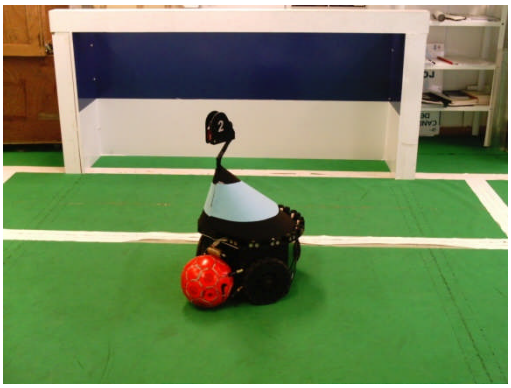
INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa



Task Specific Motion Control of Omnidirectional Robots

João Vicente Teixeira de Sousa Messias

Dissertação para a obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Júri

Presidente:	Doutor Carlos Jorge Ferreira Silvestre
Orientador:	Doutor Pedro Manuel Urbano de Almeida Lima
Vogal:	Doutor João Fernando Cardoso Silva Sequeira

Setembro de 2008

Acknowledgments

I would like to extend my thanks to Professor Pedro Lima, for giving me the opportunity to work in what is undoubtedly an important aspect of the SocRob robotic soccer project in its current state, and also for supervising my work, discussing my ideas, and reviewing this thesis.

To my father, for the instrumental help in illustrating this work, and for the long hours of his own free time in designing some of these figures, my very special thanks.

I also thank my mother, who wasn't at first particularly convinced of my professional choice, but supported me nonetheless.

Finally, I give my thanks to my friends, and to the rest of my family, who have always understood the difficulties involved in making this thesis, and may now see the result of my long-running efforts.

Abstract

This thesis addresses the most common motion control (guidance) problems that a holonomic mobile robot has to face, with particular emphasis on the tasks that must be performed in a robotic soccer environment.

The general situations where the robot must be driven to a reference posture (stabilization) or follow a reference trajectory (tracking) are considered. These problems are solved through feedback linearization, and the respective control laws are analyzed both in continuous-time and in discrete-time.

More specific tasks may require the interaction between the mobile robot and a moveable object in its environment: in a robotic soccer setting, two such tasks are those of intercepting a freely rolling ball and transporting the soccer ball. Solutions are presented to each of these problems, which explicitly take advantage of the omnidirectional capabilities of these robots. Ball interception is achieved through a solution that combines the concepts of trajectory-tracking and proportional navigation. Moving the ball is accomplished through a control scheme that determines at each instant the necessary force that must be applied to the ball, and then uses it as a reference for a hybrid force/position control law that drives the robot.

In each of these problems, the limitations of the robot's actuators are considered.

In order for the proposed control solutions to be useful in cluttered environments, the problem of obstacle avoidance is considered. The proposed solution consists in determining the locally optimal direction for movement in a reactive manner, and specifically addresses environments that are sparsely populated with fast-moving objects.

Keywords: Mobile robot motion control; Moving object interception; Nonlinear control; Object manipulation; Obstacle avoidance; Robotic soccer.

Resumo

Nesta dissertação são abordados os problemas de condução mais comuns que um robot móvel holonómico tem de enfrentar, com especial ênfase nas tarefas que são realizadas num ambiente de futebol robótico.

As situações gerais em que o robot deve ser dirigido para uma dada postura (estabilização) ou uma dada trajectória (seguimento) são considerados. Estes problemas são solucionados através de linearização exacta por retroacção, e as leis de controlo resultantes são analizadas em tempo contínuo e em tempo discreto.

Em tarefas mais específicas, pode ser necessária a interacção entre o robot e um objecto móvivel no seu ambiente: no futebol robótico, dois destes problemas são o de interceptar uma bola que rola livremente e o de transportar a bola. Apresentam-se soluções, para ambos os problemas, que tiram partido das capacidades omnidireccionais destes robots. A intercepção da bola é conseguida através de uma solução que combina seguimento de trajectórias e navegação proporcional. O transporte da bola é solucionado através de um esquema de controlo no qual a força que deve ser aplicada à bola é determinada, e em seguida serve de referência para um controlador híbrido de força/posição que conduz o robot.

Em cada um destes problemas, são consideradas as limitações dos actuadores do robot.

Para que as soluções apresentadas sejam úteis num ambiente que contenha obstáculos, também é considerado o problema de evitar esses obstáculos. A solução proposta consiste em determinar a direcção localmente óptima para o movimento, reactivamente, e é adaptada especificamente para ambientes esparsos com obstáculos em movimento.

Palavras-Chave: Condução de robots móveis; Controlo não-linear; Evitar obstáculos; Futebol robótico; Intercepção de objectos em movimento; Manipulação de objectos.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The domain of application of mobile robotics is always expanding, and with it the complexity of the tasks that a mobile robot must be prepared to accomplish also increases. In many environments, the speed at which the robot performs these tasks is critical, as well as the overall safety of the robotic system.

For the correct performance of its tasks, some form of guidance must be applied to the motion of a mobile robot. Motion control relates to the operations that must be performed, in order to obtain appropriate controls for the robot to perform its necessary movements. From these controls, the respective signals for the robot's actuators are obtained, and the effective displacement of the robot is carried out. The control techniques that are applicable to a specific robotic system are largely dependent on the robot's type of locomotion and actuator configuration.

Several approaches to the control of non-holonomic (differential-drive) robots in the robotic soccer environment were developed by the ISocRob team (the IST/ISR robotic soccer team), addressing tasks such as achieving a specific position in the field, avoiding any obstacles along the way [1], and dribbling the ball whilst taking into account the physical restrictions inherent to that process [2]. Recently, the application of holonomic robots to Middle-Sized League robotic soccer became widespread. With it, new and better ways to accomplish common tasks, such as those that involve some form of interaction with the ball, were sought by most teams. The ISocRob team has since developed a team of holonomic robotic systems, but its motion control techniques had not been updated, up until the present work, to match the capabilities of those systems, for all but the most basic tasks. As a consequence, the robots would not make use of their omnidirectionality while dribbling the ball, for example, and would instead execute these tasks by resorting to the outdated concepts developed in the non-holonomic context.

The main motivation for this work then comes from the practical necessity of obtaining new forms of motion control that fully exploit the capabilities of holonomic robots. The domain of application of these control techniques include, but are not limited to, the robotic soccer environment. This is achieved by using well known results of control theory to describe rigorous solutions to the most common tasks, the performance of which can be readily analyzed, and expanding existing works within the domain of robotics to allow holonomic mobile robots to solve more particular tasks that involve object interaction.

1.2 Objectives

It is the main objective of this work to provide a framework to holonomic robot control, which can be applied in the Middle-Sized League robotic soccer environment. This, in turn, comes as a consequence of a more particular goal of developing control solutions that allow the ISocRob holonomic robots to efficiently capture and transport a freely rolling ball in an environment populated with dynamic obstacles. To this end, particular focus is given to the tasks of ball interception and dribbling. As an extension to this original goal, the most common tasks that a robotic soccer player has to perform that do not require explicit interaction with the soccer ball are also considered, and appropriate solutions are sought for each of them. These solutions should extend and improve the existing guidance algorithms of the ISocRob team, which are not fully applicable to holonomic robots, or fail to make use of their capabilities.

More specifically, the motion control problems that should be dealt with are associated with the following tasks:

- Intercepting a freely rolling ball, with arbitrary initial conditions for the position and velocity of both the robot and the ball. The ball may suffer unpredictable deviations in its trajectory from collision with other objects in the field or from the effects of friction forces. This interception should be done as fast as possible, and in such a way that the robot is immediately able to begin transporting the ball to another position;
- Transporting the ball to a given position in the field, not necessarily stabilizing the object around the reference, in such a way that the total required time is minimal, and considering that the robot may encounter unexpected disturbances along its motion. The restrictions imposed by the physical dimensions of the robot and the ball should be taken into account, as well as the limited capabilities of the robot's actuators;
- Driving the robot to a given posture in the field, which it should then maintain (solving the stabilization problem for a reference posture).
- Following a reference trajectory, which can be thought of as a reference posture moving with a certain velocity and acceleration (solving the tracking problem for a reference trajectory).
- It should be possible to specify the stabilization and tracking problems independently for the position and orientation of the robot. This means, for example, that the robot should be able to move to a given position in the field whilst maintaining itself oriented towards a moving target.
- In all of the above tasks, it is necessary to consider that dynamic obstacles are present in the robot's environment, which must be properly avoided.

For the purposes of this work, it is assumed that the robot possesses the required sensors and the respective algorithms to obtain the relevant information from the environment, for each task, and that this information may contain noise.

It must also be considered that the motion control solutions must be implemented as part of a task-execution architecture, which, in the case of the ISocRob team, is the MeRMaID architecture, and which may affect the overall performance of those solutions.

1.3 Original Contributions

This work presents the following original content:

- The control solutions for the ball interception and dribbling tasks presented in this work are based on research that was originally developed in the context of robotic manipulators. To the best of the author's knowledge, no effort has been previously made to adapt these solutions to the field of mobile robotics;
- In the robotic soccer environment, the effects that the limitations of the robot's actuators may have on the tasks that require interaction with the ball have been left unconsidered up until now, except partly in the work of B. Damas in [2];
- The previous approaches taken by the ISocRob team [1],[2], combined motion control and local motion planning (obstacle avoidance) indistinguishably. This work maintains a clear, modular definition of both of these components of a mobile robot's navigation to provide adaptability to the specification on new tasks and robustness to changes in the robot's physical properties;
- The proposed obstacle avoidance algorithm extends the concepts introduced by existing algorithms to deal with dynamic obstacles, while focusing on reactivity and computational efficiency.

1.4 Thesis Outline

The work contained in this document is organized as follows: Chapter 2 discusses the implementation of motion control in common task-execution architectures and, specifically, in the MeRMaID architecture used by the ISocRob team, and describes the limitations that these architectures impose on the control techniques that may be applied. Chapter 3 describes the solutions to the most basic tasks that an omnidirectional robot is bound to encounter, and that do not require any explicit interaction with objects in its environment. In Chapter 4, the solutions to the specific problems of moving object interception and object transport are discussed. Chapter 5 introduces the obstacle-avoidance algorithms that allow the robot to accomplish its tasks safely. The hardware details of the ISocRob robots are presented in Chapter 6, along with the experimental setup that was used to verify the validity of the proposed solutions. In Chapter 7, experimental results are presented and discussed. Finally, in Chapter 8, conclusions are drawn, and the main topics where future work may be developed are detailed.

2 Motion Control in a Task-Execution Architecture

Generally speaking, the execution of a mobile robot's intended tasks can be broken down into a set of components, each representing a specific range of "competences" that the robot must display. These are *perception, localization, cognition* and *motion control.*

Perception relates to a mobile robot's ability to sense its environment and obtain useful information from the respective sensor data; Localization is the process of obtaining an estimate of the robot's posture, with respect to a model of its environment that is either supplied beforehand or constructed by the robot; Cognition refers to the capability of identifying the most appropriate actions to accomplish the robot's task at every given moment; Finally, Motion Control consists of the operations that must be performed in order to generate appropriate control inputs for the mobile robot to execute its necessary movements.

Though each of these elements is subject to extensive study and cover varied scientific domains, the present work is focused on the aspects of task execution directly linked to robot mobility, in a robotic soccer environment. These include motion control and the lower-level aspects of cognition. It is the goal of this section to briefly describe how these components are integrated into a robot's task-execution architecture, and the effects and limitations that such architectures can have on the performance of a mobile robot in accomplishing a given task. This is relevant for the development of efficient control solutions, which will be under study for the remainder of this work. In particular, the MeRMaID architecture, currently implemented on the ISocRob soccer robots, is explored with respect to its capabilities for rigorous motion control.

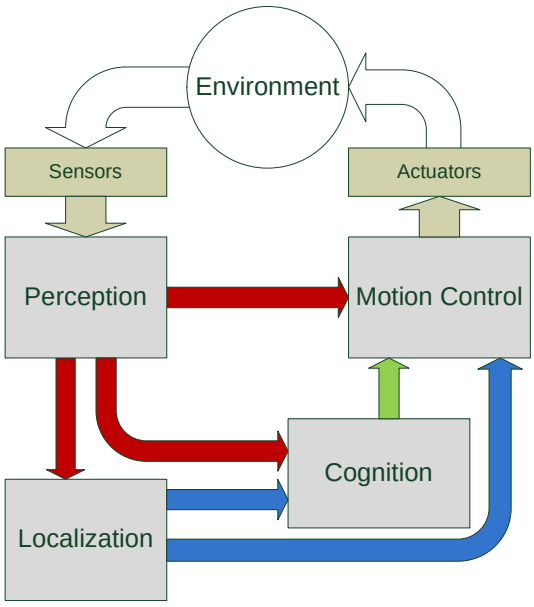


Figure 1: The main components that intervene in the execution of a task by a mobile robot.

2.1 Common Task-Execution Architectures

The most common mobile robot control architectures follow the basic *serial decomposition* (*functional*) *or parallel decomposition* (*behavior-based*) approaches [3],[4], or some combination of these. They are distinguished by the control path to the robot's actuators, i.e. by which modules are allowed to perform motion control upon the robot itself, and by the dependencies among them.

In the serial architecture, the control inputs supplied to the robot are the result of a sequential process, where the output of each module is fully dependent on the outputs of its predecessor. The sequence usually derives from the Sense-Model-Plan-Act paradigm. From a control perspective, this means that a serial architecture is a well defined control loop, and so the overall system is predictable and its performance can be analyzed with relative ease. The disadvantage of this type of architecture lies in its susceptibility to errors, which propagate throughout the process, and its inflexibility to modifications in the robotic platform, such as the addition of multiple sensors, or the extension of the robot's intended tasks. This is turn leads to its lack of usefulness in most long-term robotic applications, where modularity is sought.

Figure 2: An example of a purely serial architecture

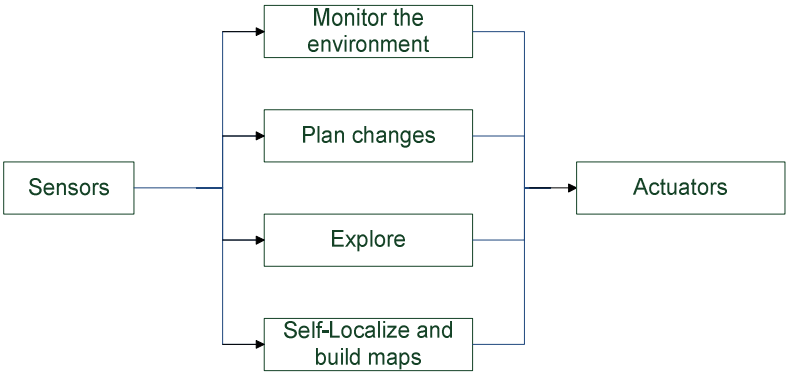


Figure 3: An example of a purely behavior-based architecture

In contrast, the control inputs sent to the robot's actuators in a purely behavior-based architecture may originate from any of its modules, which run simultaneously and at different frequencies. Each of these modules, a *behavior* in itself, constitutes an abstraction that allows the architecture to function without depending on the details of the implementation of each of said behaviors. It is important to note that in such a case, the motion control requirements of two or more of these behaviors may be in conflict – this may happen, for example, in a common situation where a robot is required to reach a given posture in its environment, but at the same time it must avoid any obstacles in its vicinity. An important aspect of this type of architecture is how to attribute control to each specific behavior. A simple approach is to permit control by a single behavior at any given time, in what is known as a *switched-parallel* architecture. An example of this is the Subsumption

architecture proposed by Brooks in [3], where the access to the robot's actuators occurs in a prioritized fashion. In these architectures, the system may exhibit undesirable transient effects when the control switches from one behavior to another, and so the frequency at which this switching occurs must be relatively low. Other possible approaches exist, such as allowing two or more modules to perform motion control simultaneously, but this is usually very disadvantageous, since the system can even be lead into instability. The main advantage of a behavior-based architecture lies in its flexibility, as each module is independent from the functionalities of other modules, and so they may be altered, added or removed without it having a large impact on the remainder of the system, which is usually very desirable for robotic applications which are subject to frequent modifications. These architectures are also more suitable to the usage of multiple sensors, which helps reduce the associated uncertainty and increases the overall robustness of the system. Insofar as motion control is concerned, however, it is difficult to verify any sort of performance specifications on the overall system. This is due not only to the unpredictability of the instants where the active behaviors are changed, but also because the involved control techniques must be specified independently for each behavior, and are therefore utilized at different time intervals. This, in turn, raises some problems: it is difficult to interact with the robot's actuators in a rigorous manner, since their control usually runs on its own, fixed frequency, and therefore any analysis performed upon the control laws of a specific behavior may lose some of its legitimacy in the presence of this additional sampling step; also, if any of the control-relevant properties of the robot is modified, such as its weight, physical dimensions, or the performance of its actuators, then all of the related behaviors will have to be revisited and updated individually, and some of the applied control laws might not even maintain their validity.

2.2 Motion Control as a Part of MeRMaID

The MeRMaID (*Multiple-Robot Middleware for Intelligent Decision-making*) architecture [5], currently in use by the ISocRob team, was developed to meet the demands of robotic applications that require cooperation between multiple robots, such as in the robotic soccer domain. It is a layered, behavior-driven architecture that allows for cooperative behaviors to be performed. Without delving too much into the details of its operation, which go beyond the scope of the present work, it is important to describe the way through which the robot's actuators are accessed.

The behaviors that the robot is intended to perform are decomposed into a set of *Primitive Actions*, which are the basic tasks, like moving to a designated posture or intercepting the ball, which cannot be further decomposed into simpler components. For each behavior that is triggered, a string of these Primitive Actions is performed, in such a way that a single action is in charge of motion control at any given instant. This scheme of operation is similar to the previously described switched-parallel approach, and is thus subject to the most of its shortcomings from a control perspective.

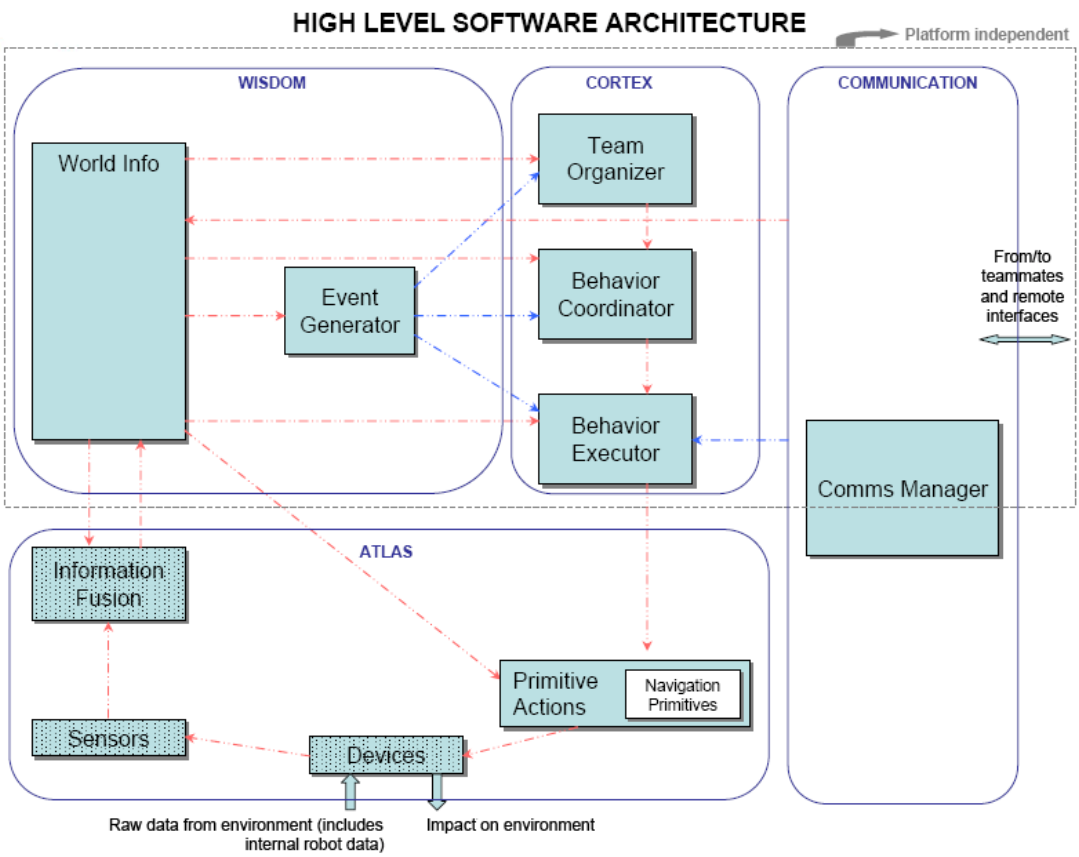


Figure 4: Layout of the MeRMaID functional architecture.

Even though these Primitive Actions cannot be broken down into more basic tasks, they can be decomposed into distinct control problems. Take, for instance, the tasks of moving to a foul-taking position or to a foul-receiving position. While these tasks serve two different purposes, and are well-distinguishable from the perspective of higher level components, they constitute the same problem of moving the robot to a given reference, and thus are equivalent in their motion control requirements. Similarly, the task of intercepting a moving ball is assigned to a single Primitive Action, but entails diverse motion control problems that must be treated separately. To account for these situations, the Primitive Actions resort to a set of *Navigation Primitives*, which constitute a common framework where each control problem is addressed individually, and thus allows for code reusability and provides (ideally) the same control performance for problems that share the same motion control requirements. The structure of the Primitive Actions themselves then reduces to selecting the appropriate navigation primitives for each situation and providing them with adequate references. Returning to the example of the foul-taking and foul-receiving tasks, this means that both of these Primitive Actions make use of the same navigation primitives, albeit with different goals.

This again defines the purpose of this work: to create a set of such navigation primitives that, by solving the most common control problems that a robotic soccer player has to face, will allow these robots to achieve their given tasks. While this constitutes an efficient way of addressing complex tasks, it also presents a set of difficulties that must be dealt with in order to perform some form of rigorous motion control. The most relevant of these is that the navigation primitives are not an active

part of the architecture itself; they are simply a generalized “control library” that the architecture, or in this case the Primitive Actions, can use. This being the case, there is no way to know exactly at what time instants will the navigation primitives be called from within the architecture itself. This is because the internal frequency of execution of a Primitive Action is not fixed during the time it is active. Moreover, the MeRMaID architecture does not currently support hard real time constraints. This poses the most significant limitation on motion control in this architecture. It is verifiable, however, that the frequency of execution of each Primitive Action does not suffer noticeable variations during the time that it is active, even though it is unknown *a priori.* It is sufficient to assume, therefore, that whenever the activation of a Primitive Action is requested by a behavior, it is put under a soft real time constraint. It is then possible to obtain an estimate of the Primitive Action’s internal execution rate, by measuring the elapsed time between consecutive iterations. This information is then used by the Navigation Primitives to enforce the stability of the applied control techniques. It is imperative, however, to analyze these control techniques in discrete-time, and, whenever possible, to maintain their characteristics (e.g. pole locations) as a function of the sampling period, so as to provide similar performance to different tasks, and to maintain their domain of application as wide as possible. It is also important to provide updated information about the physical dimensions of the robot and other relevant objects (such as the ball). In this case, even if these characteristics are modified, the functionalities of each Primitive Action remain valid, and no other modifications are necessary inside the architecture itself.

A final important detail should be mentioned: while in some tasks, such as following a trajectory, it is convenient to address the robot’s inputs as accelerations (torque control), in others it is sufficient to consider them as velocities (velocity control), like when driving the robot to a given posture. Instead of converting these inputs to a single form (by explicitly integrating the acceleration inputs, for example), the Primitive Actions allow for an abstraction of the desired control input type. These are sent in their original form to the robot’s actuators, where its controllers then perform the most adequate conversions, resulting in an overall reduction of the associated errors, since these controllers run at a fixed rate, and making the system more robust to failures in the communications between the architecture and the actuators.

3 Basic Motion Control

It is goal of this section to provide a framework for the unrestricted motion control of an omnidirectional soccer robot, by concisely addressing the most basic operations that the mobile robot must be able to perform in its environment. By unrestricted motion, it is meant the normal motion of the robot when it is not required to interact with other objects, since these cases, as it will be seen in Section 4, have to be dealt with separately. The basic actions under unrestricted motion are identified, for this case, as the *posture stabilization* and the *posture tracking* problems. While in the former, the robot is given only a static target posture in its environment as reference, which it must reach as fast as possible and then maintain it, in the latter a reference trajectory is supplied to the robot, which it must continuously follow. From these two general cases, a great variety of possible tasks may be specified (e.g. moving to a specific posture in the field, following a teammate, keeping the robot turned to the ball, etc.).

It is well known that, in the particular case of omnidirectional robots, the problem of controlling the robot's posture can be solved independently for its position and its orientation (see Annex A1 for details). By doing so, the nonlinearities present in the robot's kinematic and dynamic models are more easily tractable since they only relate to its position components. The control for orientation is then relatively straightforward through the normal tools for linear systems analysis. The above problems can then be further divided into those of *point stabilization, point tracking*, and orientation control for both static and moving references. Following this reasoning, the contents of this section are divided according to each of these control problems. Section 3.1 deals with point stabilization; Section 3.2 presents a solution to point tracking; finally, Section 3.3 discusses orientation control.

3.1 Point Stabilization

Many solutions to the point stabilization problem for mobile robots are readily available in the respective literature (e.g. [6],[7],[8],[4]), and for holonomic robots it can be solved, as it will be shown, through static state feedback linearization. The main advantage of this approach over other possible approaches such as Lyapunov-based methods (e.g. [9]), is that the behavior of the linearized system can be modified at will and readily analyzed. On the downside, the kinematic and dynamic models of the robot are assumed to be exact, which may introduce systematic errors into the closed-loop system. In the case of simple and well-known systems, such as holonomic robots, this is admissible.

Consider the position kinematic model for an omnidirectional robot,

$$\dot{\boldsymbol{p}} = B(\theta)\boldsymbol{v} \tag{3.1}$$

where $\boldsymbol{p} = [x \quad y]^T$ denotes the position of the robot in the world frame, $\boldsymbol{v} = [v_x \quad v_y]^{\boldsymbol{T}}$ is the linear velocity of the robot's chassis in the robot frame, and $B(\theta)$ can be seen as a rotation matrix that transforms two-dimensional vectors between these two frames (see Annex A1). It is assumed that the robot's position at each instant is estimated by the robot's self-localization modules, with negligible

error. Given a reference position $\boldsymbol{p}_g$, and defining the position error as $\boldsymbol{p}_e = \boldsymbol{p} - \boldsymbol{p}_g$, the goal is then to find a feedback control law $\boldsymbol{v}$ such that:

- The closed-loop system is asymptotically stable, i.e. $\lim_{t\to\infty} \boldsymbol{p}_e(t) = 0$;
- $\boldsymbol{p}(0) = \boldsymbol{p}_g \Longrightarrow \boldsymbol{p}(t) = \boldsymbol{p}_g \; \forall t > 0$.

Proposition 3.1: *(From de Wit et al. in* [8]*). A feedback linearizing control law $\boldsymbol{v}$ that solves the point stabilization problem for omnidirectional robots is:*

$$\boldsymbol{v} = B^{-1}(\theta)A(\boldsymbol{p} - \boldsymbol{p}_g) \tag{3.2}$$

for any Hurwitz matrix A.

Proof: For a non-null reference, the kinematic model (3.1) can be rewritten as,

$$\frac{d}{dt}(\boldsymbol{p} - \boldsymbol{p}_g) = B(\theta)\boldsymbol{v} \tag{3.3}$$

Applying (3.2), the closed-loop system becomes:

$$\frac{d}{dt}(\boldsymbol{p} - \boldsymbol{p}_g) = A(\boldsymbol{p} - \boldsymbol{p}_g) \tag{3.4}$$

which is a linear system whose poles are at the eigenvalues of A. ■

The usefulness of feedback linearization for omnidirectional robots becomes evident from the above, since from its well-known and invertible nonlinearities $B(\theta)$, simple control laws such as (3.2) can be obtained, and the behavior of the resulting linear system can be freely assigned by selecting proper pole locations. This in turn helps reduce the problems associated with the saturation of the robot's actuators.

As it was discussed in Section 2.2, the implementation of continuous time approximations such as the control law described by (3.2) are subject to serious limitations imposed by the varying time-step at which the controls are calculated. It is then necessary to obtain a discrete-time equivalent for the above feedback linearization procedure. Better still would be to allow the task designer to specify the desired behavior of the linearized system through matrix A in continuous-time, and have the system emulate these specifications in discrete-time. To achieve this, the discrete-time equivalent for the desired linearized system is first obtained. Though its derivations go beyond the scope of this analysis, it is easily shown [10], that the discrete-time equivalent for a linear time-invariant system $\dot{\boldsymbol{p}} = A\boldsymbol{p}$, preceded by a zero-order hold, and using a sample period T, is:

$$\boldsymbol{p}(n+1) = e^{AT}\boldsymbol{p}(n) \tag{3.5}$$

The discrete-time equivalent of the kinematic model (3.1) must also be obtained. To this end, note that the displacement in the x component of position, from t_0 to t, is given by:

$$x(t) = x(t_0) + \int_{t_0}^{t} B(\theta(\tau))v_x \, d\tau \tag{3.6}$$

Through a change of variables, (3.6) becomes,

$$x(kT+T) = x(kT) + \int_{kT}^{kT+T} B(\theta(\tau))v_x \; d\tau \tag{3.7}$$

Note that v_x is constant throughout this interval, and so (3.7) can be simplified by solving:

$$\int_{kT}^{kT+T} B(\theta(\tau))\, d\tau \;= \frac{1}{\omega}\int_{kT}^{kT+T} \begin{bmatrix} \mathrm{c}\theta & -\mathrm{s}\theta \\ \mathrm{s}\theta & \mathrm{c}\theta \end{bmatrix} \omega \, d\tau$$

$$= \frac{1}{\omega}\int_{\theta(kT)}^{\theta(kT+T)} \begin{bmatrix} \mathrm{c}\eta & -\mathrm{s}\eta \\ \mathrm{s}\eta & \mathrm{c}\eta \end{bmatrix} \omega \, d\eta \;= \frac{1}{\omega}\begin{bmatrix} \mathrm{s}\eta & \mathrm{c}\eta \\ -\mathrm{c}\eta & \mathrm{s}\eta \end{bmatrix}\Big|_{\theta(kT)}^{\theta(kT+T)}$$

$$= \frac{1}{\omega}\begin{bmatrix} \sin(\theta(kT)+\omega T) - \sin\big(\theta(kT)\big) & \cos(\theta(kT)+\omega T) - \cos\big(\theta(kT)\big) \\ -\big(\cos(\theta(kT)+\omega T) - \cos\big(\theta(kT)\big)\big) & \sin(\theta(kT)+\omega T) - \sin\big(\theta(kT)\big) \end{bmatrix} = \Gamma(\theta(kT),\omega) \tag{3.8}$$

where the result $\theta(kT+T) = \theta(kT) + \omega T$ was used, which constitutes a forward-difference approximation. Then by using the following trigonometric relation:

$$\sin a - \sin b = 2\cos\frac{a+b}{2}\sin\frac{a-b}{2} \tag{3.9}$$

Matrix $\Gamma(\theta(kT),\omega)$ then becomes,

$$\Gamma(\theta(kT),\omega) = \frac{2}{\omega T}\sin\left(\frac{\omega T}{2}\right)\begin{bmatrix} \cos\left(\theta(kT)+\frac{\omega T}{2}\right) & -\sin\left(\theta(kT)+\frac{\omega T}{2}\right) \\ \sin\left(\theta(kT)+\frac{\omega T}{2}\right) & \cos\left(\theta(kT)+\frac{\omega T}{2}\right) \end{bmatrix} T$$

$$= \mathrm{sinc}\left(\frac{\omega T}{2}\right) P(\theta(kT),\omega) T \tag{3.10}$$

The above derivations are analogous for the y component, and so, again changing variables, the discrete-time kinematic model for position can be expressed as,

$$\boldsymbol{p}_{n+1} = \boldsymbol{p}_n + \mathrm{sinc}\left(\frac{\omega_n T}{2}\right) P(\theta_n,\omega_n)\boldsymbol{v}_n T \tag{3.11}$$

where the subscript n represents the iteration number. Note that the kinematic model (3.11) has the following interesting properties:

Property 3.1: If $\omega_n \to 0$, then $\boldsymbol{p}_{n+1} \to \boldsymbol{p}_n + B(\theta_n)\boldsymbol{v}_n T$.

