robot` has been instantiated with the name `robot`. `obstacles_vect` is a vector where the distance to obstacles will be stored, so, `obstacles_vect[i]` is the distance to the obstacle in sector `i`. All the other *devices* work in a similar way, setting new values to manipulator joints positions is as simple as call `robot.set_manip_pos(joints_pos)`, or defining new values to joints velocity, `robot.set_manip_vel(joints_vel)`, where `joints_pos` and `joints_vel` are vectors with joints positions and velocities, respectively. To verbal communication, a call to `robot.speak(speech_sentence)` will send a message to the speech synthesis *device* resulting in that message to be spoken by the robot. When new values to robot navigation velocity should be sent, a call to `robot.set_locomotion(v,w)` will send the linear velocity v and angular velocity w to the robot locomotion *device*.

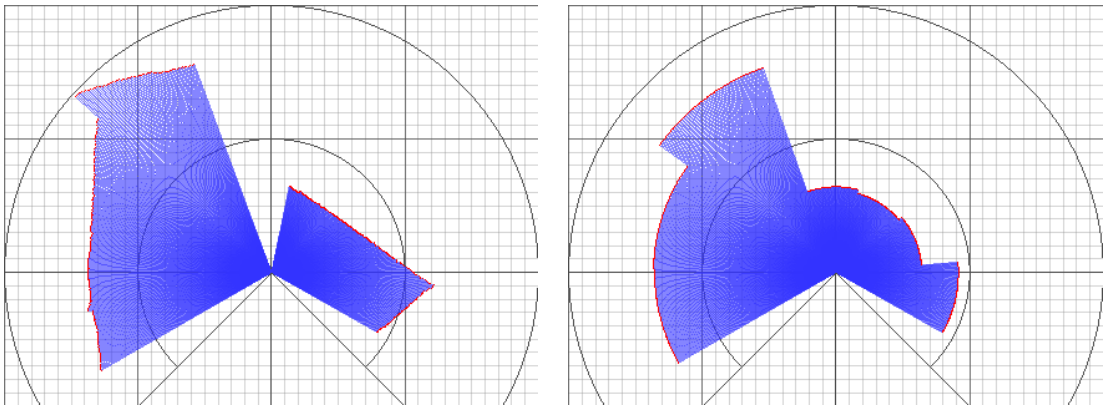
Except for obstacles *device*, all the tasks related to get data from sensors and setting new values to actuators are straightforward, since all the required work to do in `Robot` class is to compose the corresponding request and check whether errors occurred during the operation.

The complex part of obstacles *device* is its asynchronous operation mode to get the maximum achievable performance in operation and the processing task required to be performed afterwards.

Usually, the control does not need to handle obstacles with the discretization that is achieved by the laser range finder (0.35°), Figure 3.20(a). The laser range finder can be configured to return the data in clusters [1] , and this way the control may divide the full range in sectors with a wider aperture than the full discretization of the range finder, see Figure 3.20(b).

Despite this feature of the laser range finder, it can not be used by the control, since, as we can see in Figure 3.21 , two support columns of robot structure are in the range of the laser scan. The range of the laser scan can be configured, but only the start point and end point. The range must then be continuous, not allowing a

[1] Division of the data in clusters is a feature of the laser range finder that permits to specify through a cluster count value the number of adjacent steps that can be merged into single data. When cluster count is more than 1, step having minimum measurement value in the cluster will be the output data.



(a) (b)

Figure 3.20: Full discretization (a) and in clusters (b), with cluster count 99 This data was gathered from the same real world scenario, although the laser was not in the robot structure. When no distance to obstacles is presented in a sector, means that, if there is an obstacle, it is outside the range of the laser.

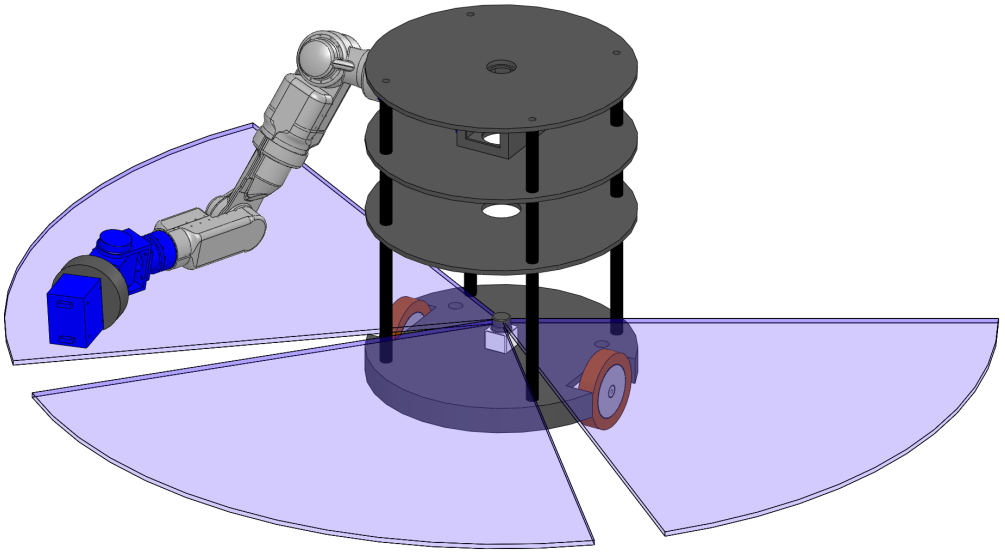


Figure 3.21: Two structure columns are in the range of the laser scan. A division in clusters be the laser range finder would return always obstacles to the left and to the right of the robot.

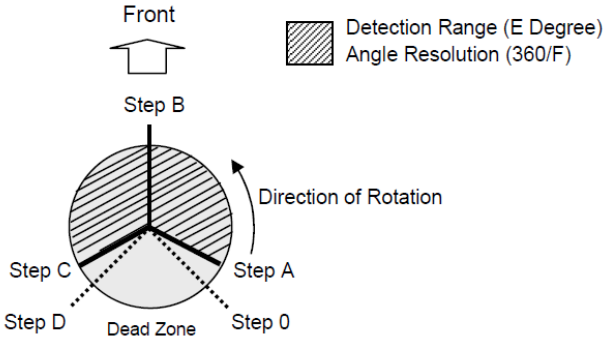


Figure 3.22: Measurement parameters

Table 3.4: Measurements Parameters for Sensor URG04-LX

	Description	Value
Step 0	First Measurement point	0
Step A	Initial Measurement Step of Detection Range	44
Step B	Sensor Front Step	384
Step C	End Point of Detection Range	725
Step D	Last Measurement Point	768
E	Detection Range (°)	239.77
F	Slit Division	1024

specification of a set of steps to be ignored. If data were collected in clusters, the control would always get obstacles to the left and to the right of robot.

To overcome those two problems, a thread inside the class of the `Robot` was implemented. It asynchronously waits for data from the laser range finder *device*. When data is received, the two set of steps that wrongly define obstacles are excluded from the measurements [2] (defining its distance to maximum) and a division in sectors is performed.

Both the two set of steps that should be ignored and the number of sectors that the measurements should be divided into are defined by the control application before a call to `robot.start_obstacles()` is performed. Those values are read from the configuration file of the application. Others parameters as Step A, Step B, Step C, Slit Division (see Figure 3.22 and Table 3.4) and desired angular range of detection must also be known. The desired angular range of detection must be

[2] A simpler approach verifying if the distance to the obstacle is smaller that the radius of the robot could not be implemented. From experience, the measured distance by the laser range finder to objects with black colors (which is the color of the robot structure) is not always accurate as expected. If the measured distance was bigger than the robot radius, it would influence the control strongly and constantly.

specified in same configuration file of ignore steps and number of sectors. The other parameters can be obtained from the *device* because those information parameters are read directly from the hardware component or from a configuration file of the *device*.

Algorithm 2 shows the steps that are performed and required information to configure the correct acquisition of obstacles. Considering the desired angular

Algorithm 2: Configure Obstacles Acquisition

input from control: Detection Range, α; Sectors Number, n
input from *device* : Step A, sa; Step B, sb; Step C, sc; Slit Division, sd
output to control : Effective Detection Range, χ; Angular Difference Between Sectors, $\Delta\theta$
output to *device* : Start Step, ss; End Step, es;

```
1  ss ← sb − (α/2)(sd/2π) + 1;
2  es ← 2 · sb − ss + 1;
3  χ ← (es − ss + 1)(2π/sd);
4  Δθ ← χ/n;
5  ns ← es − ss + 1;
6  for i ← 0 to i ≤ n do
7  |  obstacles_lim[i] = −χ/2 + i · Δθ;
8  end
9  for i ← 0 to i < ns do
10 |  found ← false;
11 |  β ← −χ/2 + (ss−a+i)/s;
12 |  while sector not found do
13 |  |  if sectors_lim[i] > β then
14 |  |  |  found ← true
15 |  |  else
16 |  |  |  increment sector
17 |  |  end
18 |  end
19 |  sector_pos_[i] ← sector
20 end
21 setup(ss,es,1);
```

range of detection, a start step (`start_step`, ss) and an end step (`end_step`, es) are determined, line 1 and 2. To gather the steps in different sectors, a vector (`sector_pos_`) with a size of the number of steps (ns) is filled with the sector number where each step should be saved. Finally, a request is sent to the device with the `start_step`, `end_step` and a cluster count of 1, since we want the maximum discretization from the laser range finder.

With the output to control from the Algorithm 2, the angle between the robot heading direction and each sector, θ_i, can be determined as Algorithm 3 suggests. Where ϕ is robot heading direction (at any instant relatively to an inertial frame

Algorithm 3: Configure Obstacles Acquisition

input : Effective Detection Range, χ; Sectors Number, n, Angular Difference Between Sectors, $\Delta\theta$
output: Angular direction of sectors θ_i, $(\Psi_i - \phi)$

```
1 for i ← 0 to i < n do
2 |  (Ψ_i − φ) = −χ/2 + Δθ/2 + i · Δθ;
3 end
```

of reference) and Ψ_i is the angle between the inertial frame of reference and the sector i.

As described before, after a call to `setup_obstacles`, calling `start_obstacles` a *yarp port* for reception of data is registered. The name of this *yarp port* is sent to the obstacles *device* and a new thread is created. Once the thread is ready, a *start_asynchronous* request is sent to the *network_wrapper* of the laser range finder *device*. This thread executes the code represented by the Algorithm 4. After some initializations, this thread waits for data from the obstacles *device*.

Algorithm 4: Thread Read Obstacles

input : Yarp Port Name, *port_name*
output: vector of distance to obstacles, *obstacles*

```
1  port ← register(port_name);
2  send start_asynchronous;
3  while not stop reading do
4  |  data ← port.read();
5  |  mutex.acquire();
6  |  obstacles ← {max_distance}_{n×1};
7  |  for index ← 0 to index < data.length() do
8  |  |  if (index + ss) ∈ Ignore_Steps then
9  |  |  |  data[index].value ← max_distance;
10 |  |  end
11 |  |  if obstacles[sector_pos_[index]] > data[index].distance then
12 |  |  |  obstacles[sector_pos_[index]] ← data[index].distance
13 |  |  end
14 |  end
15 |  mutex.release();
16 end
```

When it arrives, for each step present in the received vector, it is verified if it belongs to the ignore steps sets, line 8. If it does, the distance value is overwritten by the `maximum_distance` specified by the control. This result is then saved in the corresponding sector if its value is smaller than the one already there.

This thread works asynchronously, however, the control application can always call `robot.get_obstacles()` and the most recent set of obstacles distances received will be returned, as Algorithm 5 shows.

Algorithm 5: Control: get obstacles

output: vector of distances to obstacles, *obstacles_vect*

```
1 mutex.acquire();
2 for index ← 0 to index < obstacles.length() do
3 |   obstacles_vect[i] ← obstacles[i]
4 end
5 mutex.release();
```

Despite the straightforward algorithms presented in Algorithm 2 and Algorithm 4, the real implementation has several errors verification. For instance, the desired angular range of detection may be to wide or to close to a specific laser range finder. When data is received, error verification is performed to check if the *device* encountered an error and stopped working.

Matlab Viewer

A MATLAB GUI was developed to view the dynamics of the robot control architecture presented in Chapter 4 while tests are performed with the robot. Figure 3.23 shows a snapshot of the developed GUI which was named Matlab_Viewer. This GUI shows graphically the different contributions from the sensory input to the resulting vector field. The resulting vector field and velocity dynamics are also drawn. The GUI is able to show the distance to obstacles in a table. To the completion of this task, a communication channel was created between the GUI and the control. When the control starts, a *network_wrapper* is created (named *wrapper_viewer*), registering a */CoopTrans/Matlab_Viewer* (default name) *yarp port name* in the *name server*. As soon as the GUI is started, it will query for such *yarp port name* and connect to it if available. A request is then submitted demanding the parameters that are static for a session (e.g. number of obstacles, robot radius, between others). After this initialization, subsequent requests

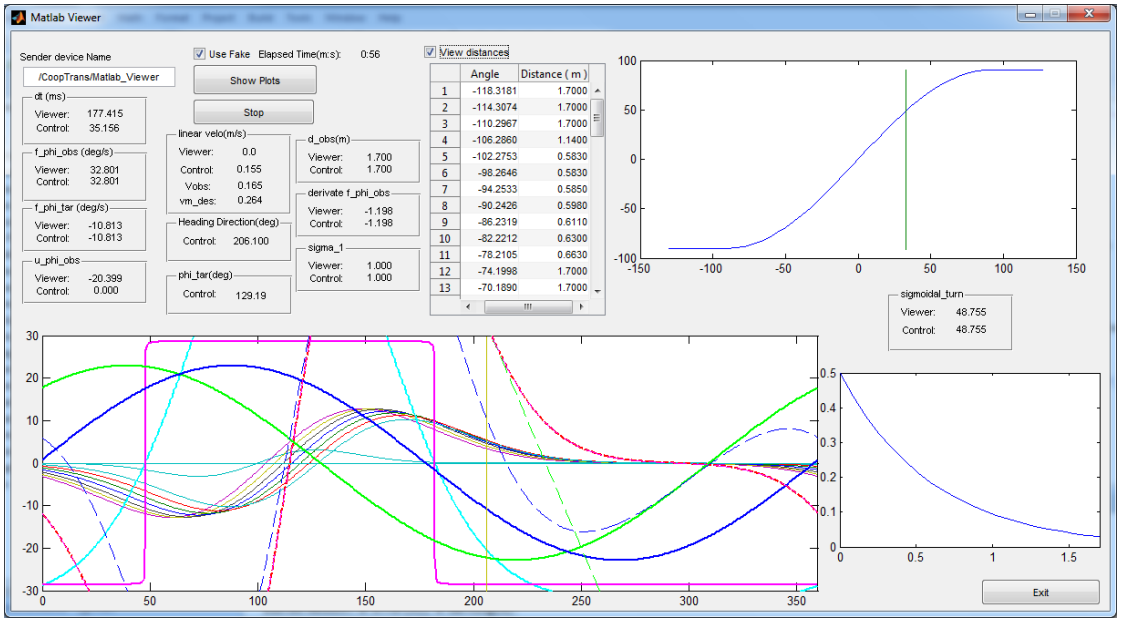


Figure 3.23: Matlab_Viewer

are performed to obtain the sensory information gathered from the Hardware Abstraction Layer that is relevant to perform the calculations. Despite the objective of the Matlab_Viewer is visualization, it has to perform more calculations than the control. In each computational cycle of the control (fetch, compute, update) it only calculates the dynamics for the current heading direction of the robot, while the GUI perform the calculations for the entire polar circle. Only actuation (update) is not performed by the viewer. The Figure 3.24 shows a typical sequence diagram of the communication between the control and the viewer.

