

Mechanisms and Machine Science

Giuseppe Carbone  
Fernando Gomez-Bravo *Editors*

# Motion and Operation Planning of Robotic Systems

Background and Practical Approaches

# **Mechanisms and Machine Science**

Volume 29

## **Series editor**

Marco Ceccarelli, Cassino, Italy

More information about this series at <http://www.springer.com/series/8779>

Giuseppe Carbone · Fernando Gomez-Bravo  
Editors

# Motion and Operation Planning of Robotic Systems

Background and Practical Approaches



Springer

*Editors*

Giuseppe Carbone  
University of Cassino  
Cassino, Frosinone  
Italy

Fernando Gomez-Bravo  
Engineering School  
University of Huelva  
La Rábida, Huelva  
Spain

ISSN 2211-0984  
Mechanisms and Machine Science  
ISBN 978-3-319-14704-8  
DOI 10.1007/978-3-319-14705-5

ISSN 2211-0992 (electronic)  
ISBN 978-3-319-14705-5 (eBook)

Library of Congress Control Number: 2015932073

Springer Cham Heidelberg New York Dordrecht London  
© Springer International Publishing Switzerland 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media  
([www.springer.com](http://www.springer.com))

# Preface

Robot motion planning and its applications have attracted the attention of the robotic community along the last decades. This book is an attempt to address this wide topic with a multidisciplinary approach. While other publications focus on describing the theoretical basis of robot motion, this work pays special attention to explain the fundamentals through real applications. Thus, it represents a perfect combination for studying this topic along with other theoretical books.

Each chapter has been authored by an expert or a team of experts in a specific area spanning from the mechanics of machinery to control theory, informatics, mechatronics. Chapters have been divided into five parts. The first one aims to give a theoretical background. Then, Parts II–V discuss the main specific issues for a proper path planning of different types of robots such as robotic manipulators, wheeled robots, legged robots, cooperation and coordination of multiple aerial or underwater robots.

This book project can be foreseen as a reference for young professionals/researchers to overview the most significant aspects in the field of path planning. Given the wideness of the topic, this book can be considered as a first edition and, as Editors, we shall be pleased to consider additional contents/suggestions for a future edition.

We wish to acknowledge all the authors and expert blind reviewers for their significant contributions to this project. Also acknowledged is the professional assistance by the staff of Springer Science+Business Media that have supported this project with their help and advice in the preparation of the book.

Last but not least we are indebted to our families. Without their patience and understanding it would not have been possible for us to work on this book.

January 2015

Giuseppe Carbone  
Fernando Gomez-Bravo

# Contents

## Part I Theoretical Background

<b>Path Planning and Trajectory Planning Algorithms:</b>	
<b>A General Overview . . . . .</b>	3
Alessandro Gasparetto, Paolo Boscaroli, Albano Lanzutti and Renato Vidoni	
<b>Off-Line and On-Line Trajectory Planning . . . . .</b>	29
Zvi Shiller	

<b>Open Architecture for Vision-Based Robot Motion Planning and Control . . . . .</b>	63
Theodor Borangiu, Florin Anton and Silvia Anton	

<b>Grasping and Manipulation of Unknown Objects Based on Visual and Tactile Feedback . . . . .</b>	91
Robert Haschke	

## Part II Motion Planning of Robotic Manipulators

<b>Obstacle Avoidance with Industrial Robots . . . . .</b>	113
T. Petrič, A. Gams, N. Likar and L. Žlajpah	

<b>Path Planning Kinematics Simulation of CNC Machine Tools Based on Parallel Manipulators . . . . .</b>	147
Luc Rolland	

<b>Planning Automatic Surgical Tasks for a Robot Assistant . . . . .</b>	193
Enrique Bauzano Nuñez, Belen Estebanez Campos, Isabel Garcia Morales and Victor F. Muñoz Martinez	

**Part III Motion and Operation Planning for Wheeled Robots**

<b>Motion Planning Using Fast Marching Squared Method . . . . .</b>	223
S. Garrido, L. Moreno and Javier V. Gómez	
<b>Motion Planning of Large Scale Vehicles for Remote Material Transportation . . . . .</b>	249
Alberto Vale and Isabel Ribeiro	
<b>Car-Like Robot Manoeuvre Generation . . . . .</b>	293
F. Gomez-Bravo	
<b>Vehicle Autonomy Using Cooperative Perception for Mobility-on-Demand Systems . . . . .</b>	331
Seong-Woo Kim, Tirthankar Bandyopadhyay, Baoxing Qin, Zhuang Jie Chong, Wei Liu, Xiaotong Shen, Scott Pendleton, James Guo Ming Fu, Marcelo H. Ang Jr., Emilio Frazzoli and Daniela Rus	
<b>Motion Planning of a Spherical Mobile Robot. . . . .</b>	361
Qiang Zhan	

**Part IV Motion Planning for Legged Robots**

<b>A Minimum Jerk-Impedance Controller for Planning Stable and Safe Walking Patterns of Biped Robots . . . . .</b>	385
Amira Aloulou and Olfa Boubaker	
<b>Online Walking Pattern Generation Using FFT for Humanoid Robots . . . . .</b>	417
Kenji Hashimoto, Hideki Kondo, Hun-Ok Lim and Atsuo Takanishi	
<b>Hexapod Walking Robot Locomotion . . . . .</b>	439
Franco Tedeschi and Giuseppe Carbone	

**Part V Robot Cooperation and Interaction**

<b>Distributed Cooperation of Multiple UAVs for Area Monitoring Missions . . . . .</b>	471
José J. Acevedo, Begoña C. Arrue, Iván Maza and Aníbal Ollero	

<b>Robotic Manipulation Within the Underwater Mission Planning Context</b> . . . . .	495
Javier Pérez, Jorge Sales, Antonio Peñalver, J. Javier Fernández, Pedro J. Sanz, Juan C. García, Jose V. Martí, Raul Marín and David Fornas	
<b>Erratum to: Motion and Operation Planning of Robotic Systems</b> . . . . .	E1
Giuseppe Carbone and Fernando Gomez-Bravo	

# **Part I**

## **Theoretical Background**

# Path Planning and Trajectory Planning Algorithms: A General Overview

Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti  
and Renato Vidoni

**Abstract** Path planning and trajectory planning are crucial issues in the field of Robotics and, more generally, in the field of Automation. Indeed, the trend for robots and automatic machines is to operate at increasingly high speed, in order to achieve shorter production times. The high operating speed may hinder the accuracy and repeatability of the robot motion, since extreme performances are required from the actuators and the control system. Therefore, particular care should be put in generating a trajectory that could be executed at high speed, but at the same time harmless for the robot, in terms of avoiding excessive accelerations of the actuators and vibrations of the mechanical structure. Such a trajectory is defined as *smooth*. For such reasons, path planning and trajectory planning algorithms assume an increasing significance in robotics. Path planning algorithms generate a geometric path, from an initial to a final point, passing through pre-defined via-points, either in the joint space or in the operating space of the robot, while trajectory planning algorithms take a given geometric path and endow it with the time information. Trajectory planning algorithms are crucial in Robotics, because defining the times of passage at the via-points influences not only the kinematic properties of the motion, but also the dynamic ones. Namely, the inertial forces (and torques), to which the robot is subjected, depend on the accelerations along the trajectory, while the vibrations of its mechanical structure are basically determined by the values of the jerk (i.e. the derivative of the acceleration). Path planning algorithms are usually divided according to the methodologies used to generate the geometric path, namely:

---

A. Gasparetto (✉) · P. Boscariol  
DIEGM – Dipartimento di Ingegneria Elettrica Gestionale E Meccanica,  
University of Udine, Via Delle Scienze, 206, 33100 Udine, UD, Italy  
e-mail: alessandro.gasparetto@uniud.it

A. Lanzutti  
MBP, Via Toscanini, 48/B, 46043 Castiglione Delle Stiviere, MN, Italy

R. Vidoni  
Faculty of Science and Technology,  
Free University of Bozen-Bolzano Piazza Università,  
39100 Bolzano, Italy

- *roadmap* techniques
- *cell decomposition* algorithms
- *artificial potential* methods.

The algorithms for trajectory planning are usually named by the function that is optimized, namely:

- *minimum time*
- *minimum energy*
- *minimum jerk*.

Examples of hybrid algorithms, which optimize more than a single function, are also found in the scientific literature. In this chapter, the general problem of path planning and trajectory planning will be addressed, and an extended overview of the algorithms belonging to the categories mentioned above will be carried out, with references to the numerous contributions to this field.

**Keywords** Path planning · Trajectory planning · Roadmap · Cell decomposition · Artificial potential · Minimum time · Minimum energy · Minimum jerk

## 1 Introduction

Human activity in many sectors is nowadays supported or substituted by robots, which range from standard robots for industrial applications to autonomous robots for complex tasks, such as space exploration. Indeed, the great versatility and flexibility of robots allows them to be employed in different sectors, to perform even very diverse tasks. Referring to the industrial environment, a robot can be defined [78] as a mechanical structure made of several rigid bodies (links) connected one to another by means of joints. Within the robot, it is possible to identify a structure that implements mobility, a wrist which provides dexterity, and an end-effector which performs the task given to the robot.

Regardless of the specific mechanical structure, in all types of applications a generic task is achieved by a robot by imposing a specific motion to the end-effector. This motion may be free or bound: the former case applies if the end-effector does not have a physical interaction with the environment, while the latter case applies if the end effector interacts with the environment by exchanging forces and/or torques.

The input of the control system of the robot is generally given by the law of motion, which is generated by a dedicated module for motion planning. Such motion planning module can operate off-line, by using a knowledge of the robot and the environment which is given *a priori*, or can operate on-line: in this case, suitable sensors must be employed to monitor the robot motion and enable the control system to adjust the movements in real time.

Ultimately, controlling the robot means determining the forces and torques that the actuators must develop at the joints, so as to ensure that the reference trajectories

are properly followed. However, this problem turns out to be very complex, because a robot is an articulated structure, so the motion of a single link arm affects the other links. Mathematically, this is expressed by the fact that the dynamic equations of a robot (with the exception of Cartesian structures), contain some terms due to the coupling effects between different links.

In most cases, robot controllers are based on closed loops, driven by the error between the reference and the actual position, which allows to achieve the accuracy required to the robot in executing the planned trajectory. In the case that, during a manipulation task, there is contact between the end-effector and the environment, the control problem is further complicated because not only the motion, but also the forces exchanged in the interaction should be monitored and controlled.

In this chapter, we will focus on the path planning and trajectory planning problems, which constitute the two main parts of the general motion planning problem. The interest for such topics is dramatically increasing, because operations at high speed are required to robots in the modern automatic systems; hence, smooth motions should be planned (where *smooth* means that such motions must avoid excessive values of accelerations of the actuators, as well as vibrations of the mechanical structure).

Many algorithms have been proposed, both for path planning and for trajectory planning, in the scientific literature of the robotic domain. The aim of this chapter is to provide a general overview of such algorithms, which have been subdivided into suitable categories.

## 2 Path Planning

Path planning is a merely geometric matter, because it is defined as the generation of a geometric path, with no mention of any specified time law. On the other hand, trajectory planning consists in assigning a time law to the geometric path. In most cases, path planning precedes trajectory planning; however, these two phases are not necessarily distinct; for instance, if point-to-point trajectories are considered (i.e. only the initial and final positions are specified), the two problems may be solved at the same time.

In this section the analysis of available works in literature deals with the case of systems without non-holonomic constraints.

Different types of paths are possible, depending on the specific case. For instance, for industrial manipulators, the standard path is usually defined by the geometry of the task, which is defined in a static way. In more advanced applications, or for robots operating in dynamic environments, some extra features, such as the need for automatic obstacle avoidance, may be added.

In applications of advanced robotics, the problem of path planning is definitely very challenging, especially for robots characterized by a large degree of autonomy or for robots that must operate in hostile environments (space, underwater, nuclear, military, etc.).

The definition of the path planning problem is very straightforward: “find a collision-free motion between an initial (start) and a final configuration (goal) within a specified environment”. The simplest situation is when the path is to be planned in a static and known environment; however, more generally, the path planning problem can be formulated for any robotic system subject to kinematic constraints, in a dynamic and unknown environment.

Much work can be found in the robotic literature, dealing with path planning. The first definitions and algorithms date back to the 1970s. In [57] a complete overview of the path planning techniques can be found. An overview of many techniques cited in this work can be found also in the classic book [23] or in the recent book [48]. Other useful reviews of path planning techniques are [49, 55].

Some basic definitions are needed to introduce the path planning problem, namely: the configuration space (*C-space*), the space of free configurations (*C-free*) and the obstacles’ representation in the C-space (*C-obs*).

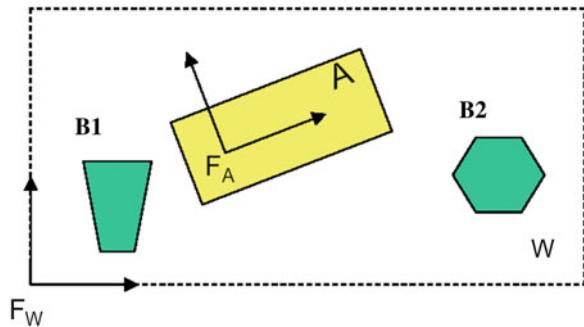
The configuration space is the space of all possible robot configurations, where a configuration  $q$  is the specification of position and orientation of the robot  $A$  with respect to a fixed reference frame  $F_W$ . Referring to Fig. 1, the C-space of the robot  $A$  is  $\mathbf{R}^3$ , since the configuration of  $A$  is specified by the origin of  $F_A$  with respect to  $F_W$ , and by its orientation.

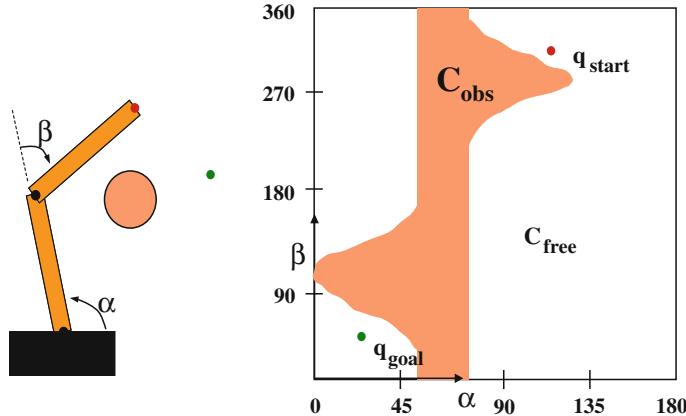
For an articulated robot (Fig. 2), the C-space is given by its joint space (in this case,  $\mathbf{R}^2$ ). The C-obs is given by the image of the obstacles in the C-space, and the C-free is defined as {C-space—C-obs}.

Path planning algorithms are usually divided in three categories, according to the methodologies used to generate the geometric path, namely:

- *roadmap* techniques
- *cell decomposition* algorithms
- *artificial potential* methods.

**Fig. 1** Mobile robot in a 2-dimensional space with obstacles





**Fig. 2** C-space, C-free and C-obs for an articulated robot with two joints

## 2.1 Roadmap Techniques

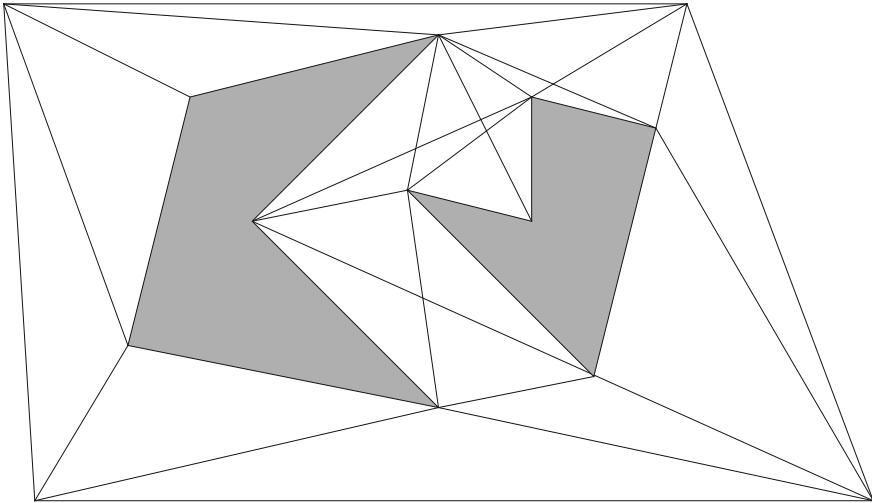
The roadmap techniques are based upon the reduction of the N-dimensional configuration space to a set of one-dimensional paths to search, possibly on a graph.

In other words, this approach maps the free space connectivity into a system of one-dimensional curves (the roadmap) in the C-free space or in its closure. The roadmap  $R$  thus obtained contains a set of paths: hence, the path planning consists in linking the initial and final configurations to  $R$ . In this way a feasible path between the two configurations is found.

It is very natural to associate a graph to the roadmap and to define some optimality index (e.g. the Euclidean length): the graph can then be searched in order to get the optimal solution to the path planning problem (in most cases, this is represented by the shortest path).

Figure 3 represents the so-called *visibility graph*, i.e. the graph whose nodes are the vertices of all the obstacles in the configuration space. Searching the graph would lead to get the shortest Euclidean path in the C-space. The nodes of the graphs indicate point locations, while edges represent visible connections between the nodes. Grey areas indicate obstacles to be avoided. The concept of visibility graph, which represents a milestone in the literature related to path planning, was introduced by Lozano-Pérez [63, 64].

Another kind of roadmap algorithms are those based on *Voronoi diagrams*, which are defined as a way to divide the space into regions having the following characteristic: given a set of points  $\{p_1, \dots, p_n\}$ , each point belonging to the  $i$ th region is closer to  $p_i$  than to any other  $p_j \neq p_i$ . This approach is dual to that based on the visibility graph, because the Voronoi diagrams enable one to obtain a path lying at the maximum distance from the obstacles, whereas the visibility graph generates a path that passes as close as possible to the obstacle vertices.



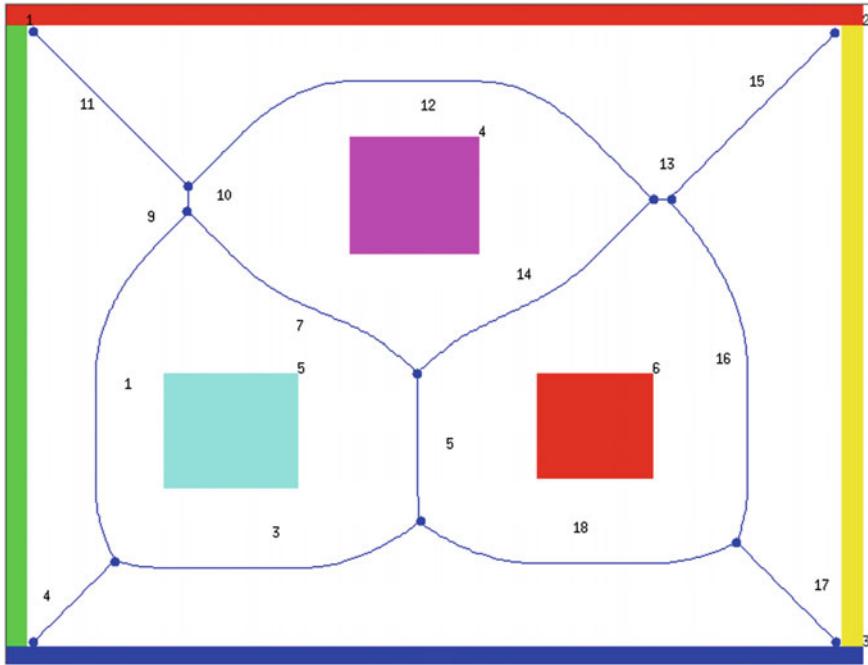
**Fig. 3** Visibility graph

Figure 4 shows some path generated by using Voronoi diagrams. The three squares in the diagram represents obstacles, while the blue lines are the set of points equidistant from at least two obstacles. Therefore the paths defined with this technique are designed to be as far away as possible from nearby obstacles. Examples of path planning algorithms may be found in [15, 35, 84].

## 2.2 Cell Decomposition Methods

According to the cell decomposition methods, the free space of the robot is subdivided into several regions, called cells, in such a way that a path between any two configurations lying in the same cell is straightforward to generate. It is then natural to define a so-called *connectivity graph*, which represents the adjacency relations between cells. Namely, the nodes of the graph represent the cells extracted from the free space, and there is an arch between two nodes are connected if and only if the corresponding cells are adjacent. The path planning problem is, again, turned into a graph searching problem, and can therefore be solved using graph-searching techniques.

Figure 5 illustrates the procedure described above, which is named *exact cell decomposition* technique, because the union of the cell represents exactly the free space. In some cases, an exact computation of the free space is not possible or convenient. *Approximate cell decomposition* methods must therefore be employed. Figure 6 shows how these techniques work:



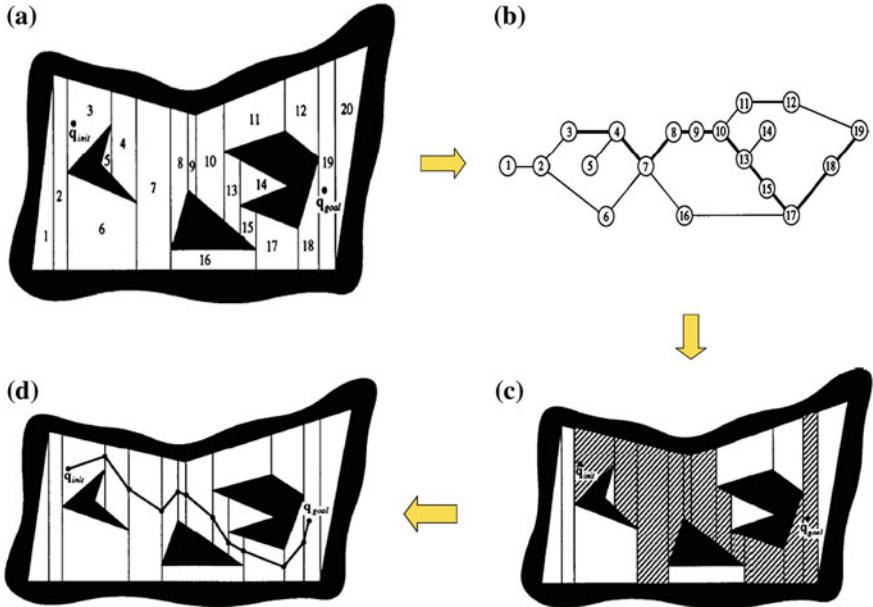
**Fig. 4** Paths resulting from Voronoi diagrams

- the whole C-space (assumed 2-dimensional) is divided into four cells;
- the algorithm checks if each cell is completely empty, completely full or mixed (such words obviously refer to the occupancy by the obstacles);
- each mixed cell is in turn divided into four subcells, and the algorithm is recursively applied to check the status of every subcell and recursively divide each mixed subcell into four sub-subcells.

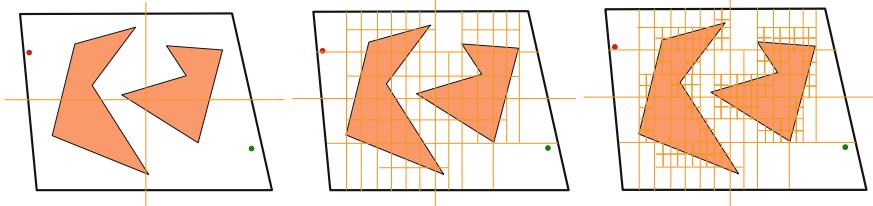
The graph that may be naturally associated to the approximate cell decomposition is a tree, named *quadtree* for 2-dimensional spaces (Fig. 7), *octree* for 3-dimensional spaces (Fig. 8), 16-tree for 4-dimensional spaces, and so forth.

### 2.3 Artificial Potential Methods

The artificial potential methodologies are a different approach to the path planning problem. The basic idea is to consider the robot in the configuration space as a moving point subject to a potential field generated by the goal configuration and the obstacles in the C-space: namely, the target configuration produces an attractive potential, while the obstacles generate a repulsive potential. The sum of these two contribution is the total potential, which can be seen as an artificial force applied



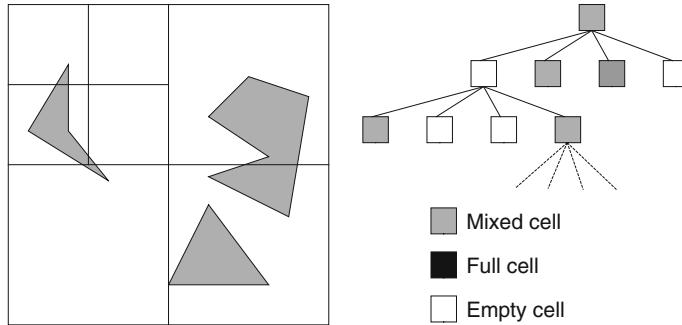
**Fig. 5** Exact cell decomposition: **a** subdivision of space into numbered polygons, **b** connectivity graph, **c** regions to be crossed, **d** path



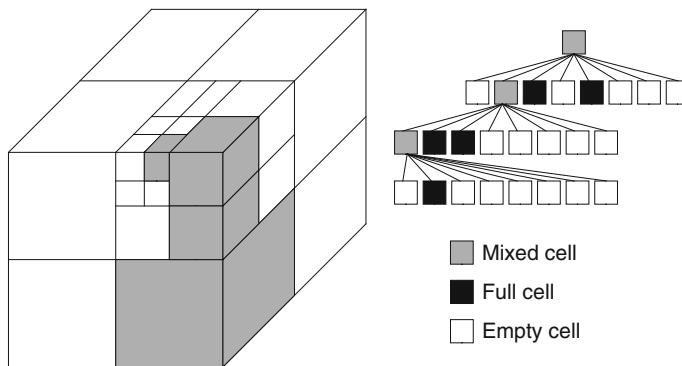
**Fig. 6** Approximate cell decomposition

to the robot, aimed at approaching the goal and avoiding the obstacles. Thus, given any configuration during the robot motion, the next configuration can be determined by the direction of the artificial force to which the robot is subjected. This normally represents the most promising direction of motion in terms of free path. An example of the application of the artificial potential method is shown in Fig. 9.

The artificial potential method was originally conceived by Khatib [50] and further developed by Volpe [91, 92]. Such a technique can find applications in many fields, because it can be successfully implemented online, thus moving the obstacle avoidance problem from the higher (and slower) level of path planners to the lower (and faster) level of online motion controllers. This implies that the good features of the artificial potential methods, especially the reactivity to environment changes,



**Fig. 7** Quadtree



**Fig. 8** Octree

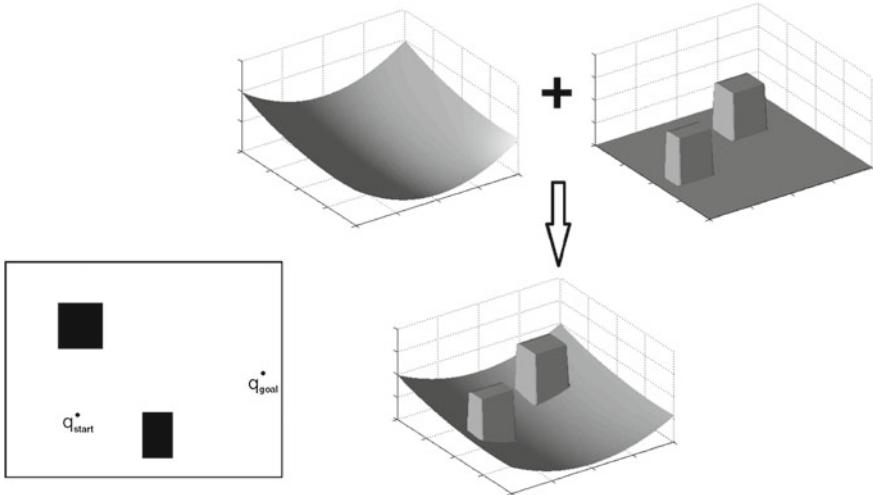
duly detected by the robot sensors, enable the robot controller to manage unexpected workspace changes in a fast way.

However, the artificial potential methods are intrinsically affected by a major problem, namely the presence of local minima, where the robot may find itself trapped. In order to overcome this problem, several solutions have been proposed: for instance, using potential functions which do not have local minima [25, 26, 51, 53]. Such functions are called *navigation functions*.

In [39, 42] alternative applications of the artificial potential method are presented.

Another approach to solve the path planning problem is found in [5], where a special kind of planners, named RPP (Random Path Planners), is proposed: local minima are avoided by combining the concepts of artificial potential field with random search techniques. Albeit with some limitations, RPP proved to be able to solve path planning problems for robots with a high number of degrees of freedom, with reasonable computation times.

Other examples of RPP can be found in [18–21].



**Fig. 9** The artificial potential method

## 2.4 Alternative Approaches to Path Planning

A possible alternative approach, which had remarkable results in very complex path planning problems, is given by the Probabilistic Roadmap Planners (PRM). It is a technique which employs probabilistic algorithms, such as random sampling, to build the roadmap. The most important advantage of PRM is that their complexity do not strictly depend on the complexity of the environment and on the dimension of the configurations space. The basic idea is to consider a graph where the nodes are given by a set of random configurations in the C-free. A local planner can then try to connect these configurations by means of a path: if a path is found, a new node is added to the graph. In this way the graph reflects the connectivity of the C-free. In order to find a path between two configurations, these configurations are added to the graph, then a graph search is performed in order to find a feasible path. Given the probabilistic nature of the algorithm, post-processing is often necessary to improve the quality of the path. PRM algorithms have been successfully applied to robotic manipulators with up to 16 degrees of freedom. Examples of PRM can be found in [1, 24, 45, 66].

There are some examples [29, 34] of path planners that take into account kinematic and dynamic constraints of the robot, in addition to the pure geometric problem of obstacles avoidance. This problem is referred to as kinodynamic motion planning. Kinodynamic and nonholonomic motion planning can be handled by the Rapidly-exploring Random Tree (RRT) method [58]. This method allows to search non-convex high-dimensional spaces by randomly building a space-filling tree.

Another important version of the general problem is given by path planning in presence of mobile obstacles. As it can be easily understood, this kind of problem

results very complex with respect to the basic version. This approach is used, for instance, in [32, 33].

A general overview of the path planning problem can be found in [54] and in [43], where the most important results achieved in the field of path planning, including PRM and RPP techniques, are reported. In [43] it is claimed that all the methodologies that have proven to be practically usable for path planning are based on a discretization of the configuration space. There are two crucial requirements in order to ensure an efficient implementation of path planning methodologies, namely: the efficiency of collision detection algorithms and the efficiency of graph searching techniques.

### 3 Trajectory Planning

Solving the trajectory planning problem means generating the reference inputs for the control system of the robot, so as to ensure that the desired motion is performed. Usually, the algorithm employed for trajectory planning takes as inputs the path generated by the path planner, as well as the kinematic and dynamic constraints of the robot. The output of the trajectory planning module is given by the trajectory of the joints, or of the end-effector, in form of a sequence of values of position, velocity and acceleration.

The geometric path is normally defined in the operating space of the robot, because the task to be performed, as well as the obstacles to avoid, are described in the operating space more naturally than in the joint space. Thus, planning the trajectory in the operative space means generating a sequence of values that specify the position and orientation that the end-effector of the robot must assume at every time interval. Planning the trajectory in the operating space is usually done when the motion follows a path with specific geometric characteristics defined in the operating space; in this case, the path can be specified in an exact form (i.e. taking the original path), or in an approximate form, by allocating some path points and connecting them by means of polynomial sequences. However, in most cases the trajectory is planned in the joint space of the robot because, since the control action on the manipulator is made on the joints, planning in the operating space requires a kinematic inversion to transform the end-effector position and orientation values into the joint values.

In order to plan a trajectory in the joint space, first a sequence of via-points should be extracted from the desired end-effector path, then a kinematic inversion is to be performed to get the corresponding values of the robot joints. The trajectory is then generated in the joint space by means of interpolation functions, taking into account the kinematic and dynamic limits imposed to the robot joints (in terms of position, velocity, acceleration and jerk). Normally, this way of planning the trajectory can also avoid the problems involved in moving near singular configurations, and can efficiently deal with the possible presence of redundant degrees of mobility. The main drawback of planning a trajectory in the joint space is given by the fact that the execution of a motion planned in the joint space is not so straightforward to predict in the operative space, due to the nonlinearities introduced by the direct

kinematics. However, no matter if the trajectory is planned in the operating space or in the joint space, it is crucial that the laws of motion resulting from the planning do not generate forces and torques at the joints that are not compatible with the given constraints: in this way the possibility of exciting mechanical resonance modes can be greatly reduced. For this reason, the planning algorithms must output smooth trajectories, i.e. trajectories represented by a curve whose derivatives are continuous up to a certain order. In particular, it is highly desirable to ensure the continuity of the accelerations of the joints, in order to get trajectories with a limited jerk, because limiting the jerk is crucial in order to reduce the vibrations induced to the robot (which may lead to considerable wear of the mechanical structure), as well as to avoid the excitation of the resonance frequencies of the robot. The vibrations caused by non-smooth trajectories may seriously damage the actuators and degrade the tracking performance of the trajectory. Furthermore, low-jerk trajectories can be executed faster and with a higher accuracy as demonstrated in [6]. In addition, there are some applications where abrupt motions can jeopardize the quality of the work or constitute a risk to the human operators working near the robot.

In order to classify the different trajectory planning methodologies into categories, it is useful to consider that a trajectory is usually planned after some optimality criterion has been set. The most significant optimality criteria that can be found in the literature are:

- minimum execution time;
- minimum energy (or actuator effort);
- minimum jerk.

In addition to the above, hybrid optimality criteria have been proposed, such as, for instance, time-energy optimal trajectory planning. With respect to the minimum energy criterion, a short clarification is necessary. In most of the cases related with trajectory planning, the term “energy” does not correspond to a physical quantity measured in Joules, but it is defined as the integral of squared torques: in other words, it measures the effort of the robot actuators. However, in the robotic literature it is possible to find also trajectory planning algorithms where the optimality index is “energy” in its strict meaning. Actually, this is not really a problem, because in the electric motors used on the robots, the torque can be assumed proportional to the current, so there is a correlation between the actuators’ effort and the energy required to the system.