This is an intuitive result, since by interpreting $B(\theta)$ as a rotation matrix that transforms vectors from the robot frame to the world frame, the system would behave as $\boldsymbol{p}_{n+1}^W = \boldsymbol{p}_n^W + \boldsymbol{v}_n^W T$, as expected, since the robot's orientation remains constant between sampling instants.

Property 3.2: If $\boldsymbol{\omega_n} \to \boldsymbol{K}\frac{\boldsymbol{2\pi}}{\boldsymbol{T}}$, with $\boldsymbol{K} \in \boldsymbol{N}$, or if $\boldsymbol{\omega_n} \to \infty$, then $\boldsymbol{p_{n+1}} \to \boldsymbol{p_n}$.

The physical interpretation of a situation where $\omega_n = K\frac{2\pi}{T}$ is that the robot performs an integer number of revolutions between two consecutive sampling instants. This implies that the robot's position kinematic model loses its controllability, since, regardless of the linear velocity of the robot, it will always return to the same position at the end of each sampling interval. On the other hand, in the theoretical situation where $\omega_n \to \infty$, it is intuitive that the robot would become unable to leave its current position, and so controllability would also be lost. For most cases, however, it is unlikely that the robotic system is capable of achieving the required angular velocity to incur in either of these situations. For example, in the ISocRob robotic soccer platform, with a typical sampling interval of $T = 0.04\,\mathrm{s}$, the robot would require an angular velocity of $\omega_n \cong 157\,\mathrm{rad/s}$ for this loss of controllability to manifest itself. This largely exceeds the capabilities of the robot, which possesses a maximum angular velocity of $\omega_{MAX} \cong 15\,\mathrm{rad/s}$.

In light of these results, and taking (3.5) as the desired linearized system, it is evident that a discrete-time feedback linearizing control law v_n is given by:

$$\boldsymbol{v}_n = \frac{1}{T}\left(\mathrm{sinc}\left(\frac{\omega_n T}{2}\right)\right)^{-1} P^{-1}(\theta_n, \omega_n)(e^{AT} - I)\boldsymbol{p}_n \tag{3.12}$$

Note that this control law possesses singularities at the values of ω_n that cause the position kinematic model to be uncontrollable. These singularities are assumed to be unreachable hereafter.

3.2 Point Tracking

Given a reference trajectory for position $\boldsymbol{p}_g(t)$, which can be thought of as a "moving" reference point in the world frame, with a certain velocity and acceleration, $\dot{\boldsymbol{p}}_g(t)$ and $\ddot{\boldsymbol{p}}_g(t)$, the objective of the trajectory-tracking primitive is to have the error between the robot's trajectory and the reference trajectory converge to zero (preferably in an exponential manner). This reference trajectory is either supplied beforehand by one of the robot's planning modules, or it corresponds to the measurable velocities and accelerations of an object moving in the robot's environment (such as in the ball interception task).

More formally, the goal is then to obtain a *point-tracking* feedback control law $\boldsymbol{a}(t)$ such that:

- $\boldsymbol{a}(t)$ and $\boldsymbol{p}_e(t) = \boldsymbol{p}(t) - \boldsymbol{p}_g(t)$ are bounded for all t;
- $\lim_{t\to\infty} \boldsymbol{p}_e(t) = 0$;
- $\boldsymbol{p}(0) = \boldsymbol{p}_g(0) \Longrightarrow \boldsymbol{p}(t) = \boldsymbol{p}_g(t)\ \forall t > 0$.

Like the point stabilization problem, the point (and posture) tracking problem has been subject to intensive study in the field of robotics, inclusively for the robotic soccer environment [9],[11]. Maintaining the same philosophy as in the previous section, this problem can be solved through feedback linearization as well. Consider the position dynamic model for the omnidirectional robot, where the dependencies on time are implicit:

$$\dot{\boldsymbol{p}} = B(\theta)\boldsymbol{v} \tag{3.13}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{a} \tag{3.14}$$

where $\boldsymbol{a} = [a_x \quad a_y]^T$ is the linear acceleration of the robot's chassis. By differentiating (3.13),

$$\ddot{\boldsymbol{p}} = B(\theta)\boldsymbol{a} + \dot{B}(\theta)\boldsymbol{v} \tag{3.15}$$

In a similar manner to the previous approach, the problem now reduces to selecting the control inputs (in this case, $\boldsymbol{a}$) that will cancel out the nonlinearities in (3.15) and result in a linear closed-loop system.

Proposition 3.2: *(From de Wit et al. in* [8]*). A feedback linearizing point-tracking torque control law for an omnidirectional robot is,*

$$\boldsymbol{a} = B^{-1}(\theta)\left(-\dot{B}(\theta)\boldsymbol{v} + \ddot{\boldsymbol{p}}_g - (\Lambda_1 + \Lambda_2)\dot{\boldsymbol{p}}_e - \Lambda_1\Lambda_2\boldsymbol{p}_e\right) \tag{3.16}$$

with,

$$\Lambda_1 = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_1 \end{bmatrix} \tag{3.17}$$

$$\Lambda_2 = \begin{bmatrix} \lambda_2 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{3.18}$$

and $\lambda_1, \lambda_2 > 0$.

Proof: By substituting (3.16) in (3.15), it results that:

$$\ddot{\boldsymbol{p}}_e + (\Lambda_1 + \Lambda_2)\dot{\boldsymbol{p}}_e + \Lambda_1\Lambda_2\boldsymbol{p}_e = 0 \tag{3.19}$$

Equation (3.19) is a stable linear differential equation, the dynamics of which can be freely assigned by selecting proper values for λ_1 and λ_2, since the closed loop system's poles are located at $s_{1,2} = -\lambda_{1,2}$. ■

It is often useful to define $\lambda_1 = \lambda_2$, since the system then becomes critically damped. A discrete-time control law can be sought by finding the discrete equivalent of equations (3.13), (3.14), and (3.15), and solving again for $\boldsymbol{a}$.

Proposition 3.3 *A discrete-time feedback linearizing control law for the point tracking problem is:*

$$\boldsymbol{a}_n = -\frac{\boldsymbol{v}_{n-1}}{T} + \frac{1}{T^2}\left(\operatorname{sinc}\left(\frac{\omega_n T}{2}\right)\right)^{-1} P^{-1}(\theta_n, \omega_n)\left(-\boldsymbol{p}_n + \boldsymbol{p}_{g_{n+1}} + (e^{-\Lambda_1 T} + e^{-\Lambda_2 T})\boldsymbol{p}_{e_n} - e^{-(\Lambda_1+\Lambda_2)T}\boldsymbol{p}_{e_{n-1}}\right) \tag{3.20}$$

where the involved variables share the same meaning as in Section 3.1.

Proof: Equation (3.19) can also be written as:

$$\left(\frac{d}{dt} - \Lambda_1\right)\left(\frac{d}{dt} - \Lambda_2\right)\boldsymbol{p}_e = 0 \tag{3.21}$$

In discrete time, an approximation of the posture dynamic model for position is represented by:

$$\boldsymbol{p}_{n+1} = \boldsymbol{p}_n + \operatorname{sinc}\left(\frac{\omega_n T}{2}\right) P(\theta_n, \omega_n)\boldsymbol{v}_n T \tag{3.22}$$

$$\boldsymbol{v}_n = \boldsymbol{v}_{n-1} + \boldsymbol{a}_n T \tag{3.23}$$

Equation (3.22) has been derived in Section 3.1. It is only exact in this context if v_n is held constant throughout each time step, otherwise the matrix P would also depend on the angular acceleration of the robot, making the problem both mathematically and computationally demanding. If the robot's hardware is performing torque control on its wheels at a higher frequency than that at

which equation (3.22) is used, some approximation error is incurred. Similarly, equation (3.23) is the backward-difference approximation of (3.14).

According to the exact discretization method proposed by Kawarai in [12], differential operators of the form $\left(\frac{d}{dt} - \alpha\right)$ can be transformed into discrete-time as follows:

$$\left(\frac{d}{dt} - \alpha\right) x(t) \rightarrow (1 - e^{\alpha T}\phi^{-1})x_n \tag{3.24}$$

where ϕ is the state transition matrix, such that:

$$x_{n+1} = \phi x_n \tag{3.25}$$

Applying the transformation described in (3.24), the desired linearized system (3.19) becomes:

$$\boldsymbol{p}_{e_{n+1}} - (e^{-\Lambda_1 T} + e^{-\Lambda_2 T})\boldsymbol{p}_{e_n} + e^{-(\Lambda_1+\Lambda_2)T}\boldsymbol{p}_{e_{n-1}} = 0 \tag{3.26}$$

Also, by combining (3.22) and (3.23):

$$\boldsymbol{p}_{n+1} = \boldsymbol{p}_n + \operatorname{sinc}\left(\frac{\omega_n T}{2}\right) P(\theta_n, \omega_n)\boldsymbol{v}_{n-1}T + \operatorname{sinc}\left(\frac{\omega_n T}{2}\right) P(\theta_n, \omega_n)\boldsymbol{a}_n T^2 \tag{3.27}$$

Using (3.26) and (3.27), and solving for $\boldsymbol{a}_n$:

$$\boldsymbol{a}_n = -\frac{\boldsymbol{v}_{n-1}}{T} + \frac{1}{T^2}\left(\operatorname{sinc}\left(\frac{\omega_n T}{2}\right)\right)^{-1} P^{-1}(\theta_n, \omega_n)\left(-\boldsymbol{p}_n + \boldsymbol{p}_{g_{n+1}} + (e^{-\Lambda_1 T} + e^{-\Lambda_2 T})\boldsymbol{p}_{e_n} - e^{-(\Lambda_1+\Lambda_2)T}\boldsymbol{p}_{e_{n-1}}\right)$$

■

It should be noted that this control law is also susceptible to the singularities discussed in Section 3.1. The conditions in which (3.20) should be used instead of the continuous-time approximation depend on the time step T and, in the cases where target tracking is intended (such as ball interception), on the motion of the target object. If T is not fixed, and if the average time step, $\bar{T}$, is sufficiently small, and the values for λ_1 and λ_2 are selected accordingly ($< 1/(10\bar{T})$), then the control law defined by (3.16) is preferred since it avoids the feed-forward of $\boldsymbol{p}_{g_{n+1}}$, which may be a source of error. If T is large, then the closed loop system's poles have to be sufficiently slow so that the continuous-time approximation still holds, which results in an overall slow transient response. Under these conditions, the discrete control law (3.20) provides more reliable control. However, if T is not held constant, then the discretization operation will lose its validity, and may result in unexpected behavior. Results are shown for both of these approaches in Chapter 7.

Note that a system using these control laws may experience overshoot for some initial conditions $\boldsymbol{p}_e(0)$ and $\dot{\boldsymbol{p}}_e(0)$. This is undesirable for most situations, and so the relationship between these initial conditions, pole selection, and the resulting overshoot of the system, must be described. Consider the situation where $\lambda_1 = \lambda_2 = \lambda$. For each component e of $\boldsymbol{p}_e$, it is verified, according to (3.19), that:

$$\ddot{e} + 2\lambda\dot{e} + \lambda^2 e = 0 \tag{3.28}$$

It can be shown that, in order for overshoot to occur, then $e(t) = 0$ must be verified in finite time[13], and that for this to occur, one must have:

$$\lambda < -\frac{\dot{e}(0)}{e(0)} \qquad if\ e(0) > 0 \tag{3.29}$$

$$\lambda > -\frac{\dot{e}(0)}{e(0)} \qquad if\ e(0) < 0 \tag{3.30}$$

Since the initial conditions are readily available, this implies that, by proper selection of λ, overshoot can be avoided altogether in most situations.

3.3 Orientation control

By treating orientation and position control separately, the orientation control problem now becomes simple, since both its kinematic and dynamic models consist solely of chained integrators:

$$\dot{\theta} = \omega \tag{3.31}$$

$$\ddot{\theta} = \alpha \tag{3.32}$$

At each iteration, an estimate of the robot's orientation, $\tilde{\theta}_n$, is supplied by its self-localization procedures, but for this purpose it will be assumed that $\tilde{\theta}_n = \theta_n$. The robot is then given a reference orientation θ_{r_n}. Depending on the task at hand, it may be necessary to provide either velocity or torque control for orientation, such as for the posture stabilization and posture tracking problems, respectively. Adequate compensation for each of these cases can be found through direct digital design, as it will be seen in 3.3.1 for velocity control and 3.3.2 for torque control.

3.3.1 Velocity Control

The transfer function in continuous-time for the kinematic orientation model is, as it can be readily seen through (3.31):

$$G_v(s) = \frac{1}{s} \tag{3.33}$$

In discrete-time, this corresponds to:

$$G_v(z) = (1 - z^{-1})Z\left\{\frac{G_v(s)}{s}\right\} = T\frac{1}{z-1} \tag{3.34}$$

where Z represents the Z-transform. The system is marginally stable, similarly to its continuous-time counterpart, but through the application of a simple proportional controller, the single closed-loop pole can be placed anywhere on the real axis inside the unit circle, as it is seen through the discrete root-locus of $G_v(z)$, shown in Figure 5. Since the reference in these cases is constant (or so it is intended), and the sensors used to determine the robot's orientation are not particularly noisy, the only design specification is that the system should be as fast as possible. This would correspond, evidently, to placing the closed loop pole at $z = 0$. However, the nonlinearities introduced by the robot's actuators (that have a certain maximum velocity and acceleration) would not allow the robot to respond sufficiently fast, and would likely result in overshoot. Therefore, the closed loop pole must be moved further to the right side of the complex plane.

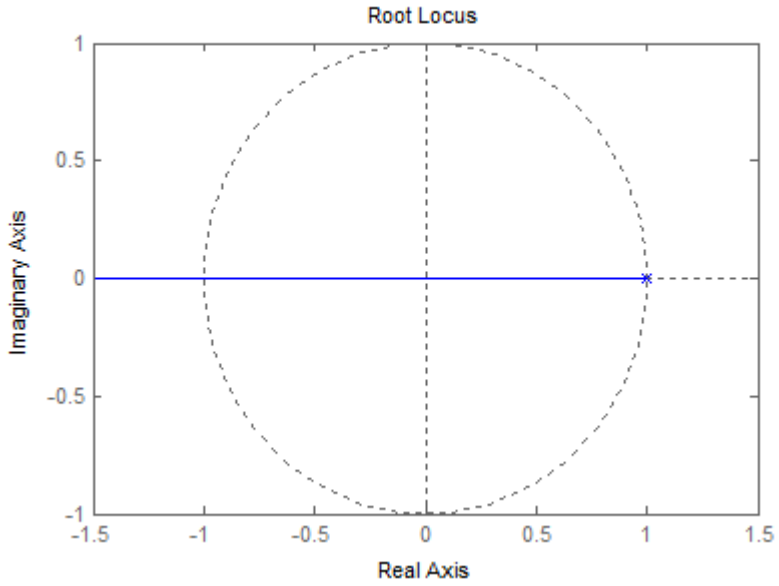


Figure 5: Root-Locus for the discrete-time kinematic model for orientation

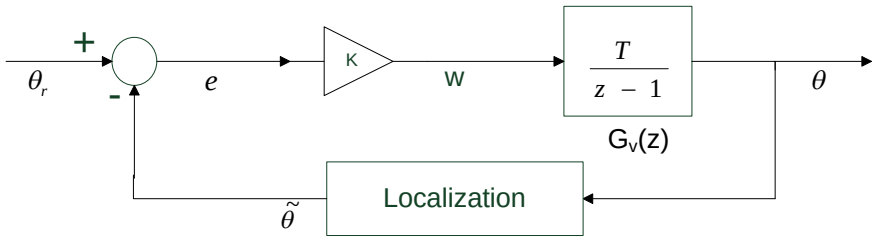


Figure 6: Block diagram for velocity-based orientation control

The necessary gain K to move the closed loop pole to $z = \alpha$, with $0 \leq \alpha < 1$, can then be obtained by determining the characteristic equation of the closed loop system:

$$1 + K\frac{T}{z-1} = 0 \Leftrightarrow z - 1 + KT = 0 \tag{3.35}$$

This, in turn, implies, for $z = \alpha$:

$$K = \frac{1-\alpha}{T} \tag{3.36}$$

The difference equation that must then be implemented in the respective navigation primitive is:

$$\omega_n = \frac{1-\alpha}{T} e_n \tag{3.37}$$

Where $e_n = \left(\theta_{r_n} - \tilde{\theta}_n\right)$. For the particular case of orientation control on the robotic platform used by the ISocRob team, it was experimentally verified that $\alpha = 0.89$ provides the fastest possible response without causing overshoot.

3.3.2 Torque Control

For this case, the continuous-time transfer function of the robot's dynamic model for orientation (3.32) is:

$$G_a(s) = \frac{1}{s^2} \tag{3.38}$$

Its corresponding equivalent in discrete-time is:

$$G_a(z) = (1 - z^{-1}) Z\left\{\frac{G_a(s)}{s}\right\} = \frac{T^2}{2}\frac{z+1}{z^2 - 2z + 1} \tag{3.39}$$

Unlike the previous case, the system quickly becomes unstable through the application of proportional feedback, as it is shown in Figure 7. To overcome this problem, lead compensation is added, of the form:

$$D_a(z) = K\frac{z-b}{z} \tag{3.40}$$

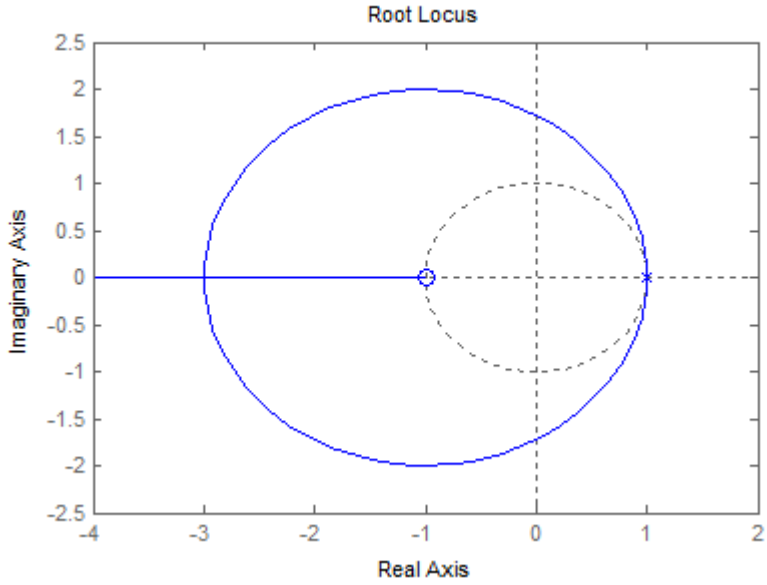


Figure 7: Root-Locus for the discrete-time dynamic model for orientation

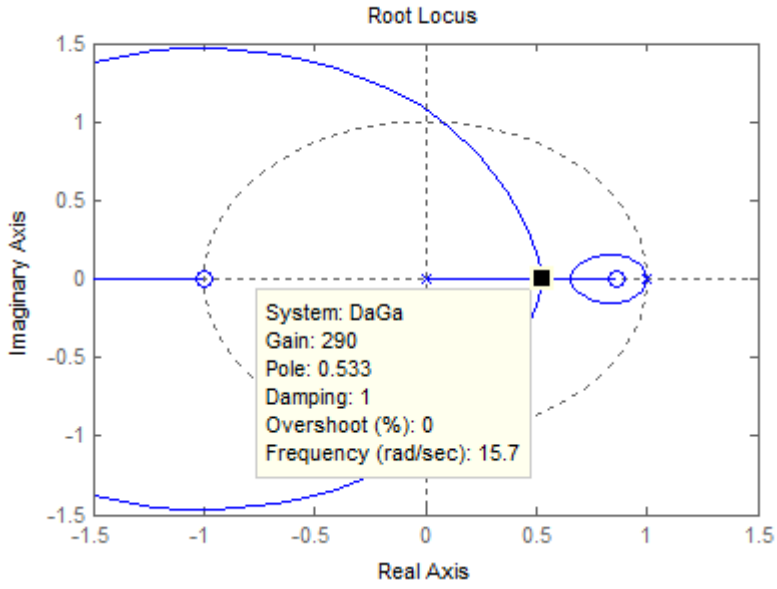


Figure 8: Root-locus for the compensated orientation dynamic model (b=0.86).

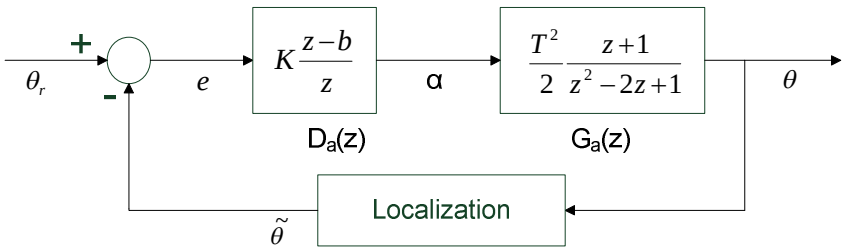


Figure 9: Block diagram for torque-based orientation control

The inclusion of this compensation term yields a closed-loop system that is stable for properly selected values of K and b. The main design difference between velocity and torque control is that,

while in the former it was assumed that the references were constant over time, in the latter the system must be prepared to follow orientation references with constant velocity (ramp inputs) and references with constant acceleration (parabola inputs). In this case, noise rejection is also an important factor, since the reference is sometimes taken as the relative orientation to a certain object in the field (when one wants to keep the robot turned to the ball during interception, for example), and the subsystems that provide these references may be subject to noise. As a consequence, the selection of the parameter b is not straightforward: by moving the zero closer to the origin, the system has greater noise rejection at higher frequencies, but its phase margin is reduced, since the breakaway point in the locus would no longer exist, and the system would have a complex pair of poles; by moving the zero farther to the right, the system becomes slower in the presence of the nonlinearities of the robot's actuators, and the steady-state error to references with constant acceleration increases. A proper value must then be determined for each specific physical system under study. For the particular case of the ISocRob robots, after some trial and error, a suitable value was found at $b = 0.86$. The corresponding root-locus is shown in Figure 8. For this particular value of b, a value of K is selected that is as high as possible while maintaining an acceptable phase margin. This corresponds to the case where two of the closed loop system's poles are located at the break-away point in the locus, $z = 0.533$, and a third real pole is located at $z = 0.703$. Then, by obtaining the closed loop system's characteristic equation for and matching the desired locations of the poles (the process is omitted since it follows the same reasoning as in Section 3.3.1, but the algebraic details are rather strenuous and do not contribute to this work), it can be shown that the relationship between K and T, follows:

$$K = \frac{\eta}{T^2} \tag{3.41}$$

where η is dependent on the selected value for b and the desired closed loop poles. In this case, $\eta = 0.661$. The difference equation that must be implemented for this controller can be obtained by noting that:

$$D_a(z) = \frac{A(z)}{E(z)} = \frac{\eta}{T^2}\frac{z-b}{z} \tag{3.42}$$

This implies that:

$$\alpha_n = \frac{\eta}{T^2}(e_n - be_{n-1}) \tag{3.43}$$

3.4 Dealing with Actuator Saturation Problems

One of the biggest obstacles to rigorous high-speed motion control is dealing with the limitations of the robot's actuators. While, for the case of omnidirectional robots, the process of obtaining motion control laws is simple since its motion models are easily manipulated, it is often left unconsidered, from a theoretical standpoint, that the nonlinearities introduced by the robot's actuators may affect the overall performance of those laws. Suppose that the robot's actuators have an absolute maximum limit for angular velocity, ω_{MAX}^{a}, and for acceleration, α_{MAX}^{a}, where the superscript is used to distinguish these quantities from the angular velocity and acceleration of the robot's chassis.

In other words, it is imposed by the actuators that:

$$\begin{aligned} &|\dot{\varphi}_i| \leq \omega_{MAX}^{a} \\ &|\ddot{\varphi}_i| \leq \alpha_{MAX}^{a},\ i = 1,2,3 \end{aligned} \tag{3.44}$$

This is in an approximation, since, in reality, the maximum acceleration that each actuator is able to display is dependent on the instantaneous angular velocity of that actuator. However, to avoid damaging the actuators by frequently driving them to their actual limits in acceleration, a *soft-limit* is imposed, which can be set to a constant value.

The linear and angular velocity of the robot chassis can then be related to the angular velocities of the actuators (see Annex A1) by:

$$r\begin{bmatrix}\dot{\varphi}_1\\ \dot{\varphi}_2\\ \dot{\varphi}_3\end{bmatrix} = \begin{bmatrix} 0 & -1 & L \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & L \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & L \end{bmatrix}\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = J\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{3.45}$$

In these conditions, it is evident that there always exists a set of initial conditions that, through the application of the motion control laws proposed in the previous sections, would always tend to violate the saturation limits of some of the actuators. If, for example, the corresponding input vector $[v_x \quad v_y \quad \omega]^T$ would require all of the actuators to spin in the same direction, albeit with different desired angular velocities, then upon saturation of all the wheels, the robot's chassis would simply spin in place with a certain maximum angular velocity, and it would therefore be prevented from reaching its target postures or trajectories. Indiveri et al. proposed in [9] a method to account for these limits, by assigning a finite "task capacity" to each individual input sent to the actuators, so that the weighted sum of these inputs never exceeds a certain maximum admissible velocity of the robot's chassis (a maximum norm of the control input vector), which supposedly would lead to actuator saturation. This however, is only an approximation: through (3.45), it can be seen that the relationship between the angular velocity of each actuator and the *norm* of the desired control input vector is non-linear. This, in turn, means that there are some directions along which the input vector is allowed to have a greater norm before at least one of the actuators enters saturation. Also, in their work, Indiveri

et al. did not consider the presence of a saturation limit for acceleration. The present work will then base itself on the approach taken in [9], but will try to address its shortcomings.

The main line of reason that will be followed is that, in the presence of certain desired velocities, the angular velocity of each actuator must be altered so that the robot still maintains its desired functionality.

Let (3.45) be rewritten as:

$$r\begin{bmatrix}\dot{\varphi}_1^d\\ \dot{\varphi}_2^d\\ \dot{\varphi}_3^d\end{bmatrix}=\begin{bmatrix}0 & -1\\ -\frac{\sqrt{3}}{2} & \frac{1}{2}\\ \frac{\sqrt{3}}{2} & \frac{1}{2}\end{bmatrix}\begin{bmatrix}v_x\\ v_y\end{bmatrix}+\begin{bmatrix}L\\ L\\ L\end{bmatrix}\omega=J_v\begin{bmatrix}v_x\\ v_y\end{bmatrix}+J_\omega\omega \tag{3.46}$$

where $[\dot{\varphi}_1^d \quad \dot{\varphi}_2^d \quad \dot{\varphi}_3^d]^T$ represents the "desired" angular velocities for each of the actuators. From these, let $\dot{\varphi}_{MAX}^d$ denote the component with the largest absolute value. At least one of the actuators will be saturated if (3.44) is not verified, i.e., if $\left|\dot{\varphi}_{MAX}^d\right| > \omega_{MAX}^a$. In these conditions, by defining a scale factor K_V as:

$$K_V=\frac{\omega_{MAX}^a}{\left|\dot{\varphi}_{MAX}^d\right|} \tag{3.47}$$

By applying this scale factor:

$$\begin{bmatrix}\dot{\varphi}_1\\ \dot{\varphi}_2\\ \dot{\varphi}_3\end{bmatrix}=K_V\begin{bmatrix}\dot{\varphi}_1^d\\ \dot{\varphi}_2^d\\ \dot{\varphi}_3^d\end{bmatrix} \tag{3.48}$$

It results that the so obtained angular velocities satisfy $|\dot{\varphi}_i| \leq \omega_{MAX}^a$, and that the associated, "real" velocities displayed by the robot are also scaled down by K_V, relative to their desired values. This means that the form of the robot's trajectory is maintained, even though the robot will pursue this trajectory in a slower manner than it is desired. The available control effort of the robot's actuators is distributed equitably through the linear and angular velocity components of the robot. Depending on the required task, it may be advantageous to give priority to one of these components over the other. This may happen, for example, in tasks where the robot is required to reach a desired position as fast as possible, but its orientation is unimportant – in this case its linear velocity would be given priority. To account for these cases, three different "priority modes" are distinguished: the above procedure, that conserves the form of the robot's trajectory but may result in a slower required time to its target, constitutes a *no-priority* situation; additionally, priority may be given to either the robot's linear velocity or its angular velocity.