When tests were performed on the robot, the Matlab_Viewer GUI was running on a computer connected by wireless to the robot. The high computational cost of the GUI (comparing to the control application is about 3 times higher) and the necessary time to complete a data request, lead us to implement the exchange of information between the Matlab_Viewer and the CoopTrans application in an asynchronous mode. The *wrapper_viewer* provides this operation mode. In each cycle time of the control loop a set of data is provided to the *wrapper_viewer*, which is running is a different thread. The next set of data will override the last one if the Matlab_Viewer haven't read it yet. This allows that, as soon as the Matlab_Viewer performs a new request (even if the last one was a long time ago) the most recent data will be replied. If, somehow, the Matlab_Viewer is faster in calculations than the control and the travel time of the data is small enough such that the GUI performs a new request to *wrapper_viewer* without the control

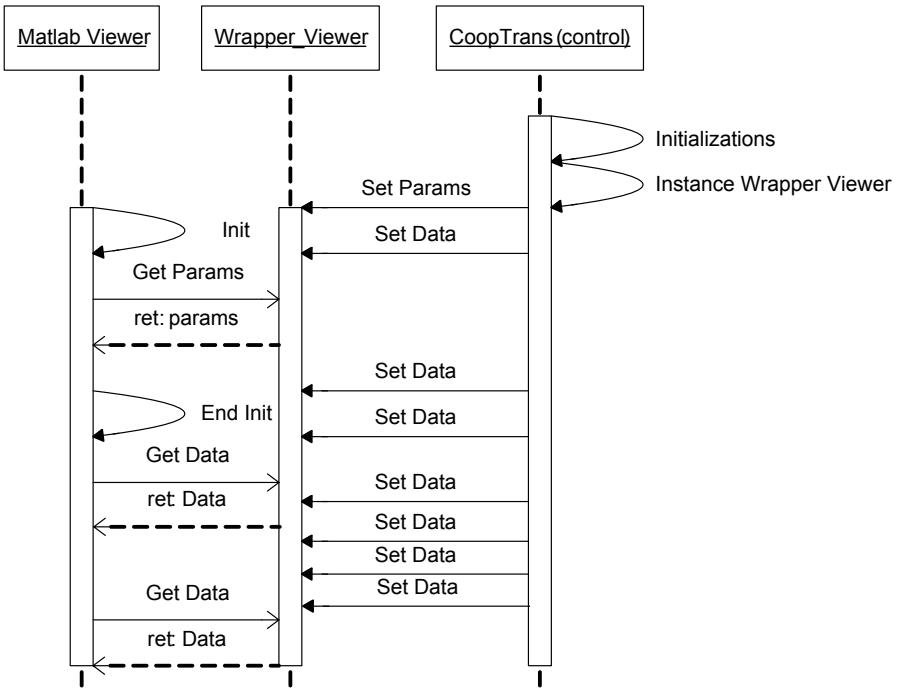


Figure 3.24: Matlab_Viewer can only be started after Cooptrans application. Because of it's high computational cost and travel time for data requests, CoopTrans sets new data much faster than the Matlab_Viewer reads it.

update the data (see Figure 3.25), *wrapper_viewer* will not reply to the GUI until the control sets new data or an error occur.

The Matlab_Viewer can be started and stopped any time during an execution of the control and *wrapper_viewer*. On the other hand, if the control is stopped, Matlab_Viewer will also stop (it recognizes that an error occurred) and has to be restarted after CoopTrans is running again.

3.3 Kinematics

Dumbo navigation in the environment is accomplished by the two motorized wheels and the two caster wheels. Since caster wheels are passive, for robot navigation, it is necessary to generate two control signals: one for angular velocity of left wheel, $w_{wheel,L}$ and another to the right, $w_{wheel,R}$.

The platform may be controlled by setting continuous values to it's linear velocity, v, or path velocity, and angular velocity, w, see Figure 3.26.

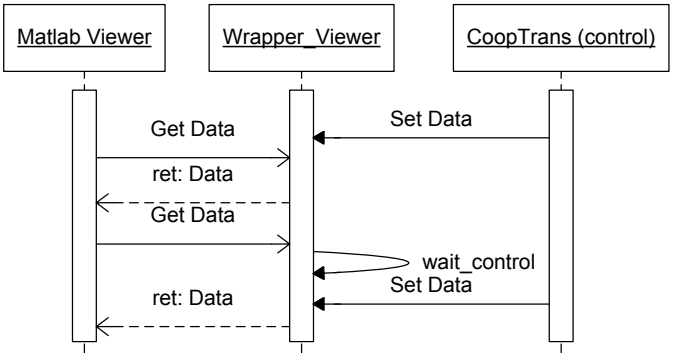


Figure 3.25: When the developed Matlab GUI is faster than the control, *wrapper_viewer* will make it wait for control.

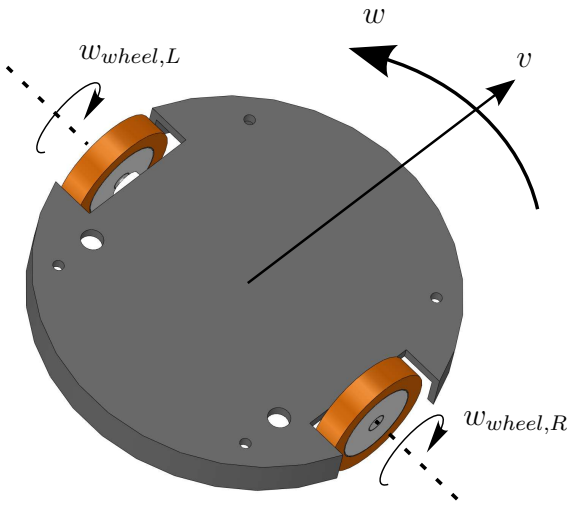


Figure 3.26: Robot Kinematics

Using the equations (3.1) and (3.2),

$$w_{wheel,L} = \frac{1}{R_{wheel}}(v - \frac{D_{wheels}}{2}w) \tag{3.1}$$

$$w_{wheel,R} = \frac{1}{R_{wheel}}(v + \frac{D_{wheels}}{2}w) \tag{3.2}$$

the angular velocity for each motorized wheel can be determined. Where R_{wheel} is the wheel's radius and D_{wheels} is the distance between the driving wheels.

3.4 Conclusion

In this chapter we have presented the hardware and software modules of the mobile manipulator that was built in the scope of the dissertation work.

Next we present the theoretical framework that was used to design the robot's dynamic control architecture.

Chapter 4

A Dynamical Architecture for Control and Coordination of the Mobile Manipulator

The task of the mobile manipulator is to transport an object in cooperation with a human. In this chapter we describe the strategy adopted to perform such task. Specifically, we adopt a *leader-follower* strategy where the leader (human) goes from an initial position to a desired goal position. The follower (robot) should help supporting the object and trail the leader agent (i.e. the human), but how to change it's navigation to avoid obstacles and how to coordinate itself with the human partner is left to the followers decision.

4.1 Strategy Adopted

The human operator and the robot must together transport a long object in an unstructured and unknown environment. The object must not fall while it is being transported to the desired goal position. A *human-follower* motion control strategy was adopted. The human assumes the leadership of the task, since it is the only agent endowed with capabilities to understand the goal of the task. The robot must be able to support the leader for smooth and safe execution of the task goal while avoiding static and dynamic obstacles.

Unlike most leader-follower scenarios, the leader does not provide to the follower

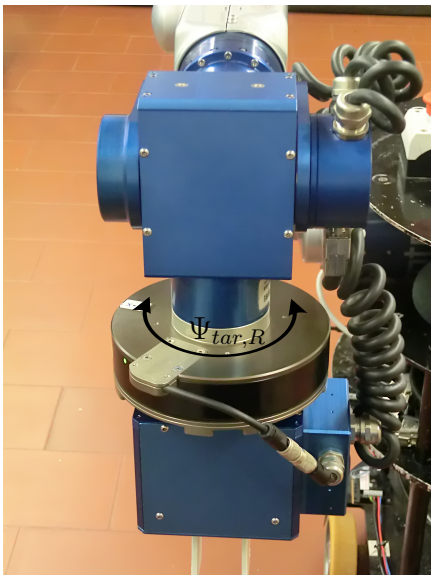


Figure 4.1: The last joint of the arm is kept rotational free

precise directions in terms of what to do, it is assumed that the robot is autonomous. In other words, the robot is able to acquire the required information to complete the task from its own sensors and has a certain degree of freedom to move as long as it does not interfere with the accomplishment of the goal or its own limitations. For instance, when the two partners are transporting a large object, the user steers to the goal position, avoiding obstacles. The robot then trails the user while supporting the object, but the robot must gather from the environment the necessary information to keep the movement of the object, avoiding obstacles and alert the human when an intended movement (e.g. pass between two obstacles) is not feasible to be performed by the robot.

As a first approach, and to make the execution of the project feasible in the stipulated time, the manipulator movements, while the task is being performed, were not considered. When the task is started, the robot moves it's arm to a prior selected posture for transportation, which is maintained until the end of the task. This (sub)optimal posture was selected to comply with some requisites:

- Allow free rotation of the wrist joint parallel to the ground, see Figures 4.1 and 4.2;
- End-Effector height enough such that the transportation of the object is comfortable to the human without requiring him to curve;
- Avoid joint limits and singularities.

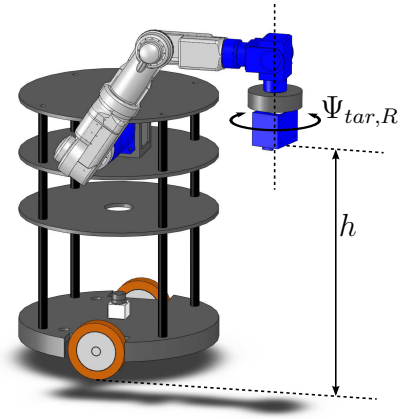


Figure 4.2: Transportation task posture for the mobile manipulator. The height, h, is about 0.72.

Table 4.1: Transportation task posture for mobile manipulator

Joint No.	Angle $^\circ$	distance to joint limit $^\circ$
1	60	105
2	0	15
3	90	75
4	150	46
5	0	165
6	90	30
7	0	165

Taking into account the height and orientation of the end-effector (in order to be able to grasp the object), joints limits and singularities, the posture shown in Figure 4.2 and Table 4.1 was selected.

This posture allows the human to steer in different directions and enable the robot to acquire the angle between the robot heading direction and human, see Figure 4.3. This angle is read from the position encoder of the free rotational joint, whose zero is aligned with the robot heading direction when the arm is in this posture.

As can be seen in Figure 4.2, the end-effector height above the floor is about $0.72m$. It was selected based on the height of the human with whom the robot is working with.

The robot agent detects a moment signal from the force/moment sensor. Figure 4.4 illustrates the measure read by the sensor and used to generate the robot navigation

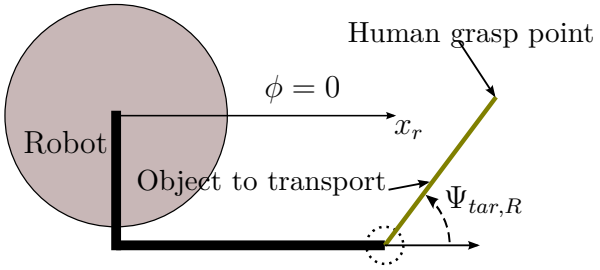


Figure 4.3: Human as Target

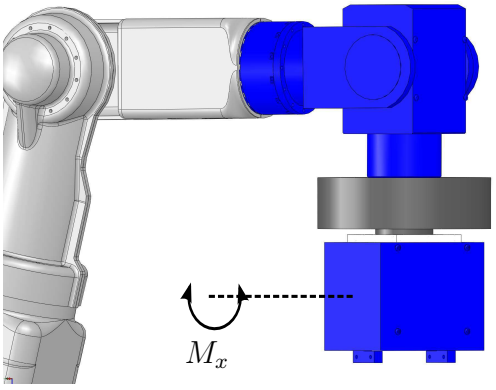


Figure 4.4: Moment Signal from force/moment sensor

path velocity.

The robot uses both the angle of human direction and this moment information, as non-verbal communication, to generate it's motion behavior to collaborate in the execution of the transportation task.

As previously described, the robot is able to sense the environment also through a laser range finder device. This device is used by the robot to acquire the obstacles that lies in it's navigation path and avoid them.