### ***3.1 Minimum Execution Time Algorithms***

The optimality criterion based on minimum execution time was the first to be considered in trajectory planning, because short execution times are strictly related to high productivity in automatized production plants in industrial environments. Thus, no wonder that many papers can be found, in the robotic literature, proposing

trajectory planning algorithms aimed at minimizing the performance index given by the execution time.

The algorithms described in [7, 80] are defined in the position-velocity phase plane. The basic idea of these algorithms is to write the dynamic equation of manipulator in a parametric form using the curvilinear abscissa  $s$  of the path as the independent parameter. The curvilinear abscissa  $s$  (path parameter) and its derivative  $s'$  (pseudo-velocity) constitute the state of the system, while the second derivative of  $s$  (i.e. the pseudo-acceleration  $s''$ ) is chosen as the control variable. In this way, it is possible to transform the constraints given by the nonlinear robot dynamics, as well as the constraints on the actuators, into constraints on the control variable depending from the state of the system. For every point on the path, the maximum admissible value for the pseudo-velocity of the end-effector is determined from the constraints; it is then possible to build in the position-velocity phase plane (i.e. in the  $(s, s')$  plane), a velocity limit curve (VLC). The optimal trajectory is then computed by finding the admissible control that yields, for each point of the path, the maximum velocity that does not exceed the limit curve. The solution turns out to be in the form of a curve (named switching curve) in the phase plane.

An alternative approach to minimum time trajectory planning consists in using dynamic programming techniques, such as those described in [2, 81]. The basic idea is to take the state space and discretize it by building a grid of points (called state points). On the basis of the limits set on velocity, acceleration and jerk, it is possible to associate to each point the set of the subsequent admissible state points, and to define the cost of each possible solution by considering the time needed for the motion. This cost is defined by assuming a constant value of acceleration for each step. Finally, an algorithm based on dynamic programming generates the minimum time trajectory. Compared with the phase plane methods, the dynamic programming methods do not require the parameterization of the path and enables to choose an arbitrary performance index. Therefore, such algorithms may be used as a general technique for trajectory optimization. On the other hand, the phase plane approach turns out to be very efficient in terms of computational load; moreover, it may also be used for on-line trajectory planning, as in [28, 67].

A model-based approach is used to maximize the speed of industrial robots by obtaining the minimum-time trajectories that satisfy various constraints commonly given in the application of industrial robots in [52]. Conventional trajectory patterns, such as trapezoidal velocity profiles and cubic polynomial functions.

The algorithms described above produce trajectories with discontinuous accelerations and joint torques, because the dynamic models used consider the robot members as perfectly rigid and do not take into account the actuator dynamics. Neglecting the link flexibility and the actuator dynamics normally leads to some undesired effects. First, in reality the robot actuators cannot generate discontinuous torques: this causes the joint motion to be delayed with respect to the reference trajectory. This accuracy in trajectory following is thus greatly reduced, and the tracking controller has to be often activated during the execution of the trajectory. Moreover, each switching of the actuators may cause the so-called chatter phenomenon, i.e. high frequency oscillations inducing vibrations of the mechanical structure of the robot. This obviously

results in wearing of the mechanical components and in a decrease of the accuracy in trajectory following. Again, the tracking controller is activated more frequently and the actuators are further stressed. Another undesired effect resulting from an inaccurate model is that, since the time-optimal control requires saturation of at least one robot actuator at any time instant, it is impossible for the controller to correct the tracking errors arising from disturbances or modelling errors.

In [26, 27] a possible solution to these kind of problems is proposed: in these works, the phase plane method is used, together with a limitation set on the torque variations (actuator jerks). The proposed algorithm takes the pseudo-jerk, defined as the third derivative of the curvilinear abscissa, as the control variable: a dynamic equation of the third order is thus obtained. The experimental results presented in [26] show that, if some upper bound is set on the pseudo-jerk, time-optimal trajectories can be practically obtained by simply employing a conventional PID controller. This proves the correlation between accuracy in trajectory following and low values of jerk.

A different way to limit the torque variations is to consider in the objective function not only the execution time, but also an energy contribution: for instance in [79] the integral of squared torques along the whole trajectory is taken into account. The experimental results presented in [79] show that the increase of the overall motion time is compensated by a greater accuracy in trajectory following, even if conventional PD controllers are used. This results in a reduction of actuator stresses, with obvious advantages in the total lifetime of the electro-mechanical components of the robot.

It is possible to approach the problem of minimum-time trajectory planning by defining *a priori* the primitives of the motion, i.e. the curves that define the trajectory in the joint space. Such curves must be smooth functions, so that the control signals and, consequently, the torque signals at the actuators, result also smooth functions. The most common situation is that in which the path is specified using a limited number of via-points: the solution is then given by spline interpolation. In the literature, several methodologies are proposed to compute time-optimal trajectories for robot manipulators based on optimization of splines, whose order may be three (cubic splines) or higher. The main differences among these techniques are:

- the type of constraints considered (either kinematic or dynamic);
- the algorithm used to compute the optimal trajectory;
- the possibility to extend the optimization problem, by taking into account other optimization criteria, in addition to the minimum time.

The distinction based on the type of constraints can be considered the most important. It can be extended to any type of trajectory planning algorithm, so that the two categories of kinematic trajectory planning and dynamic trajectory planning can be defined. The kinematic trajectory planning algorithms take as their input upper (sometimes also lower) bounds on velocity, acceleration and jerk. In most cases such bounds are considered constant. The dynamic trajectory planning algorithms consider the dynamic model of the robot and define an optimization problem taking into account dynamic constraints, such as bounds on the actuator torques, or on

the actuator jerks, defined as the variation of the torques. In some cases kinematic constraint (typically the velocity) are also considered. Both approaches have pros and cons: the kinematic trajectory planning has its main advantage in the simplicity and in the lower computational load; on the other hand, the dynamic trajectory planning features a better capacity to use the robot actuators. In other words, the kinematic methods are based on a simplified computational model that yields a non-optimal use of the robot actuators, although in most cases reasonably good trajectories are planned. Dynamic methods are based on a more accurate model and therefore produce better solutions, but at the cost of a heavier computational load, since they have to deal with non-trivial issues, such as identification of the dynamic parameters of the robot, or the efficiency in implementing efficient algorithms to solve the robot dynamic equations.

An interesting example of an algorithm based on the inverse dynamic of a parallel robot is given by [17]. In this work, a multi-objective optimisation problem is formulated and a dedicated genetic algorithm is employed to find an optimal trajectory based upon spline functions.

Splines function are therefore used as trajectory primitives in order to ensure the continuity of the acceleration. Another example can be found in [59], where a nonlinear optimization problem is set, namely the computation of the value of the time intervals between the via-points, so as to minimize the total execution time of the trajectory subject to kinematic constraints. The technique is based upon an unconstrained optimization algorithm named FPS (Flexible Polyhedron Search), in combination with an algorithm called FSC (Feasible Solution Converter), which converts the solutions that are not physically feasible (i.e. that are not compatible with the kinematic constraints) into feasible ones, by implementing a suitable time scaling of the trajectory generated by the FPS algorithm. In [93], the same optimization algorithm presented in [59] is used, but instead of cubic splines, cubic B-splines are taken as primitives of motion.

The algorithms described above produce a local optimal solution, while other minimum-time trajectory planning methods output a global optimal solution. Piazzoli and Visioli use interval analysis to calculate a minimum-time trajectory subject to kinematic constraints at the joints. Such kinematic constraints are on the maximum value of velocity, acceleration and jerk. In [71] they extend the results already presented in [71, 72]. The simulations presented in [71] showed an improvement of 18 % of the total execution time with respect to the results yielded by a local optimization algorithm.

In [40, 41] a global optimization method is presented, which combines a stochastic technique, such as a genetic algorithm, with a deterministic procedure based on interval analysis. The proposed technique can be applied to solve general global optimization problems where semi-infinite constraints are defined. In [40] this algorithm is applied to the problem of minimum-time trajectory planning with specific kinematic and dynamic constraints: namely, the trajectories, represented by cubic splines, are subject to restrictions on the maximum actuator torques, as well as on the linear and angular velocities of the end-effector in the operating space. It is remarkable

that, differently from usual, in [40] the velocity constraint is not imposed in the joint space, but in the operating space of the end-effector.

A composition of polynomial functions of different orders are used in [11, 12] to obtain jerk continuity along a trajectory planned from a set of pre-defined via-points, obtaining a global minimum time solution.

Another example of minimum-time trajectory planning for robotic manipulators can be found in [16]. In this case the objective function is made of two terms: the first term takes the squared values of the optimization variables (i.e. of the time intervals between the via-points), while the second term is the sum of the squared accelerations computed at the via-points. The introduction of this second term has the effect of increasing the trajectory smoothness with respect to a pure minimum-time approach. The optimization is performed by using the DFP (Davidon-Fletcher-Powell) algorithm, which does not consider the kinematic bounds, therefore performing an unconstrained minimization. The solution obtained by means of the DFP algorithm is then subjected to a procedure of time-scaling, until the more restrictive kinematic bound has been saturated. The resulting trajectory, although respecting the limits on velocity, acceleration and jerk, is sub-optimal with respect to time.

In [30] a technique for determining time-optimal path-constrained trajectories subject to velocity, acceleration and jerk constraints, acting on both the robot actuators and on the task to be executed, is presented. The solution of the optimization problem is based upon a hybrid optimization strategy, which takes into account the path description, the kinematic model of the robot and constraints defined by the user. The resulting trajectories are optimal with respect to time, but not with respect to smoothness.

In the work [60] a combination of spline functions up to the seventh order are used together to achieve minimum time solutions with velocities, acceleration and jerk bounds. Other examples of minimum-time algorithms subject to kinematic constraints may be found in [31, 49, 85, 86, 89]. In [74] the minimum-time trajectory problem is solved under kinematic and dynamic constraints, i.e. torque, power, jerk and energy, taking into account both the robot dynamics and the obstacle presence.

### **3.2 Minimum Energy Algorithms**

As already remarked, the minimum-time trajectory planning algorithms received a lot of consideration in the robotic literature, mainly because of the strong industrial interest to reduce the length of the production cycles. However, the minimum-time optimization criterion is not the only one that can be considered: other criteria are definitely more suitable for different needs and requirements.

The trajectory planning based on energetic criteria is interesting under many aspects. On one hand, it generates smooth trajectories which are easier to track, and reduce the stresses induced to the actuators and to the mechanical structure of the robot. On the other hand, this optimization criterion enables one to better comply with energy saving requirements, which are driven not only by mere economic

considerations, but may be imposed by specific applications in which the energy source is limited by technical factors, such as robotic applications for outer space, for underwater exploration or for military tasks.

A classical example of minimum-energy trajectory planning algorithm is contained in [65], where a trajectory is optimized with respect to energy taking into account constraints on the motion of the end-effector, as well as the physical limits of the joints. The proposed objective function is the integral of squared torques. The trajectories are expressed by cubic B-splines and, by exploiting some property of the convex hull, it is possible to transform the joint limits into some limits set on the optimization parameter, which are the control points of the B-splines. The resulting motion thus minimizes the effort of the actuators.

In [2, 79] some techniques for optimal trajectories planning, with respect to energy and time, are described: the function to optimize is made of two terms, the first related to the execution time, the second related to the energy consumption. Such algorithms are intended to reduce the stresses of the actuators and to facilitate the trajectory tracking. In [79], the integral of the squared torques along the trajectory is considered in the objective function, while in [2] the function of total energy is considered.

Other examples of optimized trajectories, with respect to energy as well as to time, are presented in [75–77, 90, 95]. In [75] the Authors consider a trajectory parameterized by cubic splines, subject to kinematic constraints set on the maximum value of velocity, acceleration and jerk, and to dynamic constraint given by the maximum torque applicable to the joints. In [76] the same Authors consider a trajectory parameterized by cubic B-splines, where the physical limits of the joints are added to the torque and kinematic constraints. The objective function includes also an additional term (penalty function), in order to avoid mobile obstacles expressed as spherical or hyperspherical safety zones. In [77], two strategies for offline 3-dimensional optimal trajectory planning of industrial robots, in presence of fixed obstacles, are presented. In [90], a nonlinear change of variables is employed to convert the time-energy optimal trajectory planning problem into a convex control problem based on only one state variable. In [95], a methodology based on the minimization of an objective function which considers both the total execution time and the total energy spent along the whole trajectory is presented; the via-points of the trajectory are interpolated by means of cubic splines. Kinematic and dynamic constraints, in terms of upper bounds on velocity, acceleration, jerk and input forces and torques are also considered. It is worth noting that in algorithms such as the one presented in [79] the energy term is added in order to produce trajectories which result slower but smoother with respect to those generated by minimum-time trajectory planning algorithms; on the other hand, in approaches such as the one presented in [76] the objective function is primarily designed to minimize the energy and to plan trajectories with no regard to the execution time.

Recently, due to the development and installation of energy recovery and redistribution devices in robotic systems, the minimum-energy topic has gained new interest among the research community, e.g. [44, 68].

### 3.3 Minimum Jerk Algorithms

The importance of generating trajectories that do not impose discontinuities of the actuator torques at the robot joints has already been remarked; for instance, in [26] and in [27] this result is obtained by imposing upper bounds to the rate of change of the actuator torques. However, this kind of approach requires the computation of the third order dynamics of the robot.

An alternative method to obtain smooth profiles of the actuator torques is based on the idea of limiting the jerk, defined as the time derivative of the acceleration. Indeed, the torque variations depend upon the dominant term of the matrix of inertia multiplied by the vector of the joint jerk. Thus, some trajectory planning methods take the jerk as the variable to be minimized, in order to obtain smooth trajectories. The minimization of the jerk yields positive results, such as: reduction of the error during the trajectory tracking phase, reduction of the excitation of resonance frequencies, reduction of the stresses induced to the mechanical structure of the robot and to the actuators.

This results in a natural and coordinated motion: indeed, some studies suggest that the movements of the human arm satisfy an optimization criterion based upon the minimization of the jerk, or of the torque variations [82]. The minimum-jerk trajectory planning for robotic manipulators are an example of optimization based on physical criteria which mimic the human ability to produce natural movements [8].

In [56] the analytical solution of a trajectory planning problem for a point-to-point path, based on a minimum-jerk optimization criterion, is presented. The optimization, performed by applying Pontryagin's principle, involves two objective functions, namely: the maximum absolute value of jerk (minimax approach) and the time integral of the squared jerk.

In some cases, the total execution time of the trajectory is not imposed, so it can be chosen so as to comply with the kinematic limits on velocity and acceleration. However, most of the minimum-jerk algorithms that can be found in the robotic literature consider an execution time imposed *a priori*.

In [82], the integral of the squared jerk is minimized along the executed trajectory. In order to have a trajectory with a smooth start and stop, the values of velocity, acceleration and jerk are set to zero at the first and at the last via-points. The proposed algorithm is based upon a stochastic optimization technique performed by means of neural networks. The algorithm does not ensure the exact interpolation of intermediate nodes, but allows a tolerance, which can be set by tuning appropriate weights. This does not constitute a problem in cases where the exact interpolation is not needed, but just the passage in the neighbourhood of the via-points is required. The main limitation of this technique is that the resulting trajectories are not analytical functions, but are numerically defined.

Another approach is contained in [83], where the interpolation of the via-points is performed by means a trigonometric spline, thus ensuring the continuity of the jerk. The algorithm assumes that the time interval between the via-points is known and constant, and takes as input the values for the velocity, the acceleration and the

jerk, at the first and at the last via-points (such values are typically all set to zero). There are some advantages in using trigonometric splines to interpolate the trajectory via-points, for instance the property of locality: namely, if a via-point is changed, it is not necessary to recalculate the whole trajectory, but only the two splines that are connected to the via-point need to be recomputed. This property allows fast computation, thus making it possible to implement obstacle avoidance procedures in real time. The most significant aspect, in terms of trajectory optimization, is that parameterizing the trajectory allows some degrees of freedom, namely those given by the values of the first three derivatives (velocity, acceleration and jerk) at the intermediate via-points. Such values can be adjusted in order to minimize an objective function, such as the time integral of the squared jerk. The optimization presented in [83] is not bounded, since no kinematic limits are imposed, and yields a closed form solution, thus not requiring iterative minimization procedures.

In [70, 73] an algorithm based on interval analysis is presented. This technique seeks the minimum of the maximum absolute value of the jerk along a trajectory whose execution time is imposed *a priori*. It is therefore a so-called *minimax* approach bounded on the trajectory execution time. The trajectories primitives are cubic splines and the intervals between the via-points are computed, so as to obtain the lowest maximum absolute jerk value. In [70] the Authors present a comparison with the method based on trigonometric splines [83], reporting the highest values of the jerk, of the torques and of the torque variations. The simulation, which calculates the robot dynamics using the MatLab<sup>TM</sup> Robotics Toolbox, highlights the efficiency of the minimax algorithm with respect to other approaches.

### 3.4 Hybrid Optimization Approaches

Optimal trajectory planning with respect to time, energy and jerk has been discussed in the foregoing. Hybrid optimization approaches have also been proposed in the robotic literature. For instance, in order to get the advantages of the jerk reduction while executing fast trajectories, hybrid time-jerk optimal techniques are proposed, for instance [9, 11, 36–38, 46, 69]. These algorithms differ from the primitives used to interpolate the path, or from the optimization procedures implemented.

In [9, 11, 36–38] a minimum time-jerk trajectory planning technique is described, based upon two algorithms aimed at the minimization of an objective function, which is designed so as to ensure fastness in execution and smoothness of the trajectory at the same time. Such an objective function is composed of a term which is proportional to the total execution time and of a term which is proportional to the integral of the squared jerk along the path. The proposed algorithm enables one to define constraints on the robot motion before the execution of the trajectory. The constraints are expressed in form of upper bounds on the velocity, acceleration and jerk values

of all robot joints. In this way, any physical limitation of the real robot can be taken into account when planning the trajectory. Unlike most jerk-minimization methods, this technique does not ask for an *a priori* setting of the total execution time.

In [61, 62], the methodology is extended by taking into account also the power consumption of the actuators and physical limits of the joints. In this way, the technique becomes a time-jerk-energy planning algorithm.

Several objectives are taken into account in the work [51]: in particular minimum electrical and kinematic energy, minimum time and maximum maniuplability are obtained with the solution of a single optimization problem.

Minimum effort trajectories planned trough model-based approaches are presented in [10, 14]. The first one includes bounds on jerk, while second one has bounded joint speed. The work [13] introduces the novel topic of robustness in trajectory planning algorithms. Such approach allows to increase the tolerance of the resulting trajectory to the inevitable mismatches between the dynamic model used for the planning and the actual robot dynamics.

The problem of finding minimum time-effort trajectories for motor-driven parallel platform manipulators, subject to the constraints imposed by the kinematics and dynamics of the manipulator structure is the topic of the paper [21]. Computational efficiency is obtained trough a hybrid scheme comprising the particle swarm optimization method and the local conjugate gradient method. Also in [22] a constrained multi-objective genetic algorithm (MOGA) based technique is proposed to address this problem for a general motor-driven parallel kinematic manipulator. The planning process is composed of searching for a motion ensuring the accomplishment of the assigned task, minimizing the traverse time, and expended energy subject to various constraints imposed by the associated kinematics and dynamics of the manipulator.

All the trajectory planning methods introduced above are applicable to rigid link robot, with either serial or parallel kinematic configurations. However, it is worthwhile to mention that also cable-driven robots application are gaining a growing interest in robotics. Among the advantages brought by this class of manipulators, low overall mass and high stiffness make them very advantageous in many applications. On the other hand, the fact that they often require to use actuation redundancy and that they operation must avoid cable interference [94], has led to the development of trajectory planning algorithms specifically designed for them. The work [87] presents a method to compute trajectories for underconstrained parallel robot that ensures positive and bounded cable tension, while in [88] a similar procedure is also experimentally validated. A detailed study of the dynamics of cable-driven parallel robot is reported in [47], as a tool for developing accurate path planning algorithms. The time-optimality of trajectories designed for cable-driven robot is the topic covered in the works [3, 4].

## 4 Conclusions

In this paper, the fundamental problems of path planning and trajectory planning in Robotics have been addressed. An overview of the most significant methods, that can be found in the robotic literature to generate collision-free paths, has been presented. Then, the problem of finding an optimal trajectory given a planned path has been discussed and the most significant approaches have been described.

## References

1. Amato NM, Wu Y (1996) A randomized roadmap method for path and manipulation planning. In: Proceedings of the 1996 IEEE international conference on robotics and automation, pp 113–120
2. Balkan T (1998) A dynamic programming approach to optimal control of robotic manipulators. *Mech Res Commun* 25(2):225–230
3. Bamdad M (2013) Time-energy optimal trajectory planning of cable-suspended manipulators. *Cable-driven parallel robots*. Springer, Berlin, pp 41–51
4. Barnett E, Gosselin C (2013) Time-optimal trajectory planning of cable-driven parallel mechanisms for fully-specified paths with g1 discontinuities. In: ASME 2013 international design engineering technical conferences and computers and information in engineering conference. American Society of Mechanical Engineers
5. Barraquand J, Latombe JC (1991) Robot motion planning: a distributed representation approach. *Int J Robot Res* 10(6):628–649
6. Barre PJ, Bearee R, Borne P, Dumetz E (2005) Influence of a jerk controlled movement law on the vibratory behaviour of high-dynamics systems. *J Intell Robot Syst* 42(3):275–293
7. Bobrow JE, Dubowsky S, Gibson JS (1985) Time-optimal control of robotic manipulators along specified paths. *Int J Robot Res* 4(3):554–561
8. Bobrow JE, Martin BJ, Sohl G, Wang EC, Kim J (2001) Optimal robot motion for physical criteria. *J Robot Syst* 18(12):785–795
9. Boscariol P, Gasparetto A, Lanzutti A, Vidoni R, Zanotto V (2011) Experimental validation of minimum time-jerk algorithms for industrial robots. *J Intell Robot Syst* 64(2):197–219
10. Boscariol P, Gasparetto A (2013) Model-based trajectory planning for flexible link mechanisms with bounded jerk. *Robot Comput Integr Manuf* 29(4):90–99
11. Boscariol P, Gasparetto A, Vidoni R (2012) Jerk-continuous trajectories for cyclic tasks. In: Proceedings of the ASME 2012 international design engineering technical conferences (IDETC), pp 1–10
12. Boscariol P, Gasparetto A, Vidoni R (2012) Planning continuous-jerk trajectories for industrial manipulators. In: Proceedings of the ESDA 2012 11th biennial conference on engineering system design and analysis, pp 1–10
13. Boscariol P, Gasparetto A, Vidoni R (2013) Robust trajectory planning for flexible robots. In: Proceedings of the 2013 ECCOMAS multibody dynamics conference, pp 293–294
14. Boscariol P, Gasparetto A, Vidoni R, Romano A (2013) A model-based trajectory planning approach for flexible-link mechanisms. In: Proceedings of the ICM 2013—IEEE international conference on mechatronics, pp 1–6
15. Canny J, Donald B (1988) Simplified voronoi diagrams. *Discret Comput Geom* 3(1):219–236
16. Cao B, Dodds GI (1994) Time-optimal and smooth constrained path planning for robot manipulators. In: Proceedings of the 1994 IEEE international conference on robotics and automation, pp 1853–1858
17. Carbone G, Ceccarelli M, Oliveira PJ, Saramago SF, Carvalho JCM (2008) An optimum path planning for Cassino parallel manipulator by using inverse dynamics. *Robotica* 26(2):229–239

18. Caselli S, Reggiani M (2000) ERPP: an experience-based randomized path planner. In: Proceedings of the ICRA'00—IEEE international conference on robotics and automation, pp 1002–1008
19. Caselli S, Reggiani M, Rocchi R (2001) Heuristic methods for randomized path planning in potential fields. In: Proceedings of the 2001 IEEE international symposium on computational intelligence in robotics and automation, pp 426–431
20. Caselli S, Reggiani M, Sbravati R (2002) Parallel path planning with multiple evasion strategies. In: Proceedings of the ICRA'02—IEEE international conference on robotics and automation, pp 260–266
21. Chen CT, Liao TT (2011) A hybrid strategy for the time-and energy-efficient trajectory planning of parallel platform manipulators. *Robot Comput-Integr Manuf* 27(1):72–81
22. Chen CT, Pham HV (2012) Trajectory planning in parallel kinematic manipulators using a constrained multi-objective evolutionary algorithm. *Nonlinear Dyn* 67(2):1669–1681
23. Choset HM, Lynch KM, Hutchinson S, Kantor GA, Burgard W, Kavraki LE, Thrun S (2005) Principles of robot motion: theory, algorithms, and implementation. MIT Press, Cambridge
24. Clark CM, Rock S (2001) Randomized motion planning for groups of nonholonomic robots. In: Proceedings of the 6th international symposium on artificial intelligence, robotics and automation in space, pp 1–8
25. Connolly CI, Burns JB (1990) Path planning using Laplace's equation. In: Proceedings of the 1985 IEEE international conference on robotics and automation, pp 2102–2106
26. Constantinescu D (1998) Smooth time optimal trajectory planning for industrial manipulators. Ph.D. Thesis, The University of British Columbia, 1998
27. Constantinescu D, Croft EA (2000) Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *J Robot Syst* 17(5):233–249
28. Croft EA, Benhabib B, Fenton RG (1995) Near time-optimal robot motion planning for on-line applications. *J Robot Syst* 12(8):553–567
29. Donald BR, Xavier PG (1990) Provably good approximation algorithms for optimal kinodynamic planning for Cartesian robots and open chain manipulators. In: Proceedings of the sixth annual symposium on computational geometry, pp 290–300
30. Dong J, Ferreira PM, Stori JA (2007) Feed-rate optimization with jerk constraints for generating minimum-time trajectories. *Int J Mach Tools Manuf* 47(12–13):1941–1955
31. Dongmei X, Daokui Q, Fang X (2006) Path constrained time-optimal robot control. In: Proceedings of the international conference on robotics and biomimetics, pp 1095–1100
32. Fiorini P, Shiller Z (1996) Time optimal trajectory planning in dynamic environments. In: Proceedings of the 1996 IEEE international conference on robotics and automation, pp 1553–1558
33. Fraichard T (1999) Trajectory planning in a dynamic workspace: a state-time space approach. *Adv Robot* 13(1):74–94
34. Fraichard T, Laugier C (1993) Dynamic trajectory planning, path-velocity decomposition and adjacent paths. In: Proceedings of the 1993 international joint conference on artificial intelligence, pp 1592–1597
35. Garrido S, Moreno L, Lima PU (2011) Robot formation motion planning using fast marching. *Robot Auton Syst* 59(9):675–683
36. Gasparetto A, Zanotto V (2007) A new method for smooth trajectory planning of robot manipulators. *Mech Mach Theor* 42(4):455–471
37. Gasparetto A, Zanotto V (2008) A technique for time-jerk optimal planning of robot trajectories. *Robot Comput-Integr Manuf* 24(3):415–426
38. Gasparetto A, Lanzutti A, Vidoni R, Zanotto V (2012) Experimental validation and comparative analysis of optimal time-jerk algorithms for trajectory planning. *Robot Comput-Integr Manuf* 28(2):164–181
39. Ge SS, Cui YJ (2000) New potential functions for mobile robot path planning. *IEEE Trans Robot Autom* 16(5):615–620
40. Guarino Lo Bianco C (2001a) A semi-infinite optimization approach to optimal spline trajectory planning of mechanical manipulators. In: Goberna MA, Lopez MA (eds) Semi-infinite programming: recent advances. Springer, pp 271–297

41. Guarino Lo Bianco C, Piazzi A (2001b) A hybrid algorithm for infinitely constrained optimization. *Int J Syst Sci* 32(1):91–102
42. Guldner J, Utkin VI (1995) Sliding mode control for gradient tracking and robot navigation using artificial potential fields. *IEEE Trans Robot Autom* 11(2):247–254
43. Gupta K, Del Pobil AP (1998) Practical motion planning in robotics: current approaches and future directions. Wiley
44. Hansen C, Oltjen J, Meike D, Ortmaier T (2012) Enhanced approach for energy-efficient trajectory generation of industrial robots. In: Proceedings of the 2012 IEEE international conference on automation science and engineering (CASE 2012), pp 1–7
45. Hsu D, Kindel R, Latombe JC, Rock S (2002) Randomized kinodynamic motion planning with moving obstacles. *Int J Robot Res* 21(3):233–255
46. Huang P, Xu Y, Liang B (2006) Global minimum-jerk trajectory planning of space manipulator. *Int J Control, Autom Syst* 4(4):405–413
47. Ismail M, Samir L, Romdhane L (2013) Dynamic in path planning of a cable driven robot. Design and modeling of mechanical systems. Springer, Berlin, pp 11–18
48. Jing XJ (2008) Edited by. Motion planning, InTech
49. Kazemi M, Gupta K, Mehrandezh M (2010) Path-planning for visual servoing: a review and issues. Visual servoing via advanced numerical methods. Springer, London, pp 189–207
50. Khatib O (1985) Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings of the 1985 IEEE international conference on robotics and automation, pp 500–505
51. Kim JO, Khosla PK (1992) Real-time obstacle avoidance using harmonic potential functions. *IEEE Trans Robot Autom* 8(3):338–349
52. Kim J, Kim SR, Kim SJ, Kim DH (2010) A practical approach for minimum-time trajectory planning for industrial robots. *Ind Robot: Int J* 37(1):51–61
53. Koditschek DE (1992) Exact robot navigation using artificial potential functions. *IEEE Trans Robot Autom* 8(5):501–518
54. Kumar V, Zefran M, Ostrowski JP (1999) Motion planning and control of robots. In: Nof Shimon Y (ed) Handbook of industrial robotics, 2nd edn, vol 2. Wiley
55. Kunchev V, Jain L, Ivancevic V, Finn A (2006) Path planning and obstacle avoidance for autonomous mobile robots: a review. Knowledge-based intelligent information and engineering systems. Springer, Berlin, pp 537–544
56. Kyriakopoulos KJ, Saridis GN (1988) Minimum jerk path generation. In: Proceedings of the 1988 IEEE international conference on robotics and automation, pp 364–369
57. Latombe JC (1991) Robot motion planning. Kluwer
58. LaValle SM (2006) Planning algorithms. Cambridge University Press
59. Lin CS, Chang PR, Luh JYS (1983) Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Trans Autom Control* 28(12):1066–1073
60. Liu H, Lai X, Wu W (2013) Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints. *Robot Comput-Integr Manuf* 29(2):309–317
61. Lombai F, Szederkenyi G (2008) Trajectory tracking control of a 6-degree-of-freedom robot arm using nonlinear optimization. In: Proceedings of the 10th IEEE international workshop on advanced motion control, pp 655–660
62. Lombai F, Szederkenyi G (2009) Throwing motion generation using nonlinear optimization on a 6-degree-of-freedom robot manipulator. In: Proceedings of the 2009 IEEE international conference on mechatronics, pp 1–6
63. Lozano-Pérez T, Wesley MA (1979) An algorithm for planning collision-free paths among polyhedral obstacles. *Commun ACM* 22(10):560–570
64. Lozano-Perez T (1983) Spatial planning: a configuration space approach. *IEEE Trans Comput* 100(2):108–120
65. Martin BJ, Bobrow JE (1999) Minimum effort motions for open chain manipulators with task-dependent end-effector constraints. *Int J Robot Res* 18(2):213–224
66. Nissoux C, Simon T, Latombe JC (1999) Visibility based probabilistic roadmaps. In: Proceedings of the 1999 IEEE international conference on intelligent robots and systems, pp 1316–1321

67. Pardo-Castellote G, Cannon RH (1996) Proximate time-optimal algorithm for on-line path parameterization and modification. In: Proceedings of the 1996 IEEE international conference on robotics and automation, pp 1539–1546
68. Pellicciari M, Berselli G, Leali F, Vergnano A (2013) A method for reducing the energy consumption of pick-and-place industrial robots. *Mechatronics* 23(3):326–334
69. Petrinec K, Kovacic Z (2007) Trajectory planning algorithm based on the continuity of jerk. In: Proceedings of the 2007 Mediterranean conference on control and automation, pp 1–5
70. Piazz A, Vissoli A (2000) Global minimum-jerk trajectory planning of robot manipulators. *IEEE Trans Ind Electron* 47(1):140–149
71. Piazz A, Vissoli A (1997b) A cutting-plane algorithm for minimum-time trajectory planning of industrial robots. In: Proceedings of the 36th Conference on decision and control, pp 1216–1218
72. Piazz A, Vissoli A (1997a) A global optimization approach to trajectory planning for industrial robots, In: Proceedings of the 1997 IEEE-RSJ international conference on intelligent robots and systems, pp 1553–1559
73. Piazz A, Vissoli A (1997c) An interval algorithm for minimum-jerk trajectory planning of robot manipulators. In: Proceedings of the 36th Conference on decision and control, pp 1924–1927
74. Rubio F, Valero F, Sunyer J, Cuadrado J (2012) Optimal time trajectories for industrial robots with torque, power, jerk and energy consumed constraints. *Ind Robot Int J* 39(1):92–100
75. Saramago SFP, Steffen V Jr (1998) Optimization of the trajectory planning of robot manipulators taking into account the dynamics of the system. *Mech Mach Theory* 33(7):883–894
76. Saramago SFP, Steffen V Jr (2000) Optimal trajectory planning of robot manipulators in the presence of moving obstacles. *Mech Mach Theory* 35(8):1079–1094
77. Saravanan R, Ramabalan R, Balamurugan C (2009) Evolutionary multi-criteria trajectory modeling of industrial robots in the presence of obstacles. *Eng Appl Artif Intell* 22(2):329–342
78. Sciavicco L, Siciliano B, Villani L, Oriolo G (2009) *Robotics. Modelling, planning and control*. Springer, London
79. Shiller Z (1996) Time-energy optimal control of articulated systems with geometric path constraints. *J Dyn Syst Meas Control* 118:139–143
80. Shin KG, McKay ND (1985) Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Trans Autom Control* 30(6):531–541
81. Shin KG, McKay ND (1986) A Dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Trans Autom Control* 31(6):491–500
82. Simon D (1993) The application of neural networks to optimal robot trajectory planning. *Robot Auton Syst* 11(1):23–34
83. Simon D, Isik C (1993) A trigonometric trajectory generator for robotic arms. *Int J Control* 57(3):505–517
84. Takahashi O, Schilling RJ (1989) Motion planning in a plane using generalized Voronoi diagrams. *IEEE Trans Robot Autom* 5(2):143–150
85. Tangpattanakul P, Meesomboon A, Artrit P (2010) Optimal trajectory of robot manipulator using harmony search algorithms. *Recent advances in harmony search algorithm*. Springer, Berlin, pp 23–36
86. Tangpattanakul P, Artrit P (2009) Minimum-time trajectory of robot manipulator using harmony search algorithm. In: Proceedings of the IEEE 6th international conference on ECTI-CON 2009, pp 354–357
87. Trevisani A (2010) Underconstrained planar cable-direct-driven robots: a trajectory planning method ensuring positive and bounded cable tensions. *Mechatronics* 20(1):113–127
88. Trevisani A (2013) Experimental validation of a trajectory planning approach avoiding cable slackness and excessive tension in underconstrained translational planar cable-driven robots. *Cable-driven parallel robots*. Springer, Berlin, pp 23–29
89. Van Dijk NJM, Van de Wouw N, Nijmeijer H, Pancras WCM (2007) Path-constrained motion planning for robotics based on kinematic constraints. In: Proceedings of the ASME 2007 international design engineering technical conference and computers and information in engineering conference, pp 1–10

90. Verschueren D, Demeulenaere B, Swevers J, De Schutter J, Diehl M (2008) Time-energy optimal path tracking for robots: a numerically efficient optimization approach. In: Proceedings of the 10th international workshop on advanced motion control, pp 727–732
91. Volpe RA (1990) Real and artificial forces in the control of manipulators: theory and experiments. The Robotics Institute, Carnegie Mellon University, Pittsburgh, 1990
92. Volpe RA, Khosla PK (1990) Manipulator control with superquadric artificial potential functions: theory and experiments. *IEEE Trans Syst, Man, Cybern* 20(6):1423–1436
93. Wang CH, Horng JG (1990) Constrained minimum-time path planning for robot manipulators via virtual knots of the cubic B-spline functions. *IEEE Trans Autom Control* 35(5):573–577
94. Williams RL, Gallina P (2002) Planar cable-direct-driven robots: design for wrench exertion. *J Intell Robot Syst* 35(2):203–219
95. Xu H, Zhuang J, Wang S, Zhu Z (2009) Global time-energy optimal planning of robot trajectories. In: Proceedings of the international conference on mechatronics and automation, pp 4034–4039

# Off-Line and On-Line Trajectory Planning

Zvi Shiller

**Abstract** The basic problem of motion planning is to select a path, or trajectory, from a given initial state to a destination state, while avoiding collisions with known static and moving obstacles. Ideally, it is desirable that the trajectory to the goal be computed online, during motion, to allow the robot react to changes in the environment, to a moving target, and to errors encountered during motion. However, the inherent difficulty in solving this problem, which stems from the high dimensionality of the search space, the geometric and kinematic properties of the obstacles, the cost function to be optimized, and the robot's kinematic and dynamic model, may hinder a sufficiently fast solution to be computed online, given reasonable computational resources. As a result, existing work on motion planning can be classified into *off-line* and *on-line* planning. Off-line planners compute the entire path or trajectory to the goal before motion begins, whereas on-line planners generate the trajectory to the goal incrementally, during motion. This chapter reviews the main approaches to off-line and on-line planning, and presents one solution for each.