Let $\dot{\varphi}_i^v$ and $\dot{\varphi}_i^\omega$ represent the contributions to the angular velocity of the actuators by, respectively, the robot's linear and angular velocities:

$$\begin{bmatrix} \dot{\varphi}_1^v \\ \dot{\varphi}_2^v \\ \dot{\varphi}_3^v \end{bmatrix} = \frac{1}{r} J_v \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{3.49}$$

$$\begin{bmatrix} \dot{\varphi}_1^\omega \\ \dot{\varphi}_2^\omega \\ \dot{\varphi}_3^\omega \end{bmatrix} = \frac{1}{r} J_\omega \omega \tag{3.50}$$

With these definitions, (3.46) becomes,

$$\begin{bmatrix} \dot{\varphi}_1^d \\ \dot{\varphi}_2^d \\ \dot{\varphi}_3^d \end{bmatrix} = \begin{bmatrix} \dot{\varphi}_1^v \\ \dot{\varphi}_2^v \\ \dot{\varphi}_3^v \end{bmatrix} + \begin{bmatrix} \dot{\varphi}_1^\omega \\ \dot{\varphi}_2^\omega \\ \dot{\varphi}_3^\omega \end{bmatrix} \tag{3.51}$$

where it can be seen that each of these components may be treated individually. When priority is given to linear velocity, the largest value of $\dot{\varphi}_i^v$ is determined, $\dot{\varphi}_{MAX}^v$. If $|\dot{\varphi}_{MAX}^v| \geq \omega_{MAX}^a$, then all of the control effort must be assigned to linear velocity, i.e. $\dot{\varphi}_i^\omega = 0$, and the desired values of $\dot{\varphi}_i^v$ are scaled down according to $K_V = \omega_{MAX}^a / |\dot{\varphi}_{MAX}^v|$, in a process similar to the no-priority situation. If $|\dot{\varphi}_{MAX}^v| < \omega_{MAX}^a$, then the actuators will not enter saturation through the application of the desired linear velocity, and so the remaining control effort may be allocated to the robot's angular velocity. In this case, the values of $\dot{\varphi}_i^v$ are left in their original form, and those of $\dot{\varphi}_i^\omega$ must be scaled down. K_V then becomes:

$$K_V = \frac{\omega_{MAX}^a - |\dot{\varphi}_{MAX}^v|}{|\dot{\varphi}_{MAX}^d| - |\dot{\varphi}_{MAX}^v|} \tag{3.52}$$

The term $\omega_{MAX}^a - |\dot{\varphi}_{MAX}^v|$ represents the available control effort. The effective angular velocities of the actuators are then:

$$\begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \end{bmatrix} = \begin{bmatrix} \dot{\varphi}_1^v \\ \dot{\varphi}_2^v \\ \dot{\varphi}_3^v \end{bmatrix} + K_V \begin{bmatrix} \dot{\varphi}_1^\omega \\ \dot{\varphi}_2^\omega \\ \dot{\varphi}_3^\omega \end{bmatrix} \tag{3.53}$$

For angular velocity priority, the situation is reversed. If $|\dot{\varphi}_{MAX}^\omega| < \omega_{MAX}^a$, then,

$$K_V = \frac{\omega_{MAX}^a - |\dot{\varphi}_{MAX}^\omega|}{|\dot{\varphi}_{MAX}^d| - |\dot{\varphi}_{MAX}^\omega|} \tag{3.54}$$

In this case,

$$\begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \end{bmatrix} = K_V \begin{bmatrix} \dot{\varphi}_1^v \\ \dot{\varphi}_2^v \\ \dot{\varphi}_3^v \end{bmatrix} + \begin{bmatrix} \dot{\varphi}_1^\omega \\ \dot{\varphi}_2^\omega \\ \dot{\varphi}_3^\omega \end{bmatrix} \tag{3.55}$$

The above procedures would account for the angular velocity limits of the robot's actuators. However, it is also important to consider the effect of the angular acceleration limits. To see this, suppose that the robot's actuators can exhibit a large angular velocity, but a reduced angular acceleration. Also, suppose that the robot is initially stopped, and the velocity input sent to the actuators implies $\dot{\varphi}_i > K$, where K is some positive constant, for all of the actuators. In this situation, before one of the actuators reaches its desired value for $\dot{\varphi}_i$, they will all display equal angular velocity values, since they all accelerate by the same maximum amount α_{MAX}^{a}. This would cause the robot to spin with $v_{x,y} = 0$ during the initial instants of its motion.

The relation between the desired accelerations of the robot's chassis and the angular accelerations of the actuators is:

$$\begin{bmatrix} \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \\ \ddot{\varphi}_3 \end{bmatrix} = \frac{1}{r} J \begin{bmatrix} a_x \\ a_y \\ \alpha \end{bmatrix} \tag{3.56}$$

Since this relation is similar to what was seen for the case of velocity inputs, all of the above derivations can be applied to treat $\ddot{\varphi}_i$, also considering the possibility of assigning priority to the different components of the robot's acceleration. Note, however, that even if no priority is specified, there is no guarantee that the robot's trajectory is preserved.

4 Task-Specific Motion Control

In the robotic soccer domain, some tasks cannot be solved efficiently through the isolated application of the motion control techniques presented in Chapter 3. Most of these tasks involve some form of interaction with the soccer ball, since in this case the robot must abide by some restrictions on its motion, imposed by the physical dimensions of both the robot and the ball, to ensure that the interaction is successful. These restrictions, in turn, invalidate the assumption of unconstrained motion of the soccer robot, and have to be dealt with specifically depending on the type of interaction that the robot must achieve. Two major tasks are identified in this sense:

- Ball interception, where the robot is required to gain possession of the ball;
- Dribbling (or ball moving), where the robot must traverse the field of play while maintaining the ball under its possession.

This chapter is organized as follows: Section 4.1 introduces the basic model of the soccer ball that will be used in the subsequent tasks; Section 4.2 describes a procedure to intercept a freely rolling ball; finally, Section 4.3 describes the motion control techniques that are used during the transport of the ball (dribbling).

4.1 Modelling the Object

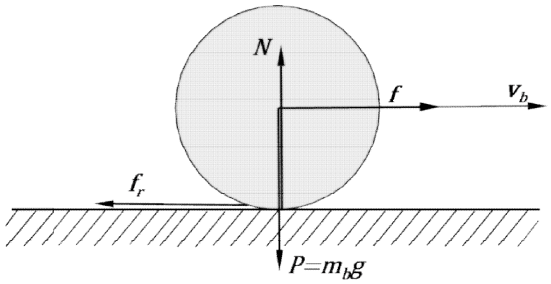


Figure 10: Diagram showing the forces involved in the motion of the ball.

Whenever interaction between a robot and an object is performed through motion control, it is often necessary to possess not only a dynamic model of the robot itself, but also of the object in question. This model describes how the motion of the object is affected when subject to the forces arising from its interaction with the robot.

The object in question, in the robotic soccer setting, is a ball that is rolling on a non-inclined plane, with position $\boldsymbol{p}_b = [x \quad y]^T$ on a fixed frame. Its basic equations of motion state that:

$$\boldsymbol{f} + \boldsymbol{f}_r = m_B \ddot{\boldsymbol{p}}_b \tag{4.1}$$

where m_B is the mass of the ball, $\boldsymbol{f}$ is an external force applied through the ball's center of mass, and $\boldsymbol{f}_r$ is the friction between the ball and the surface. The friction is proportional to the weight of the ball, and its direction is contrary to the velocity of the ball's center of mass $\boldsymbol{v}_b = \dot{\boldsymbol{p}}_b = [v_x \quad v_y]^T$:

$$\boldsymbol{f}_r = -C_{rr} m_b g\big(\cos(\alpha_b)\boldsymbol{e}_x + \sin(\alpha_b)\boldsymbol{e}_y\big) \tag{4.2}$$

where $\alpha_b = \operatorname{atan2}(v_y, v_x)$, C_{rr} is the coefficient of rolling resistance and $g = 9.81\, ms^{-2}$. C_{rr} is approximately constant in the range of feasible velocities.

Rewriting (4.1) in state-space form, where the state of the ball is $[\boldsymbol{p}_b \quad \boldsymbol{v}_b]^T$:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} + \frac{1}{m_B} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \end{bmatrix} + \frac{1}{m_B} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} f_{r_x} \\ f_{r_y} \end{bmatrix}$$

$$\boldsymbol{p}_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \tag{4.3}$$

Note that, in some tasks where the interaction forces with the ball must be controlled, $\boldsymbol{f}$ is accessible, but $\boldsymbol{f}_r$ is not, and of the latter only an estimate can be obtained.

4.2 Moving Object Interception

Consider now the set of problems where a robot must intercept a moving object, in such a way that the interaction between the robot and the object when contact occurs must respect some dynamic constraints. The interception task is then to match the position and velocity (and acceleration, if applicable) of the target object, in the shortest possible time. This time constraint is a fundamental difference from situations involving stationary objects, especially for robotic manipulators, since there is a limited time window in which the target object is passing through the manipulator's workspace. For mobile robots, the environment's configuration or the limitations of the robot's actuators could also mean that the interception of the target object would become impossible after some time.

Depending on the task at hand, the target object may be classified as a *slow-maneuvering* target, in which case its motion is predictable, such as in an industrial environment, or a *fast-maneuvering* target, implying that long-term prediction is hard to achieve. Targets of the latter type are usually free-flying or may be subject to random deviations in trajectory.

Suitable solutions for systems handling slow-maneuvering targets include the Prediction, Planning and Execution (PPE) techniques, such as [14], in which an anticipated rendezvous point is calculated based on the predicted path of the object, and a trajectory is then obtained that would drive the robot to this point. Active Prediction, Planning and Execution (APPE) tries to reduce the effect of prediction errors by continuously updating the rendezvous point and the robot's trajectory as necessary [15],[16]. These solutions present near-optimal results in the required time for interception.

For fast-maneuvering targets, however, it may not be possible to obtain a reliable long-term prediction of the target's motion. Possible solutions in this case include tracking algorithms that continuously try to minimize some measure of error between the robot and the target object. The application of visual servoing is a widely used example, such as in [17],[18], either by Image-Based Visual Servoing (IBVS), where the images produced by a camera are searched for a set of features, and the displacement to their expected location at the time of interception is measured; or by Position-Based Visual Servoing, (PBVS) where the posture of the robot (or of the robot's end-effector in the case of a manipulator) is estimated from the received images and the error to the desired posture is then used. Manchester et al. applied these techniques to the interception of moving targets by nonholonomic mobile robots in [19]. These algorithms are normally sub-optimal with respect to time. Fast-maneuvering target interception may also be accomplished through the application of *guidance* laws, which have wide application in homing missile control (e.g. [20],[21], [22]) and are designed to achieve time-optimal interception. These guidance laws do not try to match the velocity of the target at the moment of interception nor do they have the capability, by themselves, to satisfy physical constraints imposed on the interaction with the target. Therefore, when implemented in robotic interception systems, they require the application of another tracking algorithm in the vicinity of the target object to overcome these limitations.

In the robotic soccer environment, such a problem of intercepting a moving object occurs whenever a robot is required to gain possession of the soccer ball, whether it is freely rolling on the field of play, or being controlled by a robot of the opposite team. In this sense, the target is considered as fast-moving, since its trajectory may change unexpectedly due to collision with other objects in the environment, or simply from frictional forces at work that can distort its motion.

The most common approach to ball interception in the robotic soccer environment is through the application of neural networks and learning algorithms [23],[24], [25]. These techniques require a reduction in the dimensionality of the ball interception problem, and a large number of training episodes, some of which have to be performed in the physical robotic system (as opposed to simulation procedures), to achieve acceptable ball interception under most conditions. The main advantage of these techniques is that the overall system does not require modelling, and so the complexity of the problem is reduced. However, the resulting behavior of the system, after the training process, is hard to analyze. Also, for long-term robotic projects that are subject to frequent changes, this implies that the training process would often have to be repeated, and some part of it even individually for each system in the robotic soccer team.

Other solutions for robotic soccer rely on fast visual tracking [26], that are heavily dependent on the robot's hardware, or prediction techniques [27],[25], that, as discussed above, do not account for unexpected changes in the ball's motion due to collision, for example.

The proposed solution to this problem is based on the work done by Mehrandezh et al. in [28],[13],[29], which was applied in the context of fast-moving object interception for industrial manipulators, and relies on a hybrid composition of trajectory-tracking and proportional navigation

techniques to achieve near-optimal interception. This particular implementation also makes use of obstacle avoidance to ensure that the robot does not collide with the ball before it is prepared to capture it.

4.2.1 Overview of the Solution

From this point onward, the focus will be put, without loss of generality, on the particular case where a soccer ball is freely rolling on the field of play. It is assumed that the robot has, at each instant, information about the ball's position, velocity and acceleration. Given arbitrary initial conditions for both the robot and the ball, the robot must then achieve a posture such that the ball would be between its flippers (see Section 6) and in contact with robot's kicker mechanism – the robot is then said to have possession of the ball – and apply the necessary force to stop the ball. In its motion, the robot must also avoid any obstacles that are present in the environment.

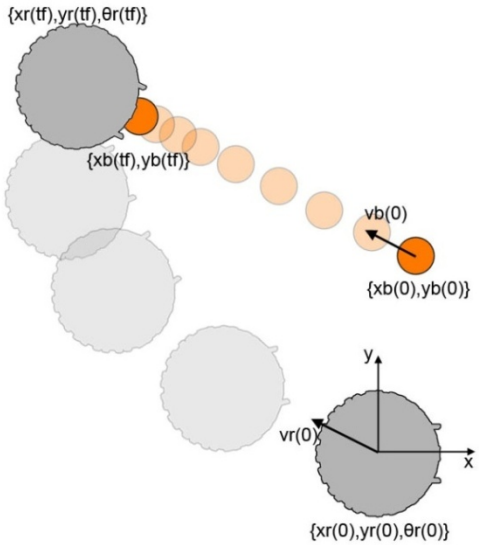


Figure 11: Representation of the ball interception task.

The control architecture is shown in Figure 12. A ball-localization sensor, the workings of which are not relevant here, supplies periodic information about the ball's position and velocity to the navigation primitives. The ball's acceleration is not accessible, and therefore an estimate based on the friction acting on the ball is also supplied; a posture is then calculated which satisfies the physical restrictions of both the ball and the robot. This posture, along with the ball's velocity and acceleration, defines a set of possible "interception trajectories", which are those that result in a successful capture by the robot. One such trajectory is obtained, and serves as reference to separate position and orientation control subsystems. For orientation control, the system described in 3.3 is used in torque-control form, since, as it will be seen, it is convenient to supply the controls for the robot as linear and angular accelerations during this task. As for position, either a point-tracking module or a *proportional navigation guidance* (described in 4.2.4) module is used, according to a set of restrictions that must be satisfied for each of them. The resulting control input for position is then passed through a ball-

avoidance module so as to assure that no undesired collisions result. The applied obstacle avoidance algorithm is discussed in Section 5.4 for moving obstacles. In the following sections each of the remaining modules is discussed in detail.

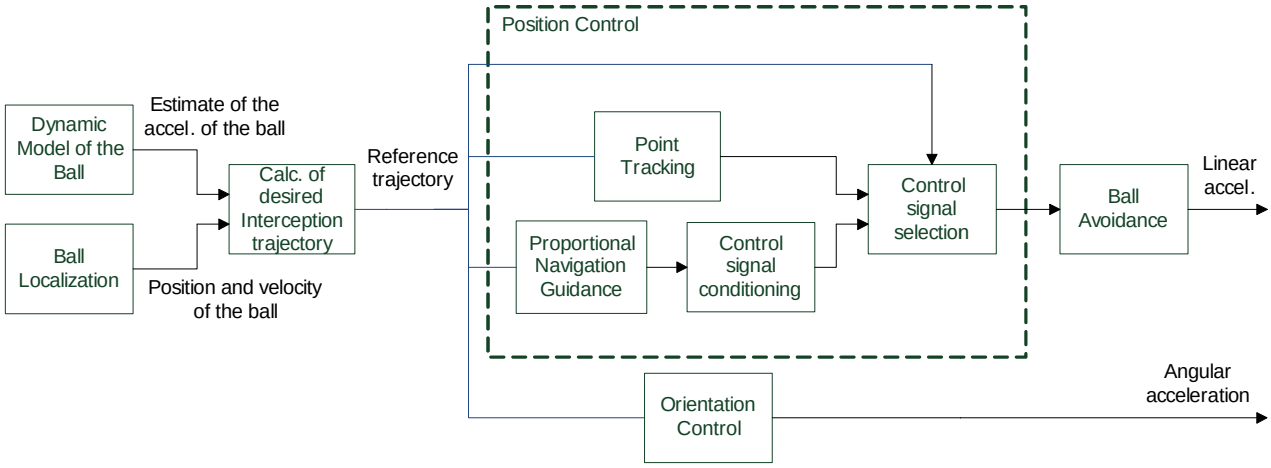


Figure 12: Control architecture for the ball interception system.

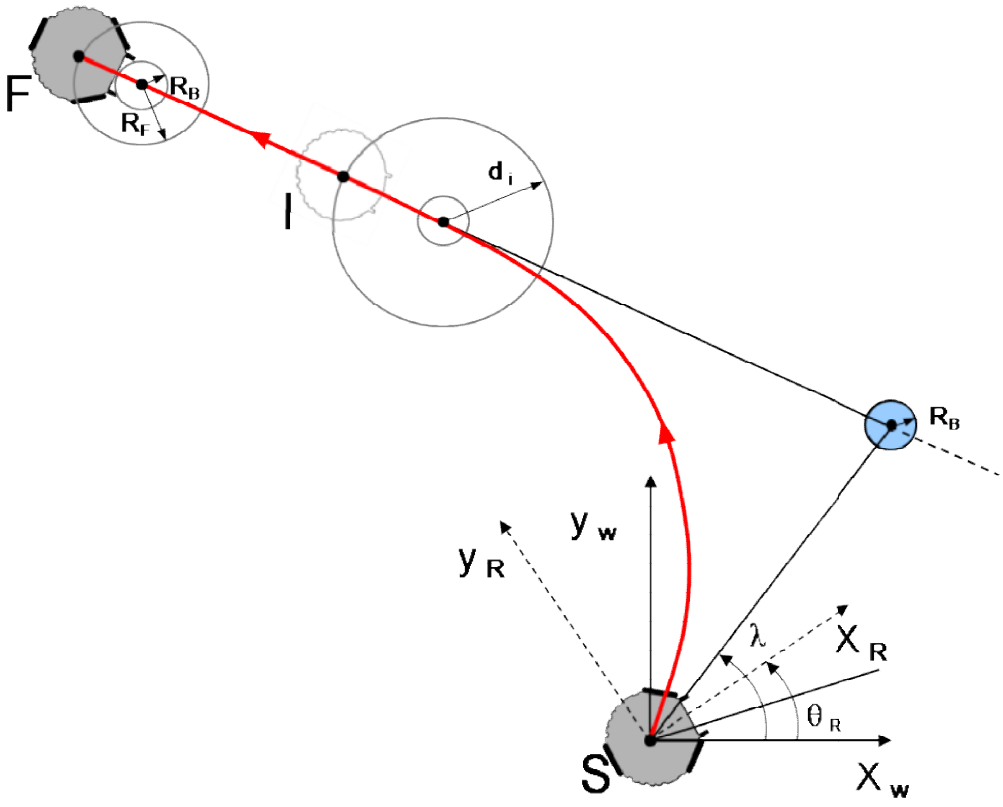


Figure 13: An example of an admissible trajectory, in the world frame, for the ball interception problem.

4.2.2 Obtaining the Desired Interception Trajectory

Regardless of the algorithm which is used to solve the ball interception problem, there is a set of restrictions imposed by the physical dimensions of both the robot and the ball that must be verified at the moment of interception. Let the position of the robot in the world frame be defined as $\boldsymbol{q}_r = [\boldsymbol{p}_r{}^T \quad \theta_r]^T$, where $\boldsymbol{p}_r = [x_r \quad y_r]^T$, and the position of the object, in this case the ball, as $\boldsymbol{p}_b = [x_b \quad y_b]^T$. In this notation, a trajectory described by the robot is represented by $\boldsymbol{q}_r(t)$, and is assumed twice differentiable. Successful interception will be attained by the trajectories $\boldsymbol{q}_r(t)\ \epsilon\ T_{int}$, where T_{int} represents the set of all trajectories that satisfy, at some time instant t_f:

1. $\|\boldsymbol{p}_r(t_f) - \boldsymbol{p}_b(t_f)\| = R_f$, where R_f is the distance between the center of the robot and the ball whenever ball possession is achieved;
2. $\|\dot{\boldsymbol{p}}_r(t_f) - \dot{\boldsymbol{p}}_b(t_f)\| \leq \delta_v$, where δ_v is a constant related to the maximum change in momentum that the ball may be subjected to without escaping;
3. $\|\ddot{\boldsymbol{p}}_r(t_f) - \ddot{\boldsymbol{p}}_b(t_f)\| \leq \delta_a$, where δ_a is a constant representing the maximum resultant force that may act on the ball for it to remain under possession, taking into account the effect any actuators that are exerting forces upon the ball;
4. $|\theta_r(t_f) - \lambda(t_f)| \leq \delta_\theta$, where $\lambda(t) = atan2\big(y_b(t) - y_r(t), x_b(t) - x_r(t)\big)$, as denoted in Figure 13, and δ_θ represents any "slack" that may exist between the robot's flippers and the ball during ball possession.

Given the particular configuration of Middle-Sized League soccer robots, an intuitive subset of T_{int} are the trajectories in which the robot is positioned along the ball's immediate direction of motion at the moment of interception, in order to facilitate ball capture, since its linear momentum is opposed by the robot. An example of such a trajectory is depicted in Figure 13, and is divided into two distinct segments, each with specific motion control requirements:

(S-I): The robot must intercept a point I along the ball's path, which is assumed to be linear in that interval, in the shortest possible time. It is also convenient (although not strictly necessary) to impose that the robot should match the ball's velocity and acceleration at this point. The robot then satisfies, at some time instant t_i:

- $\|\boldsymbol{p}_r(t_i) - \boldsymbol{p}_b(t_i)\| = d_i$, where $d_i > R_f$ is the distance between the center of the ball and point I, and should be selected for safety reasons. It is also clear that d_i directly influences the total time required for interception, and should be kept as small as possible;
- $\|\dot{\boldsymbol{p}}_r(t_i) - \dot{\boldsymbol{p}}_b(t_i)\| \cong 0$;
- $\|\ddot{\boldsymbol{p}}_r(t_i) - \ddot{\boldsymbol{p}}_b(t_i)\| \cong 0$.

Since the robot is omnidirectional and the control of its orientation can be accomplished independently from its position, as seen before, the orientation control primitives described in 3.3 can be used to satisfy condition 4 throughout the motion of the robot, by using $\lambda(t)$ as reference. Successful interception then reduces to a problem of matching the desired trajectory for position $p_i(t)$ defined by the motion of point I. From the above, this trajectory, designated as *interception trajectory*, can be related to $p_b(t)$ as:

$$\boldsymbol{p}_i(t) = \boldsymbol{p}_b(t) + d_i \begin{bmatrix} \cos\left(\text{atan2}(\dot{y}_b(t), \dot{x}_b(t)\right) \\ \sin\left(\text{atan2}(\dot{y}_b(t), \dot{x}_b(t)\right) \end{bmatrix} \tag{4.4}$$

Since the ball's path was assumed linear, it is also evident that:

$$\dot{\boldsymbol{p}}_i(t) = \dot{\boldsymbol{p}}_b(t) \tag{4.5}$$

$$\ddot{\boldsymbol{p}}_{\boldsymbol{i}}(t) = \ddot{\boldsymbol{p}}_{\boldsymbol{b}}(t) \tag{4.6}$$

The necessary quantities ($p_b(t), \dot{p}_b(t)$ and $\ddot{p}_b(t)$) are assumed to be known at each instant (in the case of the ISocRob robots, $p_b(t)$ and $\dot{p}_b(t)$ are supplied by the sensors of each robot, while an estimate of $\ddot{p}_b(t)$ is obtained through the dynamic model of the ball). In practice, this information will evidently contain some amount of error. This does not, however, prevent the successful operation of the algorithm (see Section 7.4), since, as the distance between the robot and the ball decreases, the amount of noise involved in these measurements is usually reduced.

(I-F): The robot then approaches point F in such a way that conditions 1-3 of the above are satisfied at the interception instant t_f. If point I is sufficiently close to the ball, this can be achieved through constant negative acceleration. As before, condition 4 is satisfied by the orientation controller.

This interception strategy is similar to a common approach in the control of robotic manipulators, where the end effector is first brought to the vicinity of the manipulated object as fast as possible (coarse control), and from there it approaches the object in a slower manner in which the interaction forces have priority (fine control). In their work, Mehrandezh et al. have applied a similar strategy [13]. Since the control if the robot is straightforward during the final stage (I-F) of interception, the subsequent sections address the motion control requirements of the first stage (S-I).

4.2.3 Motion Control Requirements of an Interception Task

The problem of matching a given moving reference position $p_i(t)$ during the first stage (S-I) of interception, is essentially a point-tracking problem, and could be solved by the application of the navigation primitives described in 3.2. However, this would not result in the shortest possible interception time, since this method is normally sub-optimal in that aspect. For this reason, the application of a guidance law to the interception task becomes advantageous. Not only do these guidance laws help reduce the total required time for interception under normal conditions when compared to a pure trajectory-tracking solution [28], but they also deal more efficiently with unexpected variations of the target's motion. However, when under the control of a guidance law, the robot is not capable of matching the desired interception trajectory, since these laws only ensure that the robot will eventually reach point I, but have no means to slow down the robot, in order to satisfy the conditions presented in 4.2.2. Therefore, the application of trajectory tracking is still necessary in these circumstances, and the control of the robot must then alternate between these two methods in

an appropriate manner so as to optimize the required time for interception. In the following sections, the operation of both of these methods is detailed: the guidance law applied in this work, *Ideal Proportional Navigation Guidance*, is described in Section 4.2.4; the application of trajectory tracking to the interception process is discussed in Section 4.2.6; the process of selecting the appropriate control method at each instant in order to optimize the interception time is described in Section 4.2.7.

4.2.4 Ideal Proportional Navigation Guidance

Proportional Navigation Guidance (PNG) laws have been widely applied in the context of free-flying object interception, usually in the context of aerodynamically controlled systems, since they were introduced in the 1950s [20], due to their simplicity of implementation and low level of required information. The premise behind guidance laws these is simple: two moving objects are on a collision course when the relative angle between them remains constant, and the distance between them is decreasing. To accomplish this, a control signal in acceleration form is obtained. The way through which this control signal is calculated characterizes these laws as either *pursuer-velocity referenced* or *Line-Of-Sight* (LOS) *referenced* laws. A comparison between the two classes can be found in [30].

The proposed solution utilizes a LOS-referenced guidance law known as Ideal Proportional Navigation Guidance (IPNG), which was introduced in [31] and also used in robotic interception in [13]. IPNG is superior over other existing LOS-referenced laws in terms of robustness to initial conditions and required time for interception [32]. Despite being dismissed as unpractical for missile guidance due to its physical requirements, such as lateral acceleration, it is well suited for robotic interception, particularly in the case of systems with unrestricted mobility. For omnidirectional robots, the resulting control acts solely on position, and no restrictions on orientation are imposed.

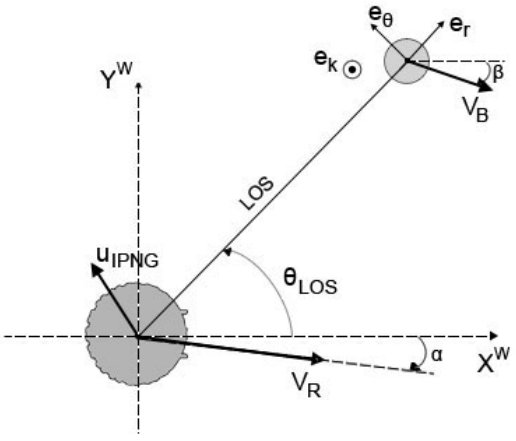


Figure 14: Interception geometry through IPNG.

Consider the interception geometry presented in Figure 14. Note that, for clarity, these derivations will be made considering that the intended target is the center of the ball. As discussed in

Section 4.2.2, the real target has a constant offset in position to accommodate for the physical restrictions of the robot. This has no other implications in the description of this method.

Let the velocity of the pursuer, which in this case is an omnidirectional robot, and of the target, which is the ball, be defined, in the world frame, as:

$$\dot{\boldsymbol{p}}_r = V_R(\cos(\alpha)\,\boldsymbol{e_x} + \sin(\alpha)\,\boldsymbol{e}_y) \tag{4.7}$$

$$\dot{\boldsymbol{p}}_b = V_B(\cos(\beta)\,\boldsymbol{e_x} + \sin(\beta)\,\boldsymbol{e}_y) \tag{4.8}$$

where α and β are, respectively, the angles of the velocity of the robot and of the ball in the world frame.

Even though this analysis is made in the world frame, the procedure is not affected by self-localization errors, since only the relative position between the robot and the ball is relevant, and this is achieved through ball localization. It is also assumed that $V_R \geq V_B$, although this is not necessary for the convergence of the control law. Note that, for a freely rolling ball, and assuming that there are no boundaries on the robot's environment, this condition will always be achieved after some time, since the ball is decelerating due to friction. In a bounded environment, such as in a robotic soccer field, it is possible that this condition is never verified, and, for this reason, interception could become impossible for some initial conditions of V_R and α.

Also, let $\boldsymbol{r}$ denote the displacement between pursuer and target, also in the world frame:

$$\boldsymbol{r} = (x_B - x_R)\boldsymbol{e_x} + (y_B - y_R)\boldsymbol{e_y} \tag{4.9}$$

The Line-Of-Sight angle, θ_{LOS}, is:

$$\theta_{LOS} = atan2(y_B - y_R, x_B - x_R) \tag{4.10}$$

Now consider the coordinate system $\{e_r, e_\theta, e_k\}$ defined as shown in Figure 14. In these coordinates, the acceleration command obtained through IPNG is, by definition:

$$\boldsymbol{a}_{IPNG} = N\dot{\boldsymbol{r}} \times \dot{\boldsymbol{\theta}}_{LOS} \tag{4.11}$$

where N is the *Navigation Constant*, and $\dot{\boldsymbol{\theta}}_{LOS} = \dot{\theta}_{LOS}\boldsymbol{e}_k$. It can be shown that with $N > 2$ interception is always achieved [31].