During the cooperative transportation task, it may occur several situations where an intended movement is not feasible to be performed by the robot. For instance, a passage between two obstacles, as can be seen in Figure 4.5, may be too narrow to the robot to pass between them. To avoid that the robot stays there indefinitely until the human partner realizes by himself the situation, the robot was endowed with a verbal communication mechanism. The robot is able to detect this occurrence and inform the human of it's own difficulties, see section 4.2.3. Others situations such as

a) when the human moves too fast and the robot is near it's maximum velocity

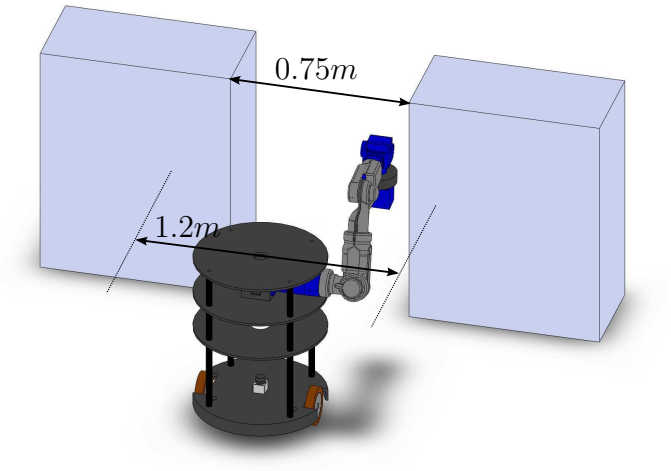


Figure 4.5: Passage too narrow for safety movements between obstacles

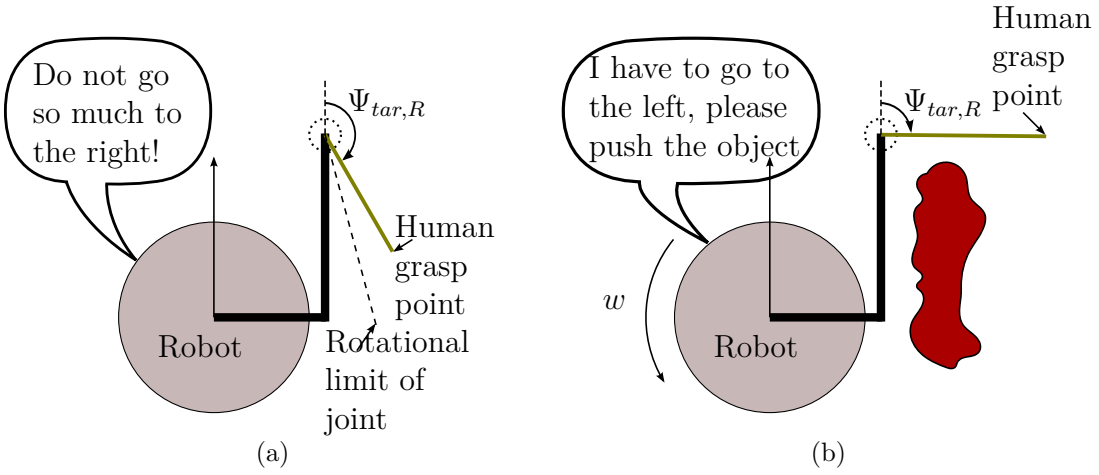


Figure 4.6: Problematic situations where verbal communication may enhance the execution of the task

or force/moment sensor limits;

b) the human turns too much to one side that the free rotational joint is near it's maximum or minimum position, see Figure 4.6a

c) because the vertical axis of rotation of the robot is not coincident (though parallel) with the vertical axis of rotation of free rotational joint, if the robot rotates too quickly to the right or to the left, it would induce the human to let the object to fall, since the human would feel a huge push or pull and not be prepared to such movement, see Figure 4.6b

can be detected by the robot and the human is alerted for such situations.

4.2 System design

The system developed here endows the mobile manipulator with a navigation functionality for object transportation in cooperation with a human in indoor environments, yet unstructured and changing.

As described in the previous section, only platform movements are considered in this project. To such movements, and following the methodology of the dynamical systems approach outlined in Chapter 2, the behavioral variables must to be defined first. Platform navigation can be expressed in terms of robot's angular velocity w and translational velocity v. Behavior constraints are usually defined by directions of objects (target or obstacles) relative to a world fixed frame of reference and restrictions on path velocity. Defining

$$\vec{x} = \begin{bmatrix} \phi \\ v \end{bmatrix} \tag{4.1}$$

as the behavioral variables allows a transparent method to derive the control variables for platform locomotion:

$$\frac{d\vec{x}}{dt} = \begin{bmatrix} \dot{\phi} \\ \dot{v} \end{bmatrix} \tag{4.2}$$

The angular velocity w may be obtained directly from equation (4.2), $w = \dot{\phi}$, and the translational velocity v obtained from integrating the acceleration $\dot{v}$ for each timestep.

From equations (2.1) and (4.2), we can obtain:

$$\begin{bmatrix} \dot{\phi} \\ \dot{v} \end{bmatrix} = \vec{f}(\vec{x}, parameters) = \begin{bmatrix} f(\phi, parameters) \\ g(v, parameters) \end{bmatrix} \tag{4.3}$$

If we keep the dynamics of ϕ and v independent of each other, the mathematical analysis of the dynamical systems will be simplified to two one-dimensional systems. Thus, the analysis of fixed-points and their stability will be simplified.

In the following subsections two dynamical systems are designed: one for heading direction and another for path velocity, respectively. In these subsections, $f(\phi, parameters)$ and $g(v, parameters)$ will be specified as $f(\phi)$, and $g(\phi)$, re-

spectively.

4.2.1 The dynamics of heading direction

Robot navigation in the environment may be constrained by targets and obstacles that define desired and to be avoided orientations for robot heading direction, see Figure 4.7. These task constraints define force-lets for the resultant vector field. Target direction ($\phi = \Psi_{tar}$) define an attractive force-let, while obstacles directions ($\phi = \Psi_{obs}$) define a repulsive force-let to the heading direction dynamical system. These angles are measured from a fixed world reference axis, so the contributions of the target and obstacles to this dynamical system does not depend on current robot heading direction, see Figure 4.8.

Regardless of the current heading direction of robot, the dynamical system will always tend to an asymptotically stable state (unless it is in an unstable state and no noise exist), thus leading the current heading direction of the robot to be always in or near a resulting attractor of the dynamics. As the robot moves (with translational velocity) the directions of the target and obstacles changes. This leads to a shift in the resulting attractor, pulling the heading direction along.

Target Acquisition

The behavior *Target Acquisition* is expected to align the robot's heading direction with the direction Ψ_{tar} of the target in the environment.

As previously described in Chapter 3, the last joint of the robotic arm is kept rotational free and it has an encoder attached to it. As can be seen if Figure 4.9 target direction relative to robot heading direction is obtained from this encoder. To get an angle relative to the world fixed axis reference, we need to sum $\Psi_{tar,R}$ to ϕ (current heading direction):

$$\Psi_{tar} = \phi + \Psi_{tar,R} \tag{4.4}$$

current heading direction is given by the digital compass device available in robot, which is referenced to the fixed world reference axis.

The target acquisition task is specified in the vector-field erecting an attractor at the orientation Ψ_{tar} and strenght λ_{tar}. Regardless of current robot heading direc-

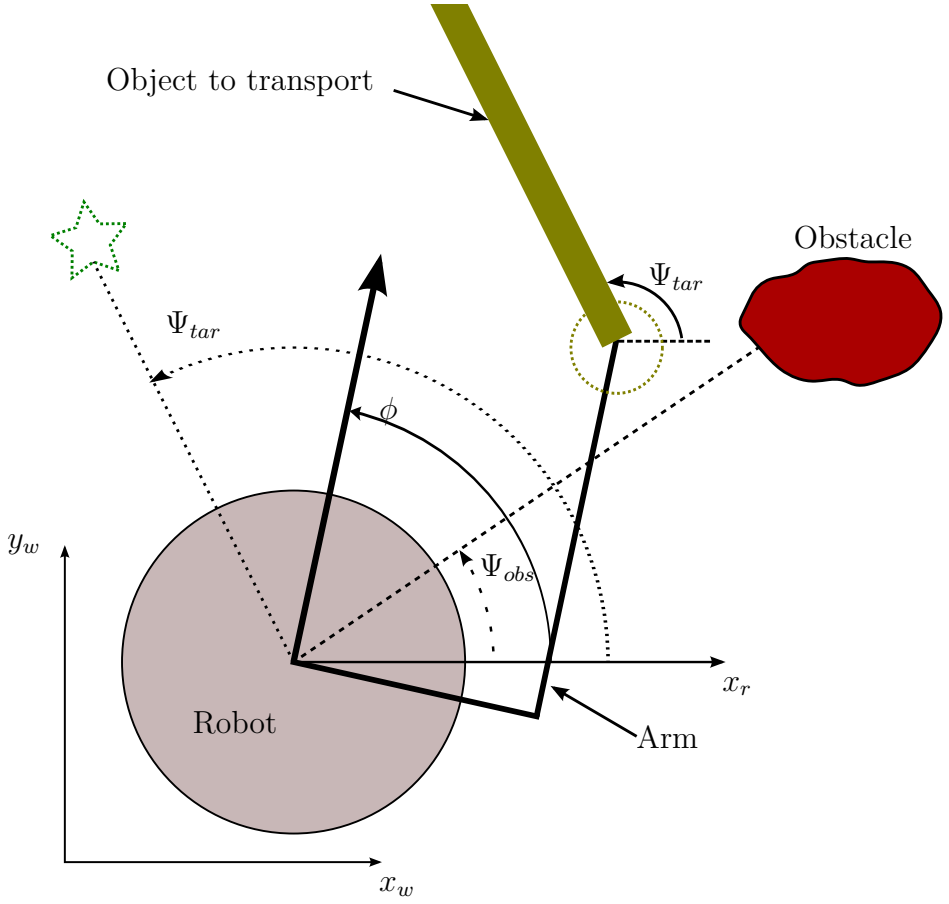


Figure 4.7: Target acquisition and obstacle avoidance task constraints. Ψ_{obs} is the direction at which the obstacles lies (to be avoided direction) from the current position of robot. Ψ_{tar} is the desired (target) direction for robot heading direction. This value is read from the arm last joint's encoder. As can be seen, this angle is the same in robot's center since the reference axis are kept parallel. The direction of x_r axis is kept parallel to a x_w during robot movements. This means that if the robot rotates about itself, the x_r axis will be parallel to x_w and so Ψ_{obs} and Ψ_{tar} will be constant. If robot moves with translational velocity, x_r will move with it, but parallel to x_w, and Ψ_{obs} and Ψ_{tar} will change accordingly

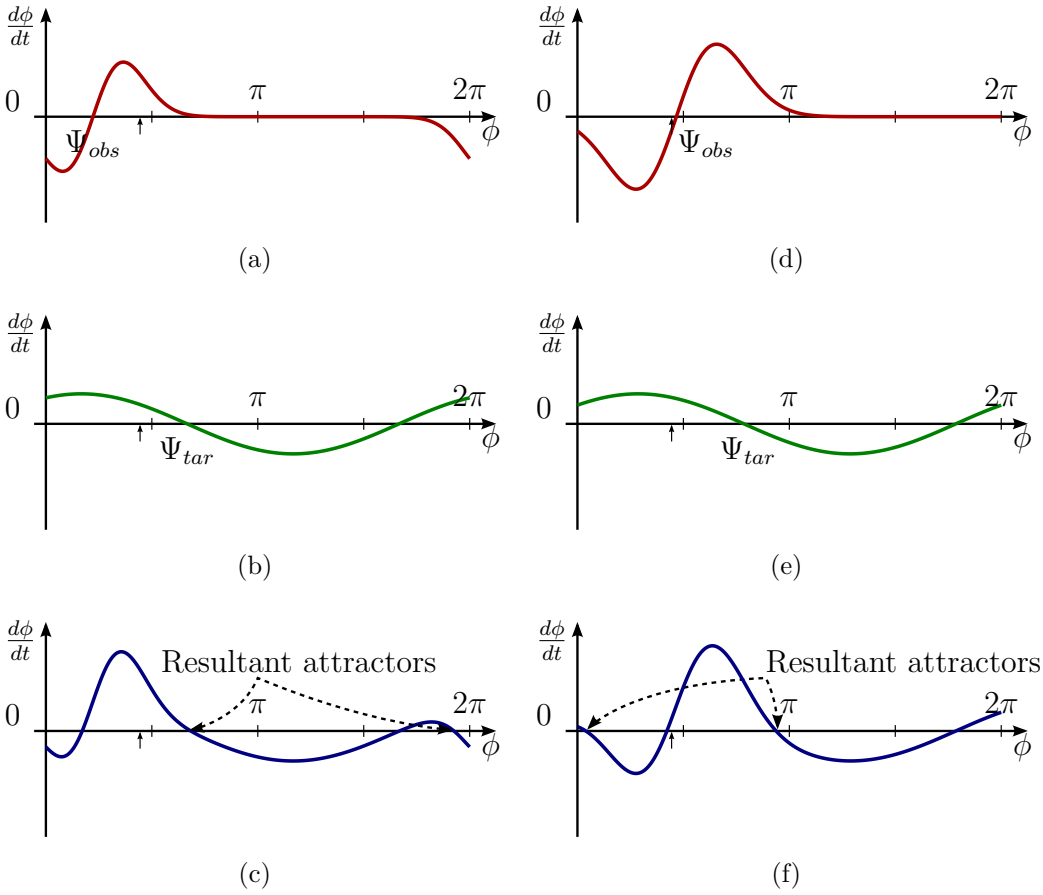


Figure 4.8: Resultant Attractors, (c) and (f), from the superposition of the repulsive force-let, (a) and (d) from obstacle constraint and attractive force-let, (b) and (e) due to target constraint. (a) to (c) are vector fields for Figure 4.7 for any robot heading direction. (d) to (f) are vector fields when target and obstacles are kept constant in the world and robot changes its position (shifting to the right).