**Keywords** Motion planning · Trajectory optimization · Online planning

## 1 Introduction

One of the basic problems in robotics is that of motion planning, which attempts to move a robot from a given initial state to a destination state, while avoiding collisions with known static and moving obstacles. We distinguish between a *path* and a *trajectory*: a path italic represents a sequence of positions, defined in the robot's configuration space, which is the space of all positions, or configurations, that the robot can achieve [1]. A trajectory italic can be viewed as a path with a velocity profile along it, defined in the higher dimensional state space, where every point defines a position, or a configuration, and the velocity vector at that point. Thus, path

---

Z. Shiller (✉)

Department of Mechanical Engineering and Mechatronics, Ariel University, Ariel, Israel  
e-mail: shiller@ariel.ac.il

planning solves a geometric, or kinematic, problem, whereas trajectory planning solves a dynamic problem [1]. In this chapter, we will focus on trajectory italic planning.

Generally, it is desirable that the trajectory to the goal be computed online, during motion, to allow the robot to react to changes in the environment to a moving target, and to errors encountered during motion. However, the inherent difficulty in solving this problem, which stems from the high dimensionality of the search space, the geometric nature of the obstacles, the cost function to be optimized, and the robot's kinematic and dynamic model, prevents it from being solved sufficiently fast to be done online, given reasonable computational resources. As a result, two branches of research have emerged in the area of motion planning: off-line planning, where the trajectory to the goal is computed before motion begins, and on-line planning where the trajectory to the goal is computed incrementally during motion. We thus associate off-line with the computation of the entire trajectory to the goal, and on-line with incremental planning, regardless of the computational resources available for the planning process.

Another distinction between off-line and on-line planners is that the former may produce globally optimal solutions if the environment is fully known, whereas the latter is locally optimal at best. The challenges in off-line planning are therefore: optimality (local and global), completeness (will a solution be found if one exists), and overall computational complexity. The challenges in online planning are: completeness (is the planner guaranteed to reach the goal if a solution exists), computational complexity at each step, and optimality (how far is a solution from the optimal and is it bounded by an upper limit).

Off-line planners are most useful for repeatable tasks in static environments where optimality is essential, as is the case in many industrial applications. On-line planners are required in applications where the target states are determined on the fly, obstacles are discovered during motion, the environment is changing during motion, the computation time required for a global solution delays the task execution, or simply as an alternative to a computationally expensive off-line search [2].

The different nature of the two types of planners has resulted in distinct strategies to reaching the goal: the off-line planner takes generally a global view of the environment to select the optimal trajectory to the goal, whereas the on-line planner may select the next move based on a partial view of the environment. Both approaches were pursued at the early stages of the development of the field of motion planning in the early 80s [3–6]. The main focus then was on *path* planning, with the goal of computing the shortest path from start to goal in the presence of static obstacles [3, 4, 6].

Focusing on the shortest path resulted in geometric planners that account for the geometry of the moving object and obstacles. While the computation of an obstacle-free path may solve many important problems in industrial settings, where the robot may move slowly, it is insufficient, and almost useless, when the robot needs to move at reasonably high speeds, such as mobile robots moving through cluttered environments, and autonomous vehicles negotiating freeway traffic. Furthermore, the computed geometric path is insufficient to move the robot along the path unless

some speed profile along the path is specified. Selecting the speed profile that can be followed without the robot deviating from the path requires knowledge of the robot dynamic behavior. This brings to focus the problem of trajectory planning, which was addressed using tools rooted in optimal control theory. The two fields of research, path planning and trajectory planning, were developed in parallel over the years until recently when geometric planners were extended to searching for trajectories in the state space [7, 8]. While the focus in this chapter is off-line and on-line trajectory planning, we begin the literature review with geometric planners.

## 1.1 Geometric Planners

The introduction of the configuration space as the basic geometric motion planning tool [4, 9, 10] reduced the search for an obstacle-free path to computing a continuous path for a point from start to goal that avoids the forbidden regions representing the physical static obstacles. Being a geometric problem, most off-line planners are based on a geometric representation of the environment, through which a global search produces the shortest path to the goal. The geometric representations may consist of roadmaps or graphs that capture the topology of the free-space, generated e.g. by a Voronoi diagram, a visibility graph, a tangent graph [11, 12], or by cell decomposition [13]. Although each representation differs in the way it represents the free space, they all consist of a connected network of path segments that can be traversed from start to goal. The main computational effort in these planners is the representation of the free-space. This includes mapping of obstacles to the configuration space and the initial construction of the roadmap. Once the roadmap was constructed, the search for the shortest path is done using standard graph search techniques such as Dijkstra's search [14] or A\* [15]. The remaining difficulties stem from the dimensionality of the search space and the number of edges (segments) in the roadmap. The main computational effort here is the construction of the roadmap.

An alternative approach to constructing roadmaps is to overlay a uniform grid over the search space and represent the entire space by an undirected graph [2]. Assigning high costs to edges that intersect obstacles, effectively separates between inaccessible nodes and nodes in the free space. As a result, this, like all approaches that are based on a discrete representation of the search space, is resolution complete, implying that at low grid resolutions one may miss paths that pass through tight spaces between obstacles. Increasing graph resolution would severely impact the computational complexity. Compared to the roadmap-based algorithms, the number of nodes for the uniform grid representation is much greater. However, this representation, which is quite general, is applicable to problems where obstacles are not clearly defined, such as for mobile robots moving over rough terrain [16], as is demonstrated later in this chapter.

The increased interest in solving high dimensional problems, such as motion planning for humanoids or multi degrees-of-freedom arms, gave rise to a class of

sampling-based planners [17]. The most popular version of sampling-based planners is based on rapidly exploring random trees (RRT) [17–19]. They search the way to the goal by probing the configuration space (can be also done in the workspace) and incrementally expanding a collision-free tree from an initial configuration. Because the entire search is done “in the dark,” the planner attempts to reach unexplored parts of the search space, resulting eventually in a uniform coverage of the free space. The efficiency of these planners stems from the incomplete coverage of the free space and from terminating the search when the goal is first reached. The solution found is feasible but not optimal in any way. RRT based planners may not produce optimal solutions even when exploring the entire search space [7, 20]. In addition, they inherently have difficulties with tight spaces. Nevertheless, the RRT algorithms were demonstrated for solving very complex problems [21].

An extension called RRT\* was developed to asymptotically produce the optimal solution [7]. The asymptotic optimality was achieved by adding a lower bound estimate to the RRT search. Optimality is achieved iteratively at a great computational cost by running the algorithm repeatedly while refining the solution until either exhausting the available computation time or reaching a desired level of optimality [7]. Despite the great promise, the RRT\* algorithm was demonstrated in [7, 20] for the kinematic avoidance of very few planar and widely spaced obstacles, producing a smooth near-optimal solution after a large number of iterations (10,000). Solving a dynamic problem in a higher dimensional state space is expected to be much more challenging. Recently, a path generated by an RRT search was further optimized using a genetic algorithm to produce the shortest path for a hybrid manipulator with six degrees-of-freedom [22]. The idea of further optimizing paths that were generated by a geometric planner is similar in nature to the off-line planner presented here, except that the objective of the off-line planner is to produce a global optimal trajectory, not only the shortest path.

The sampling-based planners represent a paradigm shift in the motion planning community by (1) accepting probabilistic completeness, which is to say that the goal may not be reached in a finite time, (2) accepting any solution, not necessarily the optimal, and (3) abandoning the explicit geometric representation of the free configuration space in terms of roadmaps or graphs. This is a significant departure from the previous practices that evaluated motion planning algorithms for completeness and optimality.

## 1.2 Trajectory Planning (Off-Line)

The trajectory planning problem concerns the computation of robot motions that move the robot between two given states, while avoiding collision with obstacles, satisfying robot dynamics and actuator constraints, and usually minimizing some cost function, such as energy or time.

Early work on trajectory planning, from the 1960s to the 1980s, was rooted in the field of optimal control theory, which provides powerful tools to characterize and

generate optimal trajectories when high speed motion is desired [23]. The elegant necessary conditions, stated by the Pontryagin's maximum principle, lead to the formulation of the optimal control problem as a two-point boundary value problem, and the development of algorithms that searched for the optimal control that generates the optimal trajectories [23]. For time optimization problems, it was shown that the time-optimal control is bang-bang. This in turn reduces the optimal control problem to a parameter optimization by iterating on the switching times between the maximal controls [24, 25].

The first attempt to use these theories for robotics was by Khan and Roth [26], who computed the multi-axis time optimal trajectory for a linearized model of robot dynamics. Solving this problem for the full robot's dynamic model was computationally very difficult. The typically non-linear and coupled robot dynamics makes such solutions computationally extensive. Adding obstacles makes the computational challenge even harder.

One approach to reducing the complexity of the problem and facilitating a practical realization of time-optimal motion planning is to decouple the problem by representing robot motions by a path and a velocity profile along the path. This decoupling allows reducing the trajectory planning problem to two smaller problems: (a) computing the optimal velocity profile along a given path, and (b) searching for the optimal path in the  $n$ -dimensional configuration space.

The time optimal velocity profile along a specified path is computed using an efficient algorithm, originally developed Bobrow, Shin and McCay and Pfeifer [27–29], and later improved by Shiller and Lu [30] and Slotine and Yang [31]. Assuming a second order system, the solution to this problem was found to be bang-bang in the acceleration, that is, applying the maximum or minimum acceleration so as to maximize the velocity along the path. The switching times are computed efficiently to avoid crossing the velocity limit curve, which reflects the actuator constraints and the robot dynamics. This approach was later extended to computing time optimal velocity profiles along specified paths for nonlinear third order systems, subject to general jerk constraints [32].

The optimal path was computed using a nonlinear parameter optimization over path parameters, such as the control points of a cubic B spline [33, 34]. In each iteration of the optimization process, the optimal velocity profile along the path is efficiently computed to produce the minimum time for that iteration. One advantage of this approach is that each iteration yields a feasible trajectory, albeit not necessarily optimal. The optimization can therefore be terminated at any time to yield an acceptable solution.

A similar approach, known as *direct optimization*, *differential inclusion* [35], and *inverse dynamics optimization* [36], was proposed by the aerospace community. Common to these methods is the direct search for the optimal trajectory as opposed to a search for the optimal control in the higher dimensional state and co-state spaces [23]. Attempts to solve the multi-axis problem using graph search techniques in the state space, solving the so called “kinodynamic” problem, did not yield practical solutions [37, 38].

### 1.3 Online Planning

Early on-line planners were developed to address the lack of apriori information about the environment. Called “sensor-based” algorithms, they navigate a point robot equipped with position and touch sensors among unknown obstacles to reach a global goal. A series of “bug” algorithms were developed, starting with the basic bug that navigates by circumventing the detected obstacle always clockwise or counterclockwise until reaching the straight line to the goal, then continuing along that line until either reaching the goal or hitting another obstacle [5]. Assuming long range vision sensors, the bug strategies were extended to the Tangent Bug algorithm, which follows the tangent line to the next obstacle that obstructs the straight line to the goal [39, 40]. It was shown that complete online navigation can be achieved with only a finite amount of memory [5, 41, 42].

Another approach to online motion planning is based on potential functions [43–45]. Representing the goal with an attractive potential, and the obstacles with repulsive potentials, the path is generated online by following the negative gradient of the potential function. While this approach is computationally efficient and is suitable for on-line feedback control, it suffers from local minima, which may cause the path to terminate at a point other than the goal. This problem was overcome using harmonic potentials [43] and navigation functions [45]. These potentials, however, address only the obstacle avoidance problem with no concern for path optimality. Furthermore, the generation of the potential function is done off-line and may be time consuming.

A similar approach generates the shortest path by following the direction of steepest descent of a discretized distance function [46]. The main computational effort is in numerically computing the distance function, which is done off-line. The computation complexity increases rapidly with the number of obstacles and with grid resolution.

The potential functions used to guide the trajectory towards the goal resemble the value function, which is the solution to the Hamilton- Jacobi-Bellman (HJB) equation [47–49]. The HJB equation states a sufficient condition of global optimality (unlike the Pontryagin Maximum Principle, which is only a necessary condition), and the value function represents the cost-to-go from any feasible state. The globally optimal trajectory is then generated by selecting the controls that minimize the time derivative of the value function. For time invariant systems, this amounts to following the negative gradient of the value function, which drives the system time-optimally to the goal from any initial state. This is similar to the potential field method, except that the value function may be regarded as the “optimal” potential function.

Although the theoretical framework exists for deriving optimal feedback controllers, it is impractical to derive a time-optimal control law, using the HJB equation, for a typical obstacle avoidance problem that accounts for robot’s dynamics.

A recently developed online algorithm navigates towards the goal by optimally avoiding one obstacle at a time [50, 51]. This transforms the multi-obstacle problem with  $m$  obstacles to  $m$  simpler sub-problems with one obstacle each, thus reducing the size of the problem from exponential to linear in the number of obstacles.

The incremental generation of the trajectories and the relatively low computational effort at each step make this algorithm an efficient on-line alternative to the computationally expensive off-line planning, thus trading optimality for efficiency. This algorithm will be later discussed in this chapter.

This chapter is organized as follows: it starts with a formal problem statement of the motion planning problem, focusing on trajectory planning rather than on path planning. It continues with the theoretical solution for the optimization problem using the Hamilton Jacobi equation. It then describes an efficient off-line planner and a very efficient online planner. Both algorithms are demonstrated for a point robot moving at high speeds over rough terrain (the off-line planner) and through very cluttered environments (the online planner).

## 2 Problem Statement

In a typical motion planning problem, we wish to solve the following optimization problem:

$$\min_u \int_0^{t_f} L(x, u) dt \quad (1)$$

subject to the system dynamics:

$$\dot{x} = f(x, u), \quad (2)$$

where  $x \in \mathbb{R}^n$  is a point in the robots state space, and  $u \in \mathbb{R}^m$  is a vector of actuator efforts, subject to the actuator constraints:

$$u_{imin} \leq u_i \leq u_{imax}, i \in \{1, \dots, m\}, \quad (3)$$

obstacle constraints:

$$g(x) \geq 0; g \in \mathbb{R}^k, \quad (4)$$

and the boundary conditions:

$$x(0) = x_0; x(t_f) = x_f, \quad (5)$$

where  $k$  is the number of obstacles and  $t_f$  is the final time. If the objective function is time, i.e.  $L(x, u) = 1$ , then  $t_f$  is free. We assume that the obstacles (4) do not overlap with each other and with the goal  $x_f$ .

Problem (1) is a two point boundary value (TPBV) problem: of all trajectories that satisfy the boundary conditions (5), select the one that minimizes the cost function (1) and satisfies system dynamics (2), control constraints (3) and obstacle constraints (4).

The global optimal trajectory can be computed using the Hamilton-Jacobi-Bellman (HJB) equation, which states a sufficient condition for global optimality [47–49]. Denoting the set of obstacles as  $O$ :

$$O = \{x : g(x) < 0\}, \quad (6)$$

The control  $u^*$  that is the solution to problem (1), satisfies, on  $\mathbb{R}^n - \{x_0\} - O$ , the HJB equation:

$$\min_u \{v_t(x, t) + \langle v_x(x, t), f(x, u) \rangle\} = -L(x, u) \quad (7)$$

subject to (3) and (4), where  $v(x, t)$  is a  $C^2$  scalar function, satisfying

$$v(x_0, t) = 0 \quad (8)$$

$$v(x, t) > 0, x \notin x_0 \quad (9)$$

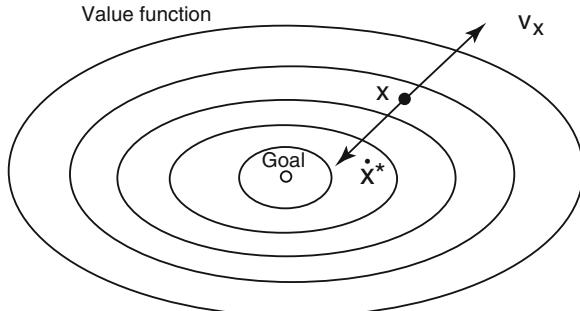
The subscripts  $x$  and  $t$  represent partial derivatives with respect to  $x$  and  $t$ , respectively, and  $\langle \cdot, \cdot \rangle$  denotes the inner product on  $\mathbb{R}^n$ .

The scalar function  $v(x, t)$  is the *value* function [47, 48, 52], representing the minimum *cost-to-go* to the origin (goal) from any given state. For an autonomous system (time-invariant) and for fixed boundary conditions,  $v_t = 0$ ; assuming in addition that the cost function to be minimized is time ( $L(x, u) = 1$ ), reduces (7) to:

$$\min_u \{\langle v_x(x), f(x, u) \rangle\} = -1 \quad (10)$$

To satisfy (10), the projection of  $\dot{x} = f(x, u)$  on  $v_x(x)$  must equal  $-1$ . It follows that the optimal control  $u^*$  that minimizes (10) drives the optimal trajectory  $\dot{x}^*(x, u)$  in the direction of the negative gradient,  $-v_x(x)$ , of the value function, as shown schematically in Fig. 1. This is similar to the trajectory generation by potential field methods [43–45, 53], except that here the potential function is the value function. Since the value function has a unique minimum at the goal, trajectories generated by following the negative gradient of the value function are globally optimal and are guaranteed to reach the goal from any initial state.

**Fig. 1** The optimal trajectory  $\dot{x}^*(x, u^*)$  slides opposite to the gradient  $v_x(x)$  of the value function



The on-line planner for the multi-obstacle avoidance problem, described later in this chapter, can be viewed in the context of the value function as following the negative gradient of an approximate value function for this problem. It generates near-optimal trajectories by avoiding obstacles *one at a time*, or equivalently, by sequentially following the negative gradient of the return function for each obstacle avoidance problem. The trajectory is generated incrementally, permitting robot motion before the entire trajectory to the goal has been computed.

Obtaining an analytical expression for the value function is practically impossible for other than for very simple cases. Computing the value function numerically would require solving the optimization problem from every point in the state space. This is essentially the approach used in [46] for solving the shortest path problem.

A discrete version of the HJB equation is the basis for the Bellman's Principle of Optimality and Dynamic Programming [54]. Dynamic programming is the optimization method used in most grid based optimizations, including the off-line optimization discussed next in this chapter.

### 3 Off-Line Planner

The off-line planner presented here computes the global time optimal trajectory between given boundary states in the presence of known static obstacles [2]. It combines a grid search in the configuration space with a continuous local optimization. In lieu of an expensive search in the  $2n$  dimensional state space for one (globally optimal) trajectory, this planner searches for many paths in the  $n$  dimensional configuration space for an  $n$  degree of freedom robot. The reduction of the search to the configuration space yields a significant (exponential) computational gain compared to a full search in the state space. The complexity of this approach is exponential in the dimension of the configuration space and linear in the number of nodes in the graph.

#### 3.1 Summary of the Approach

This planner is based on a branch-and-bound search for the global optimal trajectory between given end states in a static environment. It assumes an efficient mapping from a curve in the configuration space to the optimal traversal time along that curve. This mapping allows us to search for the optimal trajectory in the lower dimensional configuration space. We call the projection of the optimal trajectory on the configuration space the *optimal path*.

The branch-and-bound search begins by reducing the infinite set of paths between given end points to a final set by representing the configuration space by an undirected graph. The branch-and bound search then reduces this set to a small set of the most promising paths. The paths in the final set are then pruned to retain the best path in each path-neighborhood. These paths are then optimized using a nonlinear parameter

optimization to further reduce motion time. This last step significantly relaxes the grid resolution required for the initial search to ensure global optimality.

This process was proven to generate the global optimal trajectory in addition to producing a set of local minima [2]. The optimality of the solution depends on the number of paths selected in the first step, grid resolution with respect to the distance between obstacles, and the fidelity of the local optimization. This optimization was demonstrated for a six DOF manipulator moving in a cluttered environment [2] and for a mobile robot moving on general terrain [16].

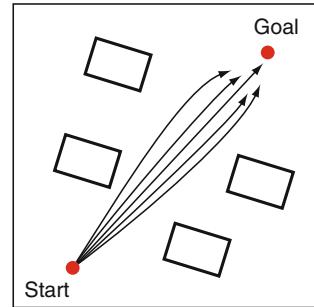
### 3.2 The Graph Search

The purpose of the graph search is to efficiently produce a set of paths that explore all regions in the configuration space and that may contain the optimal path. Obstacles are accounted for by setting high costs to edges that penetrate obstacles. For motion over rough terrain, obstacles are accounted for by considering their geometric shape and determining if the robot can safely traverse these obstacles, similarly to traversing other terrain features [16].

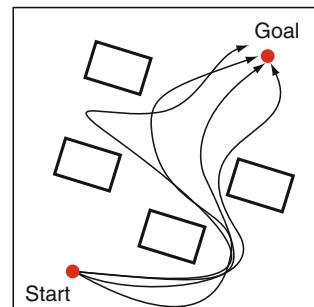
In the context of this algorithm, the optimal path is the one that can be traversed at the minimum time between given end points, subject to robot dynamics, and to control and obstacles constraints. Since metrics measured in the configuration space are not good predictors for path optimality, it is necessary to consider a large number of paths to ensure that they contain at least one path in the neighborhood of the optimal path. By representing the configuration space with a uniform grid, we reduce the infinite number of obstacle-free paths to a finite set.

One approach to generating a large set of paths, using a graph search, is to use the  $k$ -best search by Dreyfus [55] to produce a set of shortest paths. It is similar to a shortest path search except that it effectively excludes the  $k - 1$  best paths from the searched space while searching for the next  $k$ th best path. This allows us to sequentially generate the paths until some upper bound on the cost function, determined by the branch and bound search, is reached. The cost function may be path length, or some other function that produces a lower bound estimate of the optimal motion time along the path [2]. While this approach guarantees that the global and a few local minima (within grid resolution) are found, it has the drawback that it first generates a large number of paths in the neighborhood of the best path ( $k = 1$ ), usually in one homotopy class, before exploring other homotopy classes, as shown schematically in Fig. 2. A homotopy class contains all paths that can be continuously deformed into one another [1], as shown schematically in Fig. 3. Depending on grid resolution, using the  $k$ -best search may require a very large number of paths in order to cover the entire space. In addition, identifying a local minimum (that is not global optimal) is quite tedious. The difficulty arises from the regions of optimality not being easily quantified, and hence requiring that each new path be tested if it is in the neighborhood of any path generated so far. The large number of paths required by this approach thus imposes a high computational cost, first with the  $k$ -best search, which is linear in  $k$ , and then in the pruning process, which is  $O(k \log k)$ .

**Fig. 2** Near shortest paths found by the K shortest path algorithm tend to group around the shortest path



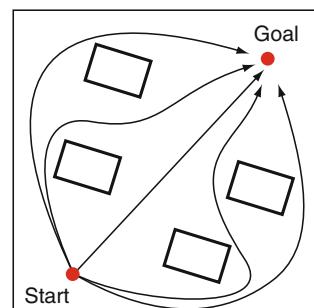
**Fig. 3** Paths in one homotopy class



The pruning process consists of selecting the best (shortest in time or distance) of all paths in the initial set of paths, then discarding all paths that are within some tube of a predefined diameter around the best path. The paths within a tube around the next best path are similarly discarded, and the process repeats until all paths in the initial set are either discarded or retained as the best path in their neighborhood. The pruning process thus reduces an initially large set of paths to a smaller set of promising paths, each is then locally optimized, as discussed later.

An alternative approach efficiently generates a large number of paths that cover the entire search space [56, 57] and can be easily reduced to the most promising path in each homotopy class, as shown schematically in Fig. 4. In two steps, each consisting of a shortest path search, it generates all shortest paths that pass through

**Fig. 4** Paths of various homotopy classes



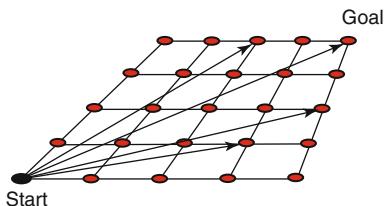
each node in the graph. This allows an efficient coverage of the entire free-space and the identification of a few promising local minima in addition to the global optimal path .

This is essentially a *single-pair* search for  $n$  constrained paths through a graph with  $n$  nodes. It starts with a single-source search, such as Dijkstra's [14], that generates the shortest path from the source  $s$  to the goal  $g$  (Fig. 5). The cost  $a_{s,i}$  stored at each node  $i$  is the cost from  $s$  to that node. Repeating this search from the goal  $g$  to  $s$  stores the cost  $b_{g,i}$  at each node  $i$  (Fig. 6). Summing the two costs  $c_{s,g,i} = a_{s,i} + b_{g,i}$  yields the optimal cost  $c_{s,g,i}$  for the path between  $s$  and  $g$  that passes through node  $i$  (Fig. 7).

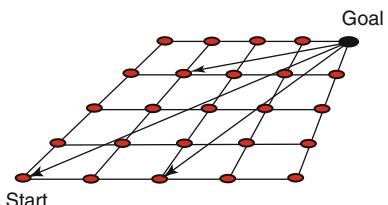
If each local minimum represents a homotopy class, the computational cost of this approach is  $O(2)$  for the initial search, and on average  $O(m/p \log m/p)$  for the pruning process, where  $m$  is the number of nodes and  $p$  is the number of homotopy classes generated by this search [57]. Compare to the  $k$ -best search,  $O(m)$  for the initial search and  $O(m \log m)$  for pruning. This efficiency is achieved at the cost of generating only a subset of all possible local minima, but at a computational cost far smaller than the alternative.

Figure 8 shows a topographic surface that is to be traversed from Start to Goal. The surface was first tessellated by a uniform grid, then the shortest paths through all nodes were computed using the algorithm discussed earlier [56, 57]. Color marking

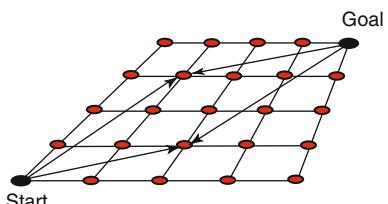
**Fig. 5** Shortest paths from start to all nodes

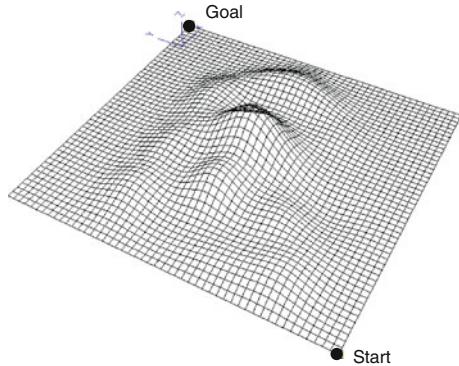
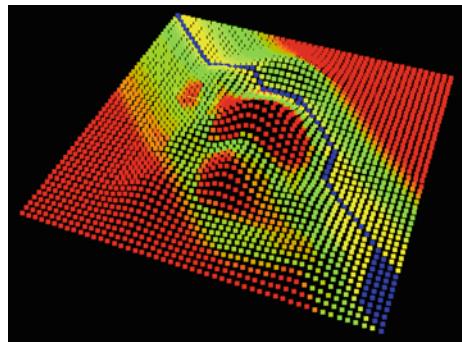


**Fig. 6** Shortest paths from goal to all nodes



**Fig. 7** Shortest paths through via points



**Fig. 8** A surface map**Fig. 9** A color coded surface map. The color at each node represents the optimal cost for passing through that node

the nodes according to the cost of passing through each node produced the *cost map* shown in Fig. 9. The cost function for this case was a traversability measure, calculated by dividing distance by the maximum safe velocity along each segment along the graph [16]. Here, blue represents the lowest cost (global minimum), then yellow, green and red represent gradually increasing costs. The *cost map* clearly shows the traversability of each region, thus offering sub-optimal alternatives to the global optimal path, which is colored blue. The blue “river”, whose nodes all have the same (optimal) cost, might be wider in regions where the neighboring edges have identical costs. In such cases, the global optimal *grid* path may not be unique, which is a common artifact of the uniform discretization of the search space.

### 3.3 Branch and Bound Search

The goal of the branch-and-bound search is to efficiently reduce the initially large set of paths in each homotopy class to a smaller set that contains the local optimal path. This is done by dividing the initial set of paths into two smaller subsets: one that contains all paths having a lower bound estimate on their cost that is higher

than the lower bound estimate of all paths in the second subset. The second subset is discarded, and the process repeats by subdividing the remaining subset using a more accurate lower bound estimate. Repeating this process, using a series of gradually increasing lower bounds, thus reduces the initial large set of paths to a much smaller set of promising paths. The search is terminated when the last subset has been shown to contain no better solution than the one already at hand. The best solution found during this search is the optimal path [15]. The fastest among the local minima found in this process is the global optimal path.

In this search, the objective function is the minimum traveling time between the two end points, whereas the initial set consists of all feasible (collision-free) paths between the given end points. It remains to determine appropriate approximations of the cost function that are guaranteed to produce lower bounds on the traveling time along a given set of paths. The computational efficiency of this approach depends on the proper selection of the lower bound estimates at each step. The most conservative but efficient approximations are used first, when the number of path candidates is large, and the more accurate but computationally expensive are used last. The last test is the exact solution, which is the optimal traveling time along the path.

We use three lower bound estimates on the optimal motion time along a given path, each represented by a different velocity profile: (1) maximum constant speed, (2) velocity limit, and (3) optimal velocity along the path. The cost estimate is computed by integrating the respective velocity profile along the path.

**Maximum Speed:** The first lower bound estimate,  $t_1$ , assumes motion everywhere at the maximum speed the robot can reach. It can be the tip velocity reached by assuming no load speeds at all joints at the most stretched configuration, or the maximum speed a mobile robot can reach on flat terrain. Dividing the distance along each edge of the graph by the maximum speed produces a lower bound estimate, obtained by the summation:

$$t_1 = \sum \frac{\Delta x_i}{v_{max}}, \quad (11)$$

where  $\Delta x_i$  is the Euclidean distance of the  $i$ th segment along the path, and  $v_{max}$  is the maximum speed.

Having assigned a fixed cost to all edges, the paths produced by the graph search are rated by a lower bound estimate on the optimal motion time along each path. Paths with lower bound estimates higher than the optimal motion time along some arbitrary path can be discarded early in the search process.

**Velocity Limit:** Once a path has been selected from the grid search, it is smoothed by cubic B splines, using the nodes of the graph along the path as control points. This eliminates the sharp corners produced by the grid segments. If the smoothed path penetrates an obstacle because of the rounded corners, it can be either discarded or kept for the next lower bound test. Eventually, the local optimization, discussed later, will divert the path away from the obstacle.

Here we assume that the speed along the path follows the velocity limit curve  $\dot{s}_{max}(s)$ ,  $s$  being the distance parameter along the path, that accounts for robot dynamics, actuator constraints, and path curvature, at every point along the path [27–29, 32].

The lower bound  $t_2$  is obtained by the integral

$$t_2 = \int_0^{s_f} \frac{ds}{\dot{s}_{max}}, \quad (12)$$

The computation of the velocity limit and the optimal velocity profile are briefly discussed later.

The value  $t_2$  is a true lower bound and greater than  $t_1$  since the velocity limit curve represents the true upper limit for the velocity profile along the path. This evaluation is computationally more demanding than the previous one but is less expensive than computing the time optimal velocity profile. This lower bound takes into account the combined effects of robot dynamics, actuator constraints, and path geometry.