As seen by equation (4.11), IPNG generates an acceleration which is orthogonal to the relative velocity between the target and pursuer, which implies that the norm of the latter will be preserved. In LOS-referenced coordinates, this means that there is a component along the normal to the LOS which tries to nullify the LOS rate, $\dot{\boldsymbol{\theta}}_{LOS}$, and a component along the direction tangent to the LOS, which acts to keep the norm of the relative velocity constant. One important property that stems from this fact is that as the LOS rate approaches zero, the closing velocity approaches constant values, and so the trajectory described by the robot, relative to the ball, is approximately linear. Also, note that if $V_R = 0$,

IPNG would generate an acceleration command orthogonal to the velocity of the ball, which is undesirable since the initial velocity of the robot in these conditions would not be directed towards the interception point, and the subsequent turning effort would increase the required time for interception. Guidance laws such as IPNG are usually derived considering that the speed of the pursuer is constant and greater than that of target [31],[32]. Although this does not prevent these laws from converging if this condition is not initially verified, other control methods (such as trajectory tracking) prove more efficient in this interval [13]. It is for this reason that IPNG is only used when $V_R > V_B$.

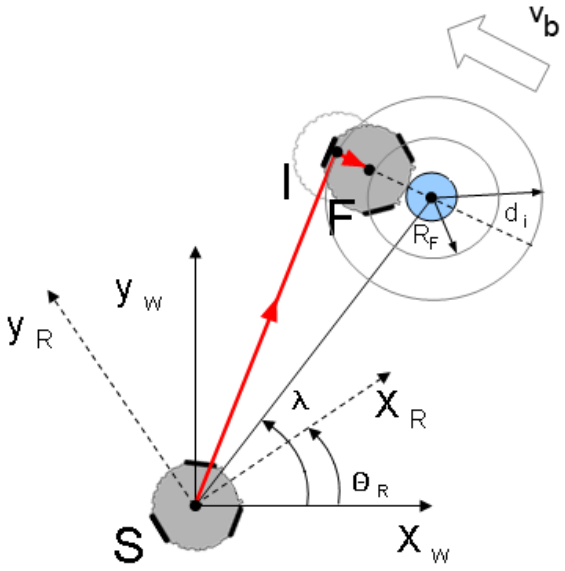


Figure 15: Representation of the trajectory described by the robot in a frame moving with the same velocity as the ball. While under the control of IPNG, this trajectory is approximately linear.

The presence of two acceleration components demands both forward and lateral accelerations by the pursuer, which are always achievable by omnidirectional robots (provided that its actuators are not saturated, as it was covered in Section 3.4), but may not be possible or desirable for other classes of mobile robots. In those cases, the application of a more conservative velocity-referenced guidance law, such as Pure Proportional Navigation Guidance (PPNG,[30]), would achieve similar, albeit slightly inferior, results (in fact, Yang et al. show in [32] that IPNG is an extreme case of PPNG).

Also in the $\{e_r, e_\theta, e_k\}$ coordinate system, it follows that:

$$\boldsymbol{r} = r\boldsymbol{e_r} \tag{4.12}$$

$$\dot{\boldsymbol{r}} = \dot{r}\boldsymbol{e_r} + \dot{\theta}_{LOS}\boldsymbol{e}_k \times \boldsymbol{r} \Leftrightarrow$$

$$\Leftrightarrow \dot{\boldsymbol{r}} = \dot{r}\boldsymbol{e_r} + r\dot{\theta}_{LOS}\boldsymbol{e_\theta} \tag{4.13}$$

Separating the target and pursuer velocities in their normal and tangent components:

$$\dot{r} = V_B \cos(\theta_{LOS} - \beta) - V_R \cos(\theta_{LOS} - \alpha) \tag{4.14}$$

$$r\dot{\theta}_{LOS} = -V_B \sin(\theta_{LOS} - \beta) + V_R \sin(\theta_{LOS} - \alpha) \Leftrightarrow$$

$$\Leftrightarrow \dot{\theta}_{LOS} = \frac{1}{r}(-V_B \sin(\theta_{LOS} - \beta) + V_R \sin(\theta_{LOS} - \alpha)) \tag{4.15}$$

Through the substitution of (4.14) and (4.15) in (4.13), the IPNG acceleration command is obtained with (4.11). In order to obtain this acceleration in the world frame, a simple rotation is applied:

$$\boldsymbol{a}_{IPNG}^{W} = B(\theta_{LOS})\left(N\dot{\boldsymbol{r}} \times \dot{\boldsymbol{\theta}}_{LOS}\right) \tag{4.16}$$

4.2.5 Control Signal Conditioning for IPNG

While the control signal obtained through (4.16) ensures that successful interception is achieved for $N > 2$, no considerations are made about the robot's dynamic restrictions. As it was seen in Section 3.4, if this control signal exceeds the limitations of the robot's actuators, its components are scaled accordingly to minimize the distortion of the robot's trajectory; however, it is also possible that the desired acceleration is too small, and fails to make use of robot's capabilities. In fact, the control signal described by (4.16) becomes null whenever a constant LOS rate is achieved, effectively "locking" the closing velocity to a constant value, along with whatever velocity the robot has at the time. Since the IPNG command itself depends solely on the relative velocity, the control signal itself has to be modified in order to reduce the required time for interception.

The most direct way to achieve this is by accelerating the robot along the tangent to the LOS.

Recall from (3.56) that:

$$\begin{bmatrix} \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \\ \ddot{\varphi}_3 \end{bmatrix} = \frac{1}{r} J \begin{bmatrix} a_x \\ a_y \\ \alpha \end{bmatrix} \tag{4.17}$$

By adding an acceleration command a_b tangent to the LOS direction, it results that:

$$\begin{bmatrix} \ddot{\varphi}'_1 \\ \ddot{\varphi}'_2 \\ \ddot{\varphi}'_3 \end{bmatrix} = \frac{1}{r} J \begin{bmatrix} a_x \\ a_y \\ \alpha \end{bmatrix} + \frac{1}{r} J \left(R^{-1}(\theta_r) \begin{bmatrix} B(\theta_{LOS}) & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{bmatrix} a_b \\ 0 \\ 0 \end{bmatrix} \Leftrightarrow$$

$$\Leftrightarrow \begin{bmatrix} \ddot{\varphi}'_1 \\ \ddot{\varphi}'_2 \\ \ddot{\varphi}'_3 \end{bmatrix} = \frac{1}{r} J \begin{bmatrix} a_x \\ a_y \\ \alpha \end{bmatrix} + \frac{a_b}{r} J \left(R^{-1}(\theta_r) \begin{bmatrix} B(\theta_{LOS}) & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{4.18}$$

which can be rewritten as,

$$\begin{bmatrix} \ddot{\varphi}_1' \\ \ddot{\varphi}_2' \\ \ddot{\varphi}_3' \end{bmatrix} = \begin{bmatrix} \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \\ \ddot{\varphi}_3 \end{bmatrix} + a_b \begin{bmatrix} \ddot{\varphi}_1^b \\ \ddot{\varphi}_2^b \\ \ddot{\varphi}_3^b \end{bmatrix} \tag{4.19}$$

The term $[\ddot{\varphi}_1^b \quad \ddot{\varphi}_2^b \quad \ddot{\varphi}_3^b]^T$ represents the contribution of a unit vector in the LOS direction to the angular accelerations of the robot's actuators. The maximum permissible value for a_b is then:

$$a_b = \min_{i=1,2,3} \frac{\alpha_{MAX}^a - \ddot{\varphi}_i}{\ddot{\varphi}_i^b} \tag{4.20}$$

4.2.6 Matching the Interception Trajectory through Point Tracking

It is important to assure that, when the robot is under the control of the point-tracking primitives, no overshoot occurs due to poor selection of the point-tracking controller's poles, and that the limits of the robot's actuators are respected. To achieve this, consider the application of (3.16) with $\boldsymbol{p}_g = \boldsymbol{p}_i$ and $\lambda_1 = \lambda_2 = \lambda$:

$$\boldsymbol{a_{PT}} = B^{-1}(\theta_r)\left(-\dot{B}(\theta_r)\boldsymbol{v} + \ddot{\boldsymbol{p}}_i - 2\Lambda(\dot{\boldsymbol{p}}_r - \dot{\boldsymbol{p}}_i) - \Lambda^2(\boldsymbol{p}_r - \boldsymbol{p}_i)\right) \tag{4.21}$$

It can be easily verified, through (4.17), if this control law produces values outside of the range of the actuators. If so, and for every iteration where this happens, a value for λ is numerically sought in the intervals defined by (3.29) and (3.30), such that the resulting signal $\boldsymbol{a_{PT}}$ is achievable by the robot. In the event that it is impossible to find a proper value for λ, then overshoot is inevitable. This may happen if, for instance, the control switches over from IPNG to point-tracking too close to the goal point, and the robot isn't able to brake sufficiently fast. These situations must be avoided by properly determining the switching instants.

4.2.7 Selection of the Proper Control Signal

Having described the application of IPNG and point tracking to the ball interception problem, it is important to analyze how these techniques can be used together in the most efficient manner, i.e., in order to minimize the total required time for interception. The total time is taken as the sum of the time during which the robot is under the control of IPNG, t_{IPNG}, point-tracking, t_{PT}, and braking, t_B:

$$t_{total} = t_{IPNG} + t_{PT} + t_B \tag{4.22}$$

As it was seen previously, the main premise behind the usage of IPNG is that it is faster in normal conditions than the point-tracking primitives, and should therefore be used as long as possible in the (S-I) segment of the interception process. As with most proportional navigation laws, however,

IPNG is not efficient when $V_R < V_B$, and so control should also be assigned to point-tracking in this case. Once the robot reaches point **I**, by verifying the conditions presented in 4.2.2, the robot then brakes until it comes to a full stop, with a constant acceleration of the form:

$$\boldsymbol{a_B} = [K\cos(\alpha + \pi) \quad K\sin(\alpha + \pi)]^T \tag{4.23}$$

where $K > 0$ and α is the angle of the robot's linear velocity in the world frame. The procedure to select the appropriate controls during ball interception in represented in Figure 16. An example of the different types of motion control applied during interception is shown in Figure 17, where the instants where the control is switched are denoted as t_{S1}, t_{S2} and t_{S3}. Of these, t_{S2} has the greatest effect on the total interception time, since t_{S1} and t_{S3} cannot be freely selected. The minimum of t_{total} is then achieved by selecting t_{S2} as late as possible, without inducing the point tracking controller into an overshoot situation, since overshoot would necessarily increase t_{PT}, and consequently t_{total}. In [13] and [29], the optimal switching instant was determined by estimating the required time to interception, which requires finding the solutions to (3.28) in real-time. In the current approach, it was instead opted to directly check for the existence of overshoot (through the concepts discussed in Section 4.2.6), since this follows from the selection of the point tracking controller's poles, which has to be performed anyway. The resulting functionality is equivalent, and in either case, in order to assert at any given iteration if control should be assigned to point tracking, it is necessary to predict the position and velocity of both the robot and the ball in the following iteration, if rigorous control is required. If the sampling period is small enough however, this prediction step can be overlooked, since by selecting t_{S2} as the first instant in which overshoot is detected, the effective overshoot around the target point is negligible.

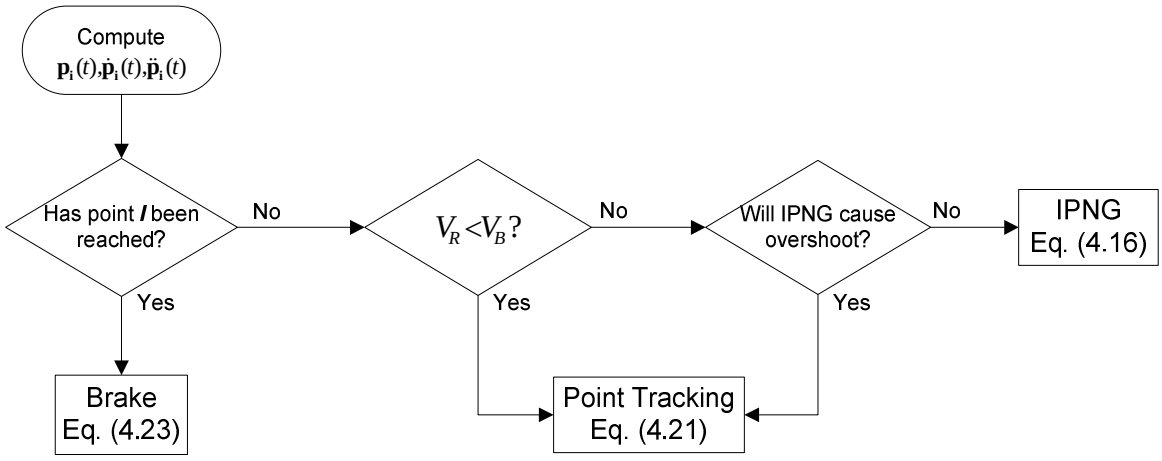
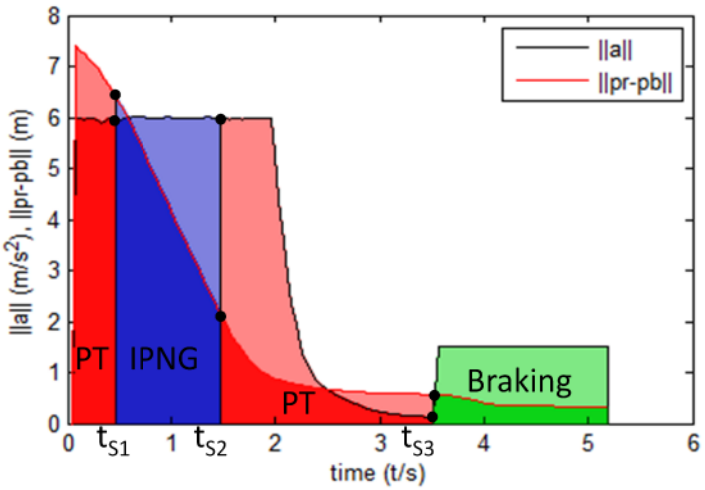
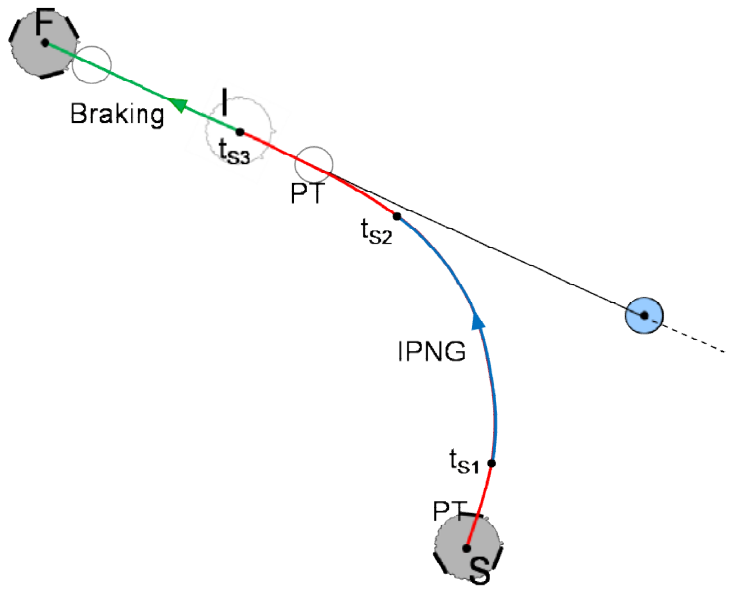


Figure 16: Algorithm to determine the suitable type of motion control during ball interception.



(a)



(b)

Figure 17: An example of a ball interception task. (a): the control signals (represented by the magnitude of the linear acceleration) applied to the robot, and the distance from the robot to the ball, during interception. (b): The trajectory described by the robot in the world frame, when these controls are applied. In both cases, the various types of motion control are also shown: Point Tracking (PT), IPNG and constant deceleration (Braking), as well as the switching instants t_{S1}, t_{S2} and t_{S3}.

4.3 Object Transport

The focus is now given to the class of problems in which a mobile robot is supposed to interact with a movable object in its environment, by displacing it from its original location to some given target point. The robot achieves this by pushing the object while it is in contact with some part of its chassis. An evident example of such a problem is the fundamental ability of moving the ball across the field of play in the robotic soccer domain, under some geometry constraints, without losing it, which is one of the most important functionalities that a robotic soccer player must possess. Depending on the robot's configuration and type of locomotion, different control strategies may be employed to achieve smooth ball movement. For a differential-drive robot, for example, the robot must always rely on some physical apparatus fixed to the robot's chassis, the so-called flippers (see Section 6) to maintain the ball under possession and within reach of its kicker mechanisms. Whenever the robot turns whilst having possession of the ball, the ball is forced to turn "with" the robot, and these flippers keep the ball from rolling away. This has been the approach taken until now by the ISocRob team. Previous to the current work, the implementation of the dribbling process was based on a potential-fields based approach, which enforced some restrictions on the robot's motion to account for the forces exerted to the ball by the robot's flippers [2]. A similar approach was taken in [33] to extend these concepts to omnidirectional robots, but required the predetermination of a path that would not infringe the physical restrictions of the dribbling process. This limits the usefulness of the solution in a highly dynamical environment such as robotic soccer, where global motion planning is usually avoided.

As with the ball interception problem, many approaches to dribbling in robotic soccer are based in reinforcement learning and neural network techniques (e.g.[34] and respective references). These approaches suffer from the same limitations of being difficult to adapt to changes in the physical dimensions that are relevant to the dribbling process.

4.3.1 Overview of the Solution

For an omnidirectional robotic soccer player, the desired behavior is such that the robot is able to turn "around" the ball whilst the ball is moving, as depicted in Figure 19. In this way, the robot is able to maintain the ball under possession regardless of the required motion (assuming that the velocity of the ball is under a certain limit, as it will be seen). It should also be noted that soccer robots usually possess specialized actuators that act to keep the ball in contact with the robot's chassis, but, by themselves, are not allowed by the game rules to be capable of maintaining the ball under possession at all times. Therefore, the dynamics of the ball should not be overlooked, and the role of such actuators should be to provide added stability to the system.

The problem then reduces to applying a pushing force to the ball with a certain magnitude and direction so that the ball will eventually be driven to its target. The proposed solution is based on the Interface-Control scheme introduced by Nakamura et al. in [35], in the context of coordinative robotic manipulation, and applied to the manipulation of free-flying objects by mobile manipulators in [36]. In this control scheme, the resultant forces that should be applied to the object at each instant are

obtained, through an “object controller” that seeks to stabilize the object around a given reference; these forces then serve as references that the robot must achieve, and through them the necessary control inputs to the robot’s actuators are found, and the object is effectively displaced. Therefore, the object controller acts only indirectly upon the dynamic model of the object, and the desired forces that it provides serve as an “interface” to the robot’s controller. To this end, the robot’s controller must be capable of following independent force and position references. A control technique particularly suited to these specifications is Hybrid force/position control [37],[38],[8], which makes use of the physical constraints placed upon the robot to reduce the dimension of the robot’s state, thereby simplifying the control problem.

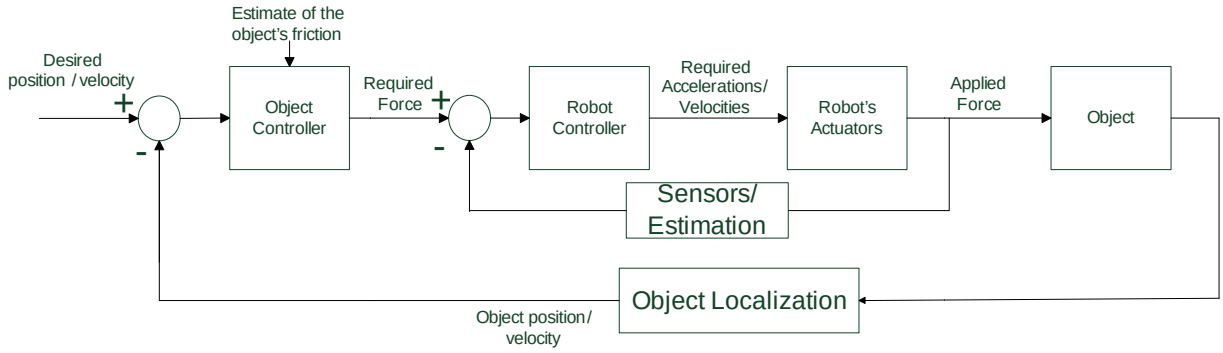


Figure 18: The Interface-Control scheme for the manipulation of an object by a single robot.

It should be pointed out that, during a robotic soccer match, the robot is not normally required to stabilize the ball around a reference position; instead, it is only required to drive the ball through a given reference as fast as possible (i.e. overall, the system must be controllable, but not necessarily stabilizable). However, to fully demonstrate the capabilities of the proposed solution, the stabilization problem will be considered, without loss of generality, since its extension to the regular requirements of a robotic soccer match is straightforward.

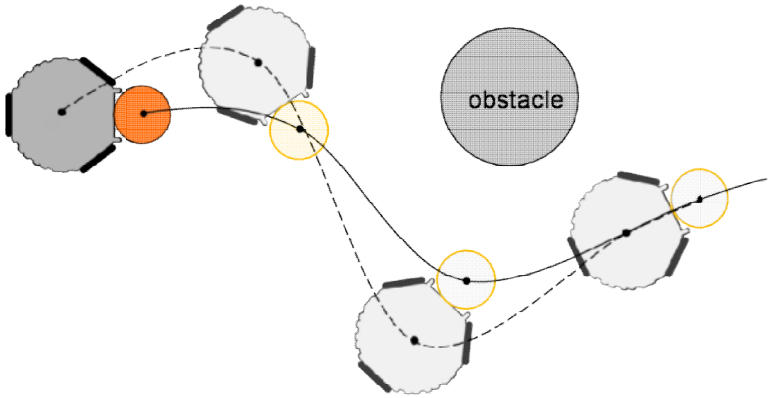


Figure 19: Representation of the dribbling process for an omnidirectional robot. An example of a trajectory described by the center of the ball is shown as a solid line. The desired trajectory described by the center of an omnidirectional robot dribbling the ball is shown as a dashed line.

4.3.2 Object Controller – PD Control

In an environment with multiple robotic systems manipulating an object in a coordinate manner, as in the original formulation of the Interface-Control scheme, when determining the desired resultant force that must be applied to the object, it is necessary to take into account its shape and the internal forces that arise from the presence of the multiple forces exerted upon its surface. However, in the dribbling problem, it is assumed that a single robot exerts a force through the ball's center of mass at any given instant. As such, any controller that is able to stabilize the dynamic model of the ball (4.3) around a reference position is suitable for a dribbling task. Taking $\boldsymbol{u} = \boldsymbol{f}$, this dynamic model can also be written in standard form as:

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + B(\boldsymbol{u} + \boldsymbol{f_r}) \\ \boldsymbol{y} &= C\boldsymbol{x} \end{aligned} \tag{4.24}$$

It is assumed that the state $\boldsymbol{x} = [\boldsymbol{p}_b \quad \boldsymbol{v}_b]^T$ is fully accessible. It can also be readily shown that the system is controllable (although that is an intuitive result). The problem is then to obtain a control law of the form:

$$\boldsymbol{u} = -K\boldsymbol{x} - \boldsymbol{f_r} \tag{4.25}$$

where K is a gain matrix such that the closed loop system described by $A - BK$ has stable poles. Recall that $\boldsymbol{f_r}$ may be estimated, since the physical dimensions of the ball, and the friction coefficient between the ball and the field of play, are known. By choosing K as:

$$K = \begin{bmatrix} k_p m_b & 0 & k_d m_b & 0 \\ 0 & k_p m_b & 0 & k_d m_b \end{bmatrix} \tag{4.26}$$

The closed loop system then becomes:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k_p & 0 & -k_d & 0 \\ 0 & -k_p & 0 & -k_d \end{bmatrix} \boldsymbol{x} \tag{4.27}$$

As a consequence, the system may be decoupled into two linear second order systems possessing the characteristic equation:

$$s^2 + k_d s + k_p \tag{4.28}$$

This is to say that both of these systems, each corresponding to a component of the ball's position, are under PD control. A well known result is that, by selecting $k_d = 2\sqrt{k_p}$, the system becomes critically damped, with poles at $s = -\sqrt{k_p} = -\frac{k_d}{2}$. The criteria for the selection of the poles is

reminiscent to that of the trajectory tracking primitive described in Section 3.2; in fact, the saturation limits of the robot's actuators impose a maximum admissible speed for the ball while dribbling, which acts as a saturation limit upon the ball itself. Above this speed value, the robot will no longer be able to accompany the ball along its path, and the ball will be lost. Likewise, a maximum admissible force exists which is related to the maximum acceleration that the robot is able to impart to the ball. The poles must then be placed in order to attenuate the effects of these nonlinearities, and to guarantee that no overshoot occurs given arbitrary initial conditions.

4.3.3 Robot Controller – Hybrid Position/Force Control

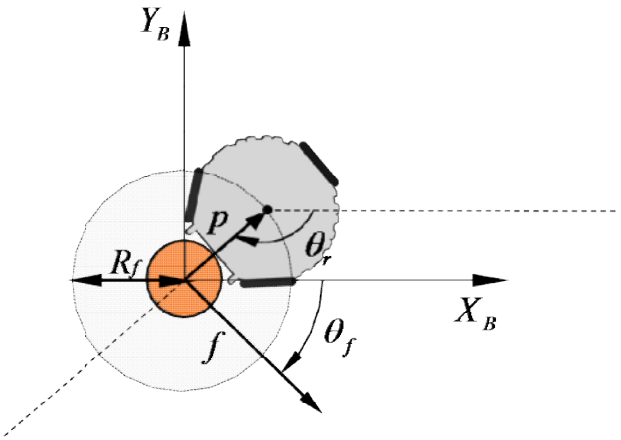


Figure 20: Geometric details of the dribbling process.

Given an instantaneous required force $\boldsymbol{f}$ that must be applied on the ball supplied by the object controller (the output of the object controller, so in the notation of Section 4.3.2, $\boldsymbol{f} = \boldsymbol{u}$), and assuming that the robot has the ball under its possession, the objective of the robot controller is to exert that force on the ball while satisfying a set of physical restrictions that allow the robot to dribble the ball continuously. In these conditions, the ball is positioned along the robot frame's *x*-axis at some distance R_f from the center of the robot. Since the required force is linear and through the ball's center of mass, and assuming that the robot's lateral flippers should not be relied on for ball handling except when strictly necessary (rather, they should provide added safety during dribbling), then this implies that any forces applied to the ball by the robot should also be along the robot frame's *x*-axis. This, in turn, means that the robot must align itself with the required force vector at all times.

Let $\boldsymbol{q}_r^W = [(\boldsymbol{p}_r^W)^T \quad \theta_r]^T$ denote the posture of the robot in the world frame, and $\boldsymbol{p}_b^W$ the position of the ball in that frame. Throughout this section, the superscript *W* will be used to denote vectors in the world frame, whereas *R* will be used to denote vectors in the robot frame. If the superscript is omitted, then the corresponding vector belongs to a frame centered on the ball, as represented in Figure 20. In this frame, the posture of the robot is given by $\boldsymbol{q} = [x \quad y \quad \theta_r]^T = [(\boldsymbol{p}_r^W - \boldsymbol{p}_b^W)^T \quad \theta_r]^T$, and the desired force is $\boldsymbol{f} = F\left(\cos(\theta_f)\, e_x + \sin(\theta_f) e_y\right)$, i.e. $F = \|\boldsymbol{f}\|$ and $\theta_f = \operatorname{atan2}(f_y, f_x)$.The physical restrictions that the robot is subject to while holding the ball, which were already introduced in the ball interception problem, are here re-written as:

$$x^2 + y^2 = {R_f}^2 \tag{4.29}$$

$$\text{atan2}(-y, -x) = \theta_r \tag{4.30}$$

These restrictions define independent constraint surfaces in $\boldsymbol{R^3}$**:**

$$c_i(\boldsymbol{q}) = 0 \quad , i = 1{,}2 \tag{4.31}$$

where $c_1(\boldsymbol{q})$ and $c_2(\boldsymbol{q})$ represents restrictions (4.29) and (4.30), respectively. Contrarily to the ball interception task, however, the only admissible displacements made by the robot are those that satisfy these constraints continuously. In this case, it is advantageous to control both the position and orientation of the robot simultaneously. The constraint-compliant control of a robotic system through hybrid position/force control has been derived thoroughly in [37]. Here, these concepts are adapted to the particular problem of dribbling by an omnidirectional robot.