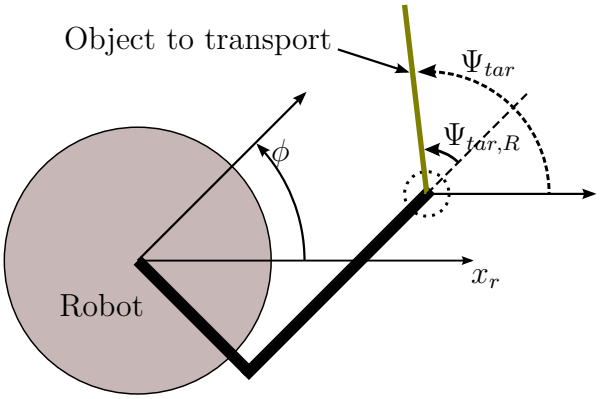


Figure 4.9: Relative angle between human and robot

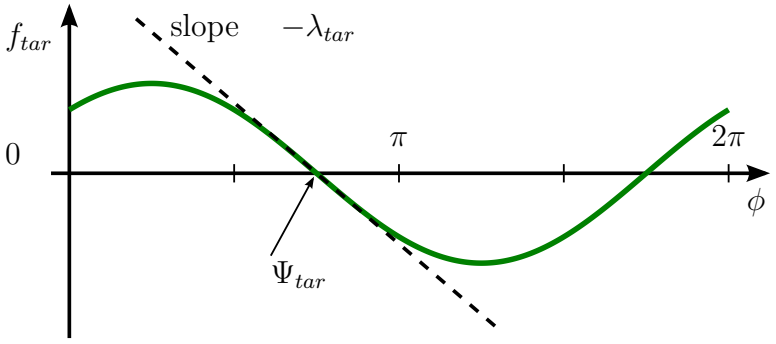


Figure 4.10: Dynamics of heading direction for target acquisition behavior. The slope $-\lambda_{tar}$ at the fixed-point $\dot{\phi} = 0$ defines an attractor at the direction Ψ_{tar}

tion, it is desired that the robot's behavior is to orientate towards this direction. Thus, this contribution should exhibit an attractive force over the entire range of heading direction, which is the full polar circle (0 to 2π). The mathematical form

$$\frac{d\phi}{dt} = f_{tar}(\phi) = -\lambda_{tar} \sin(\phi - \Psi_{tar}) \tag{4.5}$$

erects an attractor (asymptotically stable state) at Ψ_{tar} direction and an repeller (unstable state) at $2\pi - \Psi_{tar}$. A plot of this dynamical system in phase space can be seen in Figure 4.10. If we analyze equation (4.4) and the argument of the sin function in equation (4.5) we can see that this dynamical system does not depend on the current heading direction ϕ of robot, since

$$\begin{aligned} \phi - \Psi_{tar} &= \phi - (\phi + \Psi_{tar,R}) \\ &= -\Psi_{tar,R} \end{aligned} \tag{4.6}$$

where $\Psi_{tar,R}$ is acquired directly from the encoder of the free rotational joint. Since ϕ cancels out, target orientation does not need to be known relative to an external fixed world frame of reference and it is not influenced by calibration errors of digital compass.

Obstacle Avoidance

The obstacle avoidance behavior is expected to steer the robot away from obstacles that lie in robot navigation path. Dumbo robot is equipped with a laser range finder and software was developed which allows the selection of the number of sectors desired and the range detection. Each sector gets a fixed direction, θ_i

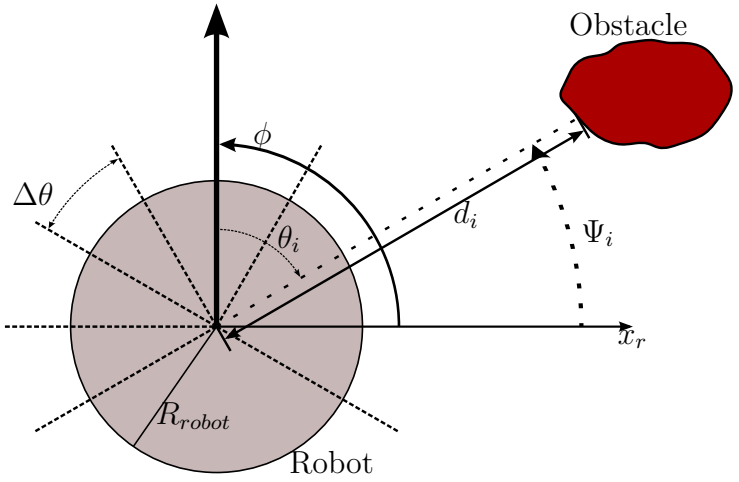


Figure 4.11: Parameters of obstacle avoidance behavior. Obstacles in robot navigation path are gathered in n sectors. Each sector i is at an angle θ_i relative to robot heading direction ϕ and measures a distance d_i to objects in direction $\Psi_i = \phi - \theta_i$ relative to a fixed world reference axis. The angular resolution is $\Delta\theta$ and the robot's radius is R_{robot}

relative to the robot heading direction and an $\Psi_i = \phi + \theta_i$ relative to the world reference axis. i denotes the number of the sector, ranging from $i = 1 \ldots n$, where n is the selected number of sectors, see Figure 4.11. For each sector, the dynamics should erect a repeller at the direction Ψ_i, see Figure 4.12:

$$\begin{aligned} f_{obs,i} &= \lambda_{obs,i}\,(\phi - \Psi_i)\exp\left[-\tfrac{(\phi-\Psi_i)^2}{2\sigma_i^2}\right] \quad , \quad i = 1 \ldots n \\ &= \lambda_{obs,i}\,(-\theta_i)\exp\left[-\tfrac{(-\theta_i)^2}{2\sigma_i^2}\right] \quad , \quad i = 1 \ldots n \end{aligned} \tag{4.7}$$

As in equation (4.5) for target acquisition behavior, also here only the relative

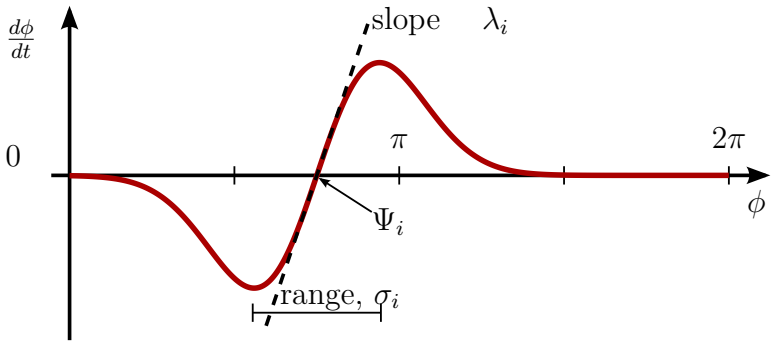


Figure 4.12: Dynamics of heading direction for single obstacle. The slope $\lambda_{obs,i}$ at the fixed point $\dot{\phi} = 0$ defines a repeller at the obstacle direction Ψ_i

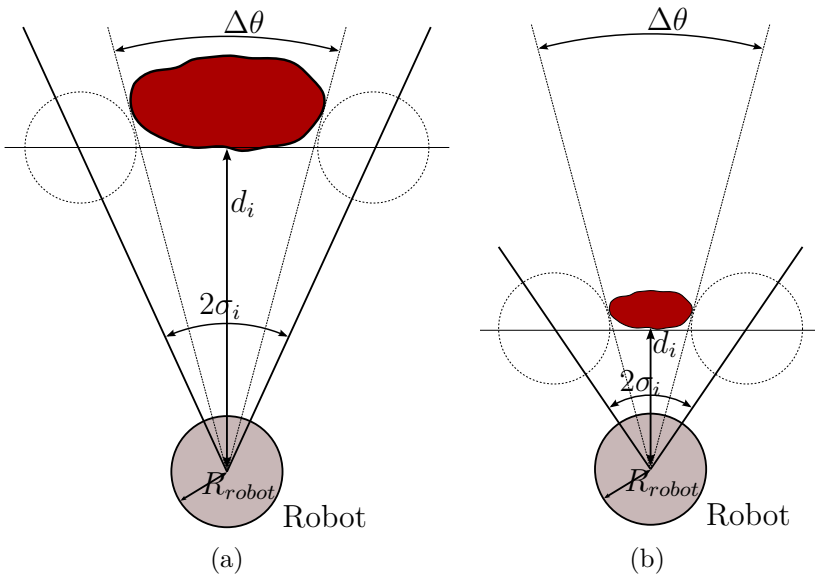


Figure 4.13: The angular range σ_i of repulsive force-let is a function of angular range of sector, $\Delta\theta$, and a safety margin on each side of obstacle

orientation θ_i of each sector i to robot's heading direction ϕ enters into the dynamics of heading direction. In equation (4.7), $\lambda_{obs,i}$ is the strength of repulsion and it's magnitude is an exponential function of the distance to the obstacles in direction Ψ_i:

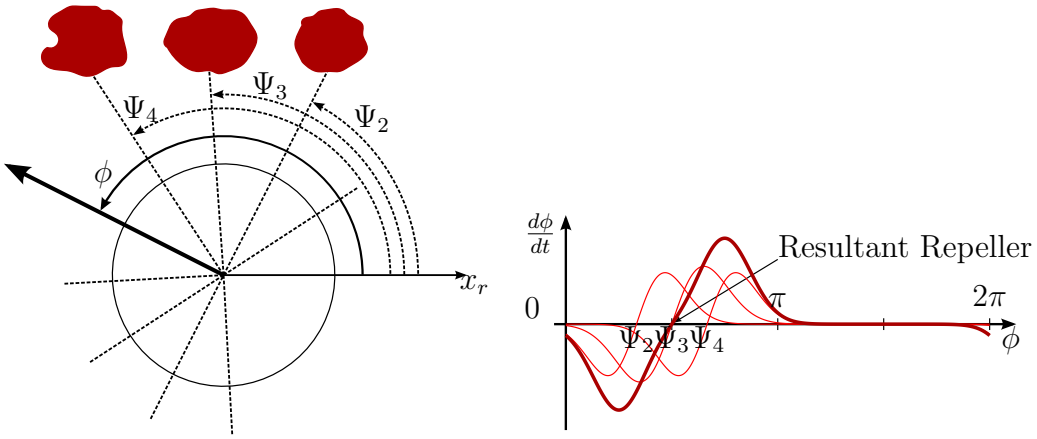
$$\lambda_{obs,i} = \beta_1 \exp\left[-\frac{d_i}{\beta_2}\right] \tag{4.8}$$

The exponential makes that the heading direction to be repelled strongly from near obstacles and weakly from far obstacles with a rate of decay with the increasing distance β_2. The parameter β_1 controls the maximum repulsion strength.

Opposite to target acquisition, the range over which the heading direction should be repelled is not the full circle. The range σ_i is a function of the distance to obstacle d_i, the angular range of each sector $\Delta\theta$ and robot radius:

$$\sigma_i = \arctan\left[\tan\left(\frac{\Delta\theta}{2}\right) + \frac{R_{robot}}{d_i}\right] \tag{4.9}$$

Within the angular range $\Delta\theta$, the sector does not define the exact direction of the object, thus, to avoid collisions, it is assumed that the object covers the entire angular range $\Delta\theta$. An angular safety margin is added to ensure a passage next to the obstacle without contact, see Figure 4.13. The final avoidance dynamics is



(a) Three different contributions to resultant vector field

(b) When contributions are summed, a resultant repeller is erected which angular range covers the three contributions

Figure 4.14: Dynamics of heading direction for obstacle avoidance behavior. The sum of all contributions may have a single resultant repulsive force-let as in (a) and (b) or, if the obstacles are far enough, multiple repellers for different Ψ_i.

obtained from the sum of the contribution of each sensor $i = 1 \ldots n$:

$$\frac{d\phi}{dt} = F_{obs}(\phi) = \sum_{i=1}^{n} f_{obs,i}(\phi) \tag{4.10}$$

The Figure 4.14 illustrate the obstacle avoidance heading dynamics for a case where three sectors detect an object. Despite the dynamical contribution of each sector erects a repulsive force-let at Ψ_2 Ψ_3 Ψ_4, the resultant dynamics has a single repeller which covers the angular range of those separate contributions. This result leads the robot to avoid the three obstacles as if there was a single one occupying the same space.

Integrating the two behaviors

The overt behavior of the robot is obtained from the following dynamical system:

$$\frac{d\phi}{dt} = f_{robot}(\phi) = -\lambda_{tar} \sin(\phi - \Psi_{tar,obs}) + f_{stoch} \tag{4.11}$$

Where $\Psi_{tar,obs}$ is the desired direction for the robot. $\Psi_{tar,obs}$ is given by summing

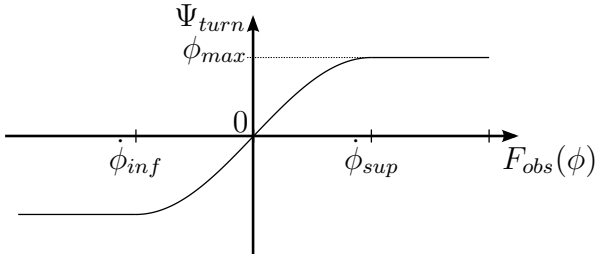


Figure 4.15: Sigmoid function for obstacles contribution

the target direction, Ψ_{tar}, and an angle function of obstacles contributions:

$$\Psi_{tar,obs} = \Psi_{tar} + \Psi_{turn} \tag{4.12}$$

The angular value Ψ_{turn} is obtained from a sigmoid function that rises smoothly from an inferior limit, $\dot{\phi}_{inf}$, to a superior limit, $\dot{\phi}_{sup}$, between a symmetric threshold value, ϕ_{max}, see Figure 4.15:

$$\begin{aligned} \Psi_{turn} &= \sigma_{\dot{\phi}_{inf},\dot{\phi}_{sup}}\left(F_{obs}\left(\phi\right)\right) \\ &= \begin{cases} -\phi_{max} & \text{if} \quad F_{obs}\left(\phi\right) \leq \dot{\phi}_{inf} \\ -\phi_{max}\cos\left(\pi\frac{F_{obs}(\phi)-\dot{\phi}_{inf}}{\dot{\phi}_{sup}-\dot{\phi}_{inf}}\right) & \text{if} \quad \dot{\phi}_{inf} < F_{obs}\left(\phi\right) < \dot{\phi}_{sup} \\ \phi_{max} & \text{if} \quad F_{obs}\left(\phi\right) \geq \dot{\phi}_{sup} \end{cases} \end{aligned} \tag{4.13}$$

$\dot{\phi}_{inf}$, $\dot{\phi}_{sup}$ and ϕ_{max} are design parameters.

A stochastic force is added to ensure that the robot is able to escape from unstable fixed points (repellers):

$$f_{stoch} = \sqrt{Q}\xi_n \tag{4.14}$$

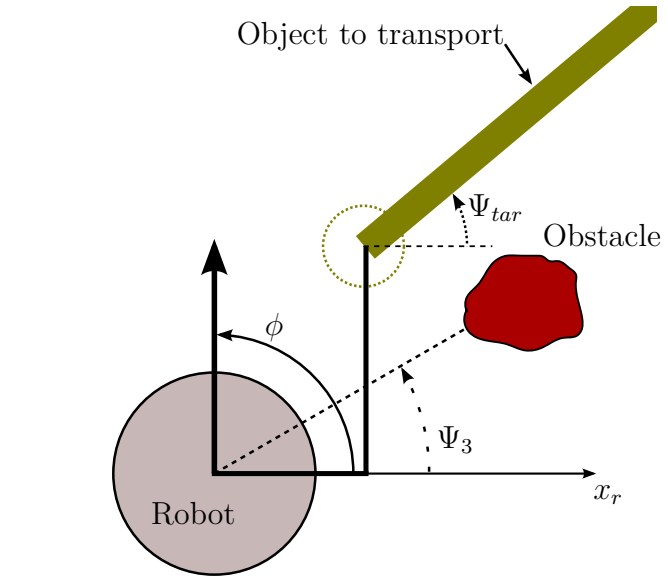
where ξ_n is Gaussian white noise of unit variance, so that Q is the effective variance of force.