**Optimal Velocity:** This is the exact solution for the optimal motion time and an upper bound to the previous lower bounds. The optimal velocity profile is always below the limit curve and at most tangent to the limit curve at a finite number of points [30]. The computation of the optimal velocity profile is briefly discussed next.

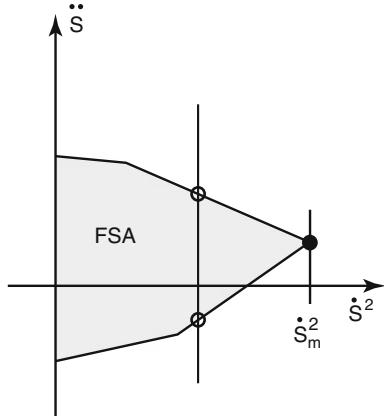
### 3.4 Time Optimal Motions Along Specified Paths

The optimal motion time along the path represents the exact cost function for the global search. It is computed using a well established algorithm [27–31, 58], which accounts for robot dynamics, actuator constraints, and path geometry. It is applicable to any fully actuated system such as industrial and mobile robots [16]. The algorithm will not be repeated here, referring the reader to the respective literature [27–31, 58].

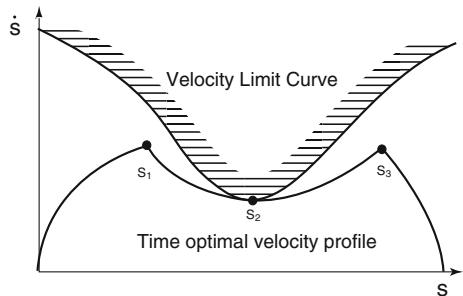
Key to this algorithm is the mapping of system dynamics to path coordinates. This reduces the multi dimensional configuration space, in which the robot operates, to a single degree-of-freedom system, where the distance and speed along the path,  $s, \dot{s}$ , are its two states, and the tangential acceleration  $\ddot{s}$  is its control input. The actuator constraints, coupled with path geometry, are mapped to constraints on  $\dot{s}$  and  $\ddot{s}$ , as shown schematically in Fig. 10 at some point  $s$  along the path. The boundary of the range of speeds and accelerations,  $FSA$ , represents states where at least one actuator reaches its limit. States outside of  $FSA$  are therefore dynamically infeasible.

At a given speed, the acceleration is bounded between its maximum and minimum values, as shown in Fig. 10. The speed  $\dot{s}_m$ , where the range of feasible accelerations reduces to a point, represents the highest speed at which the robot can still move along the prescribed path. Plotting  $\dot{s}_m$  along the path produces the velocity limit

**Fig. 10** The range of feasible speeds and accelerations (FSA)



**Fig. 11** Velocity limit curve and time optimal velocity profile



curve, as shown schematically in Fig. 11. It serves as the upper limit for any velocity profile along the path, optimal or not. Crossing the velocity limit curve implies that the robot is moving at speeds that are not sustainable by the robot's actuators or that it does not follow to prescribed path.

The time optimal velocity profile is computed using “bang-bang” control, switching between maximum acceleration and maximum deceleration along the path. The switching times are selected so that the optimal velocity profile avoids crossing the velocity limit curve [30], as shown schematically in Fig. 11. In the schematic example shown in Fig. 11, the time optimal velocity profile is integrated from the initial point at zero speed, using the maximum acceleration. At some point  $s_1$  along the path, the acceleration is switched to the maximum deceleration until point  $s_2$ , where the optimal velocity profile is tangent to the velocity limit curve. From  $s_2$ , the maximum acceleration is again integrated until some point  $s_3$ , from where the maximum deceleration is used to reach the final point at zero speed. The number of switches is usually odd for a 2nd order system, and it depends on the shape of the velocity limit curve, and the robot's dynamic properties. This algorithm is computationally very fast and can be used to efficiently assign the optimal motion time to every path in the last set of paths of the branch and bound search.

### 3.5 Local Optimization

The paths generated over the graph are forced to pass through the nodes of the graph defined by the grid used to represent the search space. To relax the demands on the grid resolution, a local optimization is used to locally alter the path to further reduce motion time [33, 34]. The optimization problem is formulated as an unconstrained parameter optimization, using the control points of cubic B splines as the optimization variables, and the optimal motion time along the path as the cost function. Obstacles are represented by penalty functions that account for the distance between the robot and the obstacles. At each iteration of the local optimization, the optimal motion time along the current path is computed using the method discussed earlier in Sect. 3.4, and the control points are modified by the optimization algorithm so as to produce paths with gradually decreasing optimal motion times. This process repeats until the optimal motion time reaches a local minimum. This optimization is obviously local since the path cannot “jump” over obstacles.

To reduce computation time and improve the convergence of the local optimization, the number of control points is reduced by retaining only a few points for each straight line segment along the grid path. It is important to note that a small number of control points may not adequately represent the true optimal path, however, a large number of parameters may be computationally costly. The true optimum can be approached asymptotically by successively increasing the number of control points and repeating the local optimization.

The local optimization is used to optimize only a small number of promising paths, selected from the paths remaining after the branch and bound search. These paths are selected as the best in each homotopy class [57] or as the best in some defined neighborhood of radius  $D_{max}$ . The classification of the paths into homotopy classes is discussed in [57] and will not be repeated here. The selection of the best path in each neighborhood is done by first discarding all paths that are contained in a tube around the best path, each satisfying the inequality:

$$D = \max |(p_i(w) - p_0(w))| < D_{max}, \quad w = [0, 1]; i = 1 \dots N, \quad (13)$$

where  $p_0(w)$  is a point along the best path in the neighborhood, with  $w$  being a normalized path distance, and  $p_i(w)$  is a point along any path in the remaining set of  $N - 1$  paths. This process is repeated for the next best path among the remaining paths until only a few paths, representing distinct regions, remain.

### 3.6 Summary of the Off-Line Planner

The off-line planner that uses the K-best search, is summarized in the following pseudo code. In the following, “best path” refers to the path along which the optimal motion time or a lower bound estimate is the smallest of all paths in the given set.

**Algorithm 1:** *Off-line planning***Step 0:** Initialize.Receive the geometric description of the workspace, robot dynamics, actuator constraints, dynamic and state constraints, current state  $x$ , target state  $x_f$ ;Determine the robot maximum speed  $v_{max}$  for Eq. (11);Set an upper bound  $t_{up}$  to be used to terminate the first search;Set diameter  $R$  for path filtering.**Step 1:** Generate a graph over the workspace

Assign cost, usually Euclidean distance, to all edges on the graph.

Assign high cost to edges that connect unreachable nodes.

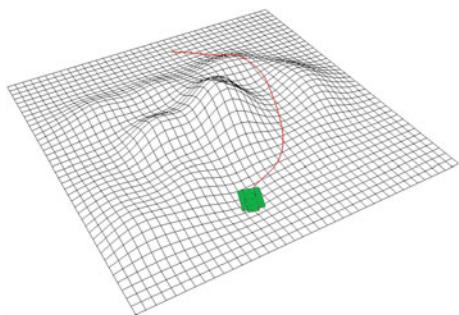
**Step 2:** Use the K-best search to generate the set  $P_0$  of shortest paths between the end points (the projections into the configuration space of the current and target states). Stop the search when  $t_1(K) \geq t_{up}$ .**Step 3:** Smoothing.Smooth all paths in  $P_0$  by B-splines, using the nodes along each path as control points.  $P_1$  is the set of  $K$  smoothed paths.**Step 4:** For all paths in  $P_1$ , compute a lower bound estimate  $t_2(i)$ ,  $i = 1, \dots, K$ , using (12).**Step 5:** Select the best path  $j$ :  $t_2(j) = \min\{t_2(i), i = 1, \dots, K\}$ . Compute the optimal motion time  $t_3(j)$ ;  $t_3(j)$  serves as the next upper bound in the branch and bound search.**Step 6:** Move all paths in  $P_1$  that satisfy  $t_2(i) \leq t_3(j)$ ,  $i = 1, \dots, K$ , to  $P_2$ .**Step 7:** Compute the optimal motion time  $t_3(i)$  for all paths in  $P_2$ .**Step 8:** Pruning.Select the best path in  $P_2$  and discard all paths that are inside a tube of radius  $R$  around that path, using (13); Move the best path to  $P_3$ ; Repeat for the next best path in  $P_2$  until  $P_2$  is empty.  $P_3$  now contains a small set of “good” paths.**Step 9:** Local optimization.Submit all paths in  $P_3$  to a local optimization. The resulting paths form the set of local minima  $P_4$ .**Step 9:** Global optimum.The best path in  $P_3$  is the global optimal path, along which the optimal motion time is globally optimal.

STOP.

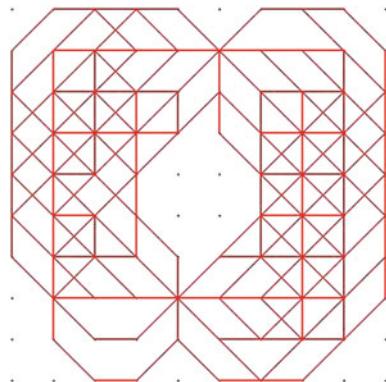
### 3.7 Example 1

Figure 12 shows the near-global time optimal trajectory, computed using the global optimization discussed here, for a vehicle moving over general terrain. For this example, the  $k$ -best search was used to generate the initial set of 500 paths, all shown in Fig. 13. The grid resolution was set low at 1 m between nodes for a  $10 \times 10$  m terrain segment. The branch and bound search retained 22 best paths, each was smoothed by

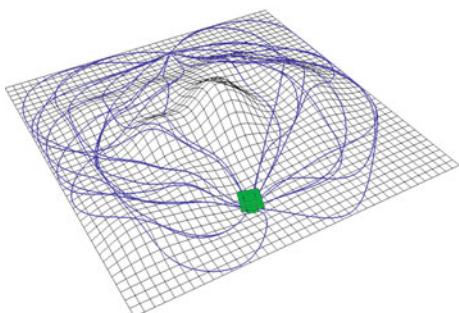
**Fig. 12** A (near) global time optimal path over general terrain, generated by the global planner



**Fig. 13** 500 shortest paths generated over the uniform grid overlayed over the terrain

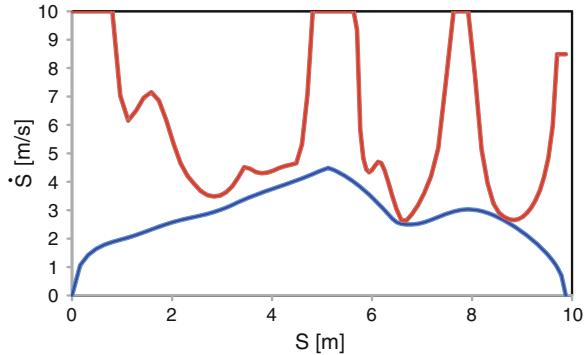


**Fig. 14** 22 best smoothed paths retained by the branch and bound search



a cubic B spline, as shown in Fig. 14. All 22 paths were locally optimized to further reduce motion time, and the best path, shown in Fig. 12, was selected as the global optimal solution. The time optimal velocity profile along the best path is shown in Fig. 15. Also shown in Fig. 15 is the velocity limit curve. Note that the vehicle slows down before accelerating again to prevent it from reaching high speeds that would cause it to airborne over the bump in the upper part of the terrain segment. The effect of the bump on the vehicle speed is reflected in the drop of the velocity limit curve.

**Fig. 15** The optimal velocity profile and the velocity limit curve along the time optimal path



The solution obtained is near global optimal due to the choices of the grid resolution and the termination condition of the local optimization.

Computation time depends on the number of paths generated in the graph search, the number of promising paths left for the local optimization, and the number of control points used to represent each path. The global planner was implemented in C and run on an Intel core-i7 3.4GHz desktop computer. For example 1, the global optimal path was computed in 20 s, most of which was spent on the local optimization of 22 paths.

The global optimization presented here is inherently off-line as it produces the complete solution to the goal. It combines a search for a set of the best paths in a grid in the configuration space with a local path optimization. This combination allows to reduce the search to the lower dimensional configuration space without compromising optimality. There are only few global planners that we can compare to, especially those computing time optimal trajectories [7, 37].

The solution produced by this planner is a global optimum if the grid is sufficiently small. The requirement on grid resolution is relaxed by assuming that the region of convergence around the optimal path is large compared to the grid size. Despite this approach being presented long ago, it is still computationally efficient compared to more recent global optimizations [7, 37]. Lacking information on the use of RRT\* to solving dynamic problems, it is difficult to compare this popular approach to ours.

## 4 Online Planner

We now address the online time-optimal obstacle avoidance problem for robots moving in cluttered environments. Motivated by the observation that the effect of an obstacle on the value function (the global cost-to-go function) in (10) is local [51], we solve the multi-obstacle problem by avoiding obstacles one at a time. This

is equivalent to approximating the value function of the multi-obstacle problem by switching between the value functions of the individual problems, each avoiding a single obstacle. Computationally, this transforms the multi-obstacle problem with  $m$  obstacles to  $m$  simpler sub-problems with one obstacle each, thus reducing the size of the problem from exponential to linear in the number of obstacles. As a result, this approach produces an on-line planner, i.e. the trajectory is generated incrementally, one step at a time, requiring a low computational effort at each step relative to the original, inherently off-line, problem.

While the approach of avoiding obstacles optimally one at a time applies to any robot dynamics, and convergence can be guaranteed for any obstacle shapes, we treat here a point mass robot in the plane and convex obstacles.

We begin with the optimal avoidance of one obstacle.

## 4.1 Optimal Avoidance of a Single Obstacle

The time optimal avoidance of a single obstacle in the plane is relatively simple. It can be computed using a global optimization [2], or by running a local optimization [34] twice (one for each side of the obstacle for a planar problem).

Consider the following point mass model:

$$\begin{aligned}\ddot{x} &= u_1 \quad ; \quad |u_1| \leq 1 \\ \ddot{y} &= u_2 \quad ; \quad |u_2| \leq 1\end{aligned}\tag{14}$$

where  $(x, y)^T \in \mathbb{R}^2$  and  $(u_1, u_2)^T \in \mathbb{R}^2$  represent the configuration space variables and actuator efforts, respectively.

We first derive the *unconstrained* trajectory, for states not affected by the presence of the obstacle.

### 4.1.1 The Unconstrained Trajectory

The unconstrained trajectory for the decoupled system (14) is determined by the minimum motion time of the *slowest* axis.

Consider first a single axis, represented by the double integrator

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \quad ; \quad |u| \leq 1.\end{aligned}\tag{15}$$

Using optimal control theory [23], it is easy to show that the time-optimal control for system (15) is bang-bang with at most one switch [50]. In the following, we denote  $x = (x_1, x_2)$  and  $x_f = (x_{1f}, x_{2f})$ .

The minimum *time-to-go* from any state  $x$  to  $x_f$  can be computed analytically [59, 60]:

$$t_f(x, x_f) = \begin{cases} -x_2 - x_{2f} + 2\sqrt{-x_1 + x_{1f} + \frac{x_2^2}{2} + \frac{x_{2f}^2}{2}}, & \text{if } x \in \mathbf{R} \\ x_2 + x_{2f} + 2\sqrt{+x_1 - x_{1f} + \frac{x_2^2}{2} + \frac{x_{2f}^2}{2}}, & \text{otherwise} \end{cases} \quad (16)$$

where

$$\mathbf{R} = \{(x) \mid S_1(x) > 0, S_2(x) < 0\}, \quad (17)$$

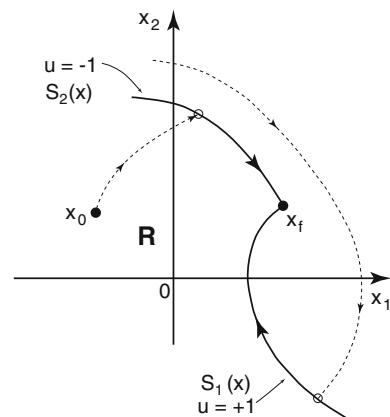
and the switching curves  $S_1(x)$ ,  $S_2(x)$ , shown in Fig. 16, are:

$$\begin{aligned} S_1(x) &= x_2^2 - 2\left(x_1 - x_{1f} + \frac{x_{2f}^2}{2}\right) = 0, \\ S_2(x) &= x_2^2 + 2\left(x_1 - x_{1f} - \frac{x_{2f}^2}{2}\right) = 0. \end{aligned} \quad (18)$$

The switching time  $t_s$  is [60]:

$$t_s(x, x_f) = \begin{cases} -x_2 + \sqrt{-x_1 + x_{1f} + \frac{x_2^2}{2} + \frac{x_{2f}^2}{2}}, & \text{if } x \in \mathbf{R} \\ x_2 + \sqrt{+x_1 - x_{1f} + \frac{x_2^2}{2} + \frac{x_{2f}^2}{2}}, & \text{otherwise} \end{cases} \quad (19)$$

**Fig. 16** Switching curves in the state space of a single axis



Equation (16) computes the optimal *time-to-go* from any given state. It is used to determine the slowest axis of a multi axis system and set the motion time for the slowest axis, as discussed later.

The time-optimal trajectory thus first follows a parabola from the initial state to the switching curve, then follows the switching curve to the target state, as shown schematically in Fig. 16. Trajectories starting from initial states left of the switching curves (region **R** in Fig. 16) begin with  $u = 1$ , and right of the switching curves with  $u = -1$ . Trajectories starting from states on the switching curve follow the switching curve to the target with no switch. The switching time is determined by the initial and final states.

Since the minimum time trajectory has only one switch (excluding trajectories that emanate from initial states on the switching curves), reaching the target at a time greater than the minimum time,  $t_f$ , using bang-bang control, requires more than one switch [50].

For the two axis system (15), each axis may reach the target at a different optimal time. Obviously, the optimal time  $t_f$  to reach the target is determined by the slowest axis. Assuming, without loss of generality, that the faster axis from any initial state  $x_0 = (x_{10}, x_{20}, y_{10}, y_{20})$  to the target state  $x_f$  is the  $y$ -axis, the time-optimal trajectory is obtained by driving the  $x$ -axis optimally, and driving the  $y$ -axis so that it reaches the target at the same final time,  $t_f$ .

The trajectory of the  $x$ -axis is unique since it is optimal and hence has only one switch, whereas the trajectory of the  $y$ -axis is not optimal and hence has *at least* two switches.<sup>1</sup> It follows that the time-optimal path between the end points is not unique. The set of all time-optimal paths is bounded by two *extremal* paths, generated by the *extremal* trajectories, which are in turn generated by the extremal controls,  $u_{max}$  and  $u_{min}$  [50]:

$$u_{max}(t) = \begin{cases} 1 & \text{if } t \in [0, t_{s1}] \\ -1 & \text{if } t \in [t_{s1}, t_{s2}] \\ 1 & \text{if } t \in [t_{s2}, T] \end{cases} \quad (20)$$

$$u_{min}(t) = \begin{cases} -1 & \text{if } t \in [0, t_{s3}] \\ 1 & \text{if } t \in [t_{s3}, t_{s4}] \\ -1 & \text{if } t \in [t_{s4}, T] \end{cases} \quad (21)$$

where  $T > t_f$  is specified, and

$$\begin{aligned} t_{s1} &= \frac{1}{2\alpha} \left( x_{1f} - x_{10} + 2\alpha T - x_{20}T - \frac{T^2}{2} - \alpha^2 \right) \\ t_{s2} &= t_{s1} + \alpha \\ \alpha &= \frac{(T + x_{20} - x_{2f})}{2}, \end{aligned} \quad (22)$$

---

<sup>1</sup>The switching time of the slowest axis occurs when its trajectory reaches one of the switching curves given in (18).

$$\begin{aligned}
t_{s3} &= \frac{1}{2\beta} \left( x_{1f} - x_{10} - 2\beta T - x_{20}T + \frac{T^2}{2} + \beta^2 \right) \\
t_{s4} &= t_{s3} + \beta \\
\beta &= \frac{(T - x_{20} + x_{2f})}{2}.
\end{aligned} \tag{23}$$

We call the two-switch trajectories the *extremal* trajectories. Note that if the optimal motion times of both axes are identical, then the time-optimal trajectory is unique.

The unconstrained trajectory of system (14) from any state  $x = (x_1, x_2, y_1, y_2)$  to the target state  $x_f = (x_{1f}, x_{2f}, y_{1f}, y_{2f})$  is thus determined by the optimal motion time of the slowest axis. It can be used to drive the system as long as at least one extremal trajectory avoids the obstacle. Otherwise, the obstacle must be avoided using the *constrained* trajectory discussed next.

## 4.2 The Constrained Trajectory

The constrained trajectory is needed for points in the state-space from which *all* unconstrained time-optimal trajectories to the target intersect the obstacle. We refer to the set of such points as the Obstacle Shadow. In the kinematic case [51], the shadow corresponds to the shadow created behind the obstacle by a point light source at the target. The physical analogy for the dynamic problem is not as obvious.

The intersection of all the *extremal* time-optimal paths with the obstacle implies the intersection of all *unconstrained* optimal paths. It is therefore sufficient to check if both extremal trajectories intersect the obstacle to conclude that an avoiding trajectory, with optimal motion time greater than the optimal motion time of each axis, should be computed. Since the motion times of both axes are non-optimal, and hence greater than the unconstrained time  $t_f$ , it follows that both axes have at least two switches.

We compute the time optimal trajectory from  $x_0$  to  $x_f$  that avoids the obstacle, numerically, using a line search over the traveling time,  $t_c = t_f + \delta$ . The search terminates when the first trajectory that reaches the goal without intersecting the obstacle is found. The computation of the constrained trajectory for one obstacle is, thus, obtained by solving the following minimization problem over the single parameter,  $\delta$ :

$$t_c(x_0, x_f, OB) = \min_{\delta} t_f(x_0, x_f) + \delta \tag{24}$$

such that there exists  $j = 1, \dots, 4$  that satisfies:

$$x_{ex,j}(t) \notin OB, \tag{25}$$

where  $t_f(x_0, x_f)$  is the unconstrained optimal time (16), and  $x_{ex,j}(t)$ ,  $t \in [0, t_f + \delta]$  represents the  $j$ th extremal trajectory. The four extremal trajectories  $x_{ex,j}(t)$

correspond to the four combinations of the initial controls of both axes:  $(1, 1)$ ,  $(-1, -1)$ ,  $(1, -1)$ ,  $(-1, 1)$ . Although only two of these four trajectories are true extremals, it is simpler to test all four. It is sufficient that only one extremal satisfies (25).

### 4.3 Multi-obstacle Avoidance

The optimal avoidance of one obstacle is relatively simple, and is hence suitable for on-line computation. We use it to solve the multi-obstacle problem by avoiding obstacles *one at a time*. Key to this approach is the selection of the *current* obstacle to be avoided at any given time, as discussed next.

### 4.4 The Current Obstacle

We select the *current* obstacle as the *maximum cost* obstacle, which takes the longest time to avoid from the current state  $x$  to the goal  $x_f$ . Denoting  $t_c(x, x_f, OB(j))$ ,  $j = [1, m]$  as the minimum time it takes to avoid obstacle  $OB(j)$  from  $x$  to  $x_f$ , the *current* obstacle,  $k$ , maximizes  $t_c$ :

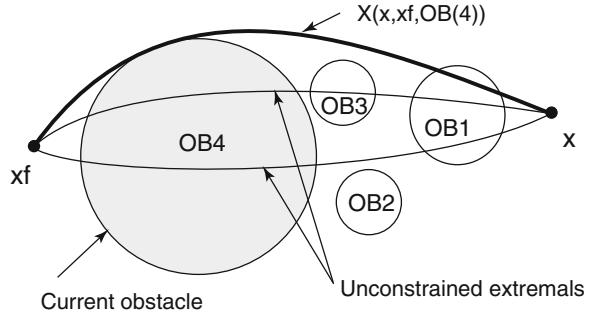
$$t_c(x, x_f, OB(k)) \geq t_c(x, x_f, OB(j)) \text{ for all } j = 1, \dots, m. \quad (26)$$

The *current* obstacle is thus selected by first determining all obstacles with shadows to the goal  $x_f$  containing the current state  $x$ , then computing the constrained trajectories avoiding each obstacle to  $x_f$ , and selecting the one with the longest motion time. If  $x$  does not lie in the shadow of any obstacle, then the cost of all obstacles equals to the unconstrained trajectory to the goal and none is selected to be avoided. One of the extremals of the set of unconstrained trajectories is then selected for navigation. The algorithm may switch between the extremals in case they collide with any obstacle, until either reaching the goal or entering the shadow of any obstacle, in which case the *current* obstacle is selected by (26).

Selecting at each step the obstacle with the highest cost to the goal produces a trajectory that is close to optimal, since the other obstacles have a smaller impact on the motion time to the goal, as is shown schematically in Fig. 17. In Fig. 17, the state  $x$  is in the shadows of obstacles 1 and 4. Of those, the trajectory avoiding  $OB(4)$ , denoted  $X(x, x_f, OB(4))$ , takes longer time than  $X(x, x_f, OB(1))$  (not shown). Hence  $OB(4)$  is selected as the *current* obstacle. Obviously, any solution to the goal must avoid obstacle 4. Hence, recognizing it early in the avoidance process increases the likelihood that the resulting trajectory will be close to optimal. The intersection of  $X(x, x_f, OB(4))$  with  $OB(1)$  will prompt a recursive process, discussed next.

While selecting the *maximum cost* obstacle is likely to result in near optimal trajectories, other selection criteria, such as the *nearest* obstacle (obstacle 1 in Fig. 17) may suffice for convergence.

**Fig. 17** Selecting the current obstacle from  $x$  to  $x_f$



#### 4.5 The Avoidance Algorithm

The avoidance algorithm assumes convex and non-overlapping obstacles (in the configuration space). It selects the current obstacle to be avoided, computes the time optimal trajectory that avoids that obstacle, selects an intermediate goal along that trajectory on the boundary of that obstacle, and attempts to reach that goal. It repeats the process recursively until reaching the closest intermediate goal.

**Algorithm 2:** *Online Avoidance*

**Step 0:** Initialize. Receive current state  $x$ , target state  $x_f$ ;

Set  $i = 0$ ,  $g(i) = x_f$ ;

**Step 1:** Determine the *current obstacle*,  $OB(k)$ , from  $x$  to  $g(i)$ .

If  $k = 0$  ( $x$  not in the shadow of any obstacle), go to Step 3.

Compute the optimal trajectory avoiding  $OB(k)$  to  $g(i)$ .

**Step 2:**  $i = i + 1$ ; Select an intermediate goal  $g(i)$  on the boundary of  $OB_k$  along the trajectory that avoids  $OB(k)$  to  $g(i - 1)$ .

Check that the velocity at  $g(i)$  is not in the *obstacle hole*<sup>2</sup> of any obstacle, consisting of *infeasible* states from which the obstacle is *unavoidable*. If it is, reduce speed at  $g(i)$  as needed.

Go to Step 1.

**Step 3:** Follow the optimal trajectory to  $g(i)$ . Set  $x = g(i)$ .

If  $i = 0$ , STOP.

$i = i - 1$

Go to Step 1.

Algorithm 2 generates a series of intermediate goals until one is reachable by a time optimal trajectory without colliding with any obstacle. Each intermediate goal  $g(i)$  ( $i \geq 1$ ) is selected along the constrained trajectory  $x_c(t)$  from the current state  $x$  to the current goal  $g(i - 1)$  at a point where  $x_c(t)$  is tangent to the current obstacle  $OB_k$ . Usually, there is just one such point. In case  $x_c(t)$  follows the obstacle for some

---

<sup>2</sup>The obstacle hole is a subset of the obstacle shadow.

distance, the point closest to the goal  $g(i - 1)$  is selected. When an intermediate goal is reached, a new avoidance problem is attempted from that intermediate goal to the next goal in the queue. Note that once an intermediate goal was reached, it is removed from the queue and a new goal may be assigned the same index  $i$ . The goals are added and removed from the queue while the trajectory gradually progresses to the final goal  $x_f = g(0)$ . Remembering the intermediate goals generated during the process is key to the convergence of this algorithm, as discussed later.

Step 2 of Algorithm 2 selects the speed at the intermediate goal  $g(i)$  that is both safe and feasible. A safe velocity is one that does not penetrate any obstacle hole, from which the obstacle is unavoidable. To simplify the search for the safe velocity, we choose to reduce it to the maximum velocity at which the robot can circle the current obstacle at its maximum lateral acceleration (the acceleration normal to its direction of motion). Denoting this velocity as the *curvature* velocity, it is easily proven that the *curvature* velocity does not lie in the obstacle hole of any obstacle.

**Definition 4.1** Curvature Velocity. The *curvature* velocity,  $v_c$ , is defined as:

$$v_c = \sqrt{u_{max} R} \quad (27)$$

where  $u_{max}$  is the maximum lateral acceleration, and  $R$  is the radius of the obstacle.

It remains to verify that the velocity at  $g(i)$  is reachable from the current state  $x$ . This is done by checking that a direct time optimal trajectory exists from  $x$  to  $g(i)$ . A direct trajectory is one that does not include loops. In case the velocity at  $g(i)$  is too high, we scale it down until it is reachable from  $x$ ; if the velocity at  $g(i)$  is too low, the current speed, which was set to the curvature velocity, can be reduced by circling the nearby obstacle at a decreasing speed. The curvature velocity (27) ensures that the obstacle can be circled to allow a safe reduction in speed when necessary. While this feature is necessary to ensure safety, it was not needed in any of the many cases tested by this algorithm.

The adjustment of speeds at the intermediate goals would ensure that any consecutive intermediate goals are connected by a feasible trajectory. This implies that a too high final velocity may be compromised for the sake of safety. Similarly, not every initial velocity is feasible for the obstacle avoidance case, even if it does not penetrate any individual obstacle hole. The speed reduction at the intermediate goals to the curvature velocity is a conservative measure to ensure safety.

## 4.6 Convergence

Convergence implies that the algorithm can reach the target state from an arbitrary feasible state, in a finite time. Since we cannot a priori determine the feasibility of arbitrary initial and target velocities, convergence of Algorithm 2 can be proven under

the assumption of zero terminal speeds (the speeds at the initial and final points), for convex obstacles that do not overlap with each other [50].

Algorithm 2 progresses incrementally towards the goal by moving through a sequence of intermediate goals. Every intermediate goal subdivides the trajectory to the goal into two smaller segments, and in fact breaks the avoidance problem into two smaller problems. Repeating this process recursively further reduces the avoidance problem until two consecutive intermediate goals are connected by an unconstrained trajectory. The motion time along each segment is finite since it is traversed at the minimum time. The number of such segments is bounded by the number of obstacles, which is assumed finite. It follows that the total travel time from start to goal is also finite, which proves convergence.

## 4.7 Optimality

The trajectory generated by Algorithm 2 is not necessarily optimal, since each step is only locally optimal. While the paths (the projection of the trajectory to the configuration space) generated by Algorithm 2 are generally close to the time optimal paths computed by a global planner [16], as demonstrated next, the motion time along the on-line trajectory is higher than the global optimal motion time due to the curvature velocity (27) imposed at the intermediate goals.

## 4.8 Numerical Examples and Experiments

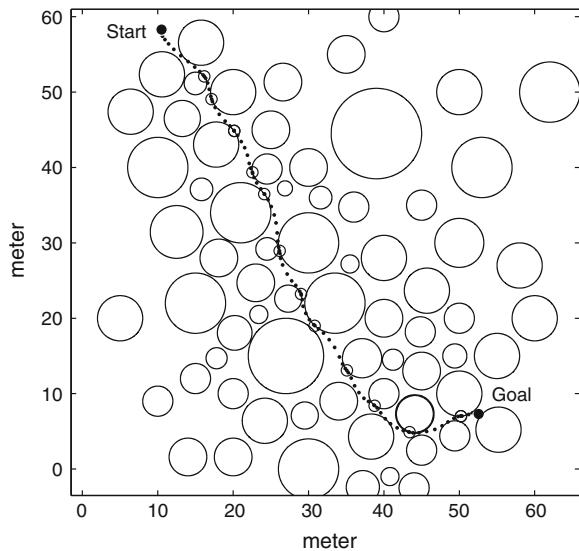
Algorithm 2 is demonstrated for a planar environment, consisting of 70 tightly spaced circular obstacles.

### 4.8.1 Example 2

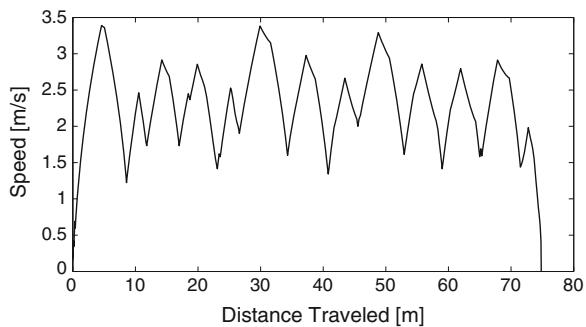
This example shows an on-line trajectory that avoids 70 obstacles, from the initial state  $(x_1, x_2, y_1, y_2) = (10.46 \text{ m}, 0.001 \text{ m/s}, 58.26 \text{ m}, 0.001 \text{ m/s})$  to the goal state  $(x_{1f}, x_{2f}, y_{1f}, y_{2f}) = (52.55 \text{ m}, 0 \text{ m/s}, 7.33 \text{ m}, 0 \text{ m/s})$ , as shown in Fig. 18. The spacing between the dots represents the speed along the path.

The motion time along this trajectory is 35.2 s, with a top speed of 3.4 m/s and an average speed of 2.1 m/s. There were 12 intermediate goals generated for this case, shown as empty circles along the trajectory. The total computation time was 4.3 s, with a time step  $\Delta t$  of 0.1 s, and an average computation time of 11 ms per-step. The speed along the trajectory, as a function of distance traveled, is shown in Fig. 19. The oscillations in the speed profile are due to the curvature velocity imposed at the intermediate goals.