Consider the Jacobian E_F of the vector-valued function $\boldsymbol{c}(\boldsymbol{q}) = [c_1(\boldsymbol{q}) \quad c_2(\boldsymbol{q})]^T$:

$$E_F = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}^{-1} \begin{bmatrix} x & y & 0 \\ \dfrac{y}{{R_f}^2} & -\dfrac{x}{{R_f}^2} & 1 \end{bmatrix} \tag{4.32}$$

where α and β are chosen so that each row of E_F has unitary norm, i.e.:

$$\begin{aligned} \alpha &= R_f = x^2 + y^2 \\ \beta &= \sqrt{\frac{1 + \alpha^2}{\alpha^2}} \end{aligned} \tag{4.33}$$

Naturally, each row vector of E_F is normal to one of the constraint surfaces, and, as can be readily seen, orthogonal to each other. By defining the complementary unit vector E_P as:

$$E_P = \begin{bmatrix} -\dfrac{y}{\alpha\beta} & \dfrac{x}{\alpha\beta} & \dfrac{1}{\alpha\beta} \end{bmatrix} \tag{4.34}$$

which is orthogonal to both of the rows of E_F and also of unit length, then an orthonormal basis to *constraint space* can be formed through E_P and the rows of E_F, and the coordinate frame defined by these vectors and centered on the robot's current posture is designated as *constraint frame*. Vectors in the constraint frame with non-zero components along either of the axes defined by E_F tend to violate the imposed constraints. Similarly, vectors that are collinear with E_P maintain the validity of these constraints. Let the square matrix E be defined as:

$$E = \begin{bmatrix} E_P \\ E_F \end{bmatrix} \tag{4.35}$$

Matrix E transforms vectors from the frame centered on the ball to the constraint frame. The objective of the robot controller, however, is to provide an acceleration control in the robot frame, $\boldsymbol{a}_r^R = \ddot{\boldsymbol{q}}_r^R$. To this end, defining $\boldsymbol{q}_b = [(\boldsymbol{p}_b)^T \quad 0]^T$:

$$\dot{\boldsymbol{q}} = R(\dot{\boldsymbol{q}}_r^R - \dot{\boldsymbol{q}}_b^R) \tag{4.36}$$

where $R = R(\theta_r)$. Differentiating once more, and rearranging terms:

$$\ddot{\boldsymbol{q}}_r^R = R^{-1}\left(\ddot{\boldsymbol{q}} - \dot{R}(\dot{\boldsymbol{q}}_r^R - \dot{\boldsymbol{q}}_b^R)\right) + \ddot{\boldsymbol{q}}_b^R \tag{4.37}$$

The acceleration of the ball in the robot's frame, $\ddot{\boldsymbol{q}}_b^R$, can be readily obtained through an estimate of the friction acting on the ball (i.e. $\boldsymbol{f_r}$). The relationship between the acceleration of the ball in the world (or ball) frame and in the robot's frame is described in [33].
Also, note that, by definition:

$$E_F\dot{\boldsymbol{q}} = 0 \tag{4.38}$$

$$E_F\ddot{\boldsymbol{q}} + \dot{E}_F\dot{\boldsymbol{q}} = 0 \tag{4.39}$$

Let $\boldsymbol{a_{rF}} = \dot{E}_F\dot{\boldsymbol{q}}$. Using (4.32), this term can be obtained as:

$$\boldsymbol{a_{rF}} = \begin{bmatrix} \dfrac{\dot{x}^2 + \dot{y}^2}{\alpha} \\ 0 \end{bmatrix} \tag{4.40}$$

In an analog manner, for E_P it can be seen that:

$$E_P\dot{\boldsymbol{q}} = \frac{1}{\alpha\beta}\dot{\theta}_r \tag{4.41}$$

$$E_P\ddot{\boldsymbol{q}} + \dot{E}_P\dot{\boldsymbol{q}} = E_P\ddot{\boldsymbol{q}} = \frac{1}{\alpha\beta}\ddot{\theta}_r \tag{4.42}$$

In which it was implied that $\dot{E}_P\dot{\boldsymbol{q}} = 0$, which can be readily checked through (4.34). These results show that any motion made by the robot which is consistent with its constraints can be specified solely in terms of its orientation θ_r, which is intuitive since the distance to the ball may not change while dribbling. By doing so, the problem of keeping the robot aligned with the required force vector reduces to that of finding an appropriate control such that θ_r converges towards θ_f, the force vector's angle in the ball-centered frame. Combining (4.39) and (4.42), it follows that:

$$\begin{bmatrix} E_P \\ E_F \end{bmatrix}\ddot{\boldsymbol{q}} = \begin{bmatrix} (1/\alpha\beta)\ddot{\theta}_r \\ -\dot{E}_F\dot{\boldsymbol{q}} \end{bmatrix} \Leftrightarrow$$

$$\Leftrightarrow \ddot{\boldsymbol{q}} = E^{-1}\begin{bmatrix}(1/\alpha\beta)\ddot{\theta}_r \\ -\boldsymbol{a}_{rF}\end{bmatrix} \tag{4.43}$$

The resulting acceleration that must be applied to the robot's actuators, and which satisfies the imposed constraints, is obtained through (4.37), and is given by:

$$\boldsymbol{a}_P = R^{-1}\left(E^{-1}\begin{bmatrix}(1/\alpha\beta)\ddot{\theta}_r \\ -\boldsymbol{a}_{rF}\end{bmatrix} - \dot{R}(\dot{\boldsymbol{q}}_r^R - \dot{\boldsymbol{q}}_b^R)\right) + \ddot{\boldsymbol{q}}_b^R \tag{4.44}$$

Note that this problem of controlling θ_r through $\ddot{\theta}_r$, is another instance of the "$\frac{1}{s^2}$ plant" problem that was already dealt with in Section 3.3.2 in the context of unrestricted motion, and so the same control techniques may be applied.

The acceleration control defined by $\boldsymbol{a}_P$ exerts no force onto the ball, as it only rotates the robot around the ball. The problem then turns to finding a similar control which results in a force of the same magnitude as $\boldsymbol{f}$ being applied to the ball's center of mass. As it was already explained, this could be attained by applying the acceleration $\ddot{\boldsymbol{q}}_r^R = [a_x \quad 0 \quad 0]^T$. However, this is heavily susceptible to errors in the orientation of the robot. A better solution is to consider that the required force will necessarily attempt to infringe the constraints placed on the motion of the robot. Any force $\boldsymbol{f}_{\boldsymbol{ef}}$ exerted by the robot on the ball while dribbling verifies, through the virtual work principle [37],[35]:

$$\dot{\boldsymbol{q}}^T \boldsymbol{f}_{\boldsymbol{ef}} = 0 \tag{4.45}$$

Using (4.38), $\boldsymbol{f}_{\boldsymbol{ef}}$ can be expressed in terms of the constraint frame components belonging to E_F:

$$\boldsymbol{f}_{\boldsymbol{ef}} = E_F^T \boldsymbol{f}_F \tag{4.46}$$

All that is left is then to relate $\boldsymbol{f}$ and $\boldsymbol{f}_F$. It should then be noted that a force directed to the center of the ball is necessarily normal to the constraint surface defined by (4.29). As a result, a force with the same magnitude as the desired force, and directed through its center of mass, can be obtained by selecting:

$$\boldsymbol{f}_F = [-F \quad 0]^T \tag{4.47}$$

This is justified by the fact that any force directed towards the ball is necessarily normal to the constraint surface defined by (4.29). This force can then be transformed into an acceleration control on the robot frame:

$$\boldsymbol{a}_F = \frac{1}{m_b} R^{-1} E_F^T \boldsymbol{f}_F \tag{4.48}$$

The total acceleration that must be applied to the robot is:

$$\boldsymbol{a_C} = \boldsymbol{a}_P + \boldsymbol{a}_F \tag{4.49}$$

The system is then able to follow independent position (through θ_f) and force (through F) trajectories. Consequently, the robot will continuously accompany the ball and apply to it the necessary force returned by the object controller.

By choosing τ_F as being independent of the robot's orientation, the system would behave strangely in the initial instants of its motion, while the robot is not yet aligned with θ_f, since the force $\boldsymbol{f_{ef}}$ would drive the ball in an unintended direction. Therefore, the system must be prevented from acting upon the ball until the orientation error is under some threshold value $T_f > 0$, i.e. $\boldsymbol{a}_F$ is redefined as:

$$\boldsymbol{a}_F = \begin{cases} \frac{1}{m_b} R^{-1} E_F^T \boldsymbol{f}_F & if\ |\theta_r - \theta_f| \leq T_f \\ 0 & otherwise \end{cases} \tag{4.50}$$

Another important problem arises from the assumption that the constraints (4.29) and (4.30) are always valid as long as the robot does not perform any movements that are inconsistent with these constraints. In reality, some amount of error may be introduced in the system, for example, due to inexact modelling and wheel slippage. This error would be accumulated throughout the dribbling process and could result in the loss of the ball. It is thus convenient to include in the position acceleration control defined by (4.50) a *recovery* term $\boldsymbol{a}_S$ that acts to maintain the dribbling constraints valid:

$$\boldsymbol{a}_S = R^{-1} E^{-1} \begin{bmatrix} 0 \\ a_{rS} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \alpha_{\theta S} \end{bmatrix} \tag{4.51}$$

The restrictions are then addressed individually: in order to maintain (4.29) valid, the term $R^{-1}E^{-1}[0 \quad a_{rS} \quad 0]^T$ is included, which acts as a force similar to $\boldsymbol{f_{ef}}$ **,** which drives the robot towards the ball; to satisfy (4.30), a simple torque compensation $[0 \quad 0 \quad \alpha_{\theta S}]^T$ is included which keeps the robot turned to the ball. The values for a_{rS} and $\alpha_{\theta S}$ are obtained through any applicable control laws (these are yet another instance of the double-integrator problem). The resulting control is then:

$$\boldsymbol{a}_C = \boldsymbol{a}_P + \boldsymbol{a}_F + \boldsymbol{a}_S \tag{4.52}$$

As a final note, it should be pointed out that the problem of hybrid position/dynamic control isn't well defined for discrete-time systems. In this case, the controls have to be taken as the finite-difference approximation of their continuous-time counterparts.

4.3.4 Actuator Limitations

It is now necessary to consider that the acceleration controls provided through (4.52) may exceed the capabilities of the robot's actuators. As it was discussed in Section 3.4 for unrestricted motion, even if these acceleration controls are always scaled down to admissible values, there is no guarantee that the form of the robot's trajectory is preserved. Even if it could be preserved by making the robot move in a slower manner, this would mean that the robot wouldn't be able to accompany the ball in these situations. This is of great significance for the dribbling problem, since it could mean that the restrictions placed on the motion of the robot would not be satisfied. Therefore, even before considering any controls that act upon the ball, it is important to identify the controls that are necessary to keep the dribbling restrictions valid at all times. These controls may not be altered or distorted in any way, since that would likely result in the loss of the ball. Using (4.44) with $\ddot{\theta}_r = 0$ (the accelerations that are related to the application of the required force on the ball are not yet considered), these necessary controls are given by:

$$\boldsymbol{a}_N = R^{-1}\left(E^{-1}\begin{bmatrix}0\\ -\boldsymbol{a}_{rF}\end{bmatrix} - \dot{R}(\dot{\boldsymbol{q}}_r^R - \dot{\boldsymbol{q}}_b^R)\right) + \ddot{\boldsymbol{q}}_b^R + \boldsymbol{a}_S = \boldsymbol{a}_{PN} + \boldsymbol{a}_S \tag{4.53}$$

The respective actuator accelerations $\ddot{\varphi}_i^N$ that are required by $\boldsymbol{a}_N$ are obtained through (4.17). If, as before, a maximum admissible acceleration is given by $\ddot{\varphi}_{MAX}$, then if any of the actuator accelerations obtained in this manner exceeds $\ddot{\varphi}_{MAX}$, the robot will not be able to satisfy the dribbling restrictions. In order to avoid these situations, a maximum admissible velocity for the ball while dribbling must be imposed, and may be determined experimentally. In addition to the controls defined by (4.53), the desired position and force controls are specified:

$$\boldsymbol{a}_D = R^{-1}E^{-1}\begin{bmatrix}(1/\alpha\beta)\ddot{\theta}_r\\ 0\end{bmatrix} + \boldsymbol{a}_F = \boldsymbol{a}_{PD} + \boldsymbol{a}_F \tag{4.54}$$

It can be easily verified that $\boldsymbol{a}_{PN} + \boldsymbol{a}_{PD} = \boldsymbol{a}_P$. As before, let $\ddot{\varphi}_i^D$ denote the actuator accelerations imposed by $\boldsymbol{a}_D$. Following now the same reasoning as in the unconstrained motion case, if $\ddot{\varphi}_i^D + \ddot{\varphi}_i^N > \alpha_{MAX}^a$ for any actuator, let K_τ be defined as:

$$K_\tau = \frac{\min\limits_{i=1,2,3}(\alpha_{MAX}^a - |\ddot{\varphi}_i^N|)}{\max\limits_{i=1,2,3}|\ddot{\varphi}_i^D|} \tag{4.55}$$

The numerator of (4.55) represents the available control effort after applying $\ddot{\varphi}_i^N$ to the actuators. Then, the resulting actuator accelerations are given by:

$$\ddot{\varphi}_i = \ddot{\varphi}_i^N + K_\tau\ddot{\varphi}_i^D \tag{4.56}$$

5 Obstacle Avoidance

In addition to motion control, another component of a mobile robot's navigation that is fundamental to the successful completion of the robot's tasks, relates to its *motion planning* competences. These competences encompass the *path planning* and *obstacle avoidance* capabilities of the robot. During path planning, also known as *global* motion planning, the mobile robot is assumed to have some degree of knowledge about its environment, which was either gathered by the robot itself or supplied beforehand, and which describes the location of certain obstacles. The objective is then to supply a safe path that will drive the robot to its goal. In contrast, the main idea behind obstacle avoidance is that the robot should make use of the available sensory information in real-time to make local decisions with respect to navigation, without any *a priori* information about its surroundings, in what is also known as *local* motion planning. In other words, the robot should alter its controls so that a safe detour is taken to avoid a detected object, whilst converging towards a goal position or trajectory supplied by the global planning modules. In most situations, it is essential for a mobile robot to possess both *global* and *local* motion planning capabilities in order to consistently reach its destinations in a safe manner. In some environments however, one of these competences can outweigh the other. Such is the case of the robotic soccer environment. Except for the goals, which should not hinder the movement of the soccer robots, there are no other permanent obstacles present in this environment. In this situation, global motion planning reduces to the problem of obtaining a proper control law that takes a robot to its destination, such as the ones described in the previous chapters, and should be computationally inexpensive. Real-time obstacle avoidance, on the other hand, is of the utmost importance, since the speed at which the robots move could even result in damage to the robot's hardware in the event of a collision.

This chapter describes an obstacle avoidance solution that can be combined with any of the motion control solutions presented in the previous chapters, to provide safety to the mobile robot during the execution of its tasks.

5.1 Related Work

The most simple obstacle avoidance algorithms employ what is known as *boundary-following*, such as the Bug algorithms [39]. The robot is expected to travel around a detected obstacle, following its edges until some leaving condition is met, at which point the robot departs from the obstacle and continues travelling to its goal. Traditionally, these methods produce inefficient paths to avoid collision, and make use of only the most recent sensor information, which can be influenced by noise or misreading. However, in many simple environments they can produce acceptable results. Additionally, they have one important property: if the goal is reachable from the robot's position, in the sense that there is at least one safe path leading to it in the environment, then the robot is always able to reach the goal after some time, regardless of its initial conditions. Such an algorithm is said to be *complete.* More elaborate algorithms often sacrifice completeness for a reduction in their computational cost.

Many improvements and variations of the Bug algorithms exist in the literature [4]. One of these, the TangentBug, described in [40], is especially relevant for the current work.

TangentBug follows the basic guidelines of the earliest Bug algorithms, which considered mainly the use of contact sensors, and extends them for systems equipped with range-sensors. It possesses two distinct navigation strategies: *motion to target* and *obstacle boundary following*. In the former mode, the robot is guided to its target normally and is not hampered by any obstacle. In the event that an obstacle is detected that prevents the robot from moving directly to its target, the TangentBug algorithm then tries to follow the boundaries of the obstacle in question. In doing so it relies on the computation of a *local tangent graph* (LTG) at each control cycle, to select the optimal direction of movement. The LTG is a subset of the *visibility graph* (see the next section for details), and it contains the shortest possible path around the detected obstacles. In order to build the LTG, the algorithm makes use of any discontinuities detected in the range sensor data. It is shown in [40] that, as the range of the sensors increases, the paths followed in this manner approach the global shortest path to the goal. This method is also complete, as its predecessors.

Another common approach to obstacle avoidance is the artificial potential fields method introduced by Khatib in [41]. In this approach, each obstacle is thought of as if it was able to exert a repulsive “force” upon the robot depending on its measured distance, and at the same time the goal position generates an attractive force that drives the robot to its target. The sum of both these repulsive and attractive force components is then used as a control for the mobile robot. Most algorithms based on this approach are subject to problems with local minima that may occur when the repulsive and attractive forces cancel each other, resulting that the robot may become “trapped” in such a situation and is thus unable to reach its goal. Another problem occurs whenever the obstacles form a narrow corridor that the robot has to move through, since the opposing repulsive forces from these obstacles drive the robot along a “valley” in the so-obtained potential field, which typically results in oscillatory motion. Numerous improvements have been made to the original method (for example [42],[43]), in order to overcome some of these limitations. Previous to the present work, the obstacle avoidance algorithm that was used by the ISocRob team consisted of a modified potential fields method that accounted for dribbling restrictions for differential-drive robots [2]. Although it presented acceptable results, the method suffered from local minima problems, and did not consider the effects of actuator saturation on the dynamical behavior of the robot. The latter problem should be properly noted: being a purely abstract concept, the artificial potential fields method bears little relation to the dynamic properties of the mobile robot. As a consequence, important control properties such as the maximum deceleration and turning capabilities of the robot, are left unconsidered (the aforementioned oscillation problem is a consequence of this fact). Minguez et al. later introduced the concept of *ego-dynamic* space, an extension to the usual *workspace* and *configuration space* representations, that takes some of the robot’s dynamic properties into account and allows the implementation of reactive algorithms that, by themselves, lack this capability, such as the potential fields method.

The Virtual Force Field (VFF) [44] method and its improved version, the Vector Field Histogram (VFH) [45] method share some of the basic concepts of artificial potential fields, but account explicitly for sensor noise and misreadings. In these approaches, a discrete grid, the

histogram grid, is used to hold the sensor readings, which over time results in an approximation to the probability distribution describing the location of the obstacles. VFH also creates a *polar histogram* that divides the space around the robot into a fixed number of angular sectors, each with a certain density value associated to the presence of obstacles. The sectors are then searched for the safest direction of motion. In their work, Borenstein et al. also considered the effects of the robot's dynamics on its overall behavior. However, by analyzing only a discrete set of possible directions, some possible safe paths are lost, and the robot may become trapped. Minguez et al. proposed the use of Nearness Diagrams [46], which are similar to the polar histograms in VFH. The method tries to overcome the limitations of VFH through an elaborate set of motion control laws that bear little consideration for the dynamics of the robot. A derived method was applied to the ISocRob team in [1]. It was specifically designed for use with differential-drive robots. It also displayed oscillation problems and performed integrated motion control. As with VFH itself, it is well suited for the use of ultrasonic sensors, but loses efficiency when applied to robots equipped with camera-based detection, such as the current OmniISocRob platform. Since obstacle detection is purely image-based, there is no simple way to obtain the constructs (like the Nearness Diagrams or polar histograms) necessary for this type of algorithms.

A more analytical approach to obstacle avoidance was presented in the Dynamic Window Approach [47] and the Curvature Velocity Method [48], which have demonstrated good results on robots moving at high speeds. These methods operate by searching the space of admissible velocities for the control inputs that maximize a given objective function. Admissible velocities are, in this sense, those that will not cause collisions with any obstacles and are within the limits of the robot's actuators. To decide whether or not a given velocity will cause a collision, the robot's kinematic and dynamic models are used to project a trajectory into the configuration space (for example, differential-drive robots can only move along arcs of circles). If these trajectories intersect obstacles, or the desired safety boundaries around them, at a distance large enough that allows for the robot to come to a full stop, it is considered safe. Although it was originally formulated for synchro-drive robots, in [49] Brock and Khatib generalized the original dynamic window approach to holonomic robots. Despite having important qualities, the dynamic window approach requires the update of the distances to all detected obstacles for all trajectories arising from the admissible velocities inside the dynamic window, as well as the subsequent maximization of the objective function. This results on a large computational overhead that, for the robots utilized in the experiments of these authors, was admissible. However, in the robotic soccer environment, especially for the current robotic platform in use by the ISocRob team, this poses a serious limitation, since there are many other subsystems running simultaneously (in this case, as part of the MeRMaID architecture) that demand a large amount of computational resources. To reduce the computational load, the velocity space would have to be discretized into larger intervals, resulting in poor dynamic response, which would defeat the purpose of the algorithm.

5.2 Overview of the Solution

The proposed solution to obstacle avoidance is intended as a replacement for the modified potential fields approach described in [2], which was designed for differential-drive robots and has since lost its validity. The method should comply with the following requirements:

- Computational efficiency – the total cycle time for the algorithm should be comparable to that of the previous method, or preferably shorter;
- Should solve both the local minima and the narrow corridor problems;
- Should be prepared to handle control inputs both in velocity and acceleration form;
- Should specifically address sparse, simple environments;
- Should explicitly account for the presence of moving obstacles with measurable velocities;
- Should be modular, in that it must be applied in tandem with the different control strategies used during a robotic soccer match, including those which are presented in this work, and provide efficient obstacle avoidance for each of them.

The development of the proposed method was at first based on the concepts introduced by Dynamic Window Approach, the Steer-Angle field method [50], and the Curvature-Velocity method, as an extension that would allow obstacle avoidance for moving obstacles. However, it was verified that, given the overall simplicity of the environment in question, the resulting behavior could be approached, for omnidirectional robots, by the simple TangentBug algorithm, which would simultaneously satisfy most of the above requirements, at a much smaller computational cost. However, the algorithm itself is not completely reactive, in that it assumes that the obstacles present in the environment can be tracked, and modelled, in successive intervals. This is undesirable for environments populated with moving obstacles. Furthermore, no methodology has been specified to build the aforementioned local tangent graph for obstacles detected from images taken by an onboard camera (i.e. considering that only the position of the obstacles is known, and no information is given about its shape, unlike what happens with range-sensors).

Therefore, the proposed algorithm draws from the main advantages and concepts of the TangentBug algorithm, but has the following core differences:

- It introduces a systematic process of obtaining locally near-optimal directions for movement based on optically-detected obstacles, that eliminates the need for explicit boundary-following modes and obstacle tracking;
- The application of the same principles to deal with moving obstacles;
- It can be applied to the tasks that involve transporting an object, such as the dribbling methods described in Section 4.3, by applying these concepts to the object's controller.

The proposed method works by calculating, at each step, the necessary deviation that must be applied to the current (linear) velocity of the robot so that the direction of this velocity is tangent to

the boundaries of any obstacle that is currently blocking the robot's path. This is similar to obtaining the most useful edges of the local tangent graph in an iterative way. By taking this approach, the control inputs of the robot are modified only in what is strictly necessary to achieve safety, and thus the interference caused to the different control strategies used throughout navigation is kept to a minimum.

The following sections review the necessary concepts and fully describe the proposed solution. Section 5.3 details the operation of the method when dealing with static obstacles, and Section 5.4 then extends the method to situations with moving obstacles.

5.3 Avoiding Static Obstacles

The obstacle detection algorithms currently in use by the ISocRob team rely on the processing of images captured by an omnidirectional camera, and provide information about the position of each obstacle relative to the robot. Current plans are to also include information about each obstacle's velocity. No information is given, however, about the shape of said obstacles. In fact, these algorithms are currently only prepared to detect either the ball or another robot player as an obstacle, based on their shape and color, and will result in misreadings when confronted with objects that do not possess these characteristics, such as walls. Considering these limitations, and for the purposes of a robotic soccer match, it is acceptable to model the expected obstacles as circles in the plane containing the field of play (the robot's workspace). The expected number of obstacles is also limited. The rules in place for the Middle Size Robot League during the development of this work limit the number of players to six robots in each team. Including the ball and assuming that there are no false detections, this implies that there is a maximum of twelve detected obstacles at any moment during a match. In reality, given the dimensions of the field and the fact that obstacle detection fails above a certain range, the average number of detected obstacles is smaller. The resulting expected environment is sparse, as stressed previously.

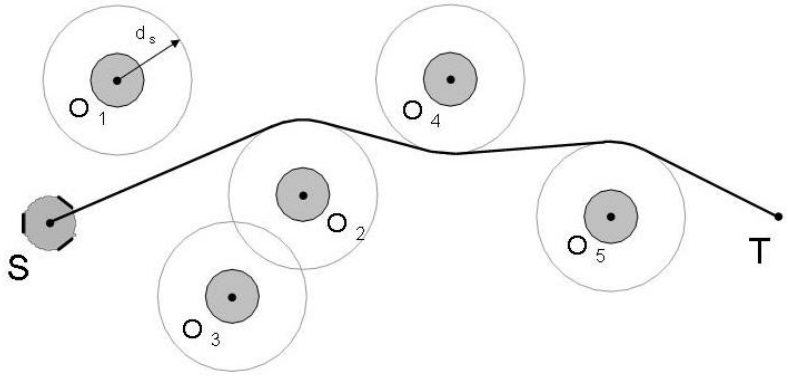


Figure 21: Typical obstacle configuration and shortest path to target.

A typical situation where obstacle avoidance is required is depicted in Figure 21. The shortest possible path between the robot's current position, S, and its goal position, T, is also shown. Ideally, the obstacle avoidance algorithm will drive the robot along this path.

At this point it is necessary to review the concepts of *visibility graph* and *tangent graph.* For an environment populated with polygonal obstacles, the visibility graph, $VG(N_V, E_V)$ is the undirected graph in which the nodes N_V represent the obstacles vertices and the starting and ending points S and T, and the edges E_V connect all nodes that are "visible" from each other, in that there is a line segment uniting them that does not intersect any other obstacle in configuration space. An important property of the visibility graph is that it always contains the shortest collision-free path to the target ([40]). However, this path will take the robot as close as possible to the obstacles it must avoid. It can be argued that due to sensor noise, the overall safety is reduced by choosing such a path. While this holds true for many robot applications, in robotic soccer it is often necessary to maneuver close to the other robotic players, and keeping a maximal distance to these obstacles isn't always the best approach. Therefore, in order to account for the presence of noise, the obstacles are inflated in configuration space by including a safety margin that the robot must respect. If this safety margin is set too large, some solutions may be overlooked, and so one has to take into consideration the minimum distance that two adjacent obstacles should have from each other, so that the robot is still allowed to pass through them. During dribbling operations, the obstacle avoidance algorithm will act upon the desired motion of the ball, instead of the robot, and so the safety margin has to be set in such a way that the robot will not collide with any obstacles while rotating around the ball (the safety margin has to account for the radius of the ball and the diameter of the robot). Other than this detail, the obstacle avoidance algorithms presented in this chapter can be directly applied to the problem of transporting a moveable object, and so, the focus will be put, without loss of generality, on the situations where the motion of the robot is unrestricted.

The tangent graph $TG(N_T, E_T)$ is a subgraph of the visibility graph which contains the convex vertices of the obstacles, the edges that are bi-tangent to them, and those that can be united to S and T without intersecting any obstacles. In the case of curved obstacles, the boundaries of the obstacles can be partitioned into convex and non-convex arcs. The edges E_T of the tangent graph are then the convex arcs of those boundaries, and the line segments that are bi-tangent to them. The nodes N_T correspond to the intersection points of these line segments with the boundaries of each obstacle, and the points S and T. Figure 22 shows the tangent graph for the situation represented in Figure 21. It is also shown in [51] that the tangent graph is the minimal subgraph of the visibility graph that is still guaranteed to contain the shortest path to the goal. By removing any of its edges, this property would be invalidated.

In turn, the local tangent graph, or LTG, is the subgraph of the tangent graph that can be constructed given the incomplete information available to the robot. It is assumed that the robot is able to detect obstacles in a certain radius around itself, but it cannot possess partial information about an obstacle, i.e., the obstacle-detection mechanism can be modeled as a binary-valued function. Recall that each obstacle is modeled as a circle in the robot's workspace. In these conditions, and in light of the previous definitions, if the LTG is non-empty, then at least one of its edges belongs to the shortest

path to the goal. The core of the algorithm then lies on selecting the local optimum (greedy) direction from the edges of the LTG. The directions selected in this way may differ from the global optimum directions. It is well known that such an algorithm is only guaranteed to converge if the problem has an *optimal* substructure, i.e. if the global optimum solution can be obtained by making local optimum decisions at each step.

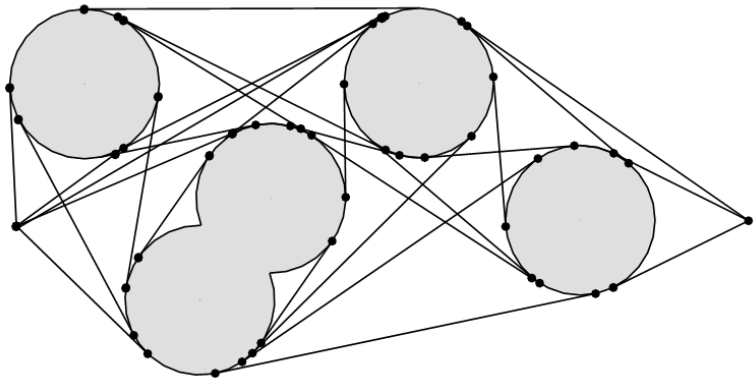


Figure 22: Tangent graph for the situation represented in Figure 21.

From the definition of the LTG, it is evident that in any situation, there are at most two edges in the graph that connect the robot's position S to nodes in the boundaries of each obstacle (since there are two lines passing through any given point that are tangent to a convex region). Let these nodes be defined as the *endpoints* $P_{1,k}, P_{2,k}$ of each obstacle O_k. Also, let the expected distance between nodes m and n of the LTG, $L_{sp}(n,m)$, be defined as the total length of the shortest path on the LTG uniting n and m (the *local* shortest path between n and m). The following result then arises:

Proposition 5.1: *If there is an endpoint* $\boldsymbol{P_{opt}}$ *on the LTG such that* $\boldsymbol{L_{sp}(S,P_{opt}) + L_{sp}(P_{opt},T) \leq L_{sp}(S,P) + L_{sp}(P,T)}$ *for every endpoint* $\boldsymbol{P}$*, then a local optimal direction for movement corresponds to the edge* $\boldsymbol{e_{opt}}$ *that connects the current position* $\boldsymbol{S}$ *to* $\boldsymbol{P_{opt}}$*.*

Proof: Note that $L_{sp}(S,P)$ is the length of the single edge that connects S and P. The proof follows from the fact that $L_{sp}(S,P_{opt}) + L_{sp}(P_{opt},T)$ is the total length of the shortest path between S and T that passes through P_{opt}. For an endpoint $P \neq P_{opt}$ that verifies $L_{sp}(S,P) + L_{sp}(P,T) > L_{sp}(S,P_{opt}) + L_{sp}(P_{opt},T)$, then, by definition, none of the edges that connect to P belong to the local shortest path from S to T. Therefore, the edge e that connects points S and P is a sub-optimal direction for movement. ■

Although the endpoints of each obstacle can be easily obtained from the assumption that the detected obstacles are circles, the calculation of $L_{sp}(P,T)$ requires building and searching the complete LTG at every iteration, for every endpoint P. Since this is computationally prohibitive, the problem must be simplified further.

Suppose that at any given instant, the line segments that connect the endpoints of each obstacle to the goal point do not intersect the boundaries of any other object. In this case, if the direct path to the goal is blocked by an obstacle, the only candidates for optimal directions are the edges that connect to the endpoints of this obstacle. This is easily verified geometrically, since any obstacle that is not blocking the robot in this manner is necessarily part of a longer path to the target. It will be seen that, by making this assumption, the robot can be mislead in certain situations, making the algorithm *incomplete.* The TangentBug algorithm deals with this situation by forcing the robot to follow the boundaries of a certain obstacle, which in turn requires a non-reactive component in the algorithm, in which the objects are modeled as walls and tracked through consecutive iterations, maintaining information about the connectivity between them. In this case however, the algorithm should be kept completely reactive, not only for computational issues, but also because most of the boundary-following concepts presented by TangentBug do not hold true for moving obstacles. By following the above assumption, however, the problem of selecting the local optimal direction reduces to obtaining the two endpoints of the obstacle (or obstacles) directly in front of the robot in configuration space. A similar implementation is described in [52].