The sigmoid function and the integration of obstacles contribution in target dynamics allows to the control to take into account the human direction even in presence of obstacles.

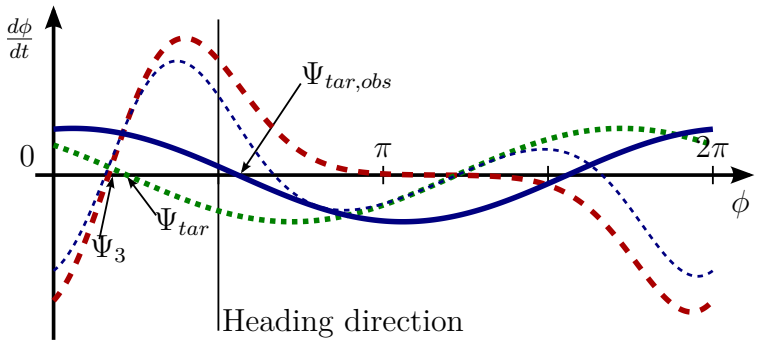
If a simpler integration of the two behaviors such as

$$\frac{d\phi}{dt} = F_{obs}\left(\phi\right) + f_{tar}\left(\phi\right) + f_{stoch} \tag{4.15}$$

would be used, when obstacles are near to the robot, very strong repellers would be erected and the target contribution would be easily superseded by obstacles



(a)



(b)

Figure 4.16: Integration of Target and Obstacles behaviors.

contribution, leading the robot to avoid the obstacles without taking into account the human direction in its behavior.

Figure 4.16 illustrates the integration of the two behaviors. The sum of obstacles contributions, dashed red line, erect two different repulsive force-let. Target acquisition dynamics contribution is represented by the dotted green line. The resultant non-linear dynamical system, solid blue line, has the same shape of target acquisition, but the attractor is shifted by the obstacles contribution. The thiner solid blue line is the resultant heading direction dynamics for equation (4.15).

4.2.2 The dynamics of path velocity

To completely define the time courses of robot behavioral variables, a dynamical system for path velocity should be specified. Every movement of the robot leads to a shift in attractors and repellers, since the sensory information changes with the moving robot. This shift also occurs when the object (target or obstacles) are moved. Despite this movements, the system must remain stable. In other words, the robot's heading direction should be in or near an attractor at all times (Bicho, 2000). Such task may be accomplished by controlling the path velocity v of the mobile platform:

$$\frac{dv}{dt} = g(v) = -c_{obs}(v - v_{obs}) - c_{tar}(v - v_{tar}) \tag{4.16}$$

The desired velocity is controlled such that the presence of obstacles influences the velocity contribution of the target. c_i ($i = tar$ or obs) indicates the strength of each contribution. A potential function indicates the presence of obstacles contributions:

$$U(\phi) = \sum_{i=1}^{n} \left(\lambda_{obs,i}\sigma_i^2 \exp\left[-\frac{(\phi - \Psi_i)^2}{2\sigma_i^2} \right] - \frac{\lambda_{obs,i}\sigma_i^2}{\sqrt{e}} \right) \tag{4.17}$$

When $U(\phi)$ is negative, no obstacles were detected, or the robot heading direction is outside the repulsion zone. The robot must then navigate with a velocity value given by the force/moment sensor, $c_{obs} = 0$ and $c_{tar} > 0$. A positive value of $U(\phi)$ indicates that the robot heading direction is inside a repulsion zone, and the robot must take into account the presence of obstacles. Now $c_{obs} > 0$ and $c_{tar} = 0$ is required and path velocity (v_{obs}) is then controlled by the minimum between the velocity to follow the human (proportional to force/moment readings) and a velocity function of the distance to obstacles:

$$v_{obs} = \min\left(\frac{d_{min}}{T_{2c,obs}}, v_{tar} \right) \tag{4.18}$$

where d_{min} is the distance to the nearest obstacle and $T_{2c,obs}$ is a parameter defining the time to contact with the obstacle. The velocity dynamics presented in equation (4.16) guarantees smooth transitions between the velocities.

c_{obs} and c_{tar} are controlled by a relaxation rate parameter, $c_{v,obs}$ and $c_{v,tar}$ respec-

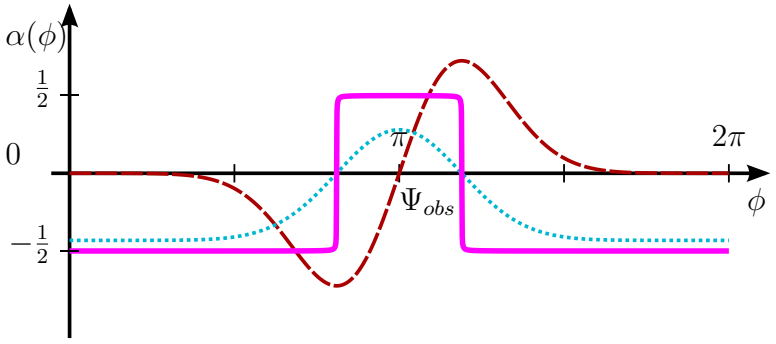


Figure 4.17: Sigmoid threshold function of potential $U(\phi)$. The dashed red line is a repulsive force-let, $f_{obs(\phi)}$, which defines a repeller at Ψ_{obs} direction. The potential function, $U(\phi)$, doted cyan line, indicates the presence of obstacles. A sigmoid threshold function, $\alpha(\phi)$, solid magenta line transforms the levels of the potential function into a range from $-1/2$ to $1/2$

tively, and a sigmoid threshold function

$$\alpha(\phi) = \arctan\left[cU(\phi)\right]/\pi \tag{4.19}$$

which transforms the levels of the potential function to value ranging from $-1/2$ and $1/2$ (see Figure 4.17). This result is feed to the equations:

$$c_{obs} = c_{v,obs}\left(1/2 + \alpha(\phi)\right) \tag{4.20}$$

$$c_{tar} = c_{v,tar}\left(1/2 - \alpha(\phi)\right) \tag{4.21}$$

where the strengths to both velocity contributions are obtained. To obtain a sharp transition in velocity behavior, a large value c should be used in equation (4.19). The following hierarchy of relaxation rates ensures that the system relaxes to the attractors and that obstacles avoidance has precedence over the target contribution:

$$\lambda_{tar} << c_{v,tar} \qquad \lambda_{obs} << c_{v,obs} \qquad \lambda_{tar} << \lambda_{obs} \tag{4.22}$$

4.2.3 Speech

The dynamical systems presented in previous sections endows the robot with an autonomous control for a safe navigation in a cooperative object transportation task context. Nevertheless, certain movements required by the human can not be performed by the robot. When the human enters in a passage, the robot may

not be able to follow him if the passage is too narrow for a safe navigation. The previously proposed architecture does not allow the robot to follow the human in such situation, since the path velocity gets low enough to the robot not move with translational velocity and heading direction dynamics tries to align the robot heading direction with the passage.

This results in a dead end, since no further movement is made by the robot. To avoid such situation, the robot was endowed with a speech synthesis mechanism to allow the robot to alert the human.

Such situations are detected by a set of conditions applied to the results of the dynamical systems.

When the robot gets trapped in a too narrow passage situation, the following conditions are verified:

$$narrow\_passage = \text{abs}\left(\alpha\left(\phi\right)\right) > 0 \wedge \text{abs}(\dot{\phi}) < \dot{\phi}_{min} \tag{4.23}$$

where $\alpha\left(\phi\right)$ is the sigmoidal threshold function, see equation (4.19), $\dot{\phi}$ is the current robot angular velocity and $\dot{\phi}_{min}$ is a design parameter. When the condition $\text{abs}\left(\alpha\left(\phi\right)\right) > 0$ is true, it means that the current heading direction is in a repulsive zone, so, obstacles are sensed near the robot heading direction and the robot's movements are restricted. The heading direction dynamics tries to align the robot with the passage, when it succeeds, the value of $\dot{\phi}$ is near zero and robot does not move (or moves slightly). The moment that these conditions are verified, the value of *narrow_passage* is true, and the sentence *"Wait! This passage is too narrow for me!"* is synthesized by the robot. The human can then take actions (e.g. going back) to continue his task and search another path to get to destiny.

In addition to the two conditions specified in equation (4.23), a third condition is added. The moment when the robot speaks is internally stored and such sentence will be synthesized again if this moment was more than a pre-specified time in past. This value is also a design parameter.

The condition of the equation (4.24) is used to detect the situations when the human is moving too fast and the robot can not follow him:

$$too\_fast = \text{abs}(v_{force} - v) > \Delta v_{max} \tag{4.24}$$

where v_{force} is the velocity attractor (when no obstacles are present) of the path

velocity dynamics, which is proportional to the force applied by the human to the robot end-effector. v is the current path velocity and Δv_{max} is a design parameter. When the distance between the desired velocity and actual velocity is higher than Δv_{max} threshold value, the sentence *"Wait! Go slower! I can not move so fast!"* is synthesized by the robot.

With equations (4.25) and (4.26) the robot is able to detect if the human (or the robot itself) movement is taking the free rotational joint getting to the limit.

$$theta\_6\_upper\_limit = \theta_6 > \theta_{6,USL} \wedge \frac{d\theta_6}{dt} \geq 0 \tag{4.25}$$

$$theta\_6\_lower\_limit = \theta_6 < \theta_{6,LSL} \wedge \frac{d\theta_6}{dt} \leq 0 \tag{4.26}$$

where θ_6 is the angle value of the free rotational joint, $\theta_{6,USL}$ and $\theta_{6,LSL}$ are the joint 6 upper safe limit and lower safe limit, respectively. When θ_6 is near the upper/lower safe limit and it's value is increasing/decreasing, the sentences *"Do not go so much to the left"* and *"Do not go so much to the right"*, respectively, are synthesized by the robot.

As for the narrow passage condition in equation (4.23), also a time restriction is applied to the too fast, equation (4.24), and joint 6 limits, equations (4.25) and (4.26), conditions. This way, the robot will only repeat itself after a period of time.

Chapter 5

Results and Discussion

In this chapter we present some experimental results of the tests performed in two scenarios that challenge the Human-Mobile Manipulator object transportation task. The first scenario is an entrance hall, were objects were placed around in order to test the overt behavior of the robot in a complex environment. A corridor scenario is presented afterwards, where the human and robot transport a long object along this corridor to a laboratory room.

In both environments, static and dynamic obstacles are in the robot's navigation path. As described in the strategy adopted, the path planning or goal position is not given to the robot. It must follow the human and help him transport the object without letting it fall.

The snapshots sequences presented below, were obtained from three videos that can be downloaded from the Mobile and Anthropomorphic Robotics Laboratory web server[1] or started directly by pressing these links, Entrance Hall and Corridor, if the media is locally available.

5.1 Scenario 1: Entrance Hall

The scenario layout of the entrance hall is presented by means of two video snapshots sequences from two different perspectives. One capturing the movements of human-robot object transportation task for the first half of the transportation task

[1] http://marl.dei.uminho.pt/Public/Human-Robot_Object_Transportation/

(a) Perspective capturing the start of the task

(b) Perspective capturing the end of the task

Figure 5.1: Entrance Hall videos perspectives

and another capturing, essentially, the second half and end. These perspectives are presented in Figure 5.1a and 5.1b, respectively.

In this section, the perspective shown in Figure 5.1a will be referenced as A and Figure 5.1b as B. In addition to these two video perspectives, the dynamics of heading direction and path velocity are presented for each snapshot.

The Figure 5.2 is organized as follows: In the left side of the figure are the snapshots of the video at the time instant in it's lower right corner. On the right side, are presented both heading direction and path velocity dynamics for the same instant. Left and lower axis are relative to heading direction, $d\phi/dt$ and ϕ, respectively. Right and upper axis are relative to path velocity dynamics, dv/dt and v, respectively. Vertical solid line and dashed line are heading direction and path velocity current values, respectively. Sinusoidal dotted green curve is the target acquisition contribution. Red dashed curve is the obstacles contribution and solid blue sinusoidal curve is the integration of both behaviors. Solid linear blue line is the path velocity dynamics. Solid magenta sigmoidal curve is the sigmoidal threshold function, see equation (4.19). ϕ units is radians, while v is m/s. Attractors and reppelers are represented by circles.

This trail starts with the robot being positioned in the start position and aligned with the human, as depicted in Figure 5.2 at the instant t=0s. Initially the robot does not sense any obstacles, thus obstacles contribution is zero. The robot is already aligned with the human partner and the latter is not making any movement, thus the behavioral variable of the heading direction dynamics is already relaxed at the attractor, see Figure 5.2b.

From the instant t=0s to t=14s, the human performs a movement in a straight

line which makes the robot to follow the human, keeping the rotation near zero, since no obstacles are sensed, and the linear velocity is proportional to the force exerted by the human.

At the instant t=14s (Figure 5.2c), the human goes through a passage that is large enough for the human but too narrow for the robot. In the same instant, the mobile manipulator starts to sense obstacles in it's navigation path. Their contributions leads to a decrease in the robot's speed, which is characterized by a value of 0.5 in the sigmoidal threshold function (magenta solid line in Figure 5.2d) at the robot heading direction (solid vertical line). The desired velocity for the linear translation of the robot is then the minimum between the velocity to follow the human and a velocity function of the distance to the obstacles, see equation (4.18).

As the human continues his movement between the two obstacles, the robot approaches the obstacles and detects, at instant t=24s, that the passage between these obstacles is too narrow for a safe movement. The robot then verbally alerts the human for such situation, synthesizing the sentence: "Wait! This passage is too narrow for me!". Figure 5.2f depicts the heading direction dynamics and path velocity in this situation.

The human acknowledges the robot's difficulty and starts to push the object. The robot, then, moves backwards (see snapshots of instants t=32s and t=38s in the Figures 5.2g and 5.2i).

At the instant t=38s, Figure 5.2i, the human starts to change his path direction, to go around the obstacle, in order to get to the destination. Obstacle contributions to the heading direction dynamics is not present, since obstacles are further away than the range of the laser range finder.

From instant t=38s to t=46s, the human changes its position, shifting the attractor along with it. As can be seen in Figure 5.2k, 5.2m and 5.2s, the robot rotates about itself to align with the human with the behavioral variable (robot's heading direction) follows very closely the moving attractor.

At the instant t=48s, an obstacle is thrown to the robot navigation path, see sequence snapshots from Figure 5.2q to 5.2v, which strongly influences the path velocity and heading direction.