**Fig. 18** Trajectory generated on-line in a tightly spaced environment with 70 circular obstacles for Example 2



**Fig. 19** Speed as a function of distance traveled along the online trajectory of Example 2

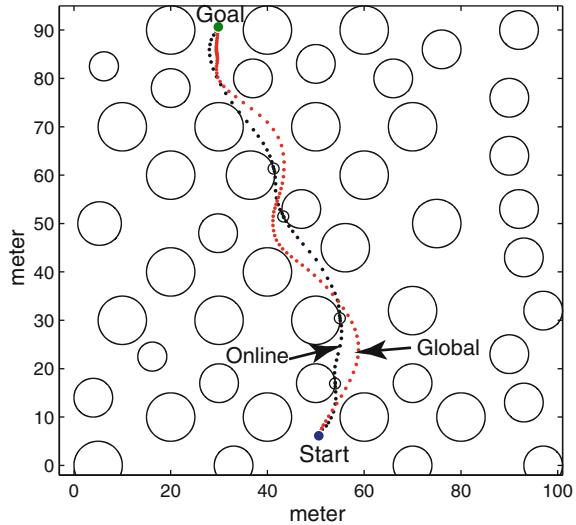


#### 4.8.2 Experiment–Global Optimality

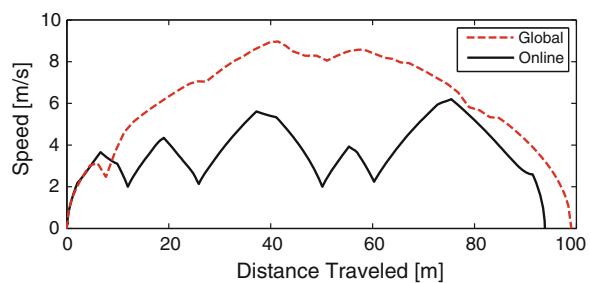
This experiment compares the online planner with the global planner [16] for the obstacle setup shown in Fig. 20 (48 obstacles).

Shown in Fig. 20 are the trajectories generated by the online planner and the global planner. The online and globally optimal paths have similar topologies as they pass between the same obstacles. The velocity profiles along both trajectories are shown in Fig. 21. The motion time along the online trajectory was 28.9 s over a total distance of 93.8 m, with an average speed of 3.2 m/s, compared to the global optimal motion time of 20.7 s over a total distance traveled of 99 m, and an average speed of 4.8 m/s. This difference is caused primarily by the reduction in speeds to the curvature velocities (27) at the intermediate goals.

**Fig. 20** The trajectories generated by the global and online planners, among 48 obstacles



**Fig. 21** Speed as a function of distance traveled for the online and global optimal trajectories



Repeating this test for 50 randomly selected end points yielded similar results, with the average motion time of the online trajectories being 30.52 s, compared to the average optimal time of 22.63 s. The average path length of the online trajectories was 111.51 m, compared with 115.26 m for the optimal trajectories. Here too, the increase in the motion time despite the comparable path lengths is due to the imposed curvature velocity at the intermediate goals, which is determined by the obstacle size.

#### 4.9 Computational Issues

The consideration of the obstacles one at a time reduces the original problem with  $m$  obstacles to  $m$  simpler sub-problems with one obstacle each.

The cost for this reduction is the loss of optimality, and the need to check at each time step if *all* obstacles intersect the unconstrained optimal path from the current state, and for those that do, solve the single obstacle problem. This may

seem excessive, but the alternative (solving the original exponential problem) is much worse. Our approach generates the trajectory incrementally, unlike the original problem that requires a complete solution before making the first move. In fact, for problems with many obstacles, such as in example 2 with 70 obstacles presented earlier, the on-line (heuristic) solution may be the only viable alternative.

Practically, it may not be necessary to consider all obstacles at all times, but instead consider only the obstacles within some radius of visibility around the robot. It would be then necessary to limit robot's speed to the stopping speed at the boundary of its visibility range to ensure that it does not collide with an unforeseen obstacle.

To appreciate the computational advantage of this approach, we attempted to compare it to the performance of efficient state-of-the-art algorithms. Currently, the most popular approach is the RRT planner, which rapidly explores a random tree to produce the first feasible solution to the goal [17–19]. The solution found is not optimal in any way, and this class of algorithms is known to have inherent difficulties with tight spaces. Yet, RRT is currently considered as the fastest algorithm to connect between two points through cluttered environments.

We compared a kinematic version of our online algorithm to the RRT and RRT\* planners for avoiding 70 tightly space obstacles, all running on similar computers [50]. Testing the algorithms for 100 randomly generated end points, the run time of the online algorithm was on average 0.5 ms, compared to 3.5 ms of the RRT planner, 7 times faster. However, the path lengths produced by the RRT planner were twice as long as those produced by the online planner, which were near global optimum. Attempting to optimize the paths using the RRT\* planner took 0.5 s to reach the optimality levels of the online planner; this is 1,000 times slower than the online algorithm. These results demonstrate the sound efficiency, in both computation time and optimality, of the online planner presented here. This is not surprising as the online planner consistently executes locally optimal paths at each incremental step, as opposed to the sampling-based planners which essentially search for a solution in the dark.

## 5 Summary

Motion planning is one of the basic problems in robotics as very few robotic tasks do not involve motion. The main challenge in motion planning is to produce a trajectory that safely and efficiently moves the robot from one state to another while accounting for its dynamic behavior. It is also desirable that the motion plan reflects the changing nature of the task or of the environment. While this is obviously the ultimate goal, early works on motion planning in the 80s settled for much less by focusing on geometric path planning with no account for robot dynamics. The resulting algorithms were useful for determining the shortest path from start to goal, but were useless for moving the robot at other than very low speeds. To account for robot dynamics, optimal control theory, developed in the late 60s, was applied then to robotics but failed because of insufficient computation power and the high sensitivity of the numerical

solutions to the initial guess. Like in many other endeavors, the solution emerged by solving a simpler problem.

The failure of the geometric algorithms to solve high dimensional problems gave rise to a class of sampling-based planners, with the goal of producing any feasible path in lieu of the shortest path expected by earlier work. The multi-dimensional optimal trajectory planning problem was eventually solved by first computing the optimal velocity profile along a given path. This lead to a local optimization of the path and eventually to a global planner that computes “good” initial guesses for the local optimization.

In this chapter, we reviewed the main approaches to off-line and on-line motion planning, and presented one solution for each with a focus on trajectory planning. It was shown that any motion planning problem can be theoretically solved using the Hamilton Jacobi Bellman (HJB) equation. If the return function is known or approximated, this approach offers an online solution. In its discrete form, the HJB equation leads to dynamic programming, which is the basis for the combinatorial optimizations used in off-line planning.

We presented an off-line planner that takes advantage of the efficient computation of the optimal motion time along any path. The on-line planner presented converts the original problem of optimally avoiding many obstacles to many simpler problems, each avoiding optimally only one obstacle. The high correlation between the solutions of the on-line and off-line planners is not surprising since both planners are based on sound optimal control theories.

As the basic problem of trajectory planning is considered solved, and as computers are becoming more powerful, the remaining challenge rests with online planning that adapts or reacts to the changing nature of real life scenarios in the industry, in the home, and on the road.

## References

1. Choset H, Lynch KM, Hutchinson S, Kantor GA, Burgard W, Kavraki LE, Thrun S (2005) Principles of robot motion: theory, algorithms, and implementations. MIT Press, Cambridge
2. Shiller Z, Dubowsky S (1991) On computing the global time optimal motions of robotic manipulators in the presence of obstacles. *IEEE Trans Robot Autom* 7(6):785–797
3. Canny JF (1988) The complexity of robot motion planning. MIT Press, Cambridge
4. Lozano-Perez T, Wesley MA (1979) An algorithm for planning collision-free paths among polyhedral obstacles. *Commun ACM* 22(10):560–570
5. Lumelsky VJ, Stepanov A (1987) Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica* 2:403–430
6. Schwartz JT, Sharir M (1983) On the piano movers’ problem: the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Commun Pure Appl Math* 36:345–398
7. Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *Int J Robot Res* 30(7):846–894
8. Manor G, Rimon E (2013) Vc-method: high-speed navigation of a uniformly braking mobile robot using position-velocity configuration space. *Auton Robot* 34(4):295–309
9. Lozano-Perez T (1987) A simple motion planning algorithm for robotic manipulators. *IEEE Trans Robot Autom RA-3(3)*:224–238

10. Wein R, van den Berg JP, Halperin D (2005) The visibility-Voronoi complex and its applications. In: Proceedings of 21st symposium on computational geometry, pp 63–72
11. Alexopoulos C, Griffin PM (1992) Path planning for a mobile robot. *IEEE Trans Syst Man Cybern* 22(2):318–322
12. Liu YH, Arimoto S (1992) Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *Int J Robot Res* 11(4):376–382
13. LaValle SM (2010) Motion planning: the essentials. *IEEE Robot Autom Mag* 110
14. Dijkstra EW (1959) A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271
15. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization, algorithms and complexity. Prentice-Hall, Englewood Cliffs
16. Shiller Z, Gwo YR (1991) Dynamic motion planning of autonomous vehicles. *IEEE Trans Robot Autom* 7(2):241–249
17. LaValle SM (1998) Rapidly-exploring random trees: a new tool for path planning. Technical Report 98-11, Department of CS, Iowa State University
18. Hsu D, Latombe JC, Motwani R (1999) Path planning in expansive configuration spaces. *Int J Comput Geom Appl* 4:495–512
19. Mazer E, Ahuactzin JM, Bessiere P (1998) The Ariadnes clew algorithm. *J Artif Intell* 9:295–316
20. Karaman S, Walter MR, Perez A, Frazzoli E, Teller S (2011) Anytime motion planning using the rrt\*. In: International conference on robotics and automation
21. Amato NM, Bayazit OB, Dale LK (2000) Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans Robot Autom* 16(4):442–447
22. Gmez-Bravo F, Carbone G, Fortes JC (2012) Collision free trajectory planning for hybrid manipulators. *Mechatronics* 22(6):836–851. Special Issue on Intelligent Mechatronics
23. Bryson AE, Ho YC (1969) Applied optimal control. Blaisdell Publishing Company, Cambridge
24. Kiriazov P, Marinov P (1985) A method for time-optimal control of dynamically constrained manipulator. Theory and practice of robotics and manipulators. MIT Press, Cambridge, pp 169–178
25. Niv M, Auslander DM (1984) Optimal control of a robot with obstacles. In: Proceedings of American control conference (San Diego, CA), June 1984, pp 280–287
26. Khan ME, Roth B (1971) The near-minimum time control of open loop articulated kinematic chains. *J Dyn Syst Meas Control* 93(3):164–172
27. Bobrow JE, Dubowsky S, Gibson JS (1985) Time-optimal control of robotic manipulators. *IJRR* 4(3):3–17
28. Pfeiffer F, Johannir R (1987) A concept for manipulator trajectory planning. *IEEE Trans Robot Autom* RA-3(3):115–123
29. Shin KG, McKay ND (1985) Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Trans Autom Control* AC-30(6):531–541
30. Shiller Z, Lu HH (1992) Computation of path constrained time-optimal motions with dynamic singularities. *ASME J Dyn Syst Meas Control* 14(1):34–40
31. Slotine JE, Yang HS (1989) Improving the efficiency of time optimal path following algorithms. *IEEE Trans Robot Autom* 5(1):118–124
32. Tarkiainen M, Shiller Z (1993) Time optimal motions of manipulators with actuator dynamics. In: Proceedings of 1993 IEEE international conference on robotics and automation, vol 2, pp 725–730
33. Bobrow JE (1988) Optimal robot path planning using the minimum time criterion. *IEEE Trans Robot Autom* 4(4):443–450
34. Shiller Z, Dubowsky S (1989) Time-optimal path-planning for robotic manipulators with obstacles, actuator, gripper and payload constraints. *IJRR* 8(6):3–18
35. Seywald H (1994) Trajectory optimization based on differential inclusion. *J Guid, Control, Dyn* 17(3):480–487
36. Bryson AE (1999) Dynamic optimization. Addison Wesley, New York

37. Donald B, Xavier P (1989) A provably good approximation algorithm for optimal-time trajectory planning. In: Proceedings of IEEE conference on robotics and automation, May 1989, pp 958–963
38. Sahar G, Hollerbach JM (1985) Planning of minimum-time trajectories for robot arms. In: Proceedings of IEEE international conference on robotics and automation (St. Louis, MO), March 1985, pp 751–758
39. Kamon I, Rimon E, Rivlin E (1998) Tangentbug: a range-sensor based navigation algorithm. *Int J Robot Res* 17(9):934–953
40. Laubach S, Burdick J, Matthies L (1998) A practical autonomous path-planner for the rocky7 prototype microrover. IEEE international conference on robotics and automation
41. Choset H, Burdick JW (1995) Sensor based planning, part ii: incremental construction of the generalized Voronoi graph. In: Proceedings of IEEE international conference on robotics and automation, ICRA'95, pp 1643–1649
42. Sankaranarayanan A, Vidyasagar M (1991) Path planning for moving a point object amidst unknown obstacles in a plane: the universal lower bound on worst case path lengths and a classification of algorithms. In: Proceedings of IEEE international conference on robotics and automation, ICRA'91, pp 1734–1941
43. Connolly CI, Burns JB, Weiss R (1991) Path planning using Laplace's equation. In: IEEE conference on robotics and automation, Cincinnati, OH, vol 1, pp 102–2106
44. Khatib O (1986) Real time obstacle avoidance for manipulators and mobile robots. *Int J Robot Res* 1:65–78
45. Rimon E, Koditschek DE (1992) Exact robot navigation using artificial potential functions. *IEEE Trans Robot Autom* 8:501–518
46. Jarvis R (1985) Collision-free trajectory planning using distance transforms. *Trans Inst Eng Aust Mech Eng ME10(3):187–191*
47. Athans M (1965) Optimal control: an introduction to the theory and it's applications. Academic Press, New York
48. Cesari L (1983) Optimization—theory and applications: problems with ordinary differential equations. Springer, New York
49. Moskalenko AI (1967) Bellman equations for optimal processes with constraints on the phase coordinates. *Autom Remote Control* 4:1853–1864
50. Shiller Z, Sharma S, Stern I, Stern A (2013) On-line obstacle avoidance at high speeds. *Int J Robot Res* 32(9–10):1030–1047
51. Sundar S, Shiller Z (1997) Optimal obstacle avoidance based on sufficient conditions of optimality. *IEEE Trans Robot Autom* 13(2):305–310
52. Lee EB, Markus L (1967) Foundations of optimal control theory. Wiley, New York
53. Koditschek DE, Rimon E (1990) Robot navigation functions on manifolds with boundary. *Adv Appl Math* 11:412–442
54. Bellman R (1957) Dynamic programming. Princeton University Press, Princeton
55. Lawler EL (1976) Combinatorial optimization. Holt, Rinehart and Winston, New York
56. Fujita Y, Nakamura Y, Shiller Z (2003) Dual dijkstra search for paths with different topologies. In: ICRA, pp 3359–3364
57. Shiller Z, Fujita Y, Ophir D, Nakamura Y (2004) Computing a set of local optimal paths through cluttered environments and over open terrain. In: ICRA, pp 4759–4764
58. Pham QC (2013) Characterizing and addressing dynamic singularities in the time-optimal path parameterization algorithm. In: 2013 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 2357–2363
59. Dreyfus S (1965) Dynamic programming and the calculus of variations. Academic Press, New York
60. Sundar S (1995) Time-optimal obstacle avoidance for robotic manipulators. Doctoral Dissertation, Mechanical and Aerospace Engineering, University of California, Los Angeles, June 1995

# Open Architecture for Vision-Based Robot Motion Planning and Control

Theodor Borangiu, Florin Anton and Silvia Anton

**Abstract** This chapter introduces a methodology for the vision-based motion control of robot manipulators. The motion control problem is decomposed into three computational stages: motion planning, trajectory generation and trajectory tracking. While the two latter activities are always executed in real time, motion is planned in traditional robot systems off line, by learning robot points or by using numerical output data from programs that plan minimal paths, avoid obstacles, etc. Guidance vision is introduced as an advanced motion control method, which provides flexibility when integrating industrial robots in computer-controlled manufacturing structures. A dynamic look-and-move system architecture is discussed, as a robot-vision system which is closed at task level. An open architecture is proposed as implementing solution for vision-based scene management and robot guidance, which integrates any types of robot controllers and image processing libraries. The chapter also presents a motion control algorithm for robots which are required to pick objects randomly moving on conveyor belts. The algorithm for visual tracking of conveyor belts for “on-the-fly” object grasping is partitioned in two stages: (i) visual planning of the instantaneous destination of the robot, (ii) dynamic re-planning of the robot’s destination while tracking the object moving on the conveyor belt. The ensemble [conveyor belt + actuator + sensor] is configured as a single-axis Cartesian robot, leading to a cooperation problem between robot manipulators subject to multitasking control. Experimental results are finally reported in what concerns the statistics of object locating errors and motion planning errors function of the size of the objects of the belt speed and of the light strobe.

**Keywords** Robot-vision system · Vision guided robot planning · Visual robot servoing · Joint-space trajectory planning

---

T. Borangiu (✉) · F. Anton · S. Anton  
Department of Automation and Applied Informatics,  
University Politehnica of Bucharest, Bucharest, Romania  
e-mail: theodor.borangiu@cimr.pub.ro

## 1 Introduction

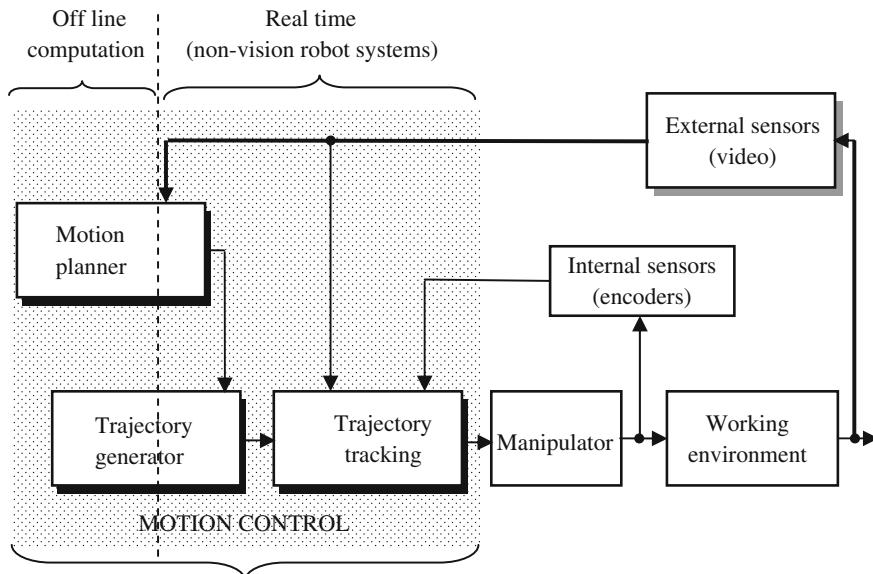
The motion control problem refers to controlling the robot manipulator such that it follows a pre-planned path. The motion control problem is generally decomposed into three computational stages (Fig. 1): (1) Motion planning; (2) Trajectory generation; (3) Trajectory tracking [1].

In the motion planning stage, desired paths are described in the  $r$ -dimensional task space  $T$  (i.e. the locus of the positions and orientations that the robot tool must attain in  $O \subseteq \mathbb{R}^m$ —the operational space,  $\mathcal{T} \subseteq O$ ), which is isomorphic to the special Euclidian group  $\mathbf{SE}^3$ .

$$\mathcal{T} = \{\mathbf{x}(t) \mid \mathbf{x} \in \mathbb{R}^r, t \in \mathbb{R}^+\}, \mathcal{T} \subseteq \mathbf{SE}^3 = \mathbb{R}^3 \times \mathbf{SO}^3$$

The vectors  $\mathbf{x} = \mathbf{x}_n^0 = [\mathbf{p} \ \phi]^T$ ,  $n = \text{no. of d.o.f.}$  express the location of the  $n$ th coordinate frame ( $x_n, y_n, z_n$ ), attached to the end-effector, relative to the world frame ( $x_0, y_0, z_0$ ) attached to the base of the robot,  $\mathbf{p} \in \mathbb{R}^3$  specifies the coordinates of the origin of the task frame (or end-effector frame), whereas the current orientation  $\phi$  of the task frame is described either by the rotation matrix  $\mathbf{R}$ —a member of the special orthogonal group  $\mathbf{SO}^3$ , or minimally by a set of 3 Euler angles (in the sequel, the *yaw*, *pitch* and *roll* angles will be considered). If the tool is a single rigid body moving arbitrarily in the Cartesian 3D workspace, then  $\mathcal{T} = \mathbf{SE}^3 = \mathbb{R}^3 \times \mathbf{SO}^3$ ,  $m = 6$ .

Because on one hand robotic tasks are specified with respect to one or more coordinate frames, and on the other hand visual servoing of robots makes intensive



**Fig. 1** Functional architecture for the global robot motion planning and control problem

use of a number of specific, additional coordinate frames, *coordinate transformations* are used in motion planning and tracking as a generalisation of *poses* to express relative locations between such frames of interest [2, 3].

Coordinate transformations must be often composed in the stage of motion planning and tracking, off line or at run time, to obtain the desired pose of the end-effector. Assuming that we are given the coordinate transformations  $\mathbf{x}_{vis}^0$  and  $\mathbf{x}_{obj}^{vis}$  expressing respectively the location of the coordinate frame  $(x_{vis}, y_{vis})$  attached to the image plane relative to the world frame  $(x_0, y_0, z_0)$  in the base of the robot, and the location of the frame  $(x_{obj}, y_{obj})$  attached to an object relative to  $(x_{vis}, y_{vis})$ , then the coordinates  $\mathbf{M}^{obj}$  of a point in the object frame can be expressed in the world frame by the composition rule (:):

$$\mathbf{M}^0 = \mathbf{x}_{vis}^0[\mathbf{x}_{obj}^{vis}[\mathbf{M}^{obj}]] = (\mathbf{x}_{vis}^0 : \mathbf{x}_{obj}^{vis})[\mathbf{M}^{obj}] = \mathbf{x}_{obj}^0[\mathbf{M}^{obj}]$$

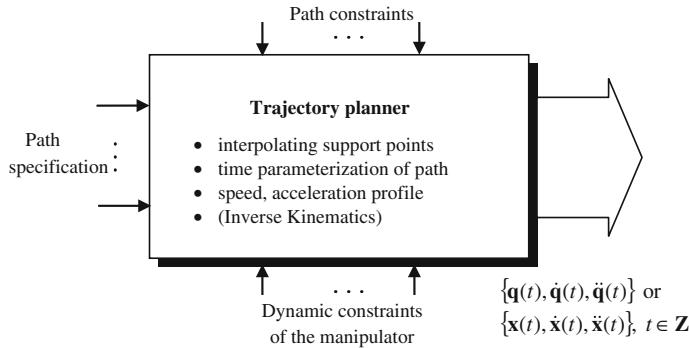
The associated relative rotation matrix and translation are given by  $\mathbf{R}_{obj}^0 = \mathbf{R}_{vis}^0 \mathbf{R}_{obj}^{vis}$ ,  $\mathbf{p}_{obj}^0 = \mathbf{R}_{vis}^0 \mathbf{p}_{obj}^{vis} + \mathbf{p}_{vis}^0$ . In the V+ structured robot programming environment, the *simple* transformations: `to.cam[cam]`—available from camera-robot calibration, and `vis.loc`—the object location computed at run time, stand respectively for  $\mathbf{x}_{vis}^0$  and  $\mathbf{x}_{obj}^{vis}$ ; the object-attached frame is related to the world frame by the *composed* transformation `obj.loc ← to.cam[cam] : vis.loc`

During *motion planning* stage, the desired paths are generated without timing information, i.e., without specifying the velocity and the acceleration along the path. Of primary concern is the definition of *collision-free paths* in the workspace. A secondary objective may be included, for example the optimization of some cost functions like: minimization of the total travel time or distance, keeping as low as possible changes in direction, continuity of velocity, etc. [4].

The trajectory planner (generator) parameterises the end-effector path *directly in the task space* either as a curve in  $\mathbf{SE}^3$ , or in  $\mathbf{R}^6$  when a minimal Euler representation is used for  $\mathbf{SO}^3$ . The trajectory planner may also compute a trajectory for the individual joints of the manipulator as a curve in the *configuration space*  $C = \{\mathbf{q}(t) \mid \mathbf{q} \in \mathbf{R}^n, t \in \mathbf{Z}^+, n = \text{no. of d.o.f.}\}$ .

The *trajectory planner* TP, represented as block—and connection diagram in Fig. 2, is a software module, component of the basic software system of the robot controller, being characterised as follows:

1. The *inputs* to the TP are the path description and constraints, and the constraints imposed by the manipulator's dynamics.
2. The *output* from the TP is the joint—or end-effector trajectory data, expressed as a discrete time sequence of the values which must be attained by the position, velocity and acceleration computed respectively in the *configuration space*  $\mathbf{q} \in C$  or in the *task space*  $\mathbf{x} = (\mathbf{p}, \phi) \in \mathbf{R}^r$ , from the initial to the final pose.
3. The *trajectory planning task* is executed by the TP in one of the two following modes:
  - Assuming that a *set of constraints* (e.g. continuity or smoothness) on position, velocity and acceleration of the manipulator's joint variables has been *explicitly*



**Fig. 2** Trajectory planner task and I/O representation

*specified* at selected joint configurations (called *support*—or *interpolating points*) along the trajectory, the TP selects then a parameterised trajectory from a class of polynomial functions in the total travelling time interval, which interpolates and satisfies the imposed constraints at the support points.

- A path that the end-effector must traverse is explicitly specified by an analytical function (e.g. a 3D straight-line path, a 2D circular-arc path in Cartesian coordinates or any computed curve), and the TP adds a time law to compute a trajectory that approximates the desired path either in joint coordinates or in Cartesian coordinates.

In the first mode, the constraint specification and the planning of the manipulator trajectory are performed in *joint coordinates*. In the second mode, the path constraints are specified in *Cartesian coordinates*, and the joint actuators are servoed in joint coordinates.

To compute a joint-space trajectory, a given end-effector path must be transformed into a joint-space path via the Inverse Kinematics (IK) mapping. Due to the difficulty of computing on line this mapping, the usual approach is to compute a discrete set of joint vectors along the end-effector path (joint support vectors), and then to interpolate in joint space between these support points in order to complete the joint-space trajectory. Common approaches to trajectory interpolation include polynomial spline interpolation using trapezoidal velocity profiles and time laws of blended polynomial type, cubic polynomial trajectories, or trajectories generated by reference models [5].

## 2 The Trajectory Generation Problem in Robot Motion Control

A path can be defined either in the joint space or in the operational space. Usually, the latter is preferred since it allows:

- a natural description of the task the manipulator has to do,

- a *simple description of the path constraints*—these are due to regions of the workspace which are forbidden to the manipulator (e.g. due to the presence of obstacles), and
- a *direct knowledge of the pose* of the end-effector in the workspace [6].

A geometric path cannot be fully specified by the user due to complexity reasons. Typically, a reduced number of parameters are specified, such as: final points, possible intermediate points, geometric primitives interpolating the points. Also, the *motion time law* is not typically specified at each point of the geometric path, but rather it regards: the total trajectory time, the constraints on the maximum velocities and accelerations or the eventual assignment of velocity and acceleration at some points of particular interest.

This section presents algorithms and implementing solutions for operational-space and joint-space and motion planning. Real-time computational aspects and performances are analysed.

## 2.1 Joint-Space Trajectory Planning

For this type of trajectory planning, the time history of all joint variables and of their first two derivatives is planned to describe the desired motion of the manipulator. Planning in the joint space has the following advantages:

- the trajectory is planned directly, in terms of the controlled joint variables  $\mathbf{q}(t)$  during motion execution;
- the trajectory planning can be done nearly in real time;
- the joint trajectories are planned with a reasonable computational effort.

The main disadvantage is the difficulty in determining the locations of the various links and of the end-effector in the operational space, a condition which is usually required to guarantee obstacle avoidance along the trajectory.

The global algorithm for generating joint-trajectory set points is given next:

```

 $t = t_0 ;$ 
loop:   wait for next control interval;
 $t = t + \Delta t ;$ 
  Update the trajectory planner  $\text{tp}(t) \rightarrow$  compute the necessary joint posture
  of the manipulator:  $\{\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t)\}$  at time  $t$  ;
  if       $t = t_{\text{final}}$     exit;
  else    go to loop.

```

Four constraints are imposed to the planned joint-space trajectory:

1. The trajectory set points must be non-iteratively readily calculable.
2. Intermediate points must be evaluated in a deterministic mode.

3. The continuity of the joint position and its first two time derivatives must be guaranteed so that the planned joint trajectory is smooth.
4. Extraneous motions must be avoided.

The constraints 1–4 for the planned trajectory will be satisfied if the *time history* of the joint variables can be specified by *polynomial sequences*.

Robot controllers use *electronic gearing* in the joint-space trajectory generator in order to synchronize the movement of one or more slave axes to the movement of a master device, which can be an encoder, A/D/C, or the trajectory of another axis, e.g. the robot's leading axis which must execute the longest displacement.

## 2.2 Operational-Space Trajectory Planning

The general case of Cartesian-space planning is considered, for which the global algorithm is given below:

```

 $t = t_0 ;$ 
loop:   wait for next control interval;
 $t = t + \Delta t ;$ 
  Update the operational hand planner  $\mathbf{TP}(t) \rightarrow$  compute the necessary
  position and orientation of the end-effector:  $\{\mathbf{p}(t), \phi(t), \dot{\mathbf{p}}(t), \omega(t)\}$ 
  in the operational space at current control time interval  $t$ ;
  Compute the closed IK joint solution – CIKS,  $\text{IK}[\mathbf{TP}(t)]$ , corresponding to
   $\mathbf{TP}(t) ;$ 
  if  $t = t_{\text{final}}$  exit;
  else go to loop.

```

In general, task-space planning is done in two steps:

- STEP 1: Generating or selecting the *set of support points in operational coordinates* according to some rules, along the operational path
- STEP 2: Specifying a *class of functions* to link the support points defined in STEP 1 (or to approximate the path segments) according to some criteria. The criteria which are chosen are often dictated by the control algorithm following the trajectory planning, which tracks the desired path.

There are two approaches which can be used for achieving STEP 2:

1. The *operational space—oriented* approach: support points are generated along the task path in operational coordinates. Then, the TP interpolates in operational space between these support points and adds the time law expressed in terms of the desired speed and acceleration profiles. The result will be the discrete

time sequence of values that must be attained by the end-effector's position and velocity computed in the task space, i.e. *the trajectory in the task space*.

Next, the task-space trajectory is converted into the corresponding *joint-space trajectory*, by applying for inverse kinematics computation. Several techniques can be used to this purpose:

- The kinematics inversion using the Jacobean pseudo-inverse  $\mathbf{J}^+$  or the Jacobean transpose  $\mathbf{J}^T$  [1, 7];
- The resolved motion rate control (RMRC) algorithm in the form:

$$\delta \mathbf{q}_c(t) = \mathbf{J}^{-1}(\mathbf{q}_c(t)) \delta \mathbf{x}_c(t)$$

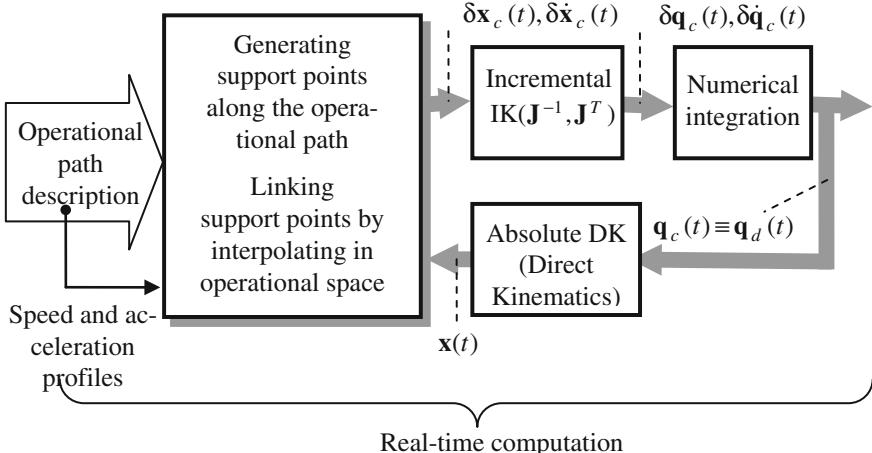
where:

$$\delta \mathbf{q}_c(t) = \mathbf{q}(t_{k+1}) - \mathbf{q}(t_k), \quad \delta \mathbf{x}_c(t) = I_{OS}(\mathbf{x}_d(t_{k+1}) - \mathbf{dk}(\mathbf{q}(t_k)))$$

This corresponds to an incremental IK task space computation  $I_{OS}$ ,  $\delta \mathbf{x}_c(t)$  being the incremental displacement along the operational path, and  $\mathbf{dk}(\mathbf{q})$  is the time function that computes the Direct Kinematics model. Hence it can be observed that the two time-consuming computing tasks:

- operational path interpolation between support points, and
- conversion of the task-space trajectory to a joint-space trajectory

are performed *incrementally* with arguments representing relative position and speed values. This will consequently reduce the computation time and augment the bandwidth of the TP (Fig. 3).



**Fig. 3** Linking support points by interpolating in operational space

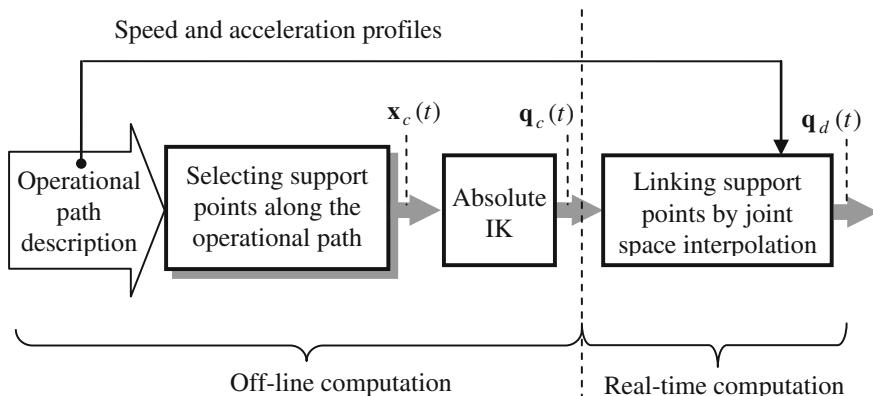
Moreover, in order to additionally reduce the computing time, truncation of results and approximations like  $\sin \theta \approx 0$ ,  $\cos \theta \approx 1 - \theta^2/2$  for  $|\theta| < \varepsilon$  with  $\varepsilon$  a small, positive quantity, or reduced-order series development are accepted for the IK computation task. This is because any induced errors will be compensated by the DK task, placed on the feedback path and operating with the absolute values of the argument “ $\mathbf{q}_c$ —the computed joint configuration on the operational path” [5, 8]. In the case of linear interpolation, the resulting output trajectory generated by the TP is a *piecewise straight line* in the task space.