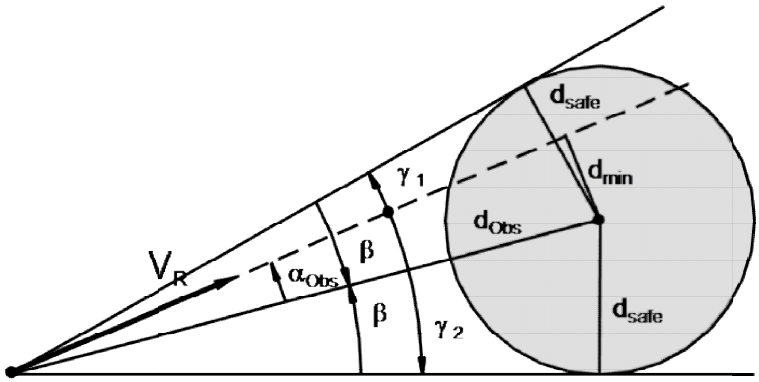


Figure 23: Relevant geometric details for obstacle avoidance.

Consider the situation depicted in Figure 23. The robot heads towards its goal T with velocity v_R, which is assumed constant between control cycles. The (Euclidean) distance between the robot and its goal is d_{Goal}. An obstacle is detected at distance d_{Obs}, and at a certain angle α_{Obs} relative to the velocity vector. The radius of the obstacle is d_{Safe}. In these conditions, the minimum distance between a line aligned with the velocity vector and the centre of the obstacle is given by:

$$d_{min} = d_{Obs}\sin(\alpha_{Obs}) \tag{5.1}$$

Consequently, if $d_{min} > d_{Safe}$ and $d_{Obs} \leq d_{Goal}$, the robot will not collide with the obstacle if its velocity remains constant. This constitutes a *safety condition* that must be verified for each detected obstacle. The minimum angle β that satisfies $d_{min} \geq d_{Safe}$ is then:

$$\beta = \begin{cases} \text{asin}\left(\frac{d_{Safe}}{d_{Obs}}\right) & if\ d_{Obs} \geq d_{Safe} \\ \frac{\pi}{2} & otherwise \end{cases} \tag{5.2}$$

The definition of $\beta = \frac{\pi}{2}$ happens whenever the robot is already *inside* the radius of the obstacle in configuration space, which may happen in the dimensions of the obstacle are overestimated. By following this direction, the robot will eventually reach the boundaries of the obstacle in configuration space. Therefore, whenever a collision is bound to happen, the robot must apply a certain *detour angle* γ to its velocity so that the safety condition is satisfied. The minimum detour angle is such that:

$$|\alpha_{Obs} + \gamma| = \beta \tag{5.3}$$

The two possible solutions, one for each endpoint, are then:

$$\gamma_{1,2} = -\alpha_{Obs} \pm \beta \tag{5.4}$$

The minimal solution is,

$$\gamma_{min} = \gamma_m\,, \qquad m = \underset{i \in \{1,2\}}{\arg\min}\ |\gamma_i| \tag{5.5}$$

Note that, for the single-obstacle case, this is in fact equivalent to calculating the endpoint P_{opt} with the minimal expected distance to the target and sending the robot along the corresponding edge of the LTG. In more complex situations, however, the endpoints must be explicitly calculated. Suppose that $\alpha_R = atan2(v_{R_y}, v_{R_x})$ is the angle of the velocity vector in the robot's frame. The endpoints $P_{(1,2),k}$ of the obstacle O_k in this frame are then:

$$P_{i,k} = [d_{Obs}\cos\beta\ \cos(\gamma_i + \alpha_{Obs}) \quad d_{Obs}\cos\beta\ \sin(\gamma_i + \alpha_{Obs})]^T \tag{5.6}$$

The expected path length $L_{sp}(P,T)$ between these endpoints and the goal can then be approximated as the length of the line segment between P and T,

$$L_{sp}(P,T) = \|P - T\| \tag{5.7}$$

The above results would provide optimal paths for obstacle avoidance in environments with a single obstacle and assuming that the robot is under velocity control. For environments with multiple obstacles, situations may arise where obstacles are close enough so that they are effectively "merged" in configuration space. This occurs whenever two obstacles are separated by a distance inferior to $2d_{Safe}$. These obstacles are, however, still considered as two separate obstacles by the robot. In these conditions, some configurations of obstacles would lead the algorithm into oscillatory motion,

namely whenever two or more endpoints have the same expected path length. To account for this, the value of $L_{sp}(P,T)$ is weighted with the detour angle γ. For every obstacle O_k, the *cost* of one of its endpoints $P_{i,k}$ is then defined as:

$$C(P_{i,k}) = c_1 \frac{L_{sp}(P_{i,k},T)}{\max\{L_{sp}(P_{1,k},T), L_{sp}(P_{2,k},T)\}} + c_2 \frac{|\gamma_i|}{\pi} \tag{5.8}$$

with $c_1, c_2 > 0$ and $c_1 + c_2 = 1$. It is evident that $C(P_{i,k})$ returns values in the $[0,1]$ interval, and represents the robot's "preference" for the paths that demand less turning effort.

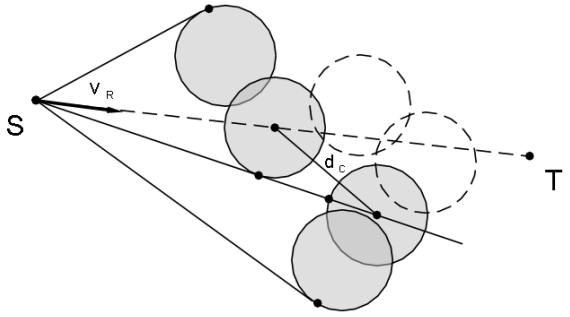


Figure 24: Obstacle avoidance for multiple obstacles and the presence of "shadowed" obstacles.

For an environment populated with multiple isolated obstacles or clusters of obstacles (i.e. for more than one detected obstacle), the safety condition must be reformulated. Consider the situation presented in Figure 24. A cluster of obstacles is preventing the robot from reaching the goal directly. However, the direction taken through the optimal endpoint (the endpoint that minimizes (5.8)) intersects a different cluster of obstacles placed farther away from the robot. In this situation, the direction remains safe if the distance between these two clusters, d_c, taken from the centers of the outermost obstacles, is greater than $2d_{Safe}$, which is to assume that these clusters are not connected in configuration space and that there is enough space between them for the robot to pass. This, in turn, may lead to a problem of "shadowing", where obstacles that are not visible from S may exist that actually connect the two clusters in configuration space, invalidating the original decision, since the direction taken by the robot does not belong to the LTG. Without expressing the connectivity between the obstacles in the configuration space, this situation is inevitable. However, the robot will be able to recover from this situation once the shadowed obstacles become visible.

The general procedure to detect the endpoints for an obstacle or a cluster of obstacles is represented in Figure 25. The following steps are performed:

1) Identify the original obstacle that violates the safety condition for the current velocity;
2) The two possible detour angles for that obstacle are calculated according to (5.4). This always results in one positive solution and one negative solution. These solutions are registered in set $\Gamma = [\gamma_1 \quad \gamma_2]$. The respective endpoints are calculated according to (5.6).

3) For the endpoint that minimizes $C(P_{i,k})$, the safety condition is re-checked:
    - If the solution is safe, update the robot's velocity and return;
    - Otherwise, the robot is in the presence of an obstacle cluster. Select the individual obstacle that currently violates the safety condition and continue to step 4;
4) Using equation (5.4), once again, two solutions are generated. One of these solutions is invalid, since it would necessarily violate the safety condition for the previously selected obstacle. Update Γ by overwriting the element with the same algebraic sign as the detour angle of the valid solution. Update the respective endpoint and re-evaluate the minimal cost solution.
    - If the optimal endpoint implies $|\gamma_i| > \pi$, the goal is unreachable. Return.
    - Otherwise, return to step 3.

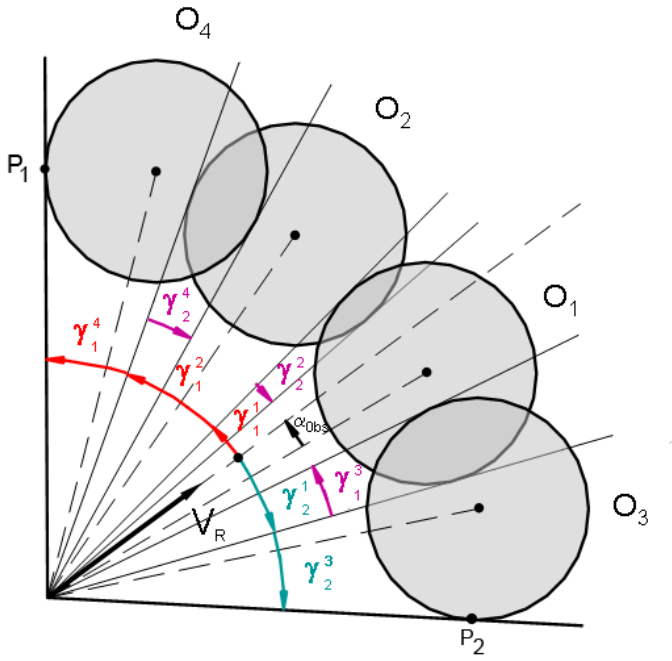


Figure 25: Endpoint calculation for an obstacle cluster. The two possible detour angles for each obstacle O_i are shown as $\gamma_{1,2}^{i}$. Note that for every obstacle except O_1 there is only one valid solution, which is either a positive (red) or negative (blue) detour.

By performing the above steps, the resulting solution will be the endpoint of the obstacle cluster that minimizes the cost function (5.8). By adjusting the robot's velocity at every iteration, obstacle avoidance will be attained for environments with static obstacles.

5.4 Avoiding Moving Obstacles

All considerations for the static obstacles case were made taking into account that the instantaneous velocity of the robot needed to be altered by a certain *detour angle* in order to avoid collisions. These concepts are easily adaptable for an environment with moving obstacles, by

addressing instead the relative velocity between these obstacles and the robot. If the obstacles possess a certain velocity v_O, which is assumed constant throughout each cycle of the obstacle avoidance algorithm, it is necessary to obtain the detour angle that must be applied to the *relative* velocity, so that the robot avoids any incoming obstacles in its frame. The remainder of this section will concern the situation where a single moving obstacle must be avoided. The extension for multiple obstacles is trivial, since it follows the same procedure as for the static obstacles case.

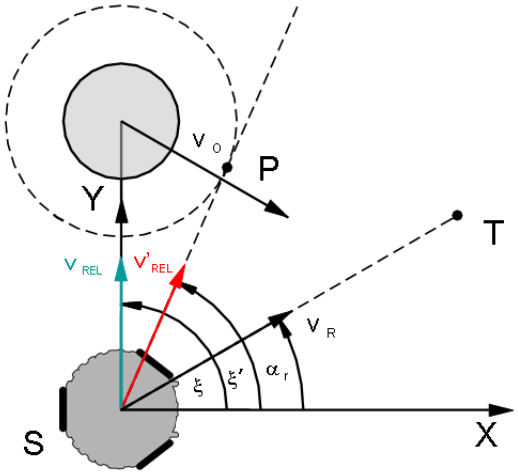


Figure 26: Avoiding a moving obstacle.

Let the robot velocity and the velocity of a moving obstacle be defined, in the robot frame, as $v_R = \left[v_{R_x}\ v_{R_y}\right]^T$ and $v_O = \left[v_{O_x}\ v_{O_y}\right]^T$ respectively. The relative velocity is then:

$$v_{REL} = [v_{R_x} - v_{O_x} \quad v_{R_y} - v_{O_y}]^T = \left[v_{REL_x}\ v_{REL_y}\right]^T \tag{5.9}$$

Also, let ξ be the angle between the robot frame's *x*-axis and the relative velocity vector, as depicted in Figure 26. The objective is then to define a direction for the robot's velocity so that the updated relative velocity satisfies the necessary conditions to avoid collision, namely that $d_{min} > d_{Safe}$ if $d_{Obs} \leq d_{Goal}$ (refer to last section for details). The updated relative velocity v'_{REL} then possesses an angle ξ, such that:

$$\xi' = \xi + \gamma = \text{atan2}\left(v'_{REL_y}, v'_{REL_x}\right) \tag{5.10}$$

and γ is obtained in the same manner as in the previous section. Note that in the moving obstacles case, the expected path length from the moving obstacle's endpoints is totally unpredictable and can lead to wrong results, so the optimality criterion becomes the necessary deviation from the original direction at each control cycle. It is imperative to note that one acts only indirectly upon the relative velocity. The robot requires an updated velocity v'_R so that v'_{REL} is achieved in turn. The following properties must be verified for v'_R:

$$\text{atan2}\left(v'_{R_y} - v_{O_y}, v'_{R_x} - v_{O_x}\right) = \xi' \tag{5.11}$$

$$\|v'_R\| = \|v_R\| = V_R \tag{5.12}$$

Again, referring to Figure 26, let α_R be the angle of v_R in the robot's frame, and V_R its norm, so that $v_{R_x} = V_R\cos(\alpha_R)$ and $v_{R_y} = V_R\sin(\alpha_R)$. Equation (5.11) can then be rewritten as:

$$\text{atan2}\left(V_R\sin\left(\alpha'_R\right) - v_{O_y}, V_R\cos(\alpha'_R) - v_{O_x}\right) = \xi' \tag{5.13}$$

The problem is now to obtain α'_R so that (5.13) is verified.

Proposition 5.2: *At each control cycle, the necessary angle* $\boldsymbol{\alpha'_R}$ *that must be applied to the robot's velocity* $\boldsymbol{v'_R}$*, so that an incoming obstacle is avoided in the conditions of Figure 26, is given by one of two possible solutions:*

$$\alpha'_R = \text{asin}\left(\frac{-\tan(\xi')\,v_{O_x} + v_{O_y}}{V_R\sqrt{1+\tan^2(\xi')}}\right) + \xi' \quad \vee \quad \alpha'_R = -\,\text{asin}\left(\frac{-\tan(\xi')\,v_{O_x} + v_{O_y}}{V_R\sqrt{1+\tan^2(\xi')}}\right) + \xi' \tag{5.14}$$

Proof: Equation (5.13) can be used carefully by noting that if the normal 2-quadrant tangent is used instead of the 4-quadrant tangent, then two possible solutions for α'_R exist. These are the solutions to the following relation:

$$\frac{V_R\sin(\alpha'_R) - v_{O_y}}{V_R\cos(\alpha'_R) - v_{O_x}} = \tan(\xi') \tag{5.15}$$

From this, and after some manipulation,

$$\sin(\alpha'_R) - \tan(\xi')\cos(\alpha'_R) = \frac{-\tan(\xi')\,v_{O_x} + v_{O_y}}{V_R} \tag{5.16}$$

Using now the following trigonometric relation,

$$a\sin(\theta) - b\cos(\theta) = \sqrt{a^2+b^2}\sin\left(\theta - \text{atan}\left(\frac{b}{a}\right)\right) \tag{5.17}$$

Equation (5.16) can then be seen to be equivalent to,

$$\sqrt{1+\tan^2(\xi')}\sin(\alpha'_R - \xi') = \frac{-\tan(\xi')\,v_{O_x} + v_{O_y}}{V_R} \tag{5.18}$$

Solving for α'_R, it results that:

$$\alpha'_R = \pm \text{asin}\left(\frac{-tan(\xi')v_{O_x} + v_{O_y}}{V_R\sqrt{1 + tan^2(\xi')}}\right) + \xi' \tag{5.19}$$

▪

5.5 Obstacle Avoidance for Dribbling and Interception

The methods for obstacle avoidance presented so far can be directly applied to tasks that involve unrestricted velocity control of an omnidirectional robot (such as posture stabilization). For situations that require torque control of the mobile robot, the proposed obstacle avoidance algorithm can still be applied, albeit with some modifications. For these cases, the robot's acceleration is piecewise constant, but the robot's trajectory is not necessarily linear between two control cycles (though it can be approximated by an arc of a circle). However, if the robot's linear acceleration maintains a constant direction over time (in the world frame), its path eventually converges towards a linear form in finite time, due to the saturation limits of its actuators. It is therefore reasonable to require that, at each control cycle, the velocity of the robot should be considered safe according to the definitions of the previous sections. This, in turn, can be assumed as a sufficient condition for safety if the time step used during navigation is small enough.

To achieve this, the linear velocity of the robot on the next control cycle is predicted given its current torque inputs $\boldsymbol{a}_n$, and the acceleration that is required to make this velocity safe, $\boldsymbol{a}'_n$, is then obtained:

$$\boldsymbol{v}_{R_{n+1}} = \boldsymbol{v}_{R_n} + \boldsymbol{a}_n T \tag{5.20}$$

$$\boldsymbol{a}'_n = \frac{1}{T}\left(\boldsymbol{v}_{R_{n+1}}' - \boldsymbol{v}_{R_n}\right) \tag{5.21}$$

where the direction of $\boldsymbol{v}_{R_{n+1}}'$ is considered "safe". Note that, the so-obtained acceleration command may exceed the dynamic capabilities of the robot. This is dealt with on a task-specific basis, such as the methods presented in 3.4.

This form of obstacle avoidance in torque-control form may then be applied to the tasks of object transport and interception. During object transport, it is sufficient to apply these concepts to the required force $\boldsymbol{f}$ supplied by the object controller (see Section 4.3.2) and to redefine the radius of the modelled obstacles in configuration space. For moving object interception, since both IPNG and trajectory tracking supply controls in acceleration form, it is also possible to use the proposed obstacle avoidance algorithms to avoid the ball until the robot satisfies the physical restrictions inherent to that task, so as to prevent the ball from escaping at the moment of collision. If other dynamic obstacles are present in its environment, the robot will also be able to avoid them whilst converging towards the desired interception trajectory.

6 Experimental Setup

In this section, the robotic platform that was used to test the motion control solutions presented in this work is briefly described with respect to its relevant characteristics for motion control, as well as the environment in which the tests were performed.

6.1 The OmniISocRob Platform

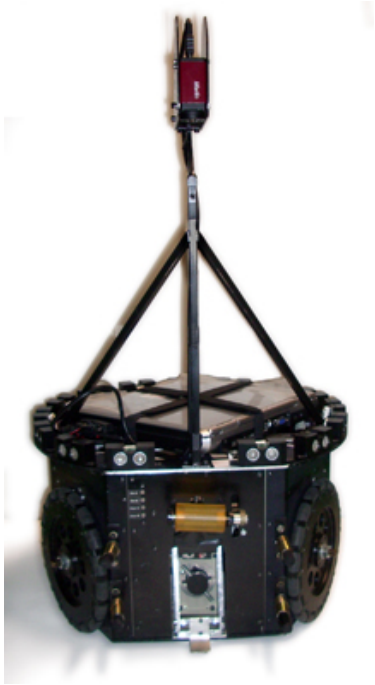


Figure 27: The robotic soccer platform currently used by the ISocRob team.

The ISocRob team is currently comprised of five omnidirectional robots, each with the following main characteristics that are relevant for motion control:

Physical dimensions:

- Each robot is equipped with three Swedish wheels. The radius of each wheel is 10 cm. A set of rollers are placed around the borders of each wheel, each with an axis of rotation orthogonal to that of the wheel;
- The robot's chassis has an hexagonal footprint, such that distance from the geometric center of the chassis and each wheel is 20 cm;
- On one of its sides, the robot possesses two pairs of passive, flexible flippers (or fingers) which are intended to keep the ball from escaping during dribbling (see Figure 28). The horizontal distance between the two flipper pairs is 22 cm;
- When fully loaded (as during a robotic soccer match), each robot weighs approximately 22.8 Kg;

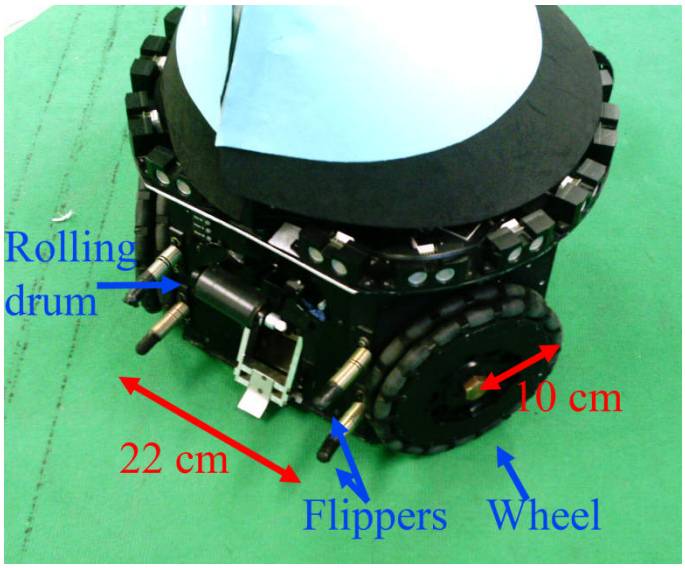


Figure 28: Relevant components and dimensions of the OmniSocRob platform

Main hardware components:

- A NEC Versa FS900 laptop, where most of the processing takes place. Each of these laptops has an Intel Centrino 1.6Ghz processor, and 512Mb of RAM;
- Each of the robot's wheels is actuated by a MAXON DC motor RE35/118776. This motor achieves a maximum angular velocity of $\omega_{MAX}^{a} \cong 30\ \mathrm{rad/s}$. A soft-limit for the maximum angular acceleration of these motors was set at $\alpha_{MAX}^{a} = 22\ \mathrm{rad/s^2}$. Between each motor and its respective wheel, a MAXON gear 203118 is placed, with a reduction of 21:1.
- The control of the motors is done through Master-Slave PIC controllers. The communication between the laptop and the Master PIC is done at (approximately) 30Hz;
- A 500 CPR encoder is present in each wheel;
- A Marlin AVT F033C firewire camera is placed, facing downward, on top of the robot (see Figure 27). The dioptric vision system utilizes a fish-eye lens, and allows for omnidirectional vision;
- An AnalogDevices XRS300EB rate-gyro.

Relevant software features:

- The self-localization of the robot is accomplished through a Monte-Carlo Localization algorithm, described in [53]. Although the robot is able to localize itself anywhere on the field of play, the algorithm itself continuously tries to correct any self-localization errors. For this reason, consecutive measures of the posture of the robot taken by this algorithm present, in a typical situation, a standard deviation of $\sigma_{Pos} \cong 4$ cm in position and $\sigma_{\theta} \cong 0.03$ rad in orientation. This value depends on the specific part of the field that the robot is on, and the velocity of the robot.
- The obstacle detection algorithm relies on the color-based segmentation of the images taken by the robot's omnidirectional camera, and the subsequent search for contiguous, black-colored, areas, which are assumed to be obstacles. To improve computational efficiency, the

image itself is divided into a discrete number of radial areas (in this particular case, 16). While this algorithm isn't particularly reliable, it provides obstacle detection up to 5 m.

- A new ball-detection algorithm is being developed concurrently to this work, which relies on a particle-filter based approach. This algorithm provides reliable detection of the ball at a distance of up to 5 m with the omnidirectional camera, and also provides information about its velocity.

6.2 The Experimental Environment

The motion control solutions presented in the previous chapters were implemented as part of the MeRMaID architecture of the ISocRob team. However, at present, the controllers of the robot's actuators were not yet adapted to provide torque control. Due to this fact, the motion control solutions that require torque control were instead tested inside the Webots simulation environment, where the OmniISocRob platform was modelled according to its physical properties and capabilities described above. The tasks that were tested inside Webots were:

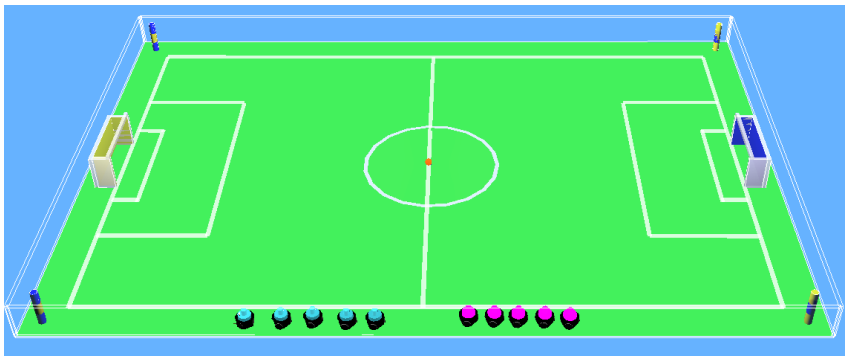
- Tracking a reference trajectory, using the control laws presented in Section 3.2, both with a continuous-time approximation and as a discrete-time control law, and the orientation control law presented in Section 3.3.2;
- The task of intercepting a moving object, described in Section 4.2;
- The task of transporting a moveable object, described in Section 4.3.

Conversely, the following tasks were tested in the real ISocRob robots, in addition to the Webots simulations:

- The task of stabilizing the robot around a reference posture, according to the control laws presented in Section 3.1 and 3.3.1;
- The obstacle avoidance methods presented in Section 5.

The experiments that were performed in the Webots simulation environment were conducted inside a robotic soccer field that complies with the current RoboCup regulations (12m x 18m). Inside the field, a maximum of ten obstacles may be detected by the robot, corresponding to the other robots and the ball. The range of detection of the ball and of the obstacles in the environment was set at 5 m.

The real field used by the ISocRob team is a rectangular (4.5 m x 9m) field shown in Figure 29. This field may be populated with at most four other obstacles.



(a)

(b)

Figure 29: The robotic soccer field.

(a): the soccer field used inside the Webots simulation environment.

(b): the soccer field used by the ISocRob team.

The soccer ball that was used while testing the object interception and transport tasks was a standard, RoboCup compliant, soccer ball, with a radius of 11 cm and a mass of 0.45 Kg. This implies that, while the robot is holding the ball, the ball fits within its flippers with a negligible amount of "slack". The coefficient of rolling friction between the ball and the soccer field was measured at $C_{rr} \approx 0.015$ (this value was also used inside Webots).

Figure 30: A soccer ball compliant with the RoboCup regulations.

7 Results

In this chapter, experimental results are shown, and discussed, for each of the motion control solutions proposed in the previous chapters, in order to ascertain the validity of those solutions.

7.1 Posture Stabilization

In the experiments that were performed to test the posture stabilization process, from the initial conditions presented in Table 1, the robot was tasked with achieving the reference posture $q_g = [0 \quad 0 \quad 0]^T$.

Table 1: Initial conditions for the posture stabilization experiments.

	x (m)	y (m)	θ (rad)	v_x (m/s)	v_y (m/s)	ω (rad/s)
Initial Value	0	3	0	0	0	0

The closed loop poles for position (a single pole for each component) were placed at $s = -1.4\ \mathrm{rad/s}$. For orientation, the closed loop pole was set directly in the Z-plane at $z = 0.89$ (see Section 3.3.1). The average time step of the control algorithms was measured at $\bar{T} = 45\ \mathrm{ms}$ with a standard deviation of $\sigma_T = 13\ \mathrm{ms}$. The response of the system is shown in Figure 31, which include the robot's position during stabilization and its velocity in the robot's frame, as obtained both in the Webots simulation environment and in the real ISocRob robots. The correlation between the simulated data, which was obtained using the discrete-time control law (3.12), and the data obtained from the physical system, through the application of both the continuous-time approximation of (3.2) and the exact control law (3.12) is described in Table 2. The settling times (to 5% of their final value) of the components of the robot's posture are also shown.

Table 2: Correlation coefficients and settling times for the point stabilization experiment.

	x	y	θ	v_x	v_y	ω
Correlation between real (exact) and simulated data, r_E	0.989	0.986	0.974	0.908	0.805	0.682
Correlation between real (approx.) and simulated data r_A	0.975	0.917	0.934	0.711	0.165	0.763
Settling time (5%, simul.), t_{SS} (s)	4.77	5.01	4.89	-	-	-
Settling time (5%, exact), t_{SE} (s)	5.80	6.67	3.33	-	-	-
Settling time (5%, approx.), t_{SA} (s)	9.07	8.94	4.28	-	-	-

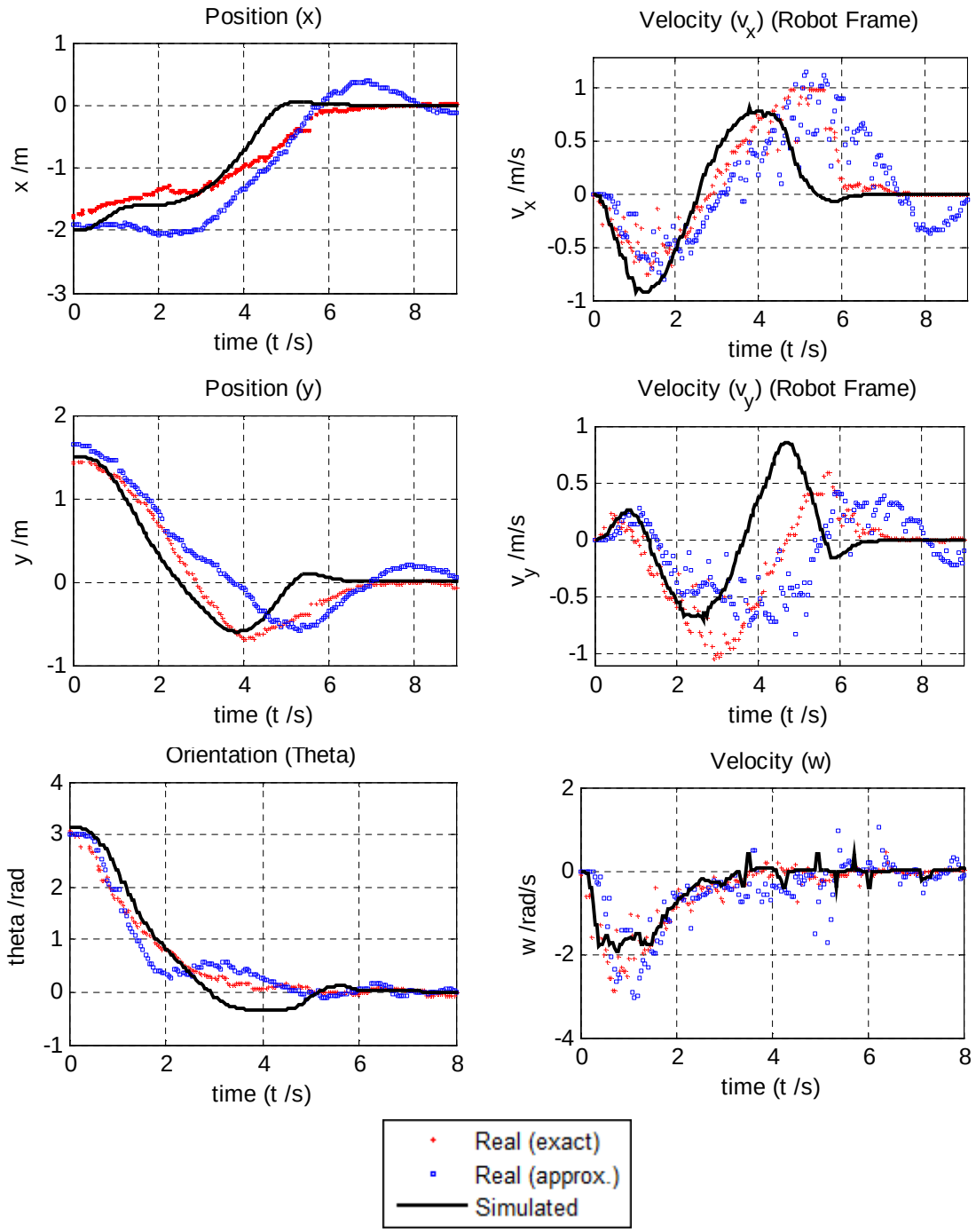


Figure 31: Response of the closed-loop system during posture stabilization. The presented velocity components are taken in the robot's frame.