The obstacle appears from the left of the robot (right side of Figure 5.2r) and the resultant direction, to be avoided (repeller) shifts with obstacle movement. When

the obstacle is near the current heading direction of the robot, linear velocity is now governed by the distance to this obstacle. Since the human and the mobile manipulator were moving with a relatively high velocity, the robot can not continue to navigate with such velocity and alerts the human partner to the situation, synthesizing "Wait! Go slower! I can not move so fast!". The human acknowledges this instruction, reducing the exerted strength, and allows the robot to safely avoid the obstacle, see snapshots sequence from instant t=53 to t=1:04m, Figures 5.2w to 5.2ab.

Afterwards, the human went through a wider passage to achieve the goal. Around instant t=1:12m, the robot was entering this passage. As can be seen in Figure 5.2ad two resultant repellers of the obstacles contribution are separated enough, thus it is safe for the robot to keep its movement and follow the human.

As the human continues his movement, the target acquisition dynamics tries to align the robot's heading direction with the human and obstacles avoidance dynamics tries to steer the robot through the passage without collision with near walls. As can be seen in instant t=1:16m, Figures 5.2ae and 5.2af, two strong repellers are erected at approximatively $90°$ and $-90°$ from current heading direction. Those are the resultant contributions of both walls . Since the contribution of each wall is approximately the same, a resultant attractor is erected at the direction that keeps the robot equally spaced from both walls.

In the same instant, the human starts to turn around the corner, shifting the attractor from the target acquisition dynamics. Since the robot can not follow him due to the left wall, the robot keeps his heading direction unchanged (behavioral variable is sitting in an attractor) until it is safe to turn (see Figure 5.2ah). At this instant, t=1:22m, the robot also starts to turn around the corner, steering to the human direction. After the corner, a new passage with walls on both sides is encountered, see Figure 5.2ai.

The task ends with the human turning in the direction of a room, t=1m:36s (see Figure 5.2ak). The mobile manipulator follows the human, see t=1m:43s, and the task is finished a few seconds later.

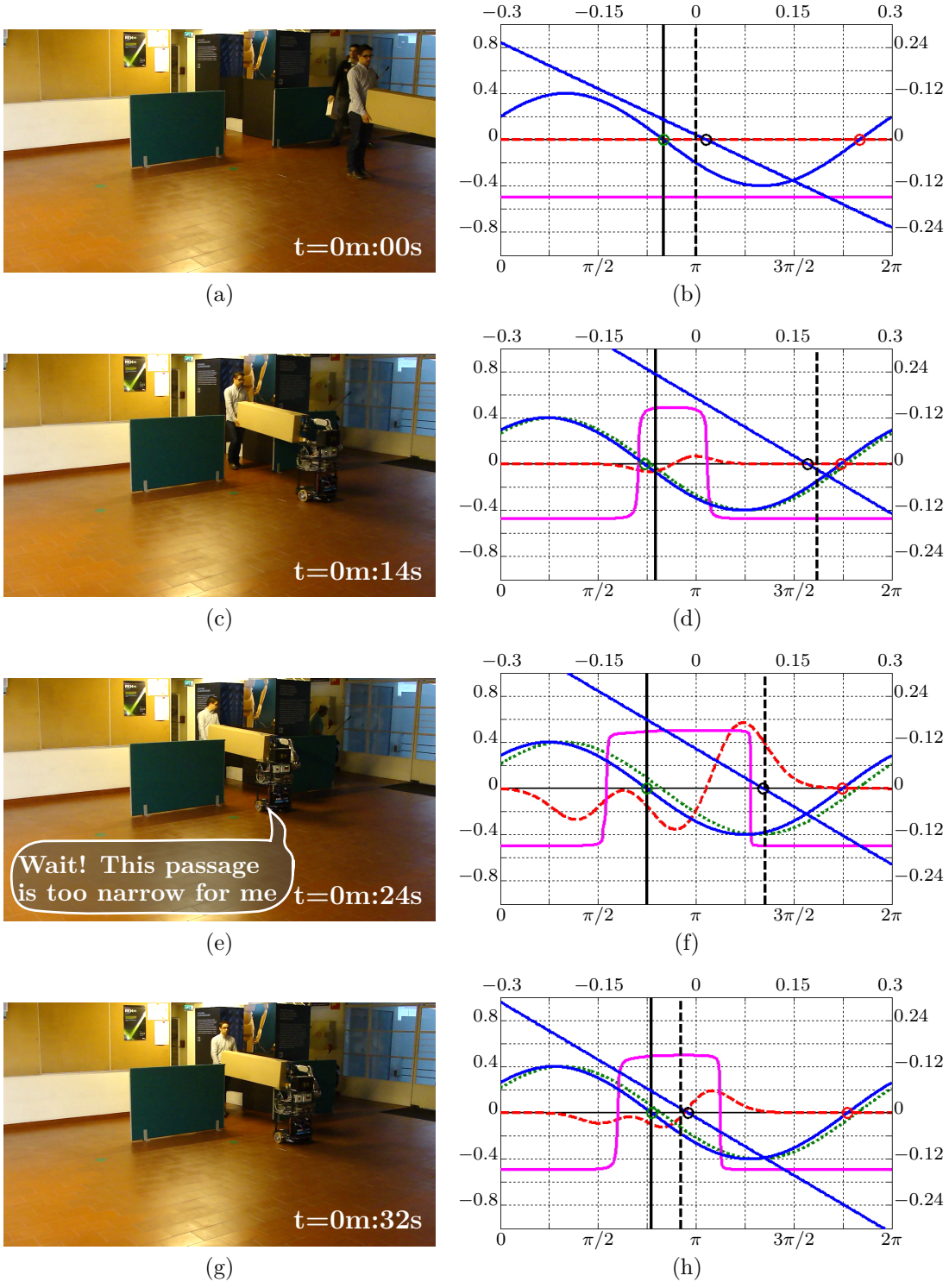
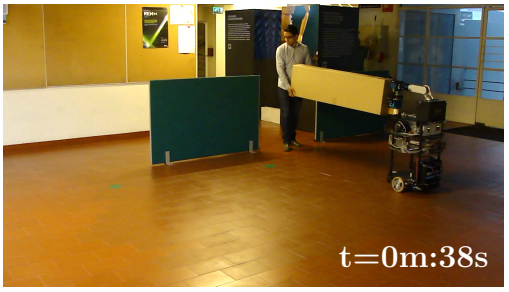
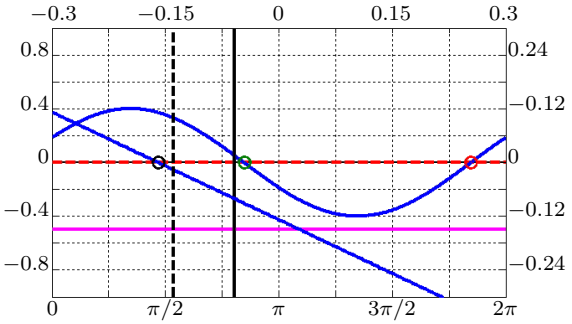


Figure 5.2: Snapshots sequence and dynamics for entrance hall scenario. Left side: snapshots at the specified time. On the right side, are presented both heading direction and path velocity dynamics for same instant. Left and lower axis are relative to heading direction, $d\phi/dt$ and ϕ, respectively. Right and upper axis are relative to path velocity dynamics, dv/dt and v, respectively. ϕ units is radians, while v is m/s.

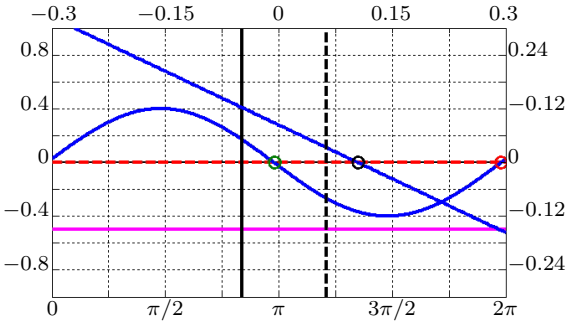


(i)



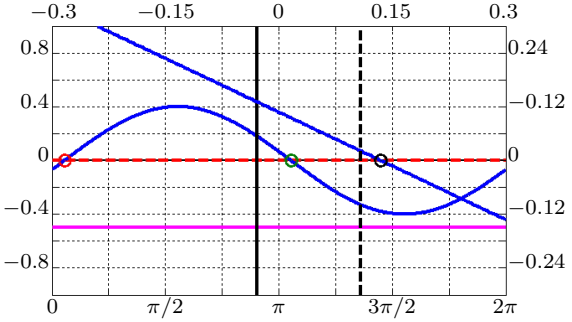
(j)

(k)



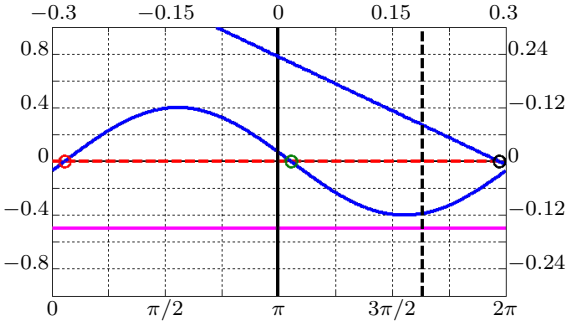
(l)

(m)



(n)

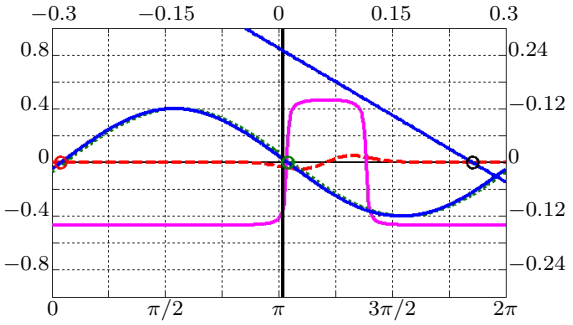
(o)



(p)

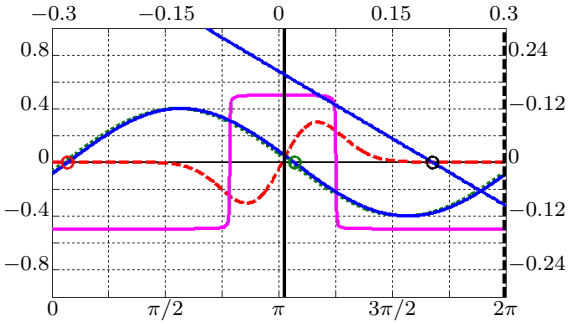
Figure 5.2: Continued.

(q)



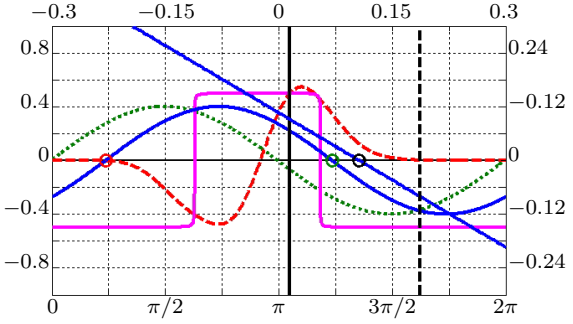
(r)

(s)



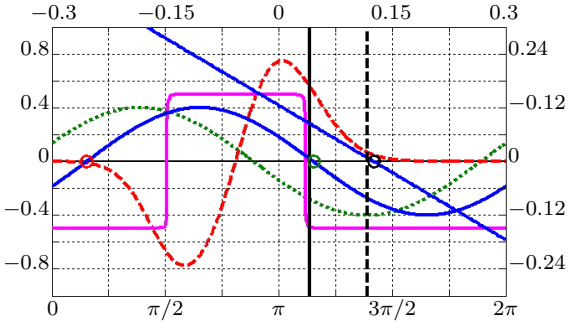
(t)

(u)



(v)

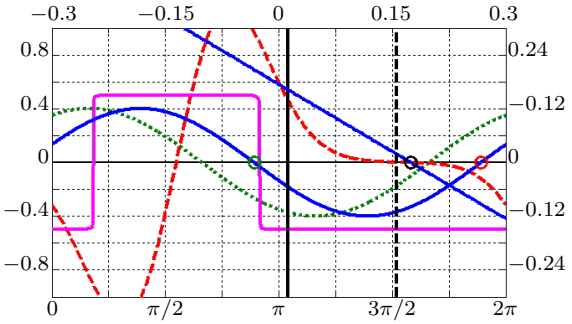
(w)



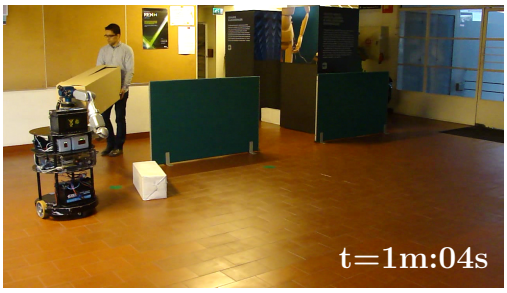
(x)

Figure 5.2: Continued.