Attention must be paid as IK transformations do not produce unique solutions; in addition, if the manipulator dynamics is included in the trajectory planning, then path constraints will be specified in operational coordinates, while physical constraints such as force, torque, velocity and acceleration limits of each joint motor will be bounded in joint coordinates.

2. The ***joint space***—oriented approach: converts first the support points that have been defined along the operational path into their corresponding joint coordinates, and the uses *low-degree polynomial functions* to interpolate between these converted support points (Fig. 4). Figures 3 and 4 represent two approaches used for interpolating between support points generated along the operational path.

If the TP must generate a linear trajectory in the task space, the support points will be on this linear path, but the linear joint-space interpolation between support points will produce a final output trajectory which is a non-piecewise straight line in the task space. According to the maximum allowed deviation in position of the planned trajectory with respect to the ideal, linear one in the task space, a certain number of support points will be defined on the operational path. The smaller the admitted deviation, the larger the number of support points to be defined on the operational path. This second approach is widely used, because of its reduced computational effort.

In the **trajectory tracking** stage, the computed reference trajectory is input to the motion controller, whose function is to determine the end-effector to *track the given trajectory as close as possible*. The trajectory tracking task is executed in real time



**Fig. 4** Linking support points by interpolating in joint space

by the motion controller and consists in computing the time history of *joint control inputs*  $\mathbf{u}$ , i.e. the vector of control voltages for the  $n$  axes' servomotors.

Task description is in most cases expressed in the  $m$ -dimensional operational space (with a particular minimal representation for the end-effector orientation), whereas control inputs (control velocities or forces/torques for the joint actuators) are generated in the  $n$ -dimensional joint space. Consequently, two types of motion control schemes have been thought of: with *joint-space* trajectory tracking and with *operational-space* trajectory tracking.

### 3 The Taxonomy of Visual Robot Servoing

The AI approach to intelligent robot automation is best characterised as the attempt to provide a robot with a symbolic representation of its environment and of its own actions, to be exploited by some kind of inference procedure. In this field, the main contributions of AI have been significant in two directions [9–11]:

- *perception*, with particular regard to object recognition and locating through vision;
- *planning*, i.e. the automatic construction of a sequence of actions capable to achieve a predefined goal.

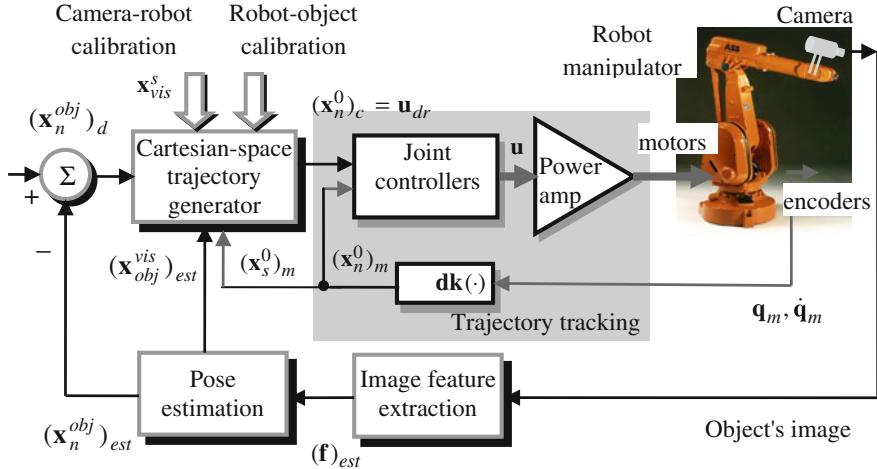
The *behavioural intelligence* of a robotic system refers to the following properties:

1. *Flexibility*: in different situations, the robot controller is able to produce appropriately different behavioural patterns in pursuit of different goals.
2. *Robustness*: the robotic system can absorb and neutralise the effects of incomplete and noisy information and of limited changes in the environment's structure and dynamics.
3. *Adaptiveness*: the ability of the robotic system to alter behaviour significantly in response to radical changes in the environment.

**Robot-vision** systems use intelligent image processing to detect, recognize or track object features and act in consequence to plan and guide the motion of the robot. The chapter introduces the Look-and-Move approach for guidance vision (visually planning the robot's motion—the industry solution), see Fig. 5.

This is a *hierarchical* motion control structure, with the vision processor providing (planning) set-points as references to the robot's joint-level controller—thus using joint data feedback to internally stabilise the robot. This structure leads to an *interlaced look-and-move* control scheme, where motion tracking and image processing are *pipelined* as follows:

- while a motion segment is executed, no image is acquired and processed, and
- while an image is taken and treated according to the specific needs of a robot task, the motion controller does not start generating a trajectory and tracking it.



**Fig. 5** Position-based look-and-move visual servoing architecture for object tracking

It can be observed that, whereas the global robotic system operates in an *open loop structure at motion control level*, it is subject to a *closed loop control at the global task level*.

*Position-based look-and-move control* is further discussed in this section. As described in Fig. 5, features are extracted from the image and used to estimate the pose  $\hat{\mathbf{x}}_{obj}^{vis} = (\mathbf{x}_{obj}^{vis})_{est}$  of the target (object, point) with respect to the camera. Using these values, an error between the current estimated and the desired pose of the robot,  $(\mathbf{x}_{obj}^{vis})_d$  is defined in the task space  $\mathcal{T}$ . Thus, position-based control neatly separates the control actions, i.e. the computation of the feedback signal  $(\mathbf{x}_s^0)_m = \mathbf{dk}(\mathbf{q}_m, s)$ ,  $n - 3 \leq s \leq n$  using the direct kinematics model  $\mathbf{dk}(\cdot)$  of the robot manipulator, from the estimation problem involved in computing position or pose  $\hat{\mathbf{x}}_{obj}^{vis}$  from visual data  $(\mathbf{f})_{est}$ .

A visual positioning task is expressed by an error function  $\mathbf{E} : \mathcal{T} \rightarrow \mathbb{R}^m$ . This function is referred to as the *virtual kinematic error function* VKE. A positioning task is fulfilled when the end-effector has been moved in pose  $\mathbf{x}_n = \mathbf{x}_n^0$  if  $\mathbf{E}(\mathbf{x}_n) = \mathbf{0}$ .

Once a suitable VKE function defined and its parameters instantiated from visual data, a compensator can be designed that reduces the value of the VKE function to zero. This compensator computes at every sampling time instant the necessary end-effector position  $(\mathbf{x}_n)_c$  that is sent as dynamic reference to the joint-space (or operational-space) motion tracking controller [1]. Since the VKE functions are defined usually in the Cartesian space, it is common sense to develop the compensator's control law through geometric insight.

## 4 Guidance Vision for Robot Motion Planning

The problem of visual feature tracking for robot motion planning and object access control will be further presented for two types of working environments: (1) fixed scene, e.g. workstation, storage, ASRS, and (2) mobile scene, e.g. conveyor belts [12, 13].

### 4.1 Open, Vision-Based Robot Motion Planning for Fixed Scene Foreground

An open, vision-based robot motion planning and control method and implementing solution is presented in this section. The method allows using any general purpose machine vision system (here an industrial camera with c-mount and AdeptSight software) with any type of industrial robot controller (here ABB), with a proper interfacing (Ethernet or serial communication).

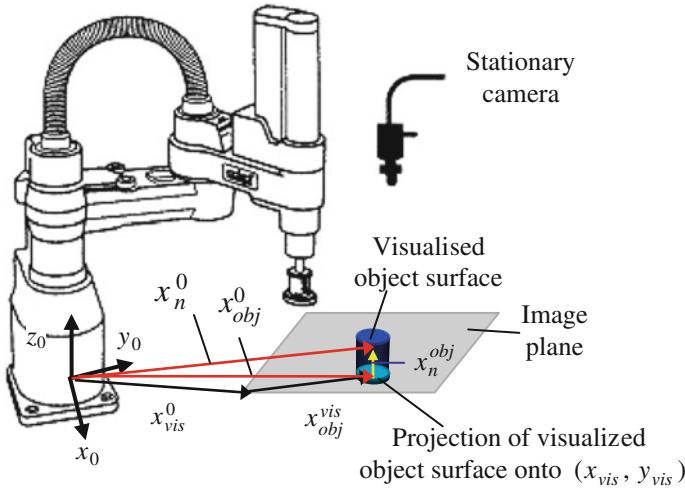
In order to be used, a camera calibration is needed (which is provided by vision any image processing library based on a calibration pattern), and also a robot-camera calibration (which must be done manually by the robot technician); the models of the objects to be accessed by the robot and the robot-object (class) grasping will be off-line taught for collision free motion at execution time.

In industrial applications of position-based dynamic look-and-move control structures, the robot-vision system works in most cases with off line learned objects which can be visually recognised and located at run time [14, 15]. It becomes thus possible:

- to recover the object's pose,  $\hat{\mathbf{x}}_{obj}$ , relative to the base frame of the robot, from the direct estimate  $\hat{\mathbf{x}}_{obj}^{vis}$  of the object's pose in the vision frame and by composing it with the camera-robot calibration estimate  $\hat{\mathbf{x}}_{vis}$ ;
- to define stationing points  $\mathbf{S}^{obj}$  on the object's image, relative to a suitable object-attached frame  $(x_{obj}, y_{obj})$ .

Figure 6 shows a fixed camera configuration and related camera-robot transformations; this is an *endpoint open-loop* (EOL) system that only observe the target object to guide the robot's motion for grasping it.

The physical camera is related to the base coordinate system of the robot by the time-invariant pose evaluated a single time during an interactive off line camera-robot calibration session, and to the object in the scene by. The camera image of the object is independent of the robot motion (unless the target is the end-effector itself, described for example by image feature of the gripper's fingerprints projected onto the image plane). The pose is computed at run time, and involves the search, recognition and locating of image features(s) on the object of interest [16–18].



**Fig. 6** Stationary camera configuration and related camera—robot relative transformations  $x_{obj}^0, x_n^0$  respectively for the *feature tracking* and *feature tracking for object grasping* tasks

For object grasping, the image features must unambiguously describe the entire object for its successful identification and locating at run time. In addition, the pose of the gripper, relative to the frame attached to the object in its current location, is required.

For a *stationary camera*, the relationship between these poses is:

$$\mathbf{x}_n^0 = \mathbf{x}_{vis}^0 : \mathbf{x}_{obj}^{vis} : \mathbf{x}_n^{obj}, \text{ for feature tracking for object grasping.}$$

Assuming a random part presentation in the robot workstation, the object's pose relative to a (unique) camera frame,  $\hat{\mathbf{x}}_{obj\_1}^{vis}$  will be estimated at run time in a *first stage* in terms of the following image feature parameters:

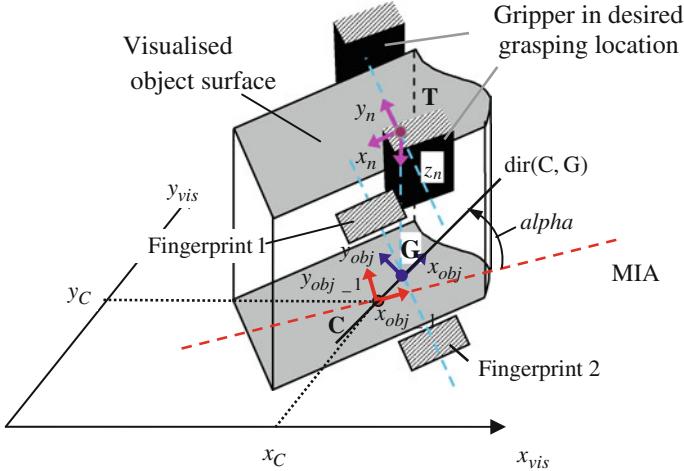
- $x_C, y_C$ : coordinates of the centre of mass C of the 2D projection of the object's visualised surface onto the image plane ( $x_{vis}, y_{vis}$ );
- $orient = \angle(\text{MIA}, x_{vis})$ : orientation angle of the object.

The object-attached frame ( $x_{obj\_1}, y_{obj\_1}$ ) has the origin in C and the abscissa  $x_{obj\_1} \equiv \text{MIA}$ , where MIA stands for the object's Minimum Inertia Axis (Fig. 7).

To move the robot to grasp objects of a certain class always in the same way, irrespective of their location in the robot scene, the desired (unique) pose of the gripper,  $\mathbf{x}_{n^*}^{obj}$ , relative to the object-attached frame must be a priori learned.

Let us denote by  $\mathbf{G}$  the projection of the end-tip point  $\mathbf{T}$ , the origin of the gripper-attached frame ( $x_n, y_n, z_n$ ), onto the image plane:  $\mathbf{G} = \text{projl}_{(x_{vis}, y_{vis})}\{\mathbf{T}\}$ .

For a desired grasping style,  $\mathbf{G}^{obj\_1}$  is a stationing point in the object's coordinates ( $x_{obj\_1}, y_{obj\_1}$ ), irrespective of the current position and orientation of the object. Its



**Fig. 7** Definition of the object-attached coordinate frame

coordinates are:  $x_G = d_{CG} \cdot \cos(\alpha)$ ;  $y_G = d_{CG} \cdot \sin(\alpha)$ , where  $d_{CG} = \text{dist}(C, G)$ , and  $\alpha = \angle(\text{dir}(C, G), \text{MIA})$  measured CCW from the Minimum Inertia Axis MIA to  $\text{dir}(C, G)$ . In a second stage, the object-attached frame will be shifted to origin  $G$ , by a translation of distance  $d_{CG}$  along  $\text{dir}(\text{MIA})$  followed by a rotation of angle  $\alpha$  about the normal in  $C$  to the image plane, as represented in Fig. 7.

Given an object pose,  $\mathbf{x}_{obj}^{vis}$  estimated visually at run time, and assuming that the object was recognised as a member of that class for which a relative grasping pose  $\mathbf{x}_{n^*}^{obj}$  was *a priori* learned using a stationary camera calibrated to the robot base frame by  $\mathbf{x}_{vis}$ , then the positioning error can be defined by the VKE function

$$\mathbf{E}(\mathbf{x}_n; \tilde{\mathbf{x}}_{n^*}^{obj}, \hat{\mathbf{x}}_{obj}^{vis}, \hat{\mathbf{x}}_{vis}) = \mathbf{x}_{n^*}^n = \hat{\mathbf{x}}_0^n : \hat{\mathbf{x}}_{vis} : \hat{\mathbf{x}}_{obj}^{vis} : \tilde{\mathbf{x}}_{n^*}^{obj},$$

where:

$$\tilde{\mathbf{x}}_{n^*}^{obj} = \begin{cases} \mathbf{x}_{n^*}^{obj} & \text{a priori known from learning, particular "grasping style"} \\ \hat{\mathbf{x}}_{n^*}^{obj} & \text{visually updated at run time, general "grasping style"} \end{cases}.$$

With an EOL system,  $\hat{\mathbf{x}}_0^n = \text{inverse}(\hat{\mathbf{x}}_n^0)$  will be dynamically updated by the trajectory generator to bring to zero the positioning error  $\mathbf{x}_{n^*}^n$ . This can be simply done applying for an IK-based *Resolved Motion Rate Control* algorithm.

The closed-loop servo control uses the visually estimated pose of the object,  $\hat{\mathbf{x}}_{obj}^{vis}$ , the estimated camera-robot calibration pose  $\hat{\mathbf{x}}_{vis}$ , and assumes that reduced-error direct kinematics ( $\hat{\mathbf{x}}_n^0$ )—and inverse kinematics ( $\hat{\mathbf{x}}_0^n$ ) models are available. As for the imposed grasping pose, for a priori unknown object location in the scene, some components in  $\hat{\mathbf{x}}_{n^*}^{obj}$  must be estimated at run time whenever the “style” in which the object will be grasped is *general*, i.e. such that  $\mathbf{G} \not\equiv \mathbf{C}$  and  $\mathbf{G} \notin \text{MIA}$  [19–21].

For object access and handling using vision, the problem is reduced to expressing the object position in the image relative to the robot base. This is done in the *robot-camera calibration* session, the result of which is a relative transformation expressing the position and orientation of the vision frame relative to the robot base. Once the calibration is executed, robot points will be computed relative to the position and orientation of the vision-attached frame, and the robot motion planning follows the procedure described in Sect. 2.

The robot-camera calibration procedure requires the usage of an object that will be handled by the robot; during the execution of the procedure the robot will move the object to different locations and will acquire pictures, generating a set of pairs of descriptions of the object's location: (a) from the camera and (b) from the joint encoders. The solution of this set of equations will describe the camera's field of view location relative to the robot base.

For testing purpose an AdeptSight system and ABB robot manipulator were used, the robot-vision calibration process and the training of the object grasping model have been integrated in a single procedure. The procedure consists in four human-robot interactive steps where the robot grasps the object and places it different positions in the workspace for image acquisition and processing [2]:

The calibration object is placed in the workspace and grasped by the robot and then released (position P1), after which the robot clears the vision plane and the object's position in the vision plane is computed by the AdeptSight library. The point P1 is the point which will be used to express all the positions of the objects in the image. For example a position of an object will be computed as Po where Po is P1 shifted with a set of offsets (for translations on X and Y and rotation on the Z axis). The position of the object in point P1 is also computed in the vision plane, having the coordinates  $P1_{Xv}$ ,  $P1_{Yv}$  (the position of the coordinate system attached to the object model) [22, 23].

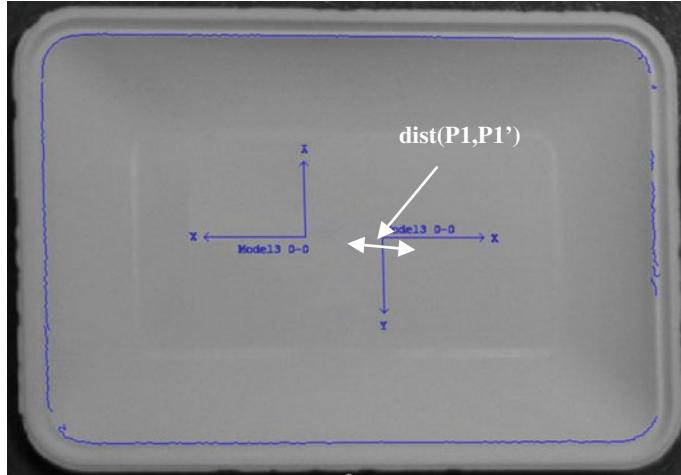
In the second step the robot grasps the object and places it in the same position, but rotated with  $180^\circ$  (point  $P1'$ ), in the vision coordinates  $P1'_{Xv}$ ,  $P1'_{Yv}$ . By comparing the position of the coordinate system of the object in these positions the system can compute the position of the mass centre of the object (the mass centre of the model relative to the grasping point). In this case the grasping point is located in the image on the middle of the segment  $[P1, P1']$  (see Fig. 8).

$$P_{gvvis} \begin{cases} P_{gvvisx} = \min(P1_x, P1'_x) + |P1_x - P1'_x| \\ P_{gvvisy} = \min(P1_y, P1'_y) + |P1_y - P1'_y| \end{cases},$$

where  $P_{gvvis}$  is the grasping point in the vision workspace.

Next the object is placed in a position P2 which is trained relative to the position P1 shifted with 100 mm on X axis of the base coordinate system of the robot.

In the final step the robot places the object in the position P3 which is trained relative to the position P1 shifted with 100 mm on Y axis of the base coordinate system of the robot. By knowing the correspondence robot-point—image-point, the system can compute now the orientation of the vision plane relative to the robot base



**Fig. 8** The relationship robot point—vision point

coordinate system, and also the distance which the robot must cover to reach an object which is placed at a certain distance from the initial point  $P1$  in the image plane.

This can be expressed as follows: for 100 mm travelling length along the X coordinate system (base coordinate system) the object moves in the image  $P2_x - P1_x$  along the  $X_{vis}$  axis, and  $P2_y - P1_y$  along  $Y_{vis}$ ; the same travelling length on the Y coordinate system generates  $P3_x - P1_x$  on  $X_{vis}$  axis, and  $P3_y - P1_y$  on  $Y_{vis}$ , in the vision workspace. It results also that the vision system is rotated with the angle:

$$\alpha = a \tan 2(P2_y - P1_y, P2_x - P1_x)$$

toward the base coordinate system. Hence for an object which is recognized in the image at the location  $P_v$ , the object will be grasped at the coordinates:

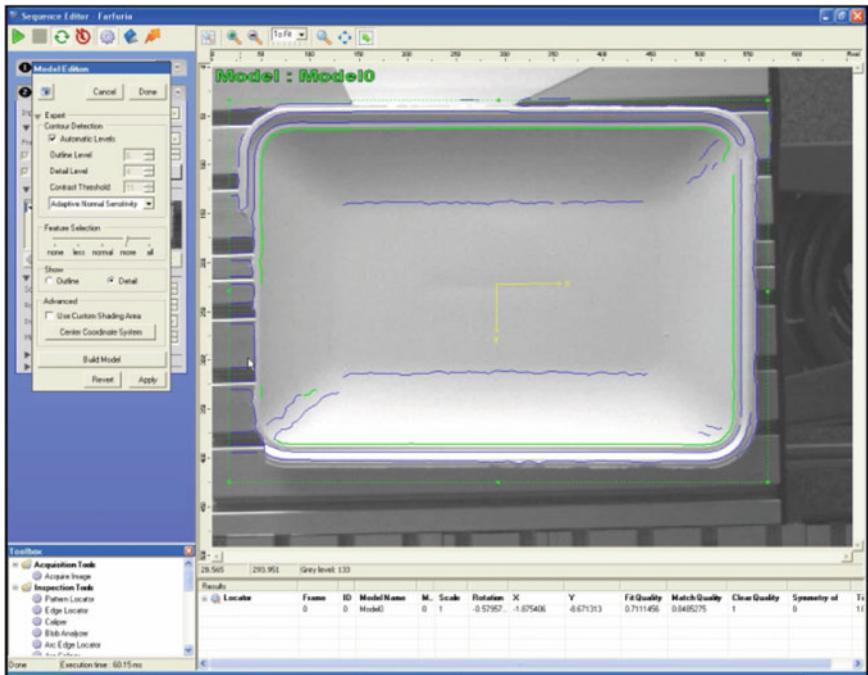
$$\begin{aligned} P_x &= P_{G\_x} + \sqrt{(P1\_x - P_{v\_x})^2 + (P1\_y - P_{v\_y})^2} \\ &\quad \cdot \cos(\alpha + a \tan 2(P_{v\_y} - P1\_y, P_{v\_x} - P1\_x)) \\ P_y &= P_{G\_y} + \sqrt{(P1\_x - P_{v\_x})^2 + (P1\_y - P_{v\_y})^2} \\ &\quad \cdot \sin(\alpha + a \tan 2(P_{v\_y} - P1\_y, P_{v\_x} - P1\_x)) \\ P_{rot} &= P_{G\_rot} + (P_{v\_rot} - P1\_rot) \end{aligned}$$

where  $P_x$ ,  $P_y$ ,  $P_{rot}$  are the position coordinates and the rotation of the grasping point of the object which was located in the vision workspace at the location  $P_v$  ( $P_{v\_x}$ ,  $P_{v\_y}$ ,  $P_{v\_rot}$ );  $P_G$  ( $P_{G\_x}$ ,  $P_{G\_y}$ ,  $P_{G\_rot}$ ) is the grasping point (in the object's centre of the mass in the base coordinates system) for the object located in the image in  $P1$  ( $P1\_x$ ,  $P1\_y$ ,  $P1\_rot$ ).

After the calibration is executed, the object model must be trained; this stage involves object edges processing in order to obtain the geometrical model of the object. The grasping position must be also trained in order to validate a collision free point for accessing the object. The grasping position (for grasping validation) is defined by two or more rectangular areas placed around the object and linked to the object frame. These areas represent the projections of the gripper fingerprints on the image plane and by processing the image colour inside these areas the program detects the presence of obstacles and can invalidate the grasping position.

These three pieces of information are used for robot motion planning; first the location of the field of view is used by extracting it from the calibration data, then the location of the object in the field of view is computed (*online*) using the object model and in the last stage the action of grasping the object is validated by using the grasping model and collision free tests. Experimental results validating the proposed solution are shown from a robotized ceramic production line (Fig. 9).

The experimental application runs two communications threads: a TCP/IP server and a serial communication thread. Both threads have the same role, they are listening and if they receive an acquisition request, they initialize the execution of the AdeptSight sequence of tools (the vision program), returning three numbers specifying the position and orientation of the plate (X, Y in mm, and the angle in degrees). The position and orientation is specified relative to the calibration object.



**Fig. 9** Real-time locating a ceramic plate for robot motion planning and grasping control

The requests are sent as ASCII characters, and they are of two types (*i*—information for debugging or *r*—real requests); when the vision server receive a request the vision sequence is executed, the object is recognized based on its boundary contours, and the values (X, Y and rotation) relative to the initial grasping point (from the calibration procedure) are computed and sent to the ABB robot.

When the robot receives the three values, it shifts the initial grasping position (from the calibration) and grasps the plate. The following pseudo-code describes how the communication is integrated with the vision server [24]:

```

Open the communication channel (Serial line)
Clear the serial line buffer
Request object coordinates from vision
Read the data streams (X, Y coordinates and rotation)
Transform the coordinates from string to real
Request object coordinates from vision
Read the data streams (X, Y coordinates and rotation)
Transform the coordinates from string to real
/*In order to avoid problems caused by communication errors
the coordinates are sent twice and only if they are the
same at the destination then the position can be computed*/
Verify if the coordinates are the same
IF YES
    Compute the grasping position
    //The position is computed relative to a predefined
    //position p1
    Close the communication
ELSE
    Repeat the request

```

The presented image processing system, AdeptSight, is robust, offers generic robot-vision functions, and can be easily integrated with controllers of other industrial equipment (robots, measuring machines, ASRS, part feeders). AdeptSight allows a rapid development of visually planned applications, based on visual tools which can be combined and configured leading to sequences which can be executed from external C# applications.

## **4.2 Multitasking Robot Motion Planning for Object Tracking on Mobile Scenes**

The problem of robot tracking objects of interest moving on conveyor belts and randomly entering the robot's dexterous space can be solved by integrating the following *devices* in a multitasking control structure, implemented on multiprocessor robot controllers:

- the robot manipulator, tracking a conveyor belt;
- the conveyor belt, driven at constant, regulated speed;
- the vision module, inspecting parts on the conveyor belt.

Conceptually, the problem is solved by defining a number of user tasks which attach two types of “robots”: the  $n$ —d.o.f. manipulator grasping on-the-fly objects moving on the conveyor belt, and one  $m \leq 3$ —axis “robot” emulating the conveyor belt under vision control;  $m$  is the number of non-null projections of the conveyor belt displacement direction on the 3 axes of an orthonormal reference frame (e.g., defined in the belt tracking robot environment). These user tasks run concurrently with the internal system tasks of a multitasking belt tracking robot controller, which are responsible for trajectory generation, axis servoing and resources management [20].

In this respect, the minimum number of tasks to be defined for the tracking problem is equal to 3:

- Task 1: Dynamic re planning of the destination location (grasping the moving object) for the robot manipulator.
- Task 2: Continuously moving (driving) the  $m$ -axis vision belt. (e.g.,  $m = 1$ )
- Task 3: Reading once the belt’s location the very moment an object of interest has been recognised, located and its grasping estimated as collision-free, and then continuously until the object is effectively picked.

#### **4.2.1 Tasks and Priorities for the Multitasking Robot Motion Planning Problem**

Consider that each control system cycle of the robot is divided into 16 time slices of one millisecond, the time slices being numbered 0 through 15. A single occurrence of all 16 time slices is referred to as *a major cycle*. For a robot system, each of these cycles corresponds to one output from the trajectory generator to the digital servos. A number of user tasks, e.g. from 0 to 6, can be used and configured to suit the needs of specific applications. Tasks are normally assigned default time slices and priorities according to the current system configuration [5, 8].

An execution cycle is terminated when a STOP instruction is executed, a RETURN instruction is executed in the top-level program, or the last defined step of the program is encountered. Tasks are scheduled to run with a specified *priority* in one or more time slices. Tasks may have priorities from  $-1$  to 64, and the priorities may be different in each time slice. The priority meanings are: 1–31 (normal user tasks); 32–62 (used by robot controller’s device drivers and system tasks); 63 (used by trajectory generator); 64 (used by the servo).

#### **4.2.2 Scheduling Program Execution Tasks with Simultaneous Belt Tracking**

An analysis of the time slice and priority allocation for the system, and of default user tasks imposes several requirements for timing and priority assignment of tasks: *vision guided robot planning* (“object recognition and locating”), and *dynamical re planning of robot destination* (“robot tracking the belt”) should always be configured on user

tasks 0 and/or 1, in “Look-and-Move” interlaced robot motion control applications, due to the continuous assignment of these two tasks, over the first 13 time slices, with high priorities [25].

Because vision guidance and motion re planning programs complete their computation in less than the 13 time slices (0–12), in order to give the chance to conveyor-associated tasks (“drive” the vision belt, “read” the current position of the vision belt) to provide the “robot tracking” programs with the necessary position update information earlier than the slice 13, and to the high-priority trajectory generation system task to effectively use this updates, a WAIT instruction should be inserted in the loop-type vision guidance and motion re planning programs of tasks 0 and/or 1.

All time slices are checked, wrapping around from slice 15 to slice 0 until the original slice is reached. If no runnable tasks are encountered, a null task executes. Whenever a 1 ms interval expires, the multitasking OS performs a similar search of the next time slice. If the next time slice does not contain a runnable task, execution of the current task continues. If more than one task in the same time slice has the same priority, they become part of a *round-robin scheduling group*. Programs that execute in continuous loops, like vision guidance and motion re planning for belt tracking, should generally execute a WAIT instruction occasionally (for example, *once through each loop execution*). This should not be done, however, if timing considerations for the tracking application preclude such execution delays in some stages of vision and motion processing [6, 26].

As previously stated, the problem of conveyor tracking with vision guiding for moving part identification and locating requires the definition of three *user tasks*, to which the following programs were associated:

1. Task 1: program “track” executes in this task, with robot 1 (e.g., SCARA) selected. This program has two main functions, carried out in a 2–stage sequence:
  - STAGE 1: Continuous *checking* whether an object travelling on the conveyor belt (it will be called in the sequel *vision belt*) entered the field of view of the camera and the reachable workspace of the SCARA robot. If such an event occurs, the vision is activated to *identify* whether the object is of interest and to *locate* it. Processing on this stage terminates with the *computation of the end-effector’s location* which would move the SCARA robot in the object picking location evaluated *once* by vision.
  - STAGE 2: Continuously *re planning* the end-effector’s location, computed when the object of interest was located by vision, by *consuming the belt position data* produced by encoder reads in the program “read” which executes on task 3, and by *dynamically altering the robot’s target* in the current motion segment.
2. Task 2: program “drive” executes in this task, with robot 2 ((m = 1)-axis robot, i.e. the conveyor belt) selected. This program *moves the belt* in linear displacement increments, at a sufficiently high rate to provide a jerk-free, continuous belt motion. This program executes in stages 1 and 2 previously defined.

3. Task 3: program “read” executes in this task, with robot 2 selected. This program executes differently in the two stages of the application:

STAGE 1: Executes *a single time* upon receiving an input signal (“la\_reco”, e.g. for “LA” objects of interest) from vision in task 1, confirming the recognition and successful locating of an “LA” part. In response, “drive” *reads the instantaneous belt position*, which from now on will be used as an offset for the position updates.

STAGE 2: Continuously *reads the belt position*, upon a request (“info” in the example of the first case study) issued by “track” in task 1, when it starts its dynamic target re planning process.

From the three user tasks, the default priority assignment is maintained. This leads to the following priority analysis for a major cycle:

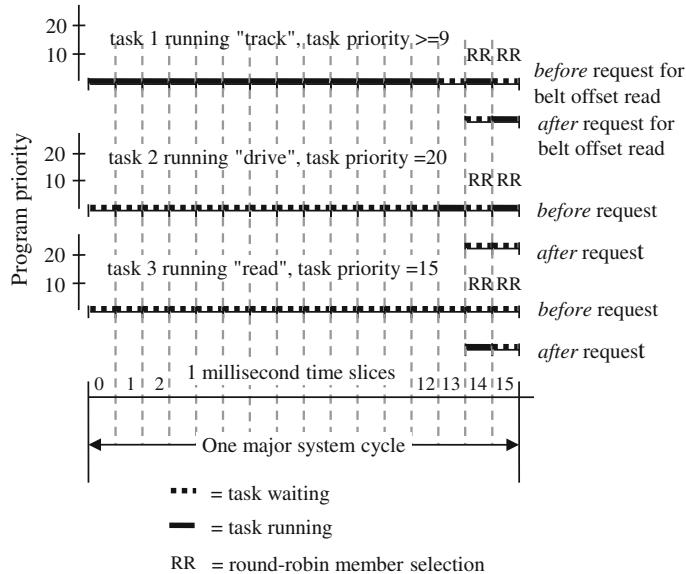
- Task 1 has the highest priority in time slices 0–12 (inclusively), with values of 19, 21, 9 and 11.
- Task 2 has the highest priority (20) in a single time slice: 13.
- Task 3 never detains a position of highest priority with respect to tasks 1 and 2.
- The three tasks become part of a round-robin group as follows:
  - tasks 2 and 3 in slices 0–12 inclusively,
  - tasks 1, 2 and 3 in slices 14 and 15.

Because tasks 2 and 3 are in more than one round-robin group on different slices, then all three tasks in the corresponding pairs of different slices appear to be in a big group. This property can cause, in general, a task to be run in a slice one does not expect; however, this risk is eliminated for task 1 in STAGE 2 since it will never be runnable in slices 14 and 15 (after generating a WAIT).

As for tasks 2 and 3, they cannot generate this risk in the remaining slices from 0–12, after “track” generates the WAIT, because they will switch continuously between them at the beginning of each new time slice.