Even though there is agreement between the real and simulated data, the most noticeable aspect of the robot's motion is that there is significant overshoot in one of the robot's position components (with this choice of initial conditions, this overshoot occurs along the y component, to about 30% of the initial value). Besides unmodeled factors such as wheel slippage, and unreliability in the determination of the robot's velocity, this is mostly due to the acceleration limits of the robot's actuators, which, by having to account for both linear and angular velocity, are unable to preserve the

expected linear trajectory of the robot. To avoid this situation altogether, the closed-loop poles would have to be moved to slower locations, which would result in longer settling times for distant goal postures. As an alternative, the acceleration limits of the actuators could be raised, but the increased effort would lead to the fast deterioration of the actuators. Notwithstanding the presence of these errors, it is clear that the discrete-time control law (3.12) provides superior performance than the finite-difference approximation of (3.2), especially with respect to the closed loop system's settling time. However, it is verified that both of these solutions successfully solve the posture stabilization problem for a holonomic robot.

7.2 Obstacle Avoidance

Two situations are presented to test the obstacle avoidance algorithm: to achieve posture stabilization in a setting with sparsely distributed static obstacles; and to escape a common local-minimum situation. Similarly to the original posture stabilization experiment, in both cases the robot must achieve the goal posture $\boldsymbol{q}_g = [0 \quad 0 \quad 0]^T$. In this case, however, the orientation of the robot is not considered, since it is not directly linked to the obstacle avoidance process, and would therefore make it harder to evaluate any sources of error.

For the first experiment, the robot is left, initially stopped, at position $\boldsymbol{p}_0 = [x_0 \quad y_0]^T$ with $x_0 = -3.5\ \mathrm{m}$ and $y_0 = 0\ \mathrm{m}$. The positions of the obstacles, as well as the path taken by the robot around them to its goal position, are shown in Figure 32.

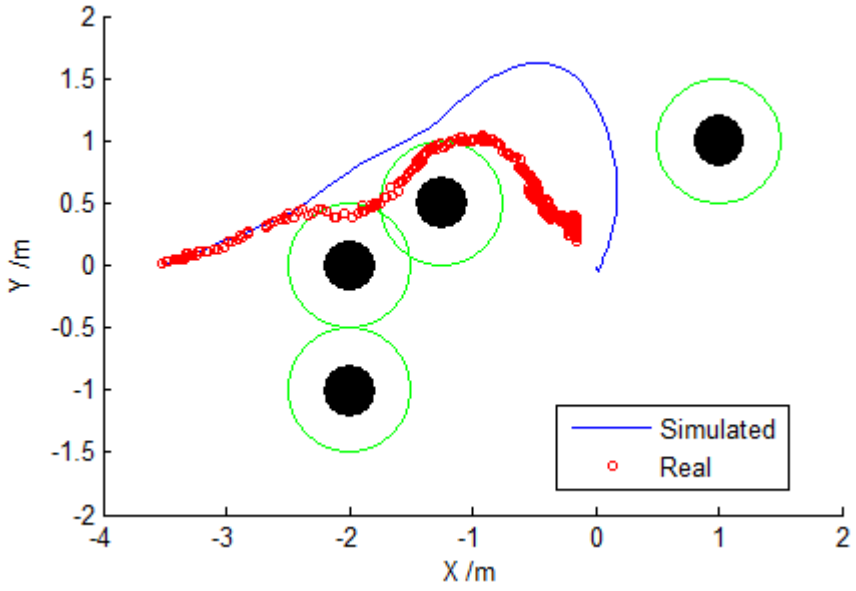


Figure 32: Obstacle avoidance with a typical obstacle configuration. The black filled circles represent the static obstacles in the environment, and the green circles around them represent the safety distance that the robot must keep.

It should be noted that, even though some of the measurements of the robot's position in the real system lie inside of the radii of safety of certain obstacles, the robot itself was verified to avoid

these obstacles and reach its goal safely, and so these situations are recognized as self-localization errors (it would otherwise be impossible for the robot to occupy those positions). Since obstacle-detection is relative to the robot's posture and therefore free from these errors, this carries no meaning to the correct execution of the obstacle avoidance algorithm.

In a second situation, the robot is placed inside a "U-shaped" formation of obstacles, as depicted in Figure 33. Its initial position is given by $x_0 = -2.5$ m and $y_0 = 0$ m. Both in the real and simulated experiments, the robot is seen to successfully detect the endpoints of this cluster of obstacles, according to the algorithm described in Section 5.3, and is thus able to escape the local minimum situation and reach its goal safely.

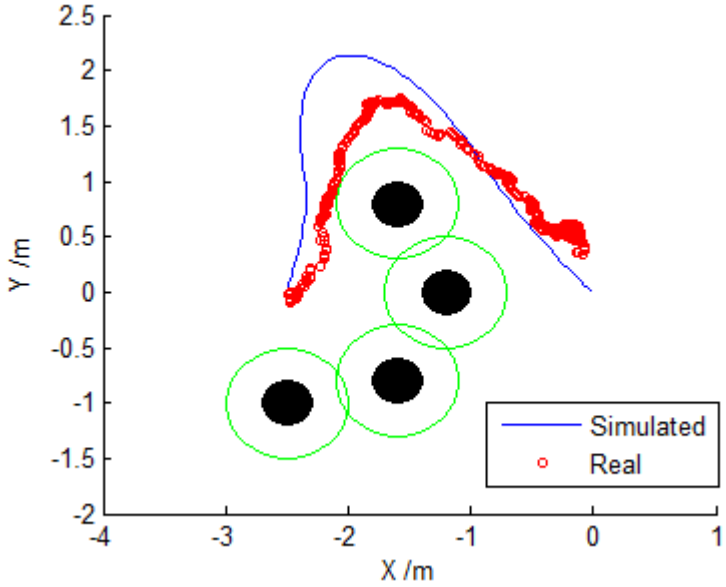


Figure 33: Escaping a local minimum situation.

7.3 Posture Tracking

To test the validity of the proposed posture tracking solution, the robot is tasked with tracking a freely moving object on the field of play with constant acceleration, by maintaining a fixed distance to the object (in this case, 1 m) and so that the robot is oriented towards it (i.e. the relative angle of the object in the robot's frame, θ_O, is zero). The initial conditions for these experiments are described in Table 3, where $\{x_I, y_I\}$ denote the position that the robot must attain, and $\{v_{O_x}, v_{O_y}\}$ the reference velocity. The acceleration of the object is given by $a_{O_x} = a_{O_y} = -0.025$ m/s^2, in the world frame.

Table 3: Initial conditions for the posture tracking experiments.

	x (m)	y (m)	θ (rad)	v_x (m/s)	v_y (m/s)	ω (rad/s)	x_I (m)	y_I (m)	v_{Ox} (m/s)	v_{Oy} (m/s)
Initial Value	0	3	0	0	0	0	1	-1	1	1

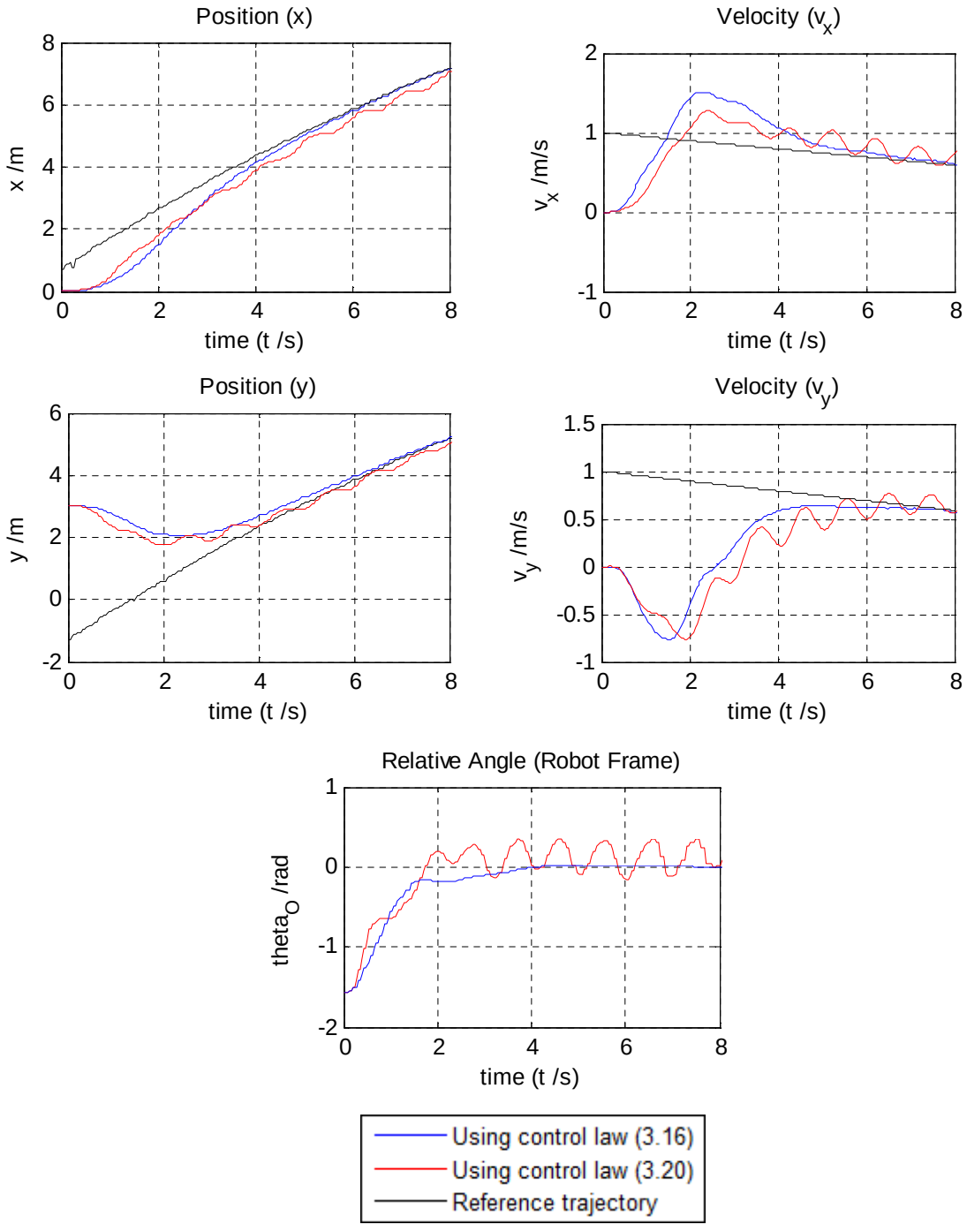


Figure 34: Position and linear velocity (world frame) of the robot during trajectory tracking, as well as the angle between the robot and the object in the robot's frame.

The closed loop system is critically damped with $\lambda_1 = \lambda_2 = 1$. The average time step was $\bar{T} = 40\ \text{ms}$ with a standard deviation of $\sigma_T = 1.5\ \text{ms}$. These results show that the robot is able to simultaneously track a given trajectory for position while maintaining an independent reference for orientation. While the finite-difference approximation of control law (3.16) is shown to successfully solve the point-tracking problem, under the direct application of the discrete-time control law (3.20) the system displays oscillatory behavior. This arises not only from the variations in the elapsed time

between consecutive iterations (since, even if these variations are not significant, they introduce error into the discretization operations performed in Section 3.2), but mainly from difficulties in synchronizing the inflow of information from the sensors (the sensors that are responsible for determining the robot's position and velocity run at different frequencies, which may vary as well). Since, in the MeRMaID architecture the sampling intervals cannot be set to a constant value, this invalidates the usage of control law (3.20) in this situation, albeit it may still retain its usefulness in more restricted environments. For the practical requirements of the ISocRob team, it is clear through this analysis that the finite-difference approximation of (3.16) displays superior results.

7.4 Moving Object Interception

The simulations made in the Webots environment, to confirm the proposed solution to the moving object interception problem, consisted of having the robot intercept a freely rolling ball, travelling along an initially linear path, as soon as it entered its detection range. To establish whether the proposed solution is more efficient, by relying on Ideal Proportional Navigation and trajectory tracking, than a solution that relies on trajectory tracking alone, both of these approaches are compared. Two situations are considered: in the first experiment, the ball is left along an unobstructed path; also, to verify the robustness of the proposed solution to unexpected variations in the motion of the ball, a situation is considered where the ball collides with an obstacle in the environment during interception. The initial conditions for both of these experiments are presented in Table 4, where, as before, $\{x_I, y_I\}$ represents the desired position of the robot, which in this case defines the interception trajectory (see Section 4.2.2), and $\left\{v_{B_x}, v_{B_y}\right\}$ are the components of the ball's velocity. As in the posture tracking experiment, the acceleration of the ball was set to $a_{B_x} = a_{B_y} = -0.025\ \mathrm{m/s^2}$. In this setting, the angle between the robot and the ball in the robot's frame is θ_B.

The results for the first experiment are shown in Figure 35 and Figure 37. Both with the proposed solution and simple trajectory tracking, the robot is able to intercept the ball and come to a full stop. The total required time for interception, using the proposed solution, was $t_{total} = 8.50$ s utilizing the approach described in Section 4.2, and $t_{total} = 9.16\ s$ using an approach based completely on trajectory tracking. The control-switching instants (refer to Section 4.2.7) are $t_{S1} = 1.59$ s, $t_{S2} = 4.58$ s and $t_{S3} = 6.87$ s. It is then verified that the application of a LOS-relative acceleration command effectively shortens the interception process. This is due to the fact that, between instants $t_{S1}$and t_{S2} (see Figure 37), when IPNG is active, the distance to the ball decreases in an approximately linear manner, which, under these circumstances, is faster than the exponential decay of the trajectory tracking control law.

It can be seen through Figure 35 that the sensors that return the velocity of the ball have some amount of associated noise, even during simulation. The measurements of the velocity of the ball were verified to possess a standard deviation of $\sigma_v \cong 0.04\ \mathrm{m/s}$. While it is expectable that there should be a greater amount of noise involved in the measurements performed by a real robotic system, this already indicates that the presented algorithms are still capable of achieving a successful interception in the presence of these errors.

Table 4: Initial conditions for the ball interception experiments.

	x (m)	y (m)	θ (rad)	v_x (m/s)	v_y (m/s)	ω (rad/s)	x_I (m)	y_I (m)	v_{Bx} (m/s)	v_{By} (m/s)
Initial Value	-2	-2	0	0	0	0	-1	-3	0.7	0.7

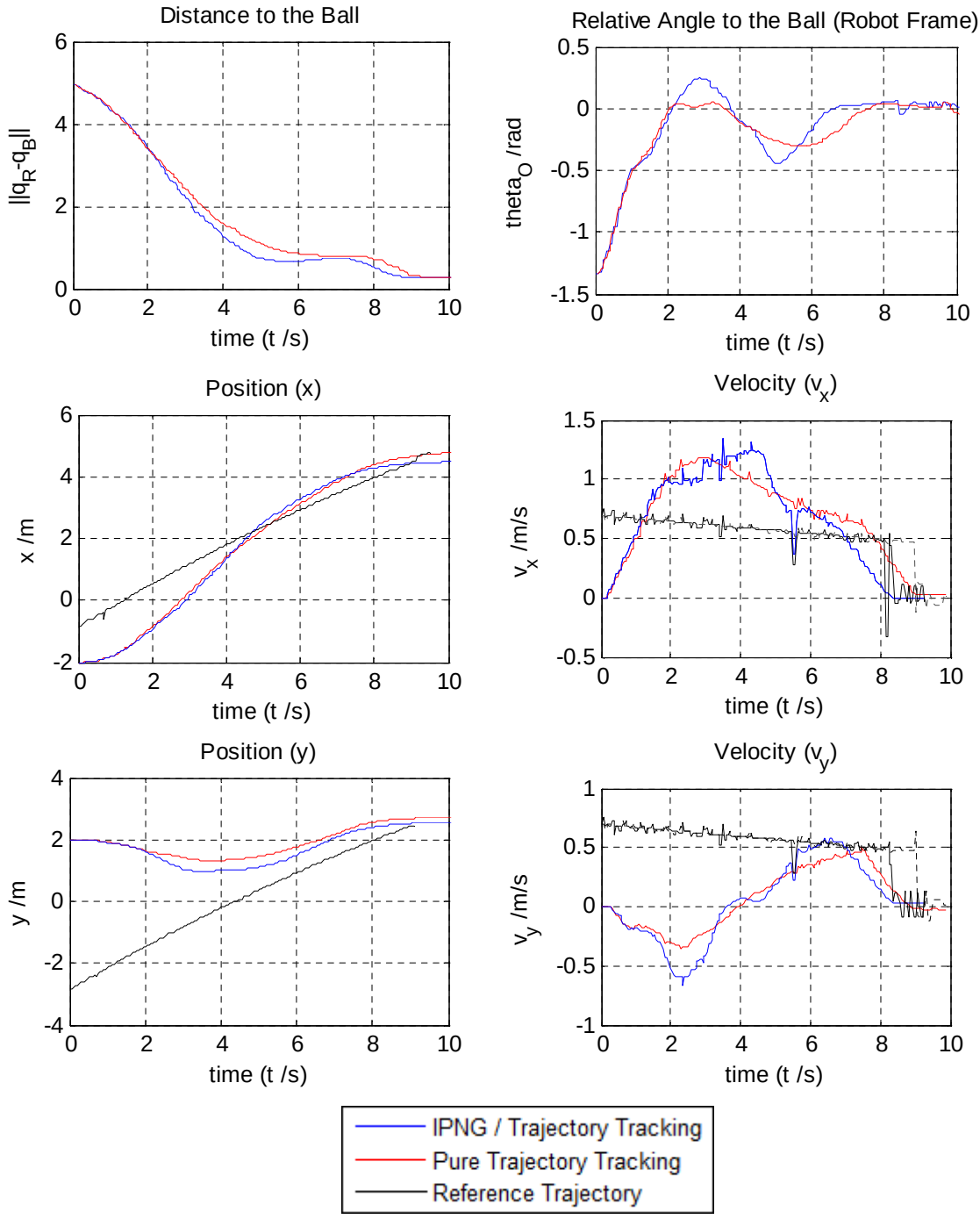


Figure 35: Position and linear velocity (world frame) of the robot, as well as the distance and relative angle to the ball, during the first interception experiment.

For the second experiment, the ball collides with an object in the environment (placed at $x = 1\ \mathrm{m}; y = -1\ \mathrm{m}$) , after being detected by the robot and thus during interception, at $t \cong 1.7\ \mathrm{s}$. The results are shown in Figure 36 and Figure 38. In this situation the advantages of proportional navigation over conventional tracking methods are more evident.

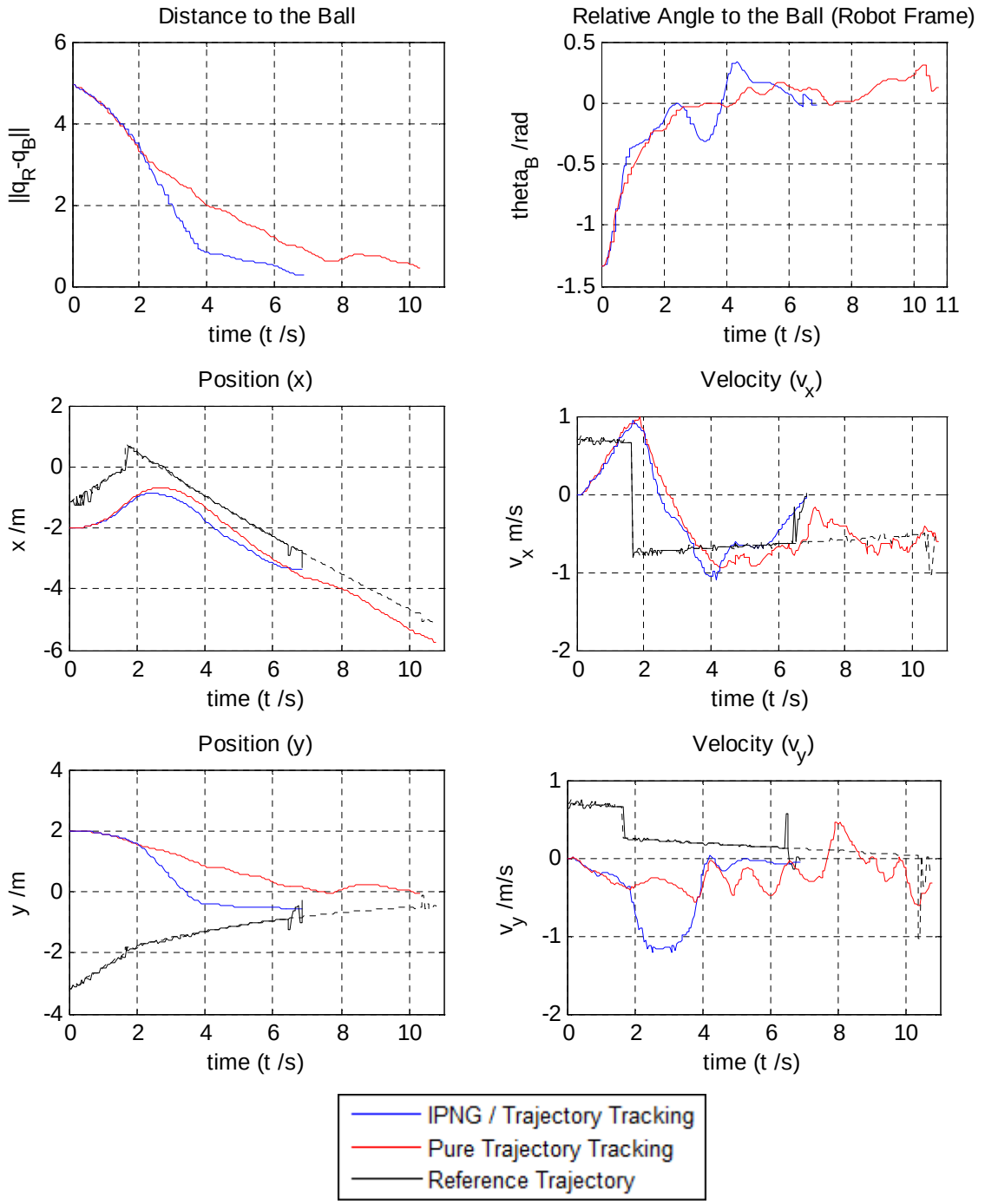


Figure 36: Position and linear velocity (world frame) of the robot, as well as the distance and relative angle to the ball, during the second interception experiment.

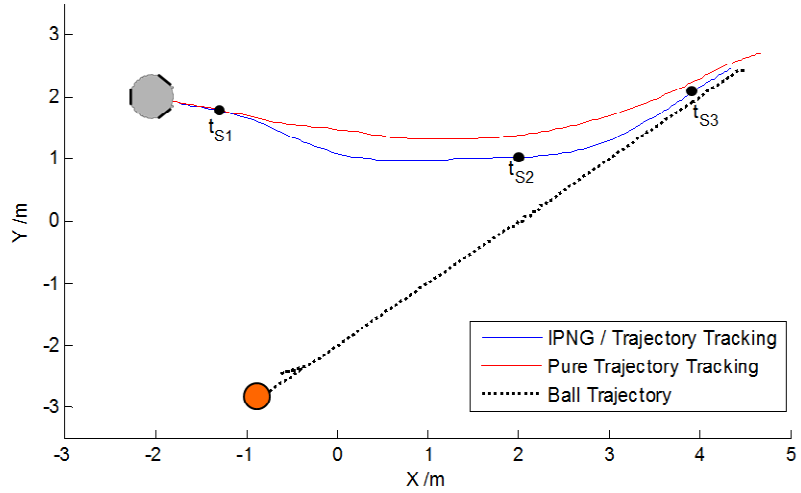


Figure 37: Trajectories described by the robot and the ball during unobstructed interception.

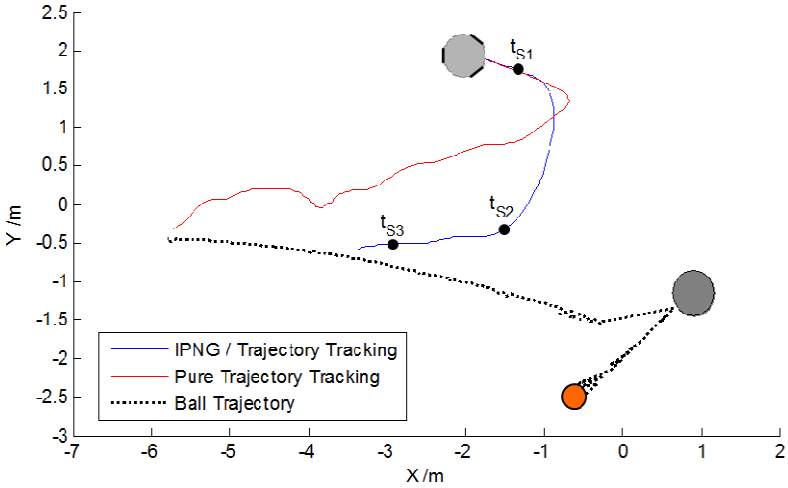


Figure 38: Trajectories described by the robot and the ball in a situation where the ball collides with an obstacle.

At the moment that the ball hits the object and its trajectory changes, the tracking algorithm is slow to respond to the sudden change in the reference velocity, and the robot is slowed down in the process, requiring more effort from the actuators to restore the robot's speed, which in turn makes it more difficult to reduce the distance to the ball (which can be seen in Figure 36). In contrast, while under the control of IPNG, the respective acceleration command acts to preserve the norm of the relative velocity between the robot and the ball, which implies that the distance to the ball is decreased more efficiently. In this case, the switching instants are $t_{S1} = 1.69$ s, $t_{S2} = 3.78$ s and $t_{S3} = 5.53$ s. The required time for interception is $t_{total} = 6.87$ s for the proposed solution and $t_{total} = 10.3$ s for trajectory tracking alone. The difference between these two values is justified not only by the reasons already presented, but also by the fact that the robot must avoid the ball (by using the concepts discussed in Section 5.4) whilst converging towards the interception trajectory, and some oscillations are caused in the robot's motion due to this fact (as can be seen in Figure 38).

7.5 Object Transport

To test the object transport solution, the robot was required to move a soccer ball, which is initially in contact with the robot's chassis, to a reference position $\boldsymbol{p}_g = [0 \quad 0]^T$. Two experiments are performed: stabilization in an unobstructed environment; and obstacle avoidance while dribbling.

For both experiments, the initial conditions are presented in Table 5.

Table 5: Initial conditions for the object transport experiments.

	x (m)	y (m)	θ (rad)	v_x (m/s)	v_y (m/s)	ω (rad/s)	x_B (m)	y_B (m)	v_{Bx} (m/s)	v_{By} (m/s)
Initial Value	-4	0	π	0	0	0	-4.3	0	0	0

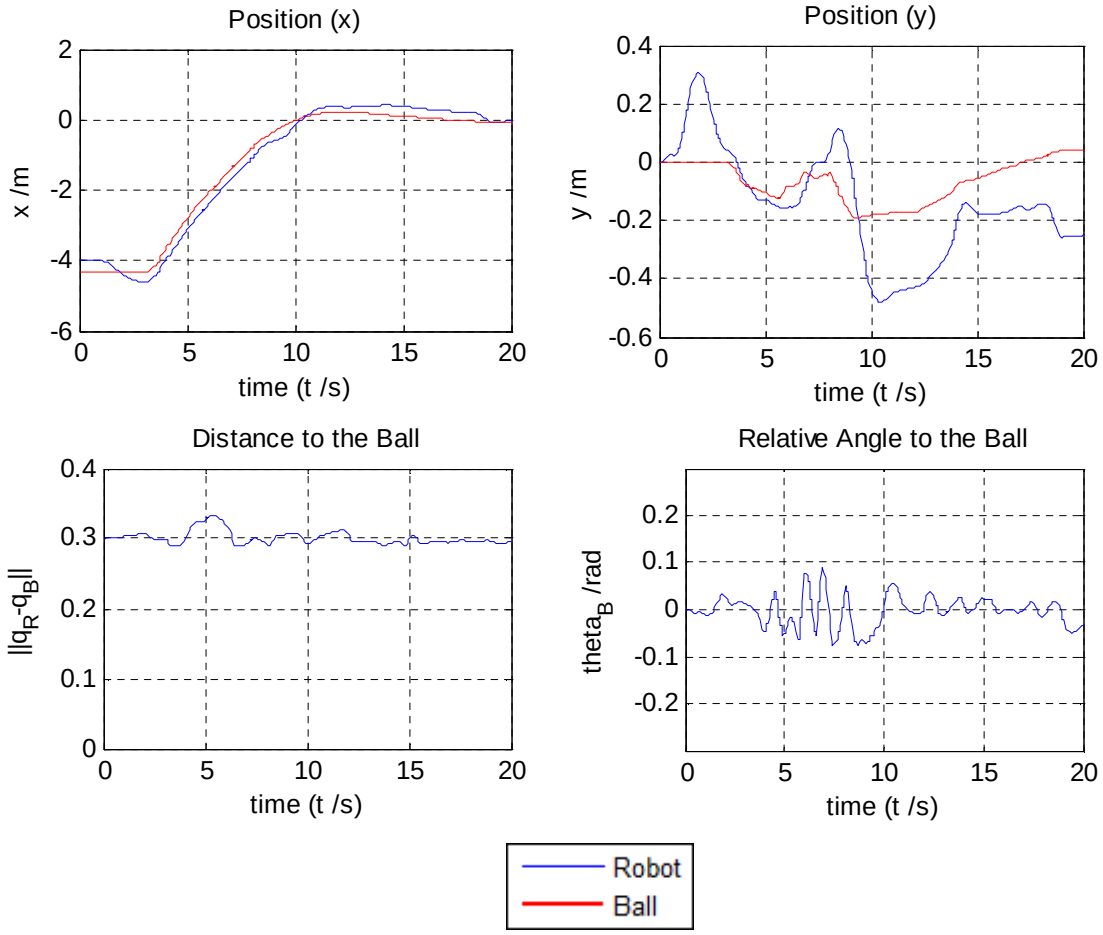


Figure 39: Position of the ball and the robot's geometric center during unobstructed dribbling, as well as the distance and relative angle between the ball and the robot.

The results of the unrestricted motion experiment are shown in Figure 39 and Figure 41. It is verified that, if nothing is preventing the ball from reaching the desired position directly, the resulting trajectory described by the ball is approximately linear, as it was expected from the model of the ball

presented in Section 4.1 and the object controller described in Section 4.3.2. The robot is able to maintain the physical restrictions of the dribbling process valid throughout the motion of the ball, which, in this case, are such that $\|\boldsymbol{q}_R - \boldsymbol{q}_B\| = 0.3$ m and $\theta_B = 0$ rad.

For the second experiment, the robot is faced with an obstacle formation, which possesses sufficient space for the robot to pass through with little effort if its motion was unrestricted, but requires significant deviation while dribbling, since the "safety distance" that must be kept from the obstacles is effectively greater in this case (see Figure 42).

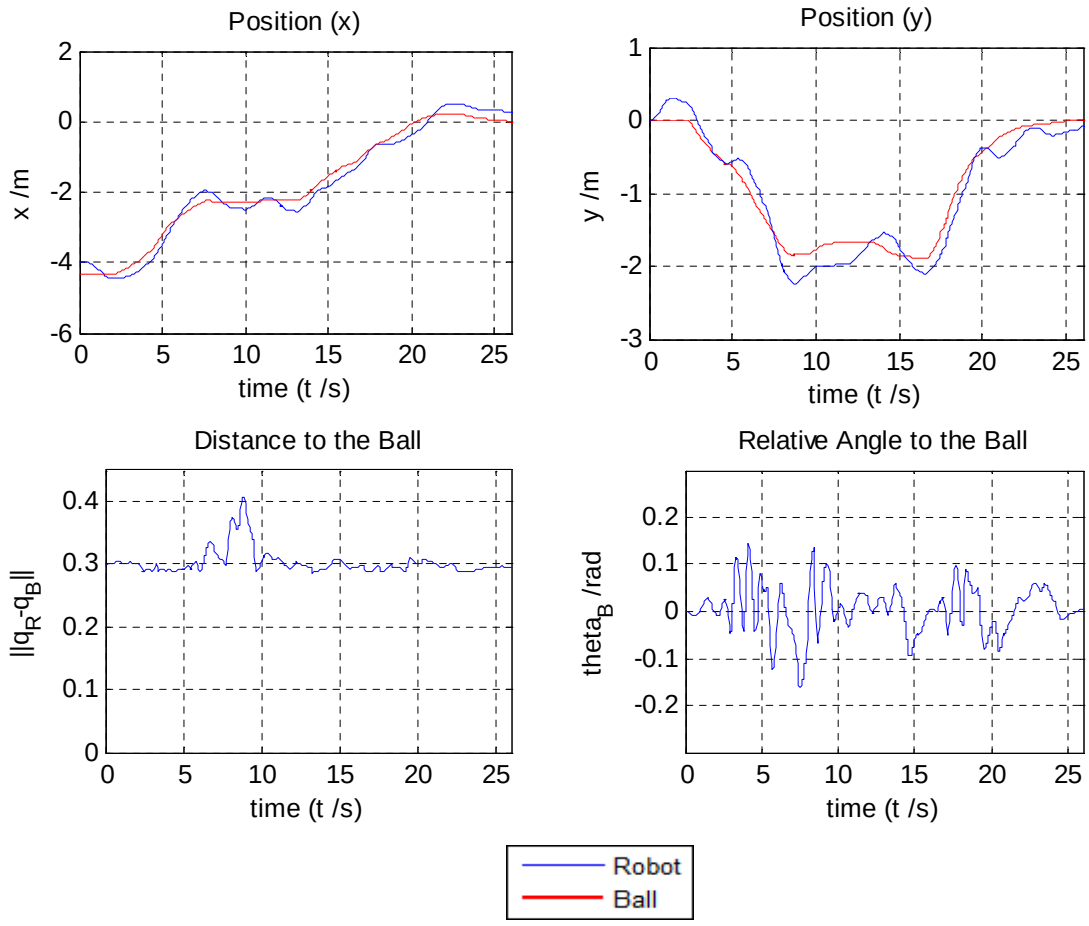


Figure 40: Position, relative distance and relative angle during transport while avoiding obstacles.

While the robot is still able to avoid these obstacles and stabilize the ball around its goal position, if an accentuated curvature is required by the ball (a rapidly varying required force vector), in some situations, the ball may briefly stop during its motion. This happens due to slight collisions with the robot while it is trying to accompany the angle of the required force vector, added to the effects of friction while the robot is unprepared to apply forces onto the ball (see Section 4.3.3). This, in turn, causes some noise in the direction of the force vectors required by the object controller, and since the robot must reposition itself in order to overcome friction and to set the ball in motion again, this may result in a situation where the robot performs a "loop" around the ball. Without possessing specialized actuators to interact with the ball, this problem may be solved by maintain the ball at a higher speed

during its motion, where the effects of these unexpected collisions are less noticeable, but this, in turn, reduces the safety of the system during its motion, and the ability of the robot to manipulate the ball.

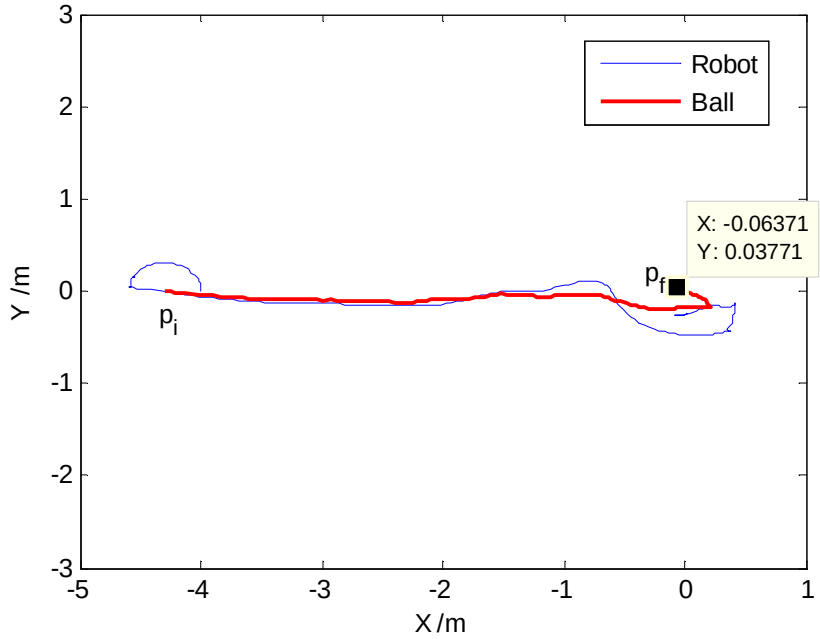


Figure 41: Trajectory described by the robot's geometric center and the ball during an unobstructed transport. The ball begins its motion at position p_i and ends at p_f.

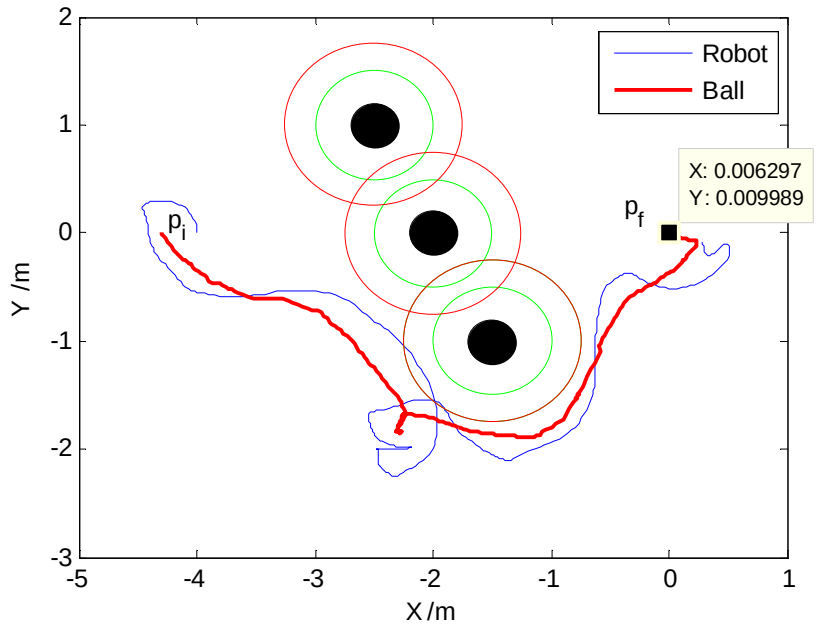


Figure 42: Trajectory described by the robot's geometric center and the ball during transport while avoiding obstacles. The obstacles are represented by the black filled circles in the environment. Around these, a green circle is drawn that symbolizes the safety distance that the robot must keep during unrestricted motion, and a red circle, which represents the necessary safety distance while dribbling. The ball begins its motion at position p_i and ends at p_f.

8 Conclusions and Future Work

8.1 Conclusions

The control algorithms that were presented in this work address the most common motion control problems for holonomic robots, namely the problems of posture stabilization, posture tracking, moving object interception and transport of moveable objects. The problem of obstacle avoidance was also considered.

Feedback linearization techniques were used to obtain control laws that solve both the point stabilization and point tracking problems for holonomic robots. Alongside these control laws for position, discrete-time solutions to the orientation control of the robot were presented, allowing velocity control for static references and torque control for references moving with constant accelerations. The solutions obtained through this approach can be readily analyzed with respect to their performance, and provide a highly adaptable framework, through which new tasks can be easily defined, and changes may be made to the robot's hardware, without requiring thorough recalibration.

It was verified that, for the posture stabilization problem, the definition of an appropriate control law in discrete time makes the system more robust to changes in the rate of execution of the control algorithms, and that the overall effectiveness of the solution is increased for tasks with a low execution rate. For posture tracking, however, a finite-difference approximation of a continuous-time control law was seen to generate better results.

A solution to the problem of moving object interception by a holonomic mobile robot was presented, based on work previously developed for robotic manipulators. Although it was not possible to test this solution in a real robotic platform, tests were performed in the Webots simulation environment on models of the ISocRob robots, and this technique was shown to allow the successful interception of a freely rolling ball, regardless of the initial conditions of the problem. This solution was also shown to be also robust to unexpected variations in the trajectory of the object. Through the proper selection of the closed-loop system's poles, it was shown that, while tracking, it is possible to avoid overshoot situations, while taking the limitations of the robot's actuators into account. This property was exploited to minimize the total required time for interception. For the problem of transporting an object through the pushing action of a mobile robot, a solution based on an Interface-Control scheme was developed, that relies on the application of a PD controller to provide the required accelerations of the object in question, and uses these accelerations as a reference for a Hybrid Position/Force controller acting upon the robot. This motion control technique explicitly takes into account the physical restrictions between the object and the robot, and allows for task versatility, since the motion of the object only depends on the controller that is used to provide its desired accelerations. This can be exploited to include, for example, obstacle avoidance while transporting the object. Compensation terms were added to the Hybrid Position/Force controller so as to maintain the physical restrictions valid throughout the motion of the object. This was shown, through simulation in the Webots environment, to allow a soccer robot to dribble a ball across the field of play, while

avoiding obstacles, even if the robot does not possess any actuators that facilitate dribbling, such as the rolling drums.

An obstacle avoidance algorithm was presented, which deviates the desired control inputs of the robot so that it passes tangent, in the configuration space, to each detected obstacle, which are modelled as circles. The solution was tested both in the real ISocRob robots and in the Webots environment, where it proved successful. The paths described by the robot while using this method approach the globally shortest paths for the most common configurations of obstacles that are encountered during a robotic soccer match. The method was extended to explicitly account for moving obstacles, and used as an integral part of the object interception methods, where it was verified that the robot successfully avoids a moving ball while converging towards the desired interception trajectory. It was verified, however, that during the application of the obstacle avoidance algorithm to situations where torque control is required, the robot may exhibit oscillations in its motion when it is near an obstacle. This is due to the nonlinearities introduced by the robot's actuators, which, as of this moment, are not taken into account during obstacle avoidance.

8.2 Future Work

This work has not yet exploited the full capabilities of any actuators that the mobile robot may have that allow interaction with moveable objects by means other than pushing those objects with the robot's chassis. This accounts, for example, for the rolling drums that are commonly used by soccer robots. This was due to the fact that, at the time of writing, the rolling drums of the ISocRob robots were not operational and no short-term measures could be taken to replace them. With such actuators, new possibilities for transport and interception appear. Namely, it would be possible to apply forces to the ball along the robot frame's *y*-axis, by applying a certain angular acceleration to the robot, while using the roll to pull the ball towards the robot's chassis. This would generate an effect of turning the ball around the robot that, while not desirable under normal conditions, could enable the robot to escape impending collisions if the restrictions of the dribbling process could not be maintained, due to the presence of obstacles.

Although the presented obstacle avoidance algorithm was shown to provide acceptable results, some algorithms, such as the Dynamic Window approach, would enable a more elegant description of the obstacle avoidance process while transporting objects. A form of implementing the concepts introduced by this approach in a computationally cheap manner could prove effective in addressing some of the limitations of the present algorithm, namely by explicitly considering the capabilities of the actuators for added safety.

The proposed solutions to moving object interception and transport are well suited to dynamic ball-pass behaviors in the robotic soccer environment. While this possibility was not yet explored, it would entail using the Hybrid Position/Force controller to turn the dribbling robot to the receiver of the pass, while maintaining the ball in motion (which is actually a trivial extension to the original problem), and using the interception techniques on the receiver to capture the ball after the passing action is executed.

9 Bibliography

[1] C. F. Marques and P. U. Lima, "Multi-sensor Navigation for Soccer Robots," in *Proceedings of the RoboCup 2001 Symposium*, Seattle, USA, 2001.

[2] B. D. Damas, P. U. Lima, and L. M. Custódio, "A Modified Potential Fields Method for Robot Navigation Applied to Dribbling in Robotic Soccer," *Lecture Notes in Computer Science*, pp. 65-77, 2003.

[3] R. A. Brooks, "Robust Layered Control System For A Mobile Robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14-23, 1986.

[4] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. MIT press, 2004.

[5] M. Barbosa, N. Ramos, and P. Lima, "Mermaid - Multiple-Robot Middleware for Intelligent Decision-Making," in *Proc. of IAV2007 - 6th IFAC Symposium on Intelligent Autonomous Vehicles*, Toulouse, France, 2007.

[6] N. H. McClamroch and D. Wang, "Feedback Stabilization and Tracking of Constrained Robots," *IEEE Transactions on Automatic Control*, vol. 33, no. 5, pp. 419-426, May 1998.

[7] C. C. de Wit and O. J. Sordalen, "Exponential stabilization of mobile robots with nonholonomic constraints," in *Proceedings of the 30th Conference on Decision and Control*, Brighton, UK, 1991, pp. 692-697.

[8] C. C. de Wit, B. Siciliano, and G. Bastin, *Theory of Robot Control*. Springer, 1996.

[9] P. G. Plöger, G. Indiveri, and J. Paulus, "Motion Control of Swedish Wheeled Mobile Robots in the Presence of Actuator Saturation," in *RoboCup 2006 Symposium, Proceedings*, Bremen, Germany, 2006.

[10] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 4th ed. Pearson Education International, 2002.

[11] S.-O. Lee, Y.-J. Cho, M. Hwang-Bo, B.-J. You, and S.-R. Oh, "A Stable Target-Tracking Control for Unicycle Mobile Robots," in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.

[12] S. Kawarai, "A Direct Method for Exact Discretization of Ordinary Differential Equations," in *Proceedings of the 47th IEEE International Midwest Symposium on Circuits and Systems*, Hiroshima, Japan, 2004, pp. 89-92.

[13] M. Mehrandezh, M. N. Sela, R. G. Fenton, and B. Benhabib, "Robotic interception of moving objects using ideal proportional navigation guidance technique," *Robotics and Autonomous Systems*, vol. 28, pp. 295-310, 1999.

[14] R. Sharma, J.-Y. Hervé, and P. Cucka, "Dynamic Robot Manipulation Using Visual Tracking," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, 1992.

[15] Z. Lin, V. Zeman, and R. V. Patel, "On-Line Robot Trajectory Planning for Catching a Moving

Object," in *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, USA, 1989, pp. 1726-1731.

[16] E. A. Croft, R. G. Fenton, and B. Benhabib, "Optimal Rendezvous-Point Selection for Robotic Interception of Moving Objects," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 28, no. 2, pp. 192-204, Apr. 1998.

[17] D. Fernandes, "Arquitecturas de Seguimento Visual e Captura por um Manipulador Robótico de Objectos em Movimento," MsC Thesis, Instituto Superior Técnico, Lisbon, 1997.

[18] J. A. Borgstadt and N. J. Ferrier, "Interception of a Projectile Using a Human Vision-Based Strategy," in *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, San Francisco, CA, USA, 2000, pp. 3189-3196.

[19] I. R. Manchester, E. M. P. Low, and A. V. Savkin, "Vision-Based Interception of a Moving Target by a Mobile Robot," in *16th IEEE International Conference on Control Applications*, Singapore, 2007, pp. 397-402.

[20] F. P. Adler, "Missile Guidance by Three-Dimensional Proportional Navigation," *Journal of Applied Physics*, vol. 27, no. 5, pp. 500-507, May 1956.

[21] V. Rajasekhar and A. G. Sreenatha, "Fuzzy Logic Implementation of Proportional Navigation Guidance," *Acta Astronautica*, vol. 46, no. 1, pp. 17-24, 2000.

[22] P.-J. Yuan, "Optimal Guidance of Proportional Navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 3, pp. 1007-1012, Jul. 1997.

[23] P. Stone and M. Veloso, " A layered approach to learning client behaviors in the robocup soccer server," *Applied Artificial Intelligence*, vol. 12, pp. 165-188, 1998.

[24] H. Müller, et al., "Making a Robot Learn to Play Soccer Using Reward and Punishment," *Lecture Notes in Computer Science*, vol. 4667/2007, pp. 220-234, 2007.

[25] T. Gabel and M. Riedmiller, "Learning a Partial Behavior for a Competitive Robotic Soccer Agent," *KI- Künstliche Intelligenz*, vol. 20, no. 2, pp. 18-23, May 2006.

[26] R. Sargent, B. Bailey, C. Witty, and A. Wright, "Dynamic Object Capture Using Fast Vision Tracking," *AI Magazine*, vol. 18, no. 1, pp. 65-72, 1997.

[27] F. Stolzenburg, O. Obst, and J. Murray, "Qualitative Velocity and Ball Interception," in *KI 2002: Advances in Artificial Intelligence, Twentyfifth Annual German Conference*, Aachen, Germany, 2002, p. 283–298.

[28] J. M. Borg, M. Mehrandezh, R. G. Fenton, and B. Benhabib, "An Ideal Proportional Navigation Guidance System for Moving Object Interception - Robotic Experiments," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2000, pp. 3247-3252.

[29] M. Mehrandezh, N. M. Sela, R. G. Fenton, and B. Benhabib, "Robotic interception of moving objects using an augmented ideal proportional navigation guidance technique," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 30, no. 3, pp. 238-250, May 2000.

[30] U. S. Shukla and P. R. Mahapatra, "The proportional navigation dilemma-pure or true?," *IEEE*

Transactions on Aerospace and Electronic Systems, vol. 26, no. 2, pp. 382-392, Mar. 1990.

[31] P.-J. Yuan and J.-S. Chern, "Ideal proportional navigation," *NASA STI/Recon Technical Report A*, vol. 95, pp. 501-512, 1993.

[32] C.-D. Yang and C.-C. Yang, "A unified approach to proportional navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 2, pp. 557-567, Apr. 1997.

[33] X. Li, M. Wang, and A. Zell, "Dribbling Control of Omnidirectional Soccer Robots," in *Proc. of the 2007 IEEE International Conference on Robotics and Automation*, Rome, Italy, 2007, pp. 2623-2628.

[34] M. Riedmiller and A. Merke, "Using Machine Learning Techniques in Complex Multi-Agent Domains," in *Adaptivity and Learning*. Springer, 2003.

[35] Y. Nakamura, K. Nagai, and T. Yoshikawa, "Mechanics of coordinative manipulation by multiple robotic mechanisms," in *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, 1987, pp. 991-998.

[36] W. C. Dickson and R. H. ,. J. Cannon, " Experimental results of two free-flying robots capturing and manipulating a free-flying object," in *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. 'Human Robot Interaction and Cooperative Robots'* , Pittsburgh, PA, USA, 1995, pp. 51-58.

[37] T. Yoshikawa, "Dynamic hybrid position/force control of robot manipulators--Description of hand constraints and calculation of joint driving force," *IEEE Journal of Robotics and Automation*, vol. 3, no. 5, pp. 386-392, Oct. 1987.

[38] T. Yoshikawa, T. Sugie, and M. Tanaka, "Dynamic Hybrid Position/Force Control of Robot Manipulators-Controller Design and Experiment," in *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, 1987, pp. 2005-2010.

[39] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," in *Autonomous robot vehicles*, I. J. Cox and G. T. Wilfong, Eds. New York, NY, USA: Springer-Verlag New York, Inc., 1990, pp. 363-390.

[40] I. Kamon, E. Rivlin, and E. Rimon, "A new range-sensor based globally convergent navigation algorithm for mobile robots," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, USA, 1996, pp. 429-435.

[41] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90-98, 1986.

[42] L. Chengqing, M. H. ,. J. Ang, H. Krishnan, and L. S. Yong, "Virtual Obstacle Concept for Local-minimum-recovery in Potential-field Based Navigation," in *Proc. of the 2000 IEEE International Conference on Robotics and Automation, ICRA apos;00.*, San Francisco, CA, USA, 2000, pp. 983-988.

[43] X. Yun and K.-C. Tan, "A Wall-Following Method for Escaping Local Minima in Potential Field Based Motion Planning," in *Proceedings of the 8th International Conference on Advanced Robotics. ICAR apos;97.* , Monterey, CA, USA, 1997, pp. 421-426.

[44] J. Borenstein and Y. Koren, "Real-time Obstacle Avoidance for Fast Mobile Robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179-1187, 1989.

[45] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278-288, Jun. 1991.

[46] J. Minguez and L. Montano, "Nearness diagram navigation (ND): a new real time collision avoidance approach," in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000).* , Takamatsu, Japan, 2000, pp. 2094-2100.

[47] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics & Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23-33, Mar. 1997.

[48] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation* , Minneapolis, Minnesota, USA, 1996, pp. 3375-3382.

[49] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proc. of the 1999 IEEE International Conference on Robotics and Automation*, Detroit, MI, USA, 1999, pp. 341-346.

[50] W. Feiten, R. Bauer, and G. Lawitzky, "Robust obstacle avoidance in unknown and cramped environments," in *Proc. of the 1994 IEEE International Conference on Robotics and Automation.*, San Diego, CA, USA, 1994, pp. 2412-2417.

[51] Y. H. Liu and S. Arimoto, "Path planning using a tangent graph for mobile robots among polygonal and curved obstacles," *International Journal of Robotic Research*, vol. 11, no. 4, pp. 376-382, Aug. 1992.

[52] M. Bowling and M. Veloso, "Motion control in dynamic multi-robot environments," in *Proceedings of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, CA, USA, 1999, pp. 168-173.

[53] J. Messias, J. Santos, J. Antunes, and P. Lima, "Monte Carlo Localization Based on Gyrodometry and Line-Detection," in *Proc. of ROBÓTICA2008 - 8th Conference on Mobile Robots and Competitions*, Aveiro, Portugal, 2008.

[54] J. C. Alexander and J. H. Maddocks, "On the Kinematics of Wheeled Mobile Robots," in *Autonomous robot vehicles*, I. J. Cox and G. T. Wilfong, Eds. New York, NY, USA: Springer-Verlag New York, Inc. , 1990, pp. 5-24.

A1. Kinematic and Dynamic Properties of an Omnidirectional Mobile Robot

In this section, a brief review is made to the fundamental concepts used throughout the remainder of the work. A basic omnidirectional robot is modelled with respect to its kinematics and dynamics, and its most relevant control properties are presented. For a more thorough description of these concepts, the reader is referred to [8],[4],[54].

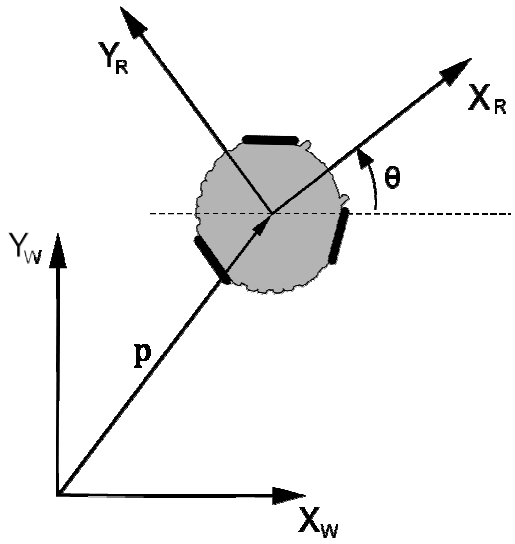


Figure 43: Representation of the world frame and the robot frame.

Let the *posture* $\boldsymbol{q}$ of a mobile robot be defined as its position $\boldsymbol{p}$ and orientation θ in an inertial frame, known as the *world frame* i.e. $\boldsymbol{q} = [\boldsymbol{p}^T \quad \theta]^T = [x \quad y \quad \theta]^T$, as depicted in Figure 43. The orientation of the mobile robot is defined as the angular difference between the world frame and a frame relative to the robot's chassis, known as the *robot frame*.

In these conditions, suppose that the robot has a certain linear velocity, which in the robot frame is $\boldsymbol{v} = [v_x \quad v_y]^T$, and a certain angular velocity ω. To obtain the displacement $\dot{\boldsymbol{q}}$ of the robot in the world frame from its velocities in the robot frame, a rotation matrix is applied, of the following form:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A9.1}$$

It then follows that:

$$\dot{\boldsymbol{q}} = R(\theta) \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{A9.2}$$

Equation (A9.2) is known as the *posture kinematic* model of the omnidirectional robot. This model has two significant properties: the rotation matrix $R(\theta)$is nonsingular for all values of θ, which simply means that any desired displacement in the world frame can be achieved by the omnidirectional robot (if it is within the capabilities of its actuators); also, $R(\theta)$ is block-diagonal, which implies that the posture kinematic model can be decoupled into its position and orientation components:

$$\begin{aligned} \dot{\boldsymbol{p}} &= B(\theta)\boldsymbol{v} \\ \dot{\theta} &= \omega \end{aligned} \tag{A9.3}$$

where $B(\theta)$ is defined as:

$$B(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{A9.4}$$

The ability to consider independent controls for position and orientation simplifies some of the motion control problems of omnidirectional robots, as seen in Chapter 3.

For some applications, it is advantageous to provide the robot's controls in the form of linear and angular accelerations rather than velocities. This type of "torque control" can be achieved by extending the posture kinematic model:

$$\begin{aligned} \dot{\boldsymbol{q}} &= R(\theta)\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \\ \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{\omega} \end{bmatrix} &= \begin{bmatrix} a_x \\ a_y \\ \alpha \end{bmatrix} \end{aligned} \tag{A9.5}$$

This model is known as the *posture dynamic* model of the omnidirectional robot. It is evident that, as before, it is possible to specify separate linear and angular acceleration control laws.

It is now necessary to determine the angular velocities and accelerations of the robot's actuators as a function of the applied controls. Consider the layout of an omnidirectional robot equipped with three Swedish wheels, as depicted in Figure 44. Suppose that the rollers attached to the Swedish wheels rotate around axes which are orthogonal to the axis of rotation of the main wheel (see Chapter 6). Then, assuming that no slippage occurs, each wheel has the following contribution to the motion of the mobile robot.

$$r\dot{\varphi}_i = \begin{bmatrix} -\sin\gamma_i & \cos\gamma_i & L \end{bmatrix}\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \quad , i = 1,2,3 \tag{A9.6}$$

where $\dot{\varphi}_i$ is the angular velocity of wheel i, γ_i is the angle between its axis of rotation and the *x*-axis of the robot frame, L is the distance between the geometric center of the robot and each wheel, and r is the wheel radius.

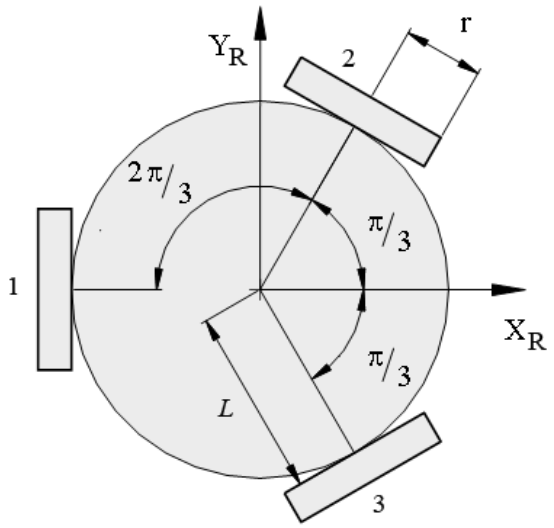


Figure 44: Basic layout of a three-wheeled omnidirectional robot.

Combining the contributions of each individual wheel, it results that:

$$r\begin{bmatrix}\dot{\varphi}_1\\ \dot{\varphi}_2\\ \dot{\varphi}_3\end{bmatrix}=\begin{bmatrix}0 & -1 & L\\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & L\\ \frac{\sqrt{3}}{2} & \frac{1}{2} & L\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ \omega\end{bmatrix}=J\begin{bmatrix}v_x\\ v_y\\ \omega\end{bmatrix} \tag{A9.7}$$

Identically, the relation between the accelerations of the robot's chassis and the angular accelerations of each wheel can be obtained by differentiating (A9.7):

$$r\begin{bmatrix}\ddot{\varphi}_1\\ \ddot{\varphi}_2\\ \ddot{\varphi}_3\end{bmatrix}=J\begin{bmatrix}a_x\\ a_y\\ \alpha\end{bmatrix} \tag{A9.8}$$

This relation is useful to identify situations where the desired controls (either in velocity or torque form) exceed the capabilities of the robot's actuators.