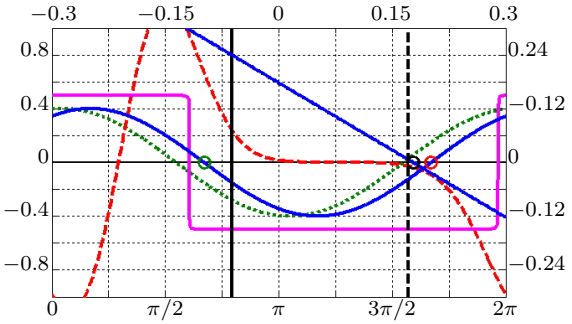
(y)



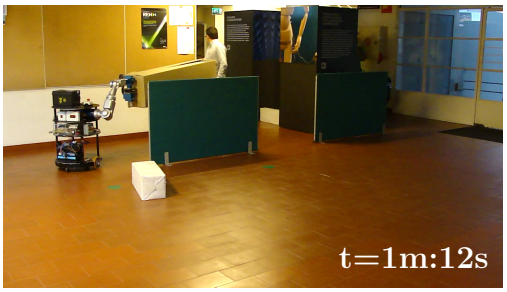
(z)



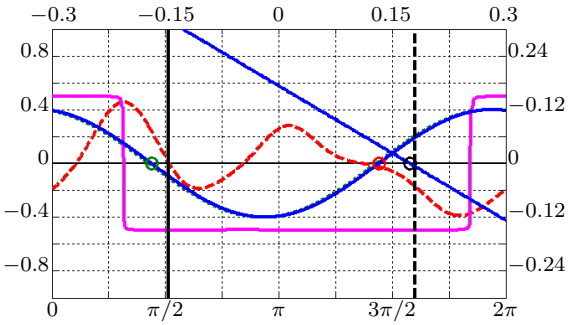
(aa)



(ab)

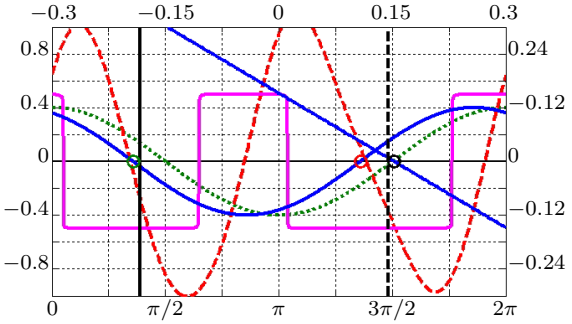


(ac)



(ad)

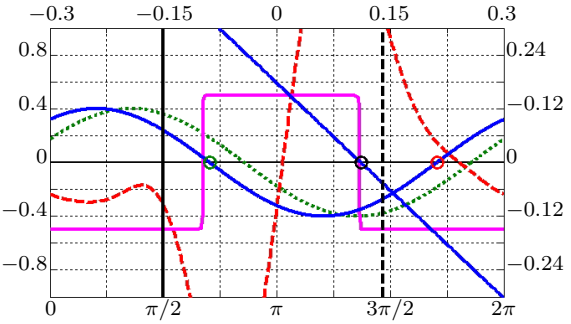
(ae)



(af)

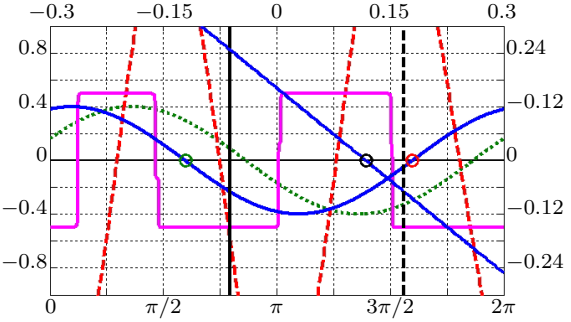
Figure 5.2: Continued.

(ag)



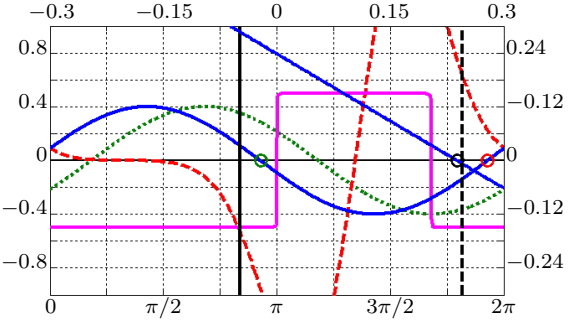
(ah)

(ai)



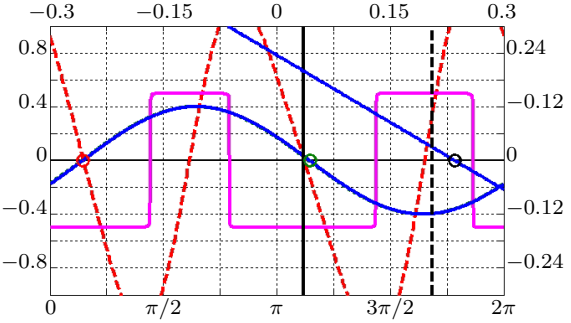
(aj)

(ak)



(al)

(am)



(an)

Figure 5.2: Continued.

5.2 Scenario 2: Corridor

The second scenario is a narrow corridor, although wide enough for the robot to move. The human and mobile manipulator are already cooperatively transporting the object from a room to another. The presented snapshots show the sequence of movements at key moments, from the halfway of the corridor until the desired location, inside the room.

At time t=0s (t=0 is relative to the first snapshot presented), see Figure 5.3, the transportation task is being carried at cruise velocity (near the maximum linear velocity of the mobile platform). As previously shown, two strong repellers are erected in the walls' directions. Since the corridor is wide enough, no repeller is erected in the robot's heading direction and the robot can follow the human. Around t=8s (Figure 5.3c), two dynamic obstacles (two humans) are sensed by the robot. It tries to go around the humans, but the passage between the wall and the humans is not wide enough for the robot to go through, instant t=10s. The robot synthesizes: "Wait! This passage is to narrow for me!". The first human acknowledges it, and tries to get out of the robot's navigation path. Since the robot has now enough space to pass, the task continues, instant t=14s, Figure 5.3h.

Around instant t=24s, the human partner starts to enter the room, while increasing its speed, at the same time, see Figure 5.3i. The robot tries to follow the human, but it is still in the corridor, so it can not follow the human's trajectory and velocity, since it needs, first, to approach the door. Hence, the robot alerts verbally the human to the situation, synthesizing: "Wait! Go slower! I cannot move so fast!". The human partner decreases his speed and the robot can now approach the entrance, without letting the object fall, and keep following the human, instant t=27s. A few seconds later the task is finished.

5.3 Stability

With the moving robot, the directions to the obstacles and the target in the world change, thus the resulting attractor of the heading direction dynamics shifts. Figures 5.4a and 5.5a show the time courses of the attractor solutions for heading direction dynamics, for scenario 1 and 2, respectively, and how the robot's heading direction tracks it. In Figures 5.4b and 5.5b the attractor solutions to the velocity dynamical system are presented.

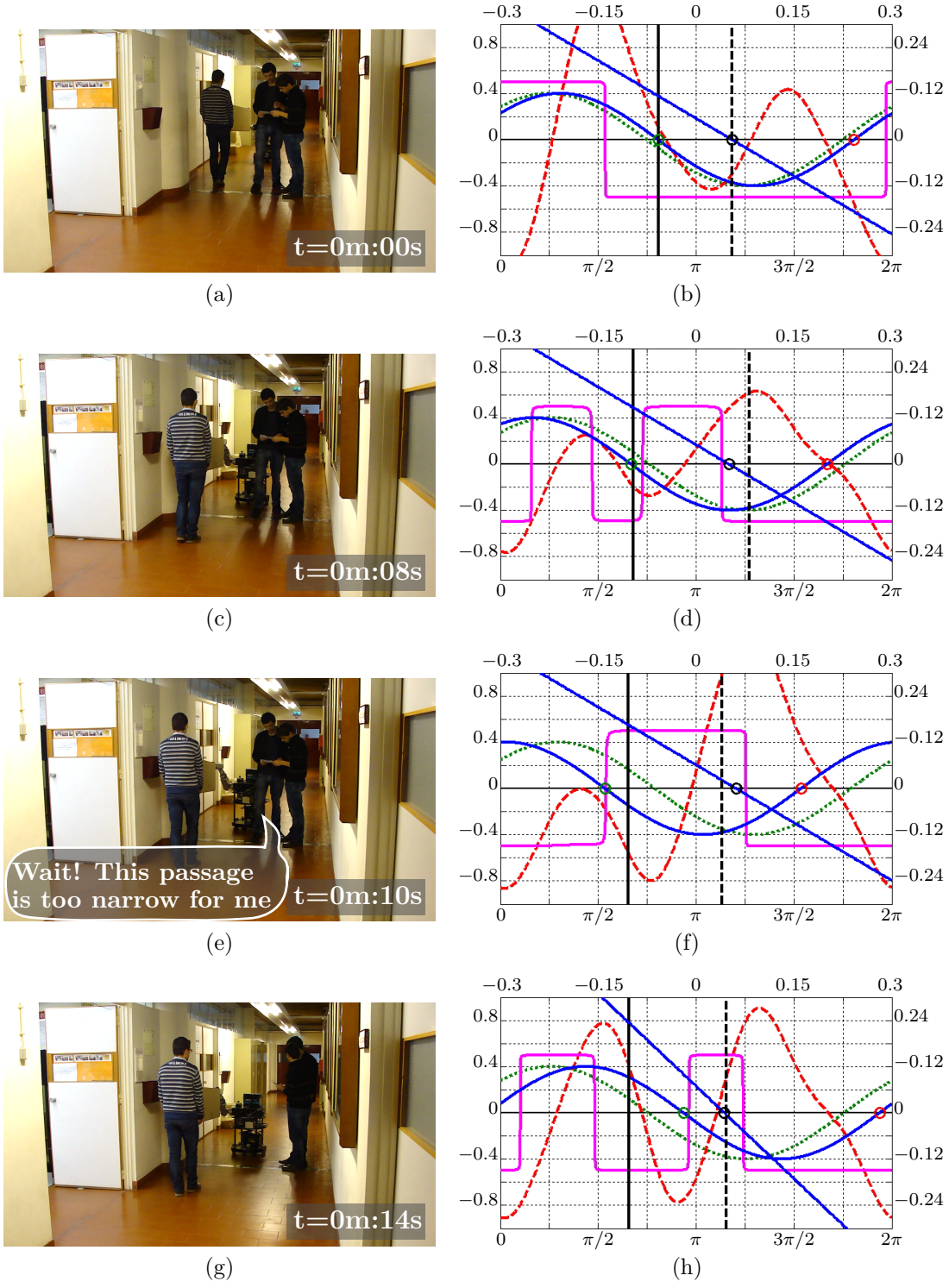
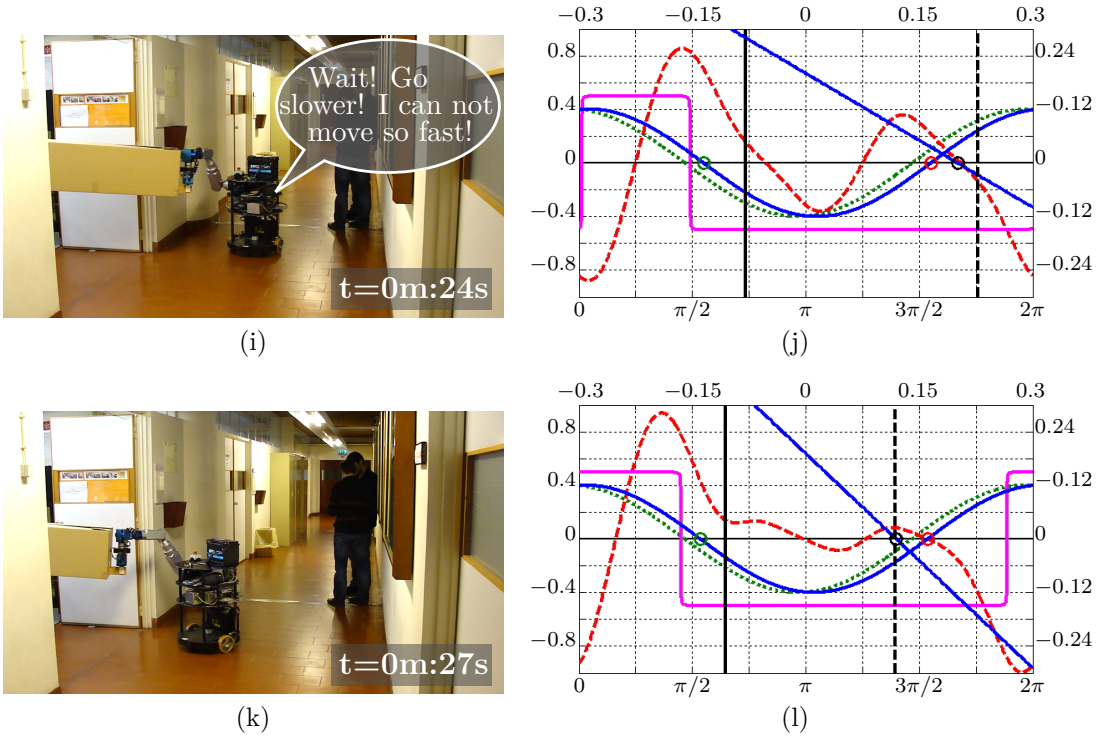


Figure 5.3: Snapshots sequence and dynamics for corridor scenario. Left side: snapshots at the specified time. On the right side, are presented both heading direction and path velocity dynamics for same instant. Left and lower axis are relative to heading direction, $d\phi/dt$ and ϕ, respectively. Right and upper axis are relative to path velocity dynamics, dv/dt and v, respectively. ϕ units is radians, while v is m/s.



(i) (j) (k) (l)

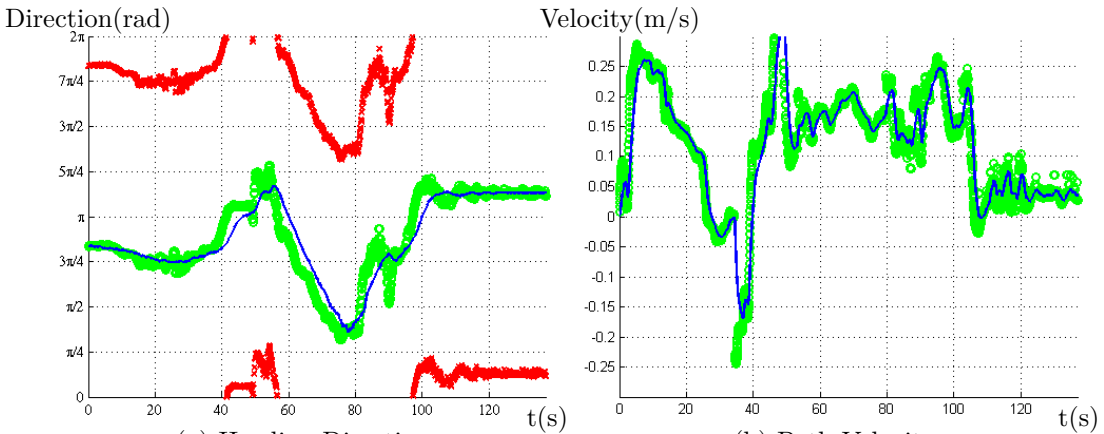
Figure 5.3: Continued.

5.4 Summary and Discussion

In this chapter we have presented some results that support the proposed dynamical architecture. To perform such tasks, no prior path planning was given to the robot, and it was able to cooperatively transport the object with the human avoiding static and dynamic obstacles. A digital compass was used to obtain the current heading direction relatively to a world reference frame, but it's value is solely for documentation proposes. Then, no calibration is necessary to be performed previously to work with the robot. As the presented snapshots and dynamics graphics illustrate, the robot navigation is very smooth and robust against perturbations, which is the case of the sudden appearance of an obstacle in the path of the robot.