As a result of the priority scan and scheduling, the programs in the three user tasks execute as follows:

- STAGE 1—vision is processing, the SCARA robot is not moving and no WAIT is issued by task 1 (Fig. 10):
- STAGE 2—*vision is not processing, the SCARA robot is moving and WAIT commands are issued in task 1 by the “track” program after each re planning of the end-effector’s target destination within a V+ major cycle of 16 ms:*
  - Task 1 runs in slices  $i - j, i \leq j, i \geq 0, j \leq 12$ , (when it detains the highest priority), i.e., starting with the time moment when it is authorised to run by the highest-priority system tasks “trajectory generation” and “servo” (in slice  $i$ ), and executing until it *accesses the position update* provided by task 3 *from the most recent belt encoder read, alters the last computed end-effector destination* and issues a WAIT (in slice  $j$ ), to give the trajectory generator a chance to execute.



**Fig. 10** Priority assignment and tasks running in STAGE 1 of vision guidance for motion planning in the belt tracking problem

- Task 2 runs: in slices  $(j + 1) - 12$  switching alternatively with task 3 whenever it is selected as the member of the round-robin group following task 3 that run most recently, in slice 13 (it detains the highest priority), and in slice 15 (it is member of the round-robin group following task 3 that run more recently—in slice 14). Task 2 runs always exactly for 1 ms whenever selected, so that the round-robin group scanning authorises task 3 to run always at the beginning of the next time slice.
- Task 3 runs in slices  $(j + 1) - 12$  switching alternatively with task 2 whenever it is selected as the member of the round-robin group following task 2 that run most recently, and in slice 14 (it is member of the round-robin group following task 2 that run more recently—in slice 13). The task 3 runs, whenever selected, for less than 1 ms and issues a RELEASE “to anyone” command.

#### 4.2.3 Dynamically Altering Belt Locations as Robot Motion References

The three previously discussed user tasks, when runnable and selected by the system’s task scheduler, attach respectively the robots:

- Task 1: **robot 1**—a SCARA-type robot (e.g. Adept Cobra 600) is considered in this case
- Task 2, 3: **robot 2**—the “vision conveyor belt” of a flexible feeding system is considered.

Program “**track**” executing in task 1 has two distinct timing aspects: during STAGE 1, “track” waits first the occurrence of the on-off transition of a signal from the photocell, indicating that an object passed over the sensor and will enter the field of view of the camera. Then, after waiting for a period of time set up function of the belt’s speed, “track” commands the vision system to acquire an image, identify an object of interest and locate it [27, 28].

During STAGE 2, “track” alters continuously, once per each major 16 ms system cycle, the target location of the end-effector, part.loc, that was computed (when one “LA”-part was located by vision and returned in the vis.loc transformation) by composing the following relative transformations (the “:” character stands for composition)

```
part.loc=to.cam[1]:vis.loc:grip.la
```

Here grip.la is the off line learned grasping transformation for the class of “LA” objects. The updating of the end-effector target location for picking-on-the-fly “LA” objects according to a predefined grasping style uses the V+ operation:

```
ALTER () Dx,Dy,Dz,Rx,Ry,Rz
```

which specifies the magnitude of the real-time path modification that is to be applied to the robot path during the next trajectory computation ( $D_x, D_y, D_z / R_x, R_y, R_z$  are the translations/ rotations respectively along the  $X, Y, Z$  axes).

This operation is executed by “track” in task 1 that is controlling the robot 1 (SCARA) in *alter mode*, enabled by the ALTON command. When *alter* mode is enabled, this instruction should be executed *once during each trajectory cycle*. The stopping decision is taken in “track” by using the STATE (select) function, which returns information about the state of robot 1 (“Motion stopped at planned location”) selected by task 1 executing the ALTER loop. The ALTOFF operation was used to terminate real-time path-modification mode (alter mode) [3, 10, 14].

Program “**drive**” executing in task 2 has a unique timing aspect in both STAGES 1 and 2: when activated by the main program, it issues continuously motion commands for the individual joint number 1 of robot 2—the vision belt.

Program “**read**” executing in task 3 evaluates the current motion of robot 2—the vision belt along its single axis, in two different timing modes. During STAGE 1, upon receiving from task 1 the request 1a\_reco (an instance of “LA” was recognised) to compute the belt’s offset, reads the current robot 2 location and extracts the component along  $Y$ .

This invariant offset component, read when the “LA” was successfully located by vision and the grasping authorised as collision-free, will be further used in STAGE 2 to estimate the updates of the  $y\_off$  motion, to alter the SCARA robot’s target location along the  $Y$  axis.

The program below shows how the STATE function is used to stop the continuous updating of the end-effector’s target location by altering at every major cycle the position along the  $Y$  axis. The altering loop will be exit *when motion stopped at planned location*, i.e. when the robot’s gripper, moving to track the part

travelling on the conveyor belt, arrives in the imposed picking position relative to the moving part.

```

ALTON () 2      ;Enable altering mode

;The robot is commanded to move towards the grasping position
;computed when the object was VLOCATED by vision.

MOVES part.loc
WHILE STATE(2)<>2 DO

;While the robot is far from the moving target (motion not
;completed at planned location...

ALTER(),-pulse.to.mm*y_off

;Continuously alter the target grasping location

WAIT

;Wait for the next major time cycle to give the trajectory
;generator a chance to execute

END
ALTOFF          ;Disable altering mode
CLOSEI          ;Robot picks the tracked object
DEPARTS        ;Robot exits the belt tracking mode
MOVES place

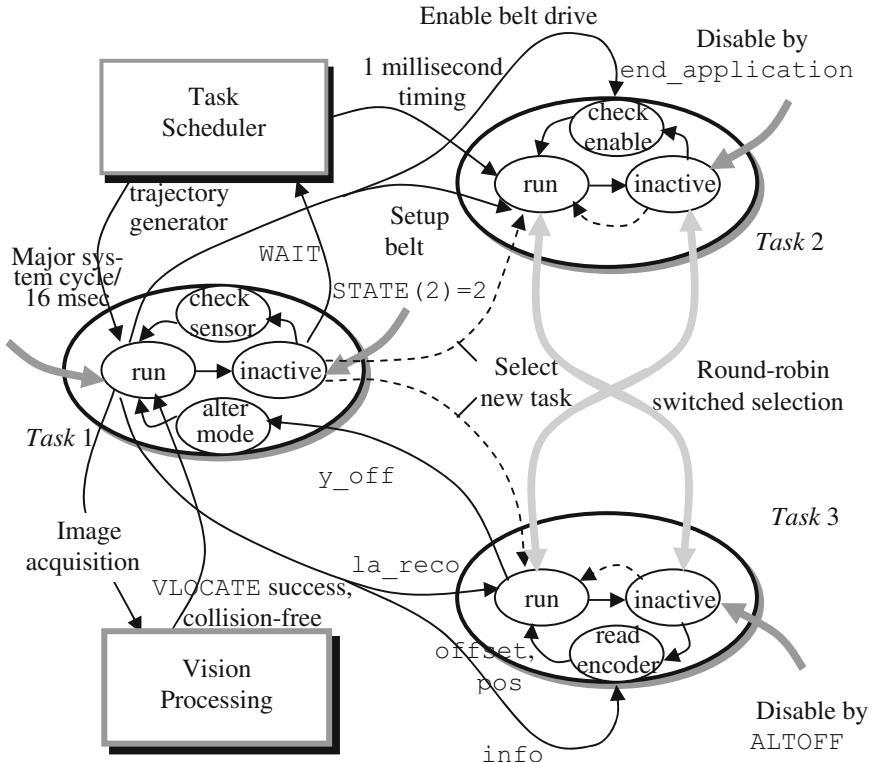
;Robot moves towards the fixed object-placing location place

```

In the example presented, the ALTOFF operation has been used to terminate real-time path-modification mode (alter mode). The instruction suspends program execution until any previous robot motion has been completed (similarly to a BREAK instruction), and then terminates real-time path-modification mode.

After *alter* mode terminates, the robot is left at a final location that reflects both the destination of the last robot motion and the total ALTER correction that has been applied [13, 17, 29].

The cooperation between the tasks on which run “track”, “drive” and “read” is shown in Fig. 11.



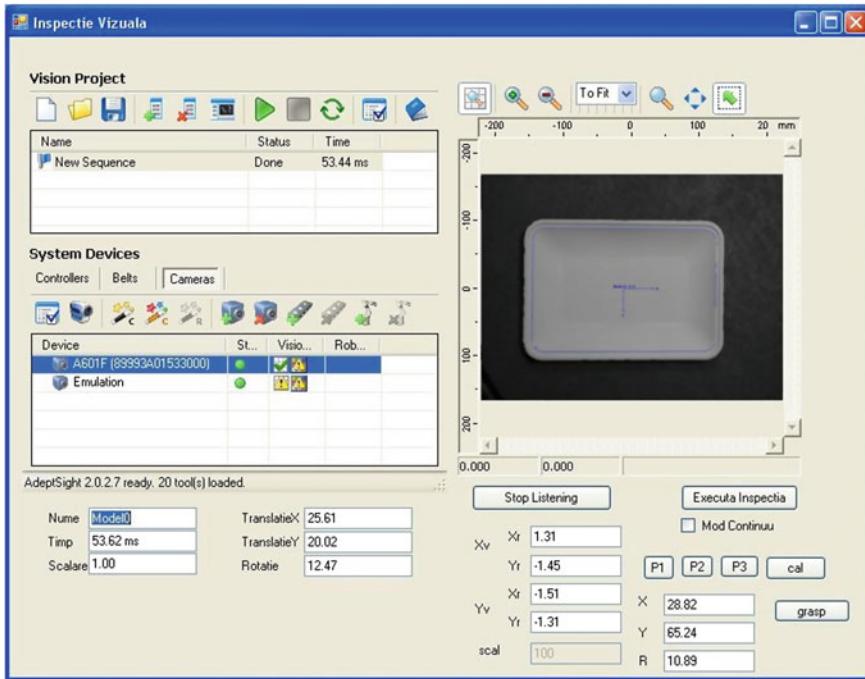
**Fig. 11** Cooperation between tasks in the robot-vision belt tracking problem

## 5 Experimental Results and Conclusions

Visual robot motion planning allows relaxation of the numerous constraints which arise when setting up a manufacturing environment, as well as the need for high-precision material transportation and presentation devices, such as conveyors, vibrating bowls, a.o. The *look-and-move motion planning* methodology offers a robust solution to create workstations with components from different manufacturers: image sensors and cameras, vision software, robot manipulators, shop floor conveyors and other mechanical devices.

The open architecture system for vision-based robot motion planning application was developed in C# and managed the robot-vision communication and sequence execution. Also the camera-robot calibration procedure and the learning of the grasping model learning were developed in the same open system concept based on standard communication means [30, 31].

Figure 12 shows a screen capture of the open architecture robot-vision user application interface. The application consists in precision locating with AdeptSight of



**Fig. 12** Screen of the application interface for high-precision plate locating and vision-based robot motion planning for stationary plate grasping

ceramic plates travelling on a conveyor belt and guiding the motion of an ABB robot with help of the location data sent via standard communication channels and interfaces.

The motion control method presented above for robots picking on-the-fly objects on moving scenes was implemented in the V+ robot programming environment with AdeptSight vision extension, and tested on a robot vision platform containing one Adept Cobra 600 SCARA-type manipulator, a 3-belt flexible feeding conveyor Adept Flex Feeder 250 and a stationary, down looking matrix camera Panasonic GP MF 650 inspecting the vision belt with backlighting [10]. The vision belt on which parts are travelling and are viewed by a fixed, down looking camera was positioned parallel to the  $Y_0$  axis of the manipulator, for a convenient robot access within a window of 460 mm. Experiments have been carried out at several speed values of the conveyor belt, in the range from 5 to 180 mm/s.

Table 1 shows the correspondence between the belt speeds and the maximum time intervals from the visual detection of a part up to its effective grasping.

It can be observed that at the maximal speed of 180 mm/s, the robot-vision multitasking controller is still able to direct the SCARA-type manipulator to access visually detected, recognised and located objects.

**Table 1** Correspondence between belt speed and part access time

Belt speed (mm/sec)	5	10	30	50	100	180
Grasping time (max) (sec)	1.4	1.6	1.9	2.0	2.3	2.5

In the experiment reported in Chap. 4.1, the vision library was successfully interfaced to an ABB 1570 vertical articulated robot.

The novelty of the research consist in developing an open architecture system for vision-based robot motion planning, allowing to use closed vision systems (here AdeptSight), that can be integrated with proprietary systems (for example AdeptSight has native functions which can be integrated only with Adept robots), with any other devices (robots, machines, feeders, a.o.) using standard communication mechanisms (serial line or Ethernet). Another novel contribution is the multitasking solution for picking objects in motion from any type of conveyor modelled as a  $m \leq 3$  degree of freedom Cartesian robot.

## References

1. Siciliano B, Sciavicco L, Villani L, Oriolo G (2010) Robotics, modelling, planning and control. Springer, Berlin
2. Borangiu Th, Ecaterina O, Manu M (2000) Multi-processor design of nonlinear robust motion control for rigid robots. Lecture notes in computer science, vol 1798. Springer, Berlin, pp 224–238
3. Borangiu Th (2002) Advanced robot motion control. Romanian Academy Press, Bucharest
4. Braun BM, Starr GP, Wood JE, Lumia R (2004) A framework for implementing cooperative motion on industrial controllers. IEEE Trans Robot Autom 20:583–589
5. Gueaieb W, Karray F, Al-Sharhan S (2003) A robust adaptive fuzzy position/force control scheme for cooperative manipulators. IEEE Trans Control Syst Technol 11:516–528
6. Battilotti S, Lanari L (1996) Tracking with disturbance attenuation for rigid robots. In: Proceedings of IEEE international conference on robot automation, Minneapolis, April 1996, pp 1570–1583
7. Borangiu Th, Anton F, Dumitrache A (2010) Robot programming. AGIR Publishing House, Bucharest
8. Kawasaki H, Ueki S, Ito S (2006) Decentralized adaptive coordinated control of multiple robot arms without using a force sensor. Automatica 42:481–488
9. Borangiu Th, Ionescu F, Manu M (2003) Visual servoing in robot motion control. In: Proceedings of 7th multi-conference on systemics, cybernetics and informatics SCI'03. Orlando, 27–30 July 2003, pp 987–992
10. Hutchinson S, Hager G, Corke P (1996) A tutorial on visual servo control. IEEE Trans Robot Autom 12:6561–6670
11. Xie WF, Li Z, Tu XW, Perron C (2009) Switching control of image based visual servoing with laser pointer in robotic assembly systems. IEEE Trans Ind Electron 520–529
12. Mendes JM, Restivo F, Leitao P, Colombo A (2010) Injecting service-orientation into multi-agent systems in industrial automation. Lecture notes in computer science, vol 6114, pp 313–320
13. Allotta B, Fioravanti D (2005) 3D motion planning for image-based visual servoing tasks. In: Proceedings of the IEEE international conference on robotics and automation, Barcelona, pp 2173–2178

14. Nelson BJ, Papanikolopoulos P, Khosla PK (1996) Robotic visual servoing and robotic assembly tasks. *IEEE Robot Autom Mag* 23:97–102
15. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *J Comput Vis* 60(2):91–110
16. Hossu A, Borangiu Th, Croicu A (1995) Robot Visionpro machine vision software for industrial training and applications. Version 2.2, Cat. #100062, Amsterdam, Tel Aviv, New Jersey, Eshed Robotec
17. Wilson W, Hulls C, Bell G (2006) Relative end-effector control using Cartesian position-based visual servoing. *IEEE Trans Robot Autom* 12:684–696
18. Miyabe T, Konno A, Uchiyama M, Yamano M (2004) An approach toward an automated object retrieval operation with a two-arm flexible manipulator. *Int J Robot Res* 23:275–291
19. ABB, Technical Reference Manual, RAPID Instructions, Functions, and Data types, 2004
20. Adept Reference Guide, V+ Programming, Adept Technology Inc. 2011
21. Bilen H, Hocaoglu M, Unel U, Sabanovic A (2012) Developing robust vision modules for microsystems applications. *Mach Vis Appl* 23(1):25–42
22. Borangiu Th, Ivanescu N, Brotac S (2002) An analytical method for visual robot -object calibration. In: Proceedings of the 7th international workshop robotics in Alpe-Adria-Danube region RAAD'98. Balatonfüred, pp 149–154
23. Chaumette F, Hutchinson S (2006) Visual servo control. Part I: basic approaches. *IEEE Robot Autom Mag* 13(4):82–90
24. Martinez-Rosas JC, Arteaga MA, Castillo-Sanchez A (2006) Decentralized control of cooperative robots without velocity-force measurements. *Automatica* 42:329–336
25. Gudifio-Lau J, Arteaga MA (2006) Dynamic model, control and simulation of cooperative robots: a case study, mobile robots, moving intelligence. ARS/pIV
26. Chaumette F, Hutchinson S (2005) A general and useful set of features for visual servoing. *IEEE Trans Robot Autom* 21:1116–1127
27. Lippiello V, Siciliano B, Villani L (2007) Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration. *IEEE Trans Robot* 23:73–86
28. Borangiu Th (2004) Intelligent image processing in robotics and manufacturing. Romanian Academy Press, Bucharest
29. Corke P, Hutchinson S (2001) A new partitioned approach to image-based visual servo control. *IEEE Trans Robot Autom* 17:507–515
30. Lazar C, Burlacu A (2009) Visual servoing of robot manipulators using model-based predictive control. In: Proceedings of the 7th IEEE international conference on industrial informatics, Cardiff, pp 690–695
31. Lazar C, Burlacu A, Copot C (2011) Predictive control architecture for visual servoing of robot manipulators. In: Proceedings of the 18th IFAC world congress, Milano, pp 9464–9469

# Grasping and Manipulation of Unknown Objects Based on Visual and Tactile Feedback

Robert Haschke

**Abstract** The sense of touch allows humans and higher animals to perform coordinated and efficient interactions within their environment. Recently, tactile sensor arrays providing high force, spatial, and temporal resolution became available for robotics, which allows us to consider new control strategies to exploit this important and valuable sensory channel for grasping and manipulation tasks. Successful dexterous manipulation strongly depends on tight feedback loops integrating proprioceptive, visual, and tactile feedback. We introduce a framework for tactile servoing that can realize specific tactile interaction patterns, for example to establish and maintain contact (grasping) or to explore and manipulate objects. We demonstrate and evaluate the capabilities of the proposed control framework in a series of preliminary experiments employing a  $16 \times 16$  tactile sensor array attached to a Kuka LWR arm as a large fingertip.

**Keywords** Grasping · Tactile servoing · Online motion planning

## 1 Introduction

The sense of touch allows humans to perform coordinated and efficient interactions within their environment. Without the sense of touch, subjects have severe difficulties maintaining a stable grasp or performing a complex action such as lightning matches [1, 2]. Also in robot applications, lacking tactile feedback results in loosing an initially grasped object or failing to robustly carry out manipulation tasks [3]. In recent years, the resolution and sensitivity of tactile sensors only sufficed for basic force feedback during blind grasping [4]. However, tactile sensor arrays providing high spatial and temporal resolution as well as high sensitivity [5, 6] emerged recently, allowing for more advanced control methods involving tactile feedback too.

---

R. Haschke (✉)  
Cognitive Interaction Technology Excellence Cluster (CITEC),  
Bielefeld University, Inspiration 1, 33619 Bielefeld, Germany  
e-mail: rhaschke@techfak.uni-bielefeld.de

Such control approaches—which we denote as tactile servoing in accordance to corresponding control approaches involving direct visual feedback—require advanced tactile perception methods and their integration into control programs for direct robot control. Tactile servoing includes important tasks like sliding a finger tip along an object’s surface, tracking specific surface structures like ridges, searching for distinctive tactile patterns, or exploring the object shape by groping. Most of these tasks are essential for both in-hand object manipulation [7], and haptic object identification [8].

Drawing on ideas for visual servoing and applying image processing algorithms to the tactile force image provided by modern tactile sensor arrays, it is possible to extract basic tactile features in real time and employ them for robot control. The challenging mission is to find generic features, which not only work in specific hard-coded control scenarios on a specific type of tactile sensor, but that generalize to a rich set of control tasks and sensor types.

We argue for a unified and open control framework that can cover many grasping and manipulation tasks including tactile exploration. The proposed control approach facilitates the exploitation of task symmetries to unleash redundancies which can be efficiently utilized by subordinated tasks. Different, challenging tasks can be easily composed from a set of basic control primitives without the need for a detailed situation modeling (object and hand shape, friction properties, etc.), thus providing the foundation to yield robust manipulation skills also in unknown and unstructured environments.

The remaining chapter is organized as follows: In the next section we introduce the general concept of the control basis framework and discuss how efficient local motion generation methods can reduce the need for explicit planning in grasping of unknown objects. The subsequent Sect. 3 will introduce some recent tactile sensor developments and vision-based feature extraction methods to yield tactile features, which are at the basis of four tactile servoing control primitives. Finally, in Sect. 4 we describe and evaluate some tactile exploration tasks that impressively demonstrate the power of the proposed control framework.

## 2 Planning-Less Grasping in the Control Basis Framework

Grupen et al. first developed the idea of the control basis framework (CBF), which allows to realize complex tasks by composition of several basic controllers [9, 10]. Each of those controllers realizes resolved motion rate control, mapping updates of task control variables  $\Delta\mathbf{x}$  to joint angle updates  $\Delta\mathbf{q}$  of the robot. An important key idea was to stack controllers by priority allowing a subordinate controller to operate in the null-space of a higher-priority controller only, which can be easily achieved using appropriate null-space projections. Given any nonlinear relationship  $\mathbf{x}(\mathbf{q})$  between joint and task-space variables, the relation of their velocities at any point  $t$  in time is linear and given as

$$\dot{\mathbf{x}}(t) = J(\mathbf{q}(t)) \cdot \dot{\mathbf{q}}(t), \quad (1)$$

where  $J_{ij}(\mathbf{q}) = \partial x_i / \partial q_j$  is the task Jacobian at time  $t$ . Then, the solution to realize three priority-ordered task space motions  $\dot{\mathbf{x}}_1, \dot{\mathbf{x}}_2, \dot{\mathbf{x}}_3$  looks like this:

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + N_1 (\dot{\mathbf{q}}_2 + N_2 \dot{\mathbf{q}}_3) \quad (2)$$

$$= J_1^+(\mathbf{q}) \dot{\mathbf{x}}_1 + N_1 (J_2^+(\mathbf{q}) \dot{\mathbf{x}}_2 + N_2 J_3^+(\mathbf{q}) \dot{\mathbf{x}}_3), \quad (3)$$

where  $J_i^+$  denotes the Moore-Penrose pseudoinverse of  $J_i$  and  $N_i = \mathbf{1} - J^+ J$  denotes the corresponding null-space projector of task  $i = 1, 2, 3$ .

To work in practice, it's important, that every controller's null-space is rich enough to accommodate lower-priority motions, i.e. that there is enough redundancy. However, classical motion planning approaches attempt to control the end-effector motion in all six degrees of freedom (dof) and thus do not leave the necessary redundancy. But, exploiting the inherent symmetry of many everyday tasks, we can restrict ourselves to a few task-relevant dofs and thus gain the required redundancy.

As a prominent example, consider the grasping of a spherical object. Nowadays grasp planning approaches attempt to generate and evaluate grasps that approach the sphere from all possible directions [11]. However, in this particular task, it's only important to drive the hand towards the sphere—no matter from which direction. This reduced task description only consumes a single dof, namely the hand-object-distance, and frees up all other dofs. The resulting task-space motion  $\dot{\mathbf{x}}$  is a straight-line towards the goal, much like in classical Cartesian control. However, the redundant space at a given goal distance is the complete sphere around the target and any null space motion is automatically projected onto this sphere. In this manner, we can easily approach spherical objects for grasping from any direction, without the need to pre-compute a multitude of feasible grasps in advance. The corresponding task Jacobian  $J_{\parallel \cdot \parallel}$  can be easily computed from the Jacobian  $J$  of the standard forward transform:

$$J_{\parallel \cdot \parallel} = (\mathbf{x} - \mathbf{x}_{\text{goal}})^t \cdot J \quad (4)$$

Similarly, grasping a cylindrical object, like a bottle, only requires to align the hand axis with the object axis—the orientation angle around this axis can freely be chosen [12]. To allow even more flexibility, one may specify a task-space interval instead of a unique target value [13]. Within the original control basis framework, Platt et al. also propose more abstract controllers, e.g. to maintain force closure, to optimize grasp quality, manipulability, or visibility [14].

## 2.1 Collision Avoidance

In the context of motion planning, an important subordinate control task is collision and joint-limit avoidance. Joint limits can be easily avoided minimizing a quadratic or higher-order polynomial function [12, 15]:

$$H_{\mathbf{q}} = \sum w_i (q_i - q_i^{\text{ref}})^p \quad w_i = (q_i^{\max} - q_i^{\min})^{-1}, \quad (5)$$

where  $\mathbf{q}^{\text{ref}}$  defines a reference pose, e.g. in the middle of the joint range, and the  $w_i$ 's weight the contribution of individual joints according to their overall motion range.

Local collision avoidance is achieved by a repelling force field originating from each object. To this end, Sugiura [16] proposed to minimize a quadratic cost function defined on the distance  $d_{\mathbf{p}} = \|\mathbf{p}_1 - \mathbf{p}_2\|$  between the two closest points  $\mathbf{p}_1$  and  $\mathbf{p}_2$  on the robot and the obstacle:

$$H_{ca}(\mathbf{p}_1, \mathbf{p}_2) = \begin{cases} \eta(d_{\mathbf{p}} - d_B)^2 & d_{\mathbf{p}} < d_B \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Here,  $d_B$  acts as a distance threshold below which the force field becomes active and  $\eta$  is a gain parameter. The gradient of this cost function directly serves as a joint-level control target and can be easily computed in terms of the body point Jacobians  $J_{\mathbf{p}_i}$  by applying the chain rule:

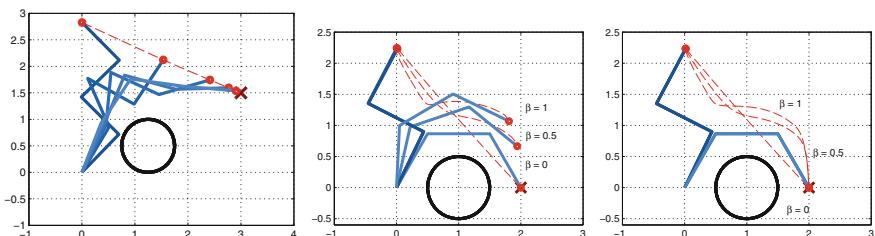
$$\dot{\mathbf{q}}_{ca} = -\nabla_{\mathbf{q}}^t H_{ca} = -2\eta(1 - d_B/d_{\mathbf{p}})(J_{\mathbf{p}_1} - J_{\mathbf{p}_2})^t(\mathbf{p}_1 - \mathbf{p}_2). \quad (7)$$

Thus we yield straight-line task-space motions (e.g. of the end-effector in Cartesian space), while the redundancy is exploited to circumvent obstacles as schematically shown in Fig. 1, left. To allow more flexible obstacle avoidance, Behnisch [17] proposed a relaxed motion control scheme, which allows deviations from straight-line motions, if the robot gets too close to obstacles:

$$\dot{\mathbf{q}} = J^+(\dot{\mathbf{x}} - \beta \dot{\mathbf{x}}_{ca}) - N(\nabla H_{ca} + \nabla H_{\mathbf{q}}). \quad (8)$$

Here, additionally to the null-space motion, which minimizes a superposition of both cost functions  $H_{\mathbf{q}}$  and  $H_{ca}$ , an obstacle avoidance motion  $\dot{\mathbf{x}}_{ca}$  is directly allowed in task-space as well. This contribution is determined by projecting the cost gradient (7) to the task space:

$$\dot{\mathbf{x}}_{ca} = J \nabla_{\mathbf{q}}^t H_{ca}. \quad (9)$$



**Fig. 1** Goal-directed task-space motion with collision avoidance. *Left* restricting avoidance motions to redundant space yields a *straight line* motion of the end-effector. *Middle* using relaxed motion control (8), the trajectory more strongly avoids the obstacle for larger weights  $\beta$ , but does not converge to the target anymore. *Right* dynamic adaption of  $\beta$  achieves both goals, target reaching and obstacle avoidance

Choosing different values of the weight  $\beta$ , we can smoothly adjust the importance of collision avoidance and target reaching as shown in Fig. 1, middle. However, because both contributions might be conflicting, the target is not always reached. To prevent this, we can ensure, that the goal-directed motion always dominates the collision avoidance motion with a margin  $\varepsilon$  by dynamically adapting  $\beta$ , such that the following condition is fulfilled:

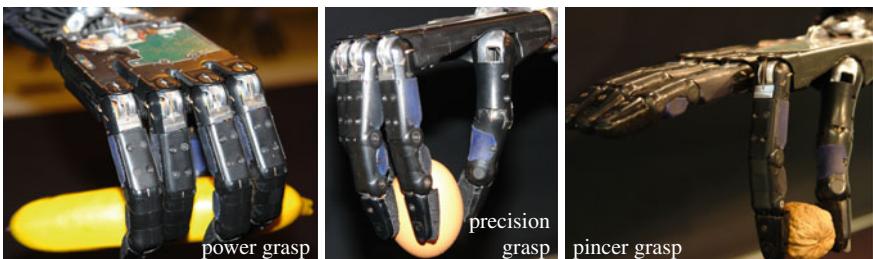
$$\|\dot{\mathbf{x}}\| - \varepsilon \geq \beta \|\dot{\mathbf{x}}_{ca}\|. \quad (10)$$

The resulting motion is shown in Fig. 1, right. Please note, that this approach—as a local method—is prone to get stuck in local minima, if a straight target-reaching motion is not collision-free. To avoid this failure, a deliberative planning method at a global level is required. To this end, Behnisch [17] proposed to augment the local motion generation with a globally acting, sampling-based planning method, that, however, searches within the low-dimensional task-space instead of the full joint space. This sharing of workload between a local, reactive planner and a global, deliberative planner turned out to be very successful and computationally efficient.

## 2.2 Vision-Based Grasp Selection

Employing the outlined control basis framework to realize approaching motions for grasping and exploiting the passive compliance of modern, often underactuated hands [18, 19], grasp planning is extremely simplified: The fingers will automatically wrap around the object due to the inherent compliance of the hand. Thus, the only task for grasp planning is to choose a suitable grasp prototype and to align the hand to the object during the approach phase.

As already observed by Cutkosky, humans employ only a very small number of grasp postures that can be roughly separated into power and precision grasps. Cutkosky's taxonomy then further subdivides grasps by the shape of the object [20]. From our experience it suffices to use the three basic grasp prototypes shown in Fig. 2 (power, precision, and pincer grasp). To chose an appropriate grasp for a given



**Fig. 2** Three basic grasp prototypes used for the Shadow Dexterous Hand. Depending on object size, estimated weight, and envisioned manipulation task we choose from a power grasp, a precision grasp, and a pincer grasp (*left to right*)

object, we employ a real-time, model-free scene segmentation method [21], which yields individual point clouds for all objects within the scene. Into each point cloud, a superquadrics model is fitted that captures the coarse shape of the object, smoothly varying between sphere, ellipsoid, cylinder, and box [22]. This model provides an estimation of the position and orientation as well as the coarse size and shape of the object. This information is utilized on the one hand to chose the grasp prototype and on the other hand to setup an appropriate approaching controller, utilizing the symmetries inherent to all recognized object shapes. A video illustrating the segmentation capabilities and the achieved grasping skills is available at youtube [22].

### 3 Tactile Servoing

In order to extend traditional grasp and manipulation planning approaches beyond a mere trajectory-centric view towards robust closed-loop controllers also integrating multi-modal feedback from proprioception, vision, and tactile sensing, in the following we discuss how the control basis framework (CBF) can be augmented by tactile servoing controllers. The main idea of these controllers is to define an inverted task Jacobian  $J_s^{-1}$  that directly maps errors in the tactile feature vector onto a suitable Cartesian velocity twist  $\mathbf{V}_s$  of the sensor frame. Subsequently we employ the power of CBF [23] to realize the computed sensor frame motion with appropriate joint motions. However, before looking into the details of these control primitives, we first review some recent developments in tactile sensing and discuss, which tactile features can be extracted from latest tactile sensing arrays.

#### 3.1 High Resolution Tactile Sensing

In the past decades tactile sensors were developed exploiting a variety of physical principles—ranging from piezo-resistive or capacitive to optical or ultrasonic effects (cf. Dahiya et al. [24] for a compact review). The BioTac® sensor can be considered a breakthrough in tactile sensing, integrating high-frequency temperature and pressure sensing with a grid of electrodes to resolve the point of contact as well as normal and shear forces [25]. Analyzing high-frequency vibrations induced by slip-stick transitions, the sensor is able to detect incipient slippage and to distinguish various materials showing characteristic vibration patterns [26, 27].

Independently, Schürmann et al. developed a modular sensor design tailored towards high-frequency sensing for slip detection too, but also providing a high spatial resolution for normal force sensing (on an array of  $16 \times 16$  tactels spaced at 5 mm) at the high frame rate of 1.9 kHz. Employing a multilayer perceptron network, trained to predict slip velocities from Fourier coefficients of the tactile time series, they were able to adjust the required grasping force to stably hold an object

without knowledge about its weight or friction properties: Every time, when incipient slippage is detected, the grasping force is increased by a fixed amount. Otherwise, it is exponentially decaying to minimize the applied contact forces [28].

While these two sensors provide excellent sensitivity to high-frequency, small amplitude vibrations, they are both rather bulky and not suited to be integrated into human-sized robotic fingertips. Although there exists an adaptation of the BioTac® sensor to the anthropomorphic Shadow Dexterous Hand™ [29], this integration design removes the distal finger joint, which is important in various manipulation tasks. Utilizing a new technology to realize 3D-shaped PCBs, Zenker et al. [30] miniaturized the tactile sensor array, integrating 12 tactels and the measurement electronics within a fingertip-shaped sensor-electrode that exactly matches the size of the robotic fingertip (cf. Fig. 3).

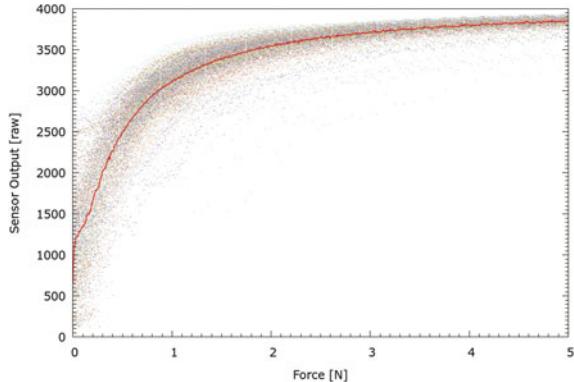
All these sensors are rather rigid and thus not suitable to be worn by a human. In order to measure interaction forces between the human hand and a manipulated object too, a more flexible sensor hardware is required. A first approach into this direction is the tactile glove developed by Büscher et al. [31] which is composed from conductive and piezo-resistive fabrics layers. In contrast to previous attempts to measure interaction forces, utilizing instrumented objects [32, 33], the sensorized glove allows to measure tactile interaction patterns with arbitrary objects. Its low construction height as well as the flexibility and stretchability of the fabrics, make this sensor concept well suited to cover larger parts of robots too, e.g. to yield a tactile-sensitive skin.

Given their high data frame rates, all sensor designs open up the opportunity to be employed for closed-loop robot control, thus for the first time offering large-scale reactivity to touch comparable to human sensitivity. Looking into the literature, only a very few approaches exist that directly utilize tactile sensor information for control, e.g. a very early [34] or a more recent one [35] on tactile contour tracking. However, a generic tactile servoing framework allowing to achieve a multitude of tasks from the composition of simple, basic controllers is missing so far. In the following sections we will get a glimpse on the enormous potential that can be unleashed when combining concepts from the control basis framework with tactile sensor information, thus lifting grasping and manipulation skills for robots to the next level of robustness and dexterity.



**Fig. 3** Recent tactile sensors from Bielefeld University. From *left* to *right* a modular, flat 16 × 16 tactile sensor array, a 3D shaped tactile fingertip suitable for the Shadow Robot Hand, and a flexible tactile glove manufactured from conductive fabrics

**Fig. 4** Sensor characteristics of all 256 tactels (and an individual one—*red solid line*) as acquired on a calibration bench



### 3.2 Feature Extraction from Tactile Images

Many tactile sensor designs propose an array of tactile sensing elements (tactels) providing *normal force* information [6, 36, 37] for each element. Sometimes it is also possible to compute contact *force directions* from this information [37]. Most array structures also have a reasonable spatial resolution to allow for an explicit control of the tactile force pattern sensed in a contact region. As a consequence, in our control framework, we assume the availability of a tactile sensor array providing a tactile image of normal force values measured by individual tactels.

Particularly, the device employed in our experiments is the  $16 \times 16$  sensor array depicted in Fig. 3, left. This sensor exploits the piezo-resistive sensing principle, measuring changes in resistance of a conductive foam due to an applied force. The analog measurement of each individual tactel is converted to a 12bit digital value covering a pressure range of 0.1–10 kPa.<sup>1</sup> Due to varying local conductive properties of the foam, every tactel has a distinguished, squashed and noisy sensor characteristics as shown in Fig. 4. To obtain a coarse force calibration, the characteristic measurement function of each individual tactel is inverted in its linear range.

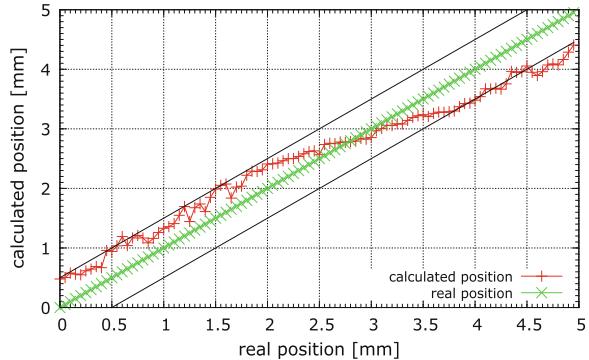
The intended tactile servoing tasks aim for controlling (a) the contact position on the fingertip, (b) the contact force, and (c) the orientation of an object edge relative to the sensor array. Hence, we propose feature extraction methods to provide the current values of these control variables.

As a first processing step, the contact region on the sensor is identified, which typically extends over several tactels due to the softness of the sensor foam. To this end, we employ connected component analysis [38], well known from image processing, to extract all connected regions in the binarized tactile image and choose the largest one as the considered contact region  $R$ —neglecting all smaller regions as originating from noise or spurious contacts. The binarization threshold is chosen rather small, just above the noise level, to consider as much tactile information as

---

<sup>1</sup>The sensor's sensitivity and force range can be adjusted to the task. Here, we have chosen the characteristics to provide a linear range from 0.1–1 kPa.

**Fig. 5** Estimated (red) and expected (green) contact position (COP) of a 2 mm-diameter probe tip



possible. Subsequently, the overall contact (normal) force  $f$  is determined as the sum of forces  $f_{ij}$  within the contact region and the contact position  $\mathbf{c}$  as the force-weighted center of pressure (COP) of  $R$ :

$$f = \sum_{ij \in R} f_{ij} \quad \mathbf{c} = f^{-1} \sum_{ij \in R} f_{ij} \mathbf{c}_{ij}, \quad (11)$$

where  $c_{ij}$  are the discrete coordinates of the tactels on the sensor surface. Due to the averaging effect from multiple tactels composing a contact region, we obtain a sub-tactel resolution for the contact position as illustrated in Fig. 5. In this experiment a probe tip, 2 mm in diameter, was moved across the sensor from one tactel to another, i.e. about a distance of 5 mm. At every point, the estimated and real probe position (obtained from the robot's end-effector pose) are compared.

Usually, we want to control the contact pressure instead of the overall contact force. Considering manipulation of fragile objects, like an egg, it is the local pressure that should be limited to not damage the object. To obtain a pressure value, we normalize the overall measured force by the size of the contact region (measured as the number of pixels in  $R$ ):

$$p = \frac{f}{|R|} \quad (12)$$

To extract the orientation of an object edge that maps onto a line-shaped contact region, we utilize the Hough transform, also well known from image processing [39].

### 3.3 Tactile Control Primitives

The proposed tactile servoing controller aims at realizing sliding and rolling motions about the contact point while maintaining a specified normal contact force during manipulation. Dependent on the actual task at hand, specific control primitives can be selectively turned on or off. Additionally to this purely tactile-driven motion,

an external task planner can provide a motion component  $\mathbf{V}_s^{\text{ext}}$ , which is a twist expressed in terms of the sensor frame  $O_s$ . This motion component allows to realize externally controlled tactile object exploration, e.g. to follow an object edge or to run the sensor over the whole object surface as detailed in the experimental Sect. 4.

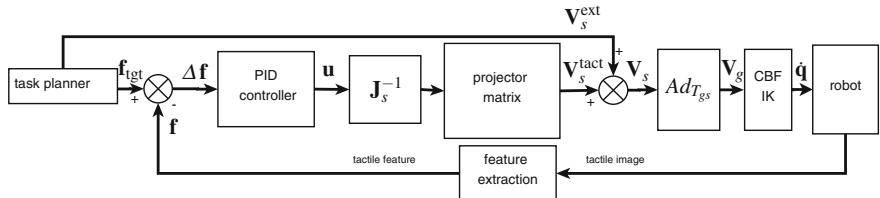
The general control scheme of our proposed controller is depicted in Fig. 6. The control cycle starts by computing the deviation of the current tactile feature vector  $\mathbf{f}$  from the targeted one. This error is fed into PID-type controllers, acting independently on all feature-error components. The resulting control variable  $\mathbf{u}$  is a linearly transformed version of  $\Delta\mathbf{f}$ . Please notice, that for effective force control a non-zero integral component is required to compensate for static errors caused by a pure P-controller. Additionally, the derivative component is necessary to suppress undesired oscillations.

Subsequently, we compute a sensor motion  $\mathbf{V}_s^{\text{tact}}$  aiming to reduce the feature error. This is realized with a fixed, task-*independent*, inverted Jacobian matrix  $J_s^{-1}$ . Both entities are expressed in terms of the sensor coordinate frame  $O_s$ , which is located in the center of the sensor surface and aligned with the sensor such that the  $z$ -axis equals the surface normal. This choice tremendously facilitates the determination of  $J_s^{-1}$ , which maps feature errors onto sensor motions.

The subsequent application of a task-*dependent* projector matrix  $P$  selecting certain twist-components for control and neglecting others, allows to selectively switch on or off specific motion components. To this end,  $P$  is a simple  $6 \times 6$  diagonal matrix, where ones and zeros indicate, that the corresponding twist component is or is not used for control. Summarizing, the feedback-part of the tactile servoing controller is determined by the following equation:

$$\mathbf{V}_s^{\text{tact}} = P \cdot J_s^{-1} \cdot \left( K_P \cdot \Delta\mathbf{f}(t) + K_I \cdot \int \Delta\mathbf{f}(t) dt + K_D \cdot (\Delta\mathbf{f}(t) - \Delta\mathbf{f}(t-1)) \right). \quad (13)$$

Here  $\mathbf{V}_s = [\mathbf{v}_s, \boldsymbol{\omega}_s]$  denotes the 6-dimensional twist vector composed of linear and angular velocity components  $\mathbf{v}_s, \boldsymbol{\omega}_s$ .  $K_{P,I,D}$  denote diagonal matrices of PID-controller gains and  $\Delta\mathbf{f}(t) = [\Delta x_s, \Delta y_s, \Delta f, \Delta \alpha]$  denotes the deviation of the feature vector composed of the positional error  $\Delta x_s$ ,  $\Delta y_s$ , the normal force error  $\Delta f$ , and the



**Fig. 6** Control scheme for tactile servoing: the core feedback part computes a sensor motion  $\mathbf{V}_s^{\text{tact}}$  from tactile-feature deviations  $\Delta\mathbf{f}$ , which is superimposed with an external motion signal  $\mathbf{V}_s^{\text{ext}}$  and subsequently fed into CBF's inverse kinematics

angular error  $\Delta\alpha$  of the line orientation. Note, that the latter one is measured modulo  $\pi$  in order to obtain angular errors in the range  $(-\frac{\pi}{2}, \frac{\pi}{2}]$  and thus circumventing singularities due to their circular nature. The rotational symmetry allows to restrict the errors to this range instead of  $(-\pi, \pi]$ .

Finally, the twists originating from the tactile feedback-loop and the external task planner are superimposed and fed to the inverse kinematics module of the control basis framework. To this end, the twist  $\mathbf{V}_s$  expressed in terms of the sensor frame  $O_s$  needs to be transformed to the global frame  $O_g$ , which is realized by the adjoint matrix derived from the forward kinematics  $T_{gs} = (R_{gs}, \mathbf{p}_{gs})$ :

$$Ad_{T_{gs}} = \begin{pmatrix} R_{gs} & \hat{\mathbf{p}}_{gs} R_{gs} \\ 0 & R_{gs} \end{pmatrix} \quad (14)$$

At the core of the tactile-feedback controller is the inverse Jacobian that maps feature deviations onto a motion twist of the tactile sensor array:

$$\mathbf{V}_s^{\text{tact}} = J_s^{-1} \cdot \Delta \mathbf{f} = \begin{pmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ \mathbf{0} & \mathbf{1} & 0 & 0 \\ \mathbf{1} & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{pmatrix} \cdot \begin{pmatrix} \Delta x_s \\ \Delta y_s \\ \Delta f \\ \Delta \alpha \end{pmatrix} \quad (15)$$

This matrix can be easily determined in the sensor coordinate frame  $O_s$ : Positional deviations are simply mapped onto corrective tangential motions in the  $x$ - $y$ -plane of the sensor. Normal force errors are mapped onto a corrective translational motion along the  $z$ -axis of the sensor frame, which is normal to the sensor plane, pointing towards the object. These linear motion components are determined by the first three rows of  $J_s^{-1}$ . The rotational error  $\Delta\alpha$  is mapped onto a rotational velocity around the  $z$ -axis (last row). The motion components corresponding to the fourth and fifth row of the inverted Jacobian realize a rolling motion of the sensor. These are triggered by positional deviations again. Thus, an error  $\Delta x_s$  is not only reduced by an appropriate tangential linear motion of the sensor, but also by a rolling motion around the  $y$ -axis of the sensor, that also moves the COP of the contact region closer towards its target location.

The task-dependent projector matrices  $P$  can be used to toggle these individual twist components on and off. For example, if contact position control is desired, one will choose  $P = \text{diag}(1, 1, 0, 0, 0, 0)$ . When additionally force control is required, the third diagonal entry should be set to 1 too. In order to enable or disable the orientation tracking of an object edge, you will set the last diagonal entry to 1 or 0 respectively. Finally, the fourth and fifth entries in the diagonal projector matrix determine, whether rolling is enabled or not. In the following section, we will discuss several application scenarios of the proposed tactile-servoing framework.

## 4 Experimental Evaluation

As shown in Fig. 7, we mounted the tactile sensor pad as a large fingertip to a 7-dof Kuka lightweight robot arm operated in joint-space compliance mode. The control basis framework maps Cartesian-space twists into joint-angle velocities, thus changing the equilibrium posture of the robot controller. The tactile sensor pad provides an array of  $16 \times 16$  tactels measuring contact forces with 12bits resolution [6]. The sampling frequency of the tactile sensor as well as the control cycle frequency of the robot arm are set to 250Hz. We use manually tuned PID parameters for the tactile servoing controller.

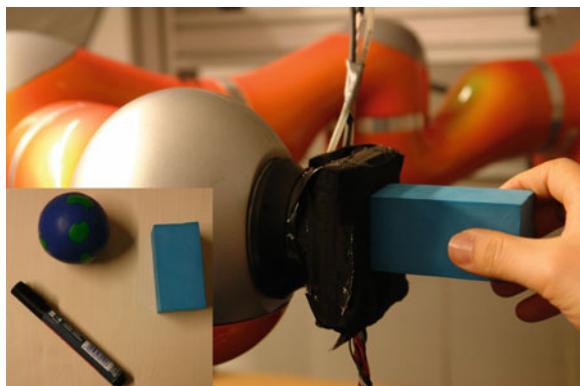
All the experiments discussed in the following are also shown in a youtube video [40] and follow the same course: Initially the robot is moved to its working area, holding this posture until object contact is established. As soon as a pressure threshold is exceeded, the robot switches to a specific, previously determined tactile servoing task.

In order to reduce the noisiness of the feature signals, we apply a smoothing filter to both the force/pressure feature and the line orientation feature  $\alpha$ . To this end, we average the ten most recent measurements, i.e. in a time window of 40 ms. The position feature is smooth enough due to the averaging of Eq.(11).

### 4.1 Tracking Contact Points

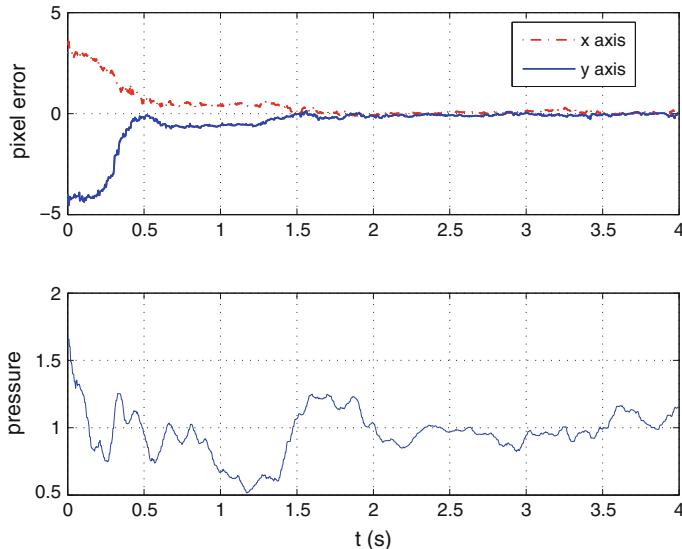
Contact point tracking has an important application for multi-finger grasping and manipulation. In both cases, fingers need to maintain object contact with a given contact force and they should ensure, that the contact location remains on the fingertip area—optimally in its center—to avoid slipping off. Consequently, the task-dependent projector matrix has the form  $P = \text{diag}(1, 1, 1, 0, 0, 0)$  enabling contact position and force control.

**Fig. 7** Experimental setup:  
tactile sensor mounted on  
Kuka LWR



**Table 1** Statistical tracking results for force and position control

Object	Steady state error	Standard deviation	Response time
Rigid pen	0.0032	0.039	2.5 s
Toy box	0.0026	0.039	2.0 s
Soft ball	0.0010	0.043	2.0 s
X	0.0041 pixel	0.1146 pixel	1.8 s
Y	0.0082 pixel	0.1158 pixel	1.8 s

**Fig. 8** Tracking results for combined position and force control

Please notice, that the quality of force control depends on the stiffness of objects (softer objects allow for a larger motion range given a fixed force range). We evaluated the control performance on various objects of different stiffnesses: a rigid pen, a toy box from stiff foam, and a soft ball. The results for maintaining a desired pressure level of  $p = 1$  are shown in Table 1. As expected, stiffer objects take longer to converge to a stable tracking result (response time) and exhibit stronger force oscillations given similar deflections. However, in all cases the desired force level will eventually be well maintained with a small steady state error<sup>2</sup>.

For contact position tracking, the goal is to maintain the COP of the contact region at the center of the tactile sensor frame. The evolution of the errors in contact position and force are shown in Fig. 8. As can be seen from the top sub-figure an initial position offset is corrected within half a second. The steady state error and

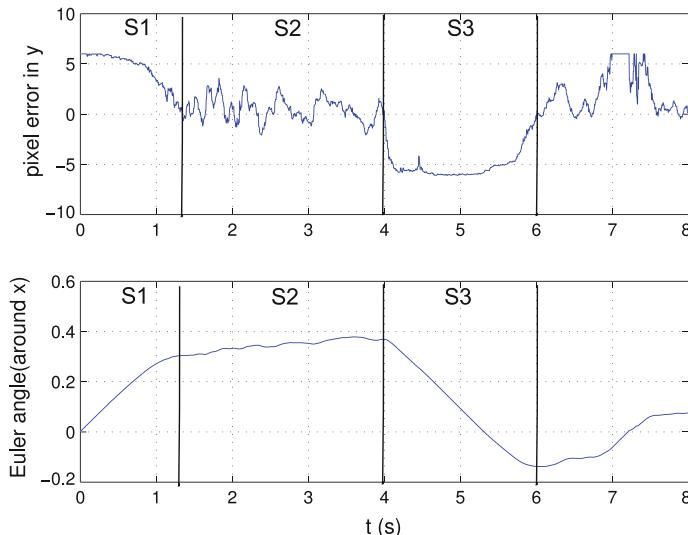
<sup>2</sup>The steady state errors and standard deviations are computed from a time series of 20 s duration starting after convergence (response time). All values are obtained by averaging over 20 trials.

response time are summarized in Table 1. As can be seen from bottom sub-figure the normal force applied in this experiment evolves randomly as it is not controlled. Note, that a large normal force—due to friction—will also cause large tangential forces, rendering the sliding motion more difficult. Hence, normal force control should be generally enabled.

## 4.2 Track Contact Point and Increase Contact Area by Rolling

The fourth and fifth row of the task Jacobian (Eq. 13) provide another mode of operation to compensate for positional errors of the COP: Instead of realizing a translational sliding motion, this control behavior realizes a rolling motion, thus changing the contact point both on the tactile sensor and the object’s surface. While previous approaches to realize rolling employed complex algorithms to determine the point of revolution and a corresponding joint-space robot motion [41], the tactile servoing approach proposed here, is conceptually much easier: a deviation in contact position is simply mapped to a rotational twist within the tangential plane of the sensor. Because we do not explicitly compute the point of revolution and do not know the shape of the object, the normal force will probably be disturbed due to this motion. However, the normal force controller, running in parallel, will counteract and maintain a pre-defined force level. The employed projector matrix equals  $P = \text{diag}(1, 1, 1, 1, 1, 0)$ , i.e. simultaneously realizing sliding and rolling as well as force control.

The resulting rolling motion is visualized in Fig. 9. An initial positional offset along the  $y$ -axis is compensated by a rolling motion about the sensor’s  $x$ -axis



**Fig. 9** Orientation control of surface normals by rolling

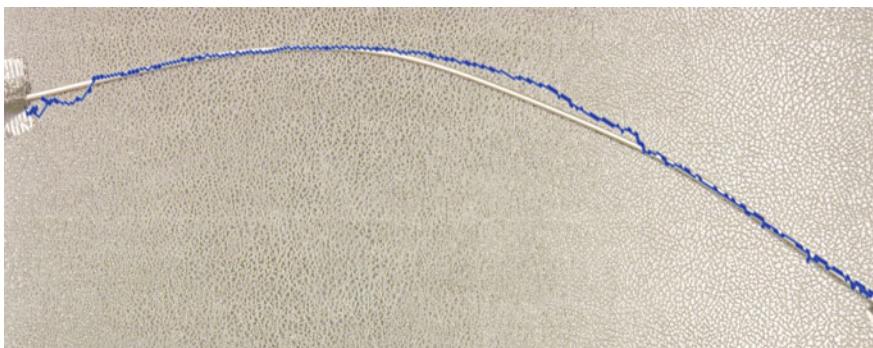
(stage S1). When the contact point error decays, the rolling motion ceases as well (stage S2). After 4 s the object was displaced yielding a negative position offset that was compensated by a rolling motion into the opposite direction (stage S3). This behavior can nicely be seen in the video [40] as well.

The rolling behavior has the beneficial side-effect of increasing the area of contact between the finger tip and the object. This is an important capability for grasp stabilization. Although classical grasp planning considers point contacts only, a large contact area naturally increases the grasp wrench space and thus increases the ability to resist to external disturbances. Furthermore, a prerequisite for successful tactile object exploration will be to maintain a large contact area during exploration in order to collect as much shape information about the object as possible.

How this side effect is achieved? Assuming large object and sensor surfaces, a small contact area typically results from a badly tilted sensor w.r.t. the object surface. In this situation the sensor only touches an object edge instead of the whole surface. This contact is often located off-center on the sensor array. The corrective rolling motion to move the COP into the sensor's center will also reduce the tilting and eventually result in the desired surface contact. This state also constitutes a fixed point of the controller dynamics, because the COP will be in the center of the tactile array in this case.

### 4.3 Tracking an Object Edge on the Sensor Surface

The orientation around the normal axis is controlled using the orientation angle  $\alpha$  of a line in the tactile image emerging from an object edge on the sensor. For this control task the last row of the Jacobian matrix is important, resulting in a projector matrix  $P = \text{diag}(0, 0, 1, 0, 0, 1)$ . The tracking result for this experiment is qualitatively shown in the video [40] only. However, the next experiment also employs this control primitive and provides an evaluation in Fig. 10.



**Fig. 10** Tracking of a cable of unknown shape: tracking result is superimposed onto a scene photo as a *blue* trajectory

#### 4.4 Tracking of an Unknown Object Edge

The previous experiments illustrated the performance of the proposed tactile servoing controllers in various scenarios, neglecting external motion commands  $\mathbf{V}_s^{\text{ext}}$ . However, the aim of the following two tasks is to illustrate, that complex exploration behavior emerges if the tactile servoing motion and some externally provided guidance motion are combined.

In the first experiment, we consider the task of tracking the unknown shape of a cable lying flat on the table. To this end, the sensor should (i) be aligned to the local orientation of the cable, (ii) maintain the tactile imprint within its sensor boundaries (optimally in the center), and (iii) actively control the contact force. Accordingly we choose a projector matrix  $P = \text{diag}(1, 1, 1, 0, 0, 1)$  selecting those subtasks. In order to follow the cable in space, we additionally impose an external tangential motion onto the sensor along its  $y$ -axis, which coincides with the desired orientation of the cable. Thus  $\mathbf{V}_s^{\text{ext}} = [0, 1, 0, 0, 0, 0]^t$ .

Figure 10 shows a photo of the tracked cable superimposed with the object shape (blue line) estimated from the forward kinematics of the robot arm when tracking the cable with tactile servoing. After some initial oscillations, the robot manages to align the cable imprint on the sensor with its  $y$ -axis.

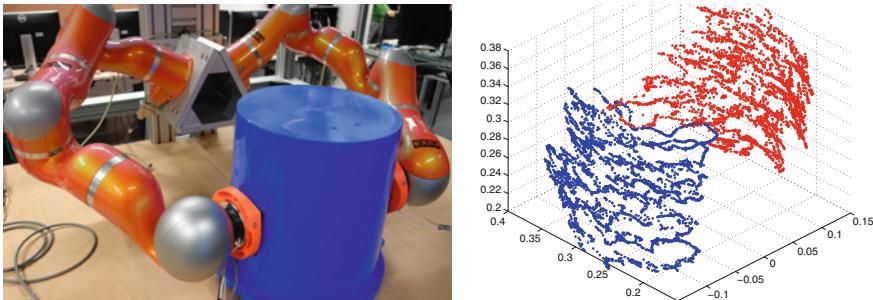
#### 4.5 Exploring the Shape of an Unknown Object

The second experiment illustrating the power of the proposed tactile servoing framework, aims at tactile object exploration: The sensor should slide over the unknown surface of the object in order to accumulate a dense shape model. Lacking an appropriate control framework, previous work acquired the corresponding tactile information by repeated establishment and breaking of object contact [42].

To realize this complex task, we decompose it into several phases: after establishing contact to the object, the robot maximizes the sensed contact area and aligns its  $y$ -axis with the major axis of the contact region applying the control schemes of Sects. 4.2 and 4.3 simultaneously.

Subsequently, by imposing a tangential motion along the sensor's  $x$ -axis (orthogonal to the major axis of contact region), we induce the exploratory motion. The tactile servoing controller maintains the optimal orientation and position of the tactile sensor on the object's surface by generating appropriate sliding and rolling motions. This task exploits all tactile servoing behaviors employing the projector matrix  $P = \mathbf{1}$ . As a result the object exploration behavior emerges automatically.

Similarly we can explore the object along the other direction, if we follow the contact's major axis instead (cf. previous task in Sect. 4.4). Please notice, that in the accompanying video [40] we change the direction of the external guidance motion  $\mathbf{V}_s^{\text{ext}}$  in order to realize a scanning of the object into both directions. Figure 11 shows, how this exploration behavior can be utilized to construct an object shape estimation by touch.



**Fig. 11** Tactile object exploration using two tactile sensor arrays mounted onto Kuka LWR arms (*left*) and the resulting tactile point cloud as a local estimation of object shape (*right*)

## 5 Conclusion

The introduced tactile servoing control framework allows to realize a large range of tactile tracking and exploration tasks. To this end, it's only necessary to choose the task-specific projector matrix  $P$  choosing which tactile servoing primitives (sliding, rolling, turning, force control) should be applied.

The integration of an externally driven guidance motion  $\mathbf{V}_s^{\text{ext}}$  allows to realize complex exploratory behavior. In the shown example tasks, we only used very simple, static guidance motions. However, if those guidance motions are computed from tactile feedback as well, one can easily realize even more complex exploration behavior, e.g. to drive the tactile sensor towards interesting spots on the object's surface, like ridges, edges or corners.

As you have seen, the formulation of tasks as a clever chosen set of primitive controllers relaxes the need for explicit planning and modeling to a large extend, such that both grasping and manipulation tasks become feasible also for unknown objects. Such situations frequently occur in unstructured human environments, like homes or hospitals, which are natural environments for service robots.

## References

1. Jenmalm P, Johansson RS (1997) Visual and somatosensory information about object shape control manipulative fingertip forces. *J Neurosci* 17:4486–4499
2. Johansson RS, Westling G (1984) Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects. *Exp Brain Res* 56:550–564
3. Steffen JF, Elbrechter C, Haschke R, Ritter H (2010) Bio-inspired motion strategies for a bimanual manipulation task. In: Proceedings of international conference on humanoid robots
4. Dang H, Weisz J, Allen PK (2011) Blind grasping: stable robotic grasping using tactile feedback and hand kinematics. In: Proceedings of ICRA
5. Ho V, Nagatani T, Noda A, Hirai Sh (2012) What can be inferred from a tactile arrayed sensor in autonomous in-hand manipulation? In: Proceedings of CASE, p 461

6. Schürmann C, Kõiva R, Haschke R (2011) A modular high-speed tactile sensor for human manipulation research. In: World haptics conference
7. Li Q, Haschke R, Bolder B, Ritter H (2012) Grasp point optimization by online exploration of unknown object surface. In: Proceedings of international conference on humanoid robots
8. Pezzementi Z, Plaku E, Reyda C, Hager GD (2011) Tactile-object recognition from appearance information. *Trans Robot* 27(3):473–487
9. Hart S, Sen S, Ou S, Grupen R (2009) The control basis API—a layered software architecture for autonomous robot learning. In: 2009 workshop on software development and integration in robotics at ICRA
10. Huber M (2000) A hybrid architecture for adaptive robot control. PhD thesis, University of Massachusetts
11. León B, Ulbrich S, Diankov R, Puche G, Przybylski M, Morales A, Asfour T, Moisio S, Bohg J, Kuffner J (2010) OpenGRASP: a toolkit for robot grasping simulation. In: Proceedings of SIMPAR. Springer, Darmstadt, pp 109–120
12. Gienger M, Toussaint M, Goerick C (2010) Whole-body motion planning—building blocks for intelligent systems. In: Harada K, Yoshida E, Yokoi K (eds) Motion planning for humanoid robots. Springer, London, pp 67–98
13. Gienger M, Janßen H, Goerick C (2006) Exploiting task intervals for whole body robot control. In: Proceedings of IROS, pp 2484–2490
14. Platt R, Fagg AH, Grupen RA (2010) Null-space grasp control: theory and experiments. *IEEE Trans Robot* 26(2):282–295
15. Liegeois A (1977) Automatic supervisory control of configuration and behavior of multibody mechanisms. *IEEE Trans Syst, Man Cybern* 7(12):861–871
16. Sugiura H, Gienger M, Jannsen H, Goerick C (2010) Reactive self collision avoidance with dynamic task prioritization for humanoid robots. *Int J Humanoid Robot* 7(01):31–54
17. Behnisch M, Haschke R, Ritter H, Gienger M (2011) Deformable trees—exploiting local obstacle avoidance. In: Proceedings of international conference on humanoid robots
18. Catalano MG, Grioli G, Farnioli E, Serio A, Piazza A, Bicchi C (2014) Adaptive synergies for the design and control of the Pisa/IIT SoftHand. *Int J Robot Res* 33(5):768–782
19. Odhner LU, Ma RR, Dollar AM (2013) Open-loop precision grasping with underactuated hands inspired by a human manipulation strategy. *IEEE Trans Autom Sci Eng* 10(3):625–633
20. Cutkosky M, Howe RD (1990) Human grasp choice and robotic grasp analysis. In: Venkataraman ST, Iberall T (eds) Dextrous robot hands. Springer, New York
21. Ückermann A, Haschke R, Ritter H (2013) Realtime 3D segmentation for human-robot interaction. In: Proceedings of IROS
22. Ückermann A, Haschke R, Ritter H (2012) Real-time 3D segmentation of cluttered scenes for robot grasping. In: Proceedings of international conference on humanoid robots. Video: [www.youtube.com/watch?v=Z2SwggQTBC8](http://www.youtube.com/watch?v=Z2SwggQTBC8)
23. Schöpfer M, Schmidt F, Pardowitz M, Ritter H (2010) Open source real-time control software for the Kuka light weight robot. In: Proceedings of WCICA, pp 444–449
24. Dahiya RS, Metta G, Valle M, Sandini G (2010) Tactile sensing: from humans to humanoids. *IEEE Trans Robot* 26(1):1–20
25. Wettels N, Santos VJ, Johansson RS, Loeb GE (2008) Biomimetic tactile sensor array. *Adv Robot* 22(8):829–849
26. Fishel JA, Loeb GE (2012) Sensing tactile microvibrations with the BioTac—comparison with human sensitivity. In: International conference on biomedical robotics and biomechatronics (BioRob), pp 1122–1127
27. Xu D, Loeb GE, Fishel JA (2013) Tactile identification of objects using Bayesian exploration. In: Proceedings of ICRA, pp 3056–3061
28. Schürmann C, Schöpfer M, Haschke R, Ritter H (2012) A high-speed tactile sensor for slip detection. In: Prassler E, Burgard W, Handmann U, Haschke R, Hägele M, Lawitzky G, Nebel B, Nowak W, Plöger P, Reiser U, Zöllner M (eds) Towards service robots for everyday environments, vol 76. Springer, New York, pp 403–415. Video: [www.youtube.com/watch?v=mSq8e4PU90s](http://www.youtube.com/watch?v=mSq8e4PU90s)

29. Shadow Robot Company. Shadow Dexterous Hand (2013). <http://www.shadowrobot.com/products/dexterous-hand>
30. Kõiva R, Zenker M, Schürmann C, Haschke R, Ritter H (2013) A highly sensitive 3D-shaped tactile sensor. In: International conference on advanced intelligent mechatronics (AIM)
31. Büscher G, Kõiva R, Schürmann C, Haschke R, Ritter H (2012) Tactile dataglove with fabric-based sensors. In: Proceedings of international conference on humanoid robots
32. Maycock J, Essig K, Haschke R, Schack T, Ritter H (2011) Towards an understanding of grasping using a multi-sensing approach. In: Proceedings of ICRA, pp 1–8
33. Roa M, Kõiva R, Castellini C (2012) Experimental evaluation of human grasps using a sensorized object. In: International conference on biomedical robotics and biomechatronics (BioRob)
34. Chen N, Zhang H, Rink R (1995) Edge tracking using tactile servo. In: Proceedings of IROS, vol 2. August 1995, pp 84–89
35. Martinez-Hernandez U, Lepora NF, Barron-Gonzalez H, Dodd TJ, Prescott TJ (2012) Towards contour following exploration based on tactile sensing with the iCub fingertip. In: Herrmann G, Studley M, Pearson M, Conn A, Melhuish C, Witkowski M, Kim J-H, Vadakkepat P (eds) Advances in autonomous robotics. Lecture notes in computer science, vol 7429. Springer, Berlin, pp 459–460
36. Schmitz A, Maiolino P, Maggiali M, Natale L, Cannata G, Metta G (2011) Methods and technologies for the implementation of large-scale robot tactile sensors. Trans Robot 27(3):389–400
37. Wettels N, Loeb GE (2011) Haptic feature extraction from a biomimetic tactile sensor: force, contact location and curvature. In: Proceedings of ROBIO, pp 2471–2478
38. Suzuki K, Horiba I, Sugie N (2003) Linear-time connected-component labeling based on sequential local operations. Comput Vis Image Underst 89(1):1–23
39. Duda RO, Hart PE (1972) Use of the Hough transformation to detect lines and curves in pictures. Commun ACM 15(1):11–15
40. Li Q (2013) A control framework for tactile sensing. Video: <https://www.youtube.com/watch?v=TcWipks3