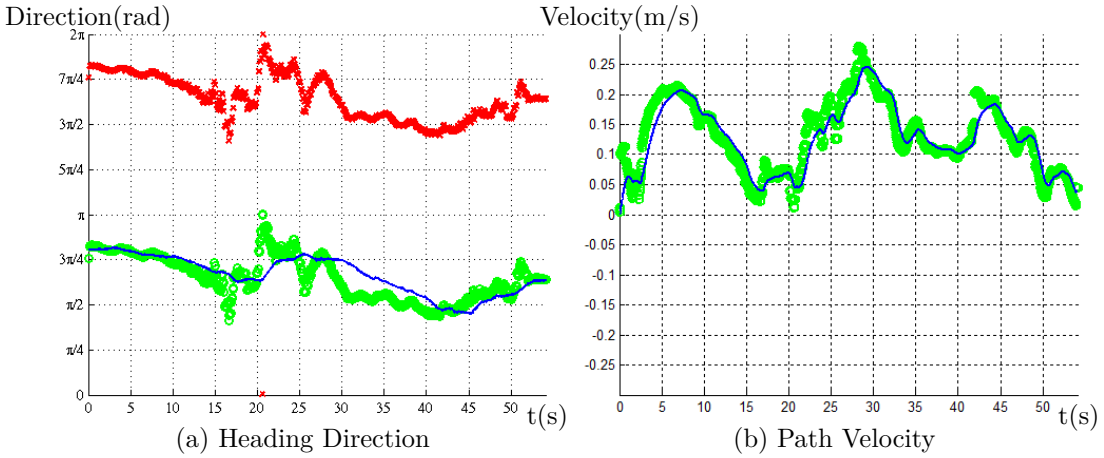
The integration of speech synthesis in the robot enhanced the execution of the task, since the robot could alert the human to some situations where if the human kept his movement, the robot would may let the object to fall.

It was shown also that the control variables (behavioral variables), heading direction and path velocity were always in or very close to the attractor of the corresponding dynamics, which makes the overall system asymptotically stable.



(a) Heading Direction (b) Path Velocity

Figure 5.4: Evolution of the fixed points over time for scenario of the entrance hall



(a) Heading Direction (b) Path Velocity

Figure 5.5: Evolution of the fixed points over time for scenario of the corridor

Chapter 6

Conclusion and Outlook

In this work we have addressed a Human-Robot object transportation task. A mobile manipulator was the robot selected to such cooperation in order to enable the team to transport the object from an initial position to a goal position.

The mobile manipulator was built under the scope of this dissertation where several software modules were developed to access and work with the hardware components of the mobile manipulator. The developed software was of major importance for the completion of this dissertation during tests of the developed dynamical control architecture on robot, since it allowed an abstract interface with hardware and all the required tasks necessary to initialize and error verification of hardware. Nevertheless, the developed platform has some limitations in the maximum achievable velocity. The robot structure and all the attached hardware are very heavy, thus the gear of locomotion motors has a rate of 1:167, which drastically reduces the maximum velocity of the platform to 0.2 m/s.

We have proposed an approach to Human-Robot (mobile manipulator) object transportation task based on dynamical systems. A leader-follower strategy was adopted and the results, documented in videos and log files, of the implementation shows the overt smooth and stable behavior of the developed system. It was shown that the robot is able to generate its own behavior without specific orientations of the human partner. The necessary information to generate such movements was gathered from the sensory system of the robot. As described previously, the robot does not have a map of the working environment and yet was able to avoid static and dynamic obstacles.

In the face of complex situations, the robot was able to express itself verbally to

alert the human to the situation. This feature was of major importance during Human-Robot object transportation, since the environment is unknown, unstructured and changing and some situations, as the human moving too fast may induce the robot to let the object to fall.

The designed dynamical architecture endowed then the mobile manipulator with autonomous capabilities to complete a Human-Robot Object Transportation task.

Nevertheless, the robot arm was kept stationary in order to make the execution of this dissertation feasible to be performed in the allocated time to it. The issue of controlling simultaneously the movement of the robotic arm will be addressed in the very near future.

In addition to arm issue, further research in Human-Robot Object Transportation may be addressed. During transportation task it was registered that the human has to continuously exert strength pushing or pulling the object to keep the robot in movement. An active cooperation may enhance this task. An intention recognition capability based on signal measured from the arm gripper may reduce the human effort to keep the robot in movement.

In future work it may be desirable to add a speech recognition mechanism to the robot in order to improve the verbal interaction between the human and robot to accomplish a more elaborated task. Simple commands such as turn-right, turn-left, wait and move faster may enhance the human experience and the robot can have a feedback from the human to his behavior. In addition to those simple commands, it can be included an adaptation/learning mechanism which allows the robot to tune its internal design parameters on behalf of human commands such as *"No! Here you can not pass!"* or *"You are going too close to obstacles, please stay away!"*. The association of the intention recognition and bilateral communication also allows the human to specify how he wants the robot to acknowledge his movements. A *"You are not helping too much! I am pushing too hard the object!"* message may induce the robot to tune it's intention recognition mechanism to reduce the human strength. The robot can then adapt to the human partner.

Bibliography

O. M. Al-Jarrah and Y. F. Zheng, "Arm-manipulator coordination for load sharing using reflexive motion control," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 3, 1997, pp. 2326 —-2331 vol.3.

——, "Arm-manipulator coordination for load sharing using variable compliance control," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 1, 1997, pp. 895 —-900 vol.1.

P. Althaus, "Indoor Navigation for Mobile Robots : Control and Representations," Ph.D. dissertation, Royal Institute of Technology, 2003.

H. Arai, T. Takubo, Y. Hayashibara, and K. Tanie, "Human-robot cooperative manipulation using a virtual nonholonomic constraint," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 4, 2000, pp. 4063 —-4069 vol.4.

E. Bicho and G. Schöner, "Target position estimation, target acquisition, and obstacle avoidance," in *Industrial Electronics, 1997. ISIE '97., Proceedings of the IEEE International Symposium on*, vol. 1, 1997, pp. SS13 —-SS20 vol.1.

E. Bicho, *Dynamic Approach to Behavior-Based Robotics: Design, Specification, Analysis, Simulation and Implementation.* Shaker Verlag, 2000.

E. Bicho and G. Schöner, "The dynamic approach to autonomous robotics demonstrated on a low-level vehicle platform," *Robotics and Autonomous Systems*, vol. 21, no. 1, pp. 23–35, 1997.

E. Bicho, P. Mallet, and G. Schöner, "Using attractor dynamics to control autonomous vehicle motion," in *Industrial Electronics Society, 1998. IECON '98. Proceedings of the 24th Annual Conference of the IEEE*, vol. 2, 1998, pp. 1176 –1181 vol.2.

——, "Target Representation on an Autonomous Vehicle with Low-Level Sensors," *The International Journal of Robotics Research*, vol. 19, no. 5, pp. 424–447, 2000.

E. Bicho, L. Louro, N. Hipolito, S. Monteiro, and W. Erlhagen, "Motion control of a mobile robot transporting a large size object in cooperation with a human: a nonlinear dynamical systems approach," in *Proc. of the IEEE 11th Intl. Conf. on Advanced Robotics*, 2003, pp. 197–203.

M. H. Choi, B. H. Lee, and M. S. Ko, "An application of force ellipsoid to the optimal load distribution for two cooperating robots," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, 1992, pp. 461 —-466 vol.1.

J. D. Crawford, "Introduction to bifurcation theory," *Reviews of Modern Physics*, vol. 63, no. 4, pp. 991–1037, 1991.

D. De Carli, E. Hohert, C. A. C. Parker, S. Zoghbi, S. Leonard, E. Croft, and A. Bicchi, "Measuring intent in human-robot cooperative manipulation," in *Haptic Audio visual Environments and Games, 2009. HAVE 2009. IEEE International Workshop on*, 2009, pp. 159–163.

L.-P. Ellekilde and H. I. Christensen, "Control of mobile manipulator using the dynamical systems approach," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on.* Ieee, 2009, pp. 1370–1376.

V. Fernandez, C. Balaguer, D. Blanco, and M. A. Salichs, "Active human-mobile manipulator cooperation through intention recognition," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3, 2001, pp. 2668 —- 2673 vol.3.

K. Fukaya, Y. Hirata, Z. Wang, and K. Kosuge, "Design and Control of A Passive Mobile Robot System for Object Transportation," in *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, 2006, pp. 31–36.

A. Goswami, M. A. Peshkin, and J. E. Colgate, "Passive robotics: an exploration of mechanical computation," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, 1990, pp. 279 —-284 vol.1.

S. A. Green, M. Billinghurst, X. Q. Chen, and G. J. Chase, "Human-robot collaboration: A literature review and augmented reality approach in design," *International Journal of Advanced Robotic Systems*, vol. 5, no. 1, pp. 1–18, 2008.

Y. Hirata and K. Kosuge, "Distributed robot helpers handling a single object in cooperation with a human," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 1, 2000, pp. 458 —-463 vol.1.

Y. Hirata, T. Takagi, K. Kosuge, H. Asama, H. Kaetsu, and K. Kawabata, "Map-based control of distributed robot helpers for transporting an object in cooperation with a human," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3, 2001, pp. 3010 —- 3015 vol.3.

Y. Hirata, Z. Wang, and K. Kosuge, "Human-Robot Interaction Based on Passive Robotics," in *SICE-ICASE, 2006. International Joint Conference*, 2006, pp. 4206–4209.

Y. Hirata, Y. Ojima, and K. Kosuge, "Coordinated motion control of multiple passive object handling robots based on environment information," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, 2009, pp. 2338–2343.

Y. Hirata, K. Suzuki, and K. Kosuge, "Improvement in the performance of passive motion support system with wires based on analysis of brake control," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 4272–4277.

N. Hogan, "Impedance Control: An Approach to Manipulation: Parts {I}, {II}, and {III}," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–24, 1985.

R. Ikeura and H. Inooka, "Variable impedance control of a robot for cooperation with a human," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3, 1995, pp. 3097 —-3102 vol.3.

R. Ikeura, H. Monden, and H. Inooka, "Cooperative motion control of a robot and a human," in *Robot and Human Communication, 1994. RO-MAN '94 Nagoya, Proceedings., 3rd IEEE International Workshop on*, 1994, pp. 112–117.

K. I. Kim and Y. F. Zheng, "Unknown load distribution of two industrial robots," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, 1991, pp. 992 —-997 vol.2.

K. Kosuge, H. Yoshida, and T. Fukuda, "Dynamic control for robot-human collaboration," in *Robot and Human Communication, 1993. Proceedings., 2nd IEEE International Workshop on*, 1993, pp. 398–401.

E. W. Large, H. I. Christensen, and R. Bajcsy, "Scaling the Dynamic Approach to Path Planning and Control: Competition among Behavioral Constraints," *The International Journal of Robotics Research*, vol. 18, no. 1, pp. 37–58, 1999.

M. Lawitzky, A. Mörtl, and S. Hirche, "Load sharing in human-robot cooperative manipulation," in *RO-MAN, 2010 IEEE*, 2010, pp. 185–191.

K. M. Lynch and C. Liu, "Designing motion guides for ergonomic collaborative manipulation," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 3, 2000, pp. 2709–2715.

Y. Maeda, T. Hara, and T. Arai, "Human-robot cooperative manipulation with motion estimation," *Proceedings 2001 IEEERSJ International Conference on Intelligent Robots and Systems Expanding the Societal Role of Robotics in the the Next Millennium Cat No01CH37180*, vol. 4, pp. 2240–2245, 2001.

G. Metta, P. Fitzpatrick, and L. Natale, "YARP: Yet Another Robot Platform," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 43–48, 2006.

Microsoft, "Microsoft Speech API," 2011. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms723627(v=vs.85).aspx

S. Monteiro and E. Bicho, "Attractor dynamics approach to formation control: theory and application," *Auton. Robots*, vol. 29, no. 3-4, pp. 331–355, 2010.

H. Neven and G. Schöner, "Dynamics parametrically controlled by image correlations organize robot navigation," *Biological Cybernetics*, vol. 75, no. 4, pp. 293–307, 1996.

F. G. Pereira, F. B. Sá, D. B. Ferreira, and R. F. Vassallo, "Object transportation task by a human and a mobile robot," in *Industrial Technology (ICIT), 2010 IEEE International Conference on*, 2010, pp. 1445–1450.

L. Perko, *Differential Equations and Dynamical Systems*. Berlin: Springer Verlag, 1991.

E. R. Scheinerman, *Invitation to Dynamical Systems*. Prentice Hall, 1996.

G. Schöner, M. Dose, and C. Engels, "Dynamics of behavior: Theory and applications for autonomous robot architectures," *Robotics and Autonomous Systems*, vol. 16, no. 2-4, pp. 213–245, 1995.

G. Schöner and M. Dose, "A dynamical systems approach to task-level system integration used to plan and control autonomous vehicle motion," *Robotics and Autonomous Systems*, vol. 10, no. 4, pp. 253–267, 1992.

R. Soares and E. Bicho, "Using attractor dynamics to generate decentralized motion control of two mobile robots transporting a long object in coordination," in *Proc. of the workshop on cooperative robotics, in IROS, 2002: 2002 IEEE/RSJ intl. conf. on intelligent robots and systems*, 2002.

R. Soares, E. Bicho, T. Machado, and W. Erlhagen, "Object transportation by multiple mobile robots controlled by attractor dynamics: theory and implementation," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, pp. 937–944.

R. Soares, "Transporte de Objectos por equipas de robôs móveis autónomos: estratégias de controlo distribuidas baseadas em sistemas dinâmicos não lineares," Ph.D. dissertation, University of Minho, 2007.

A. Steinhage, "Dynamical Systems for the Generation of Navigation Behavior," Ph.D. dissertation, Ruhr-Universität Bochum, 1997.

T. Takubo, H. Arai, Y. Hayashibara, and K. Tanie, "Human-Robot Cooperative Manipulation Using a Virtual Nonholonomic Constraint," *The International Journal of Robotics Research*, vol. 21, no. 5-6, pp. 541–553, 2002.

W. Wannasuphoprasit, R. B. Gillespie, J. E. Colgate, and M. A. Peshkin, "Cobot control," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 4, 1997, pp. 3571 —-3576 vol.4.

Y. Yamamoto, H. Eda, and X. Yun, "Coordinated task execution of a human and a mobile manipulator," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 2, 1996, pp. 1006 —-1011 vol.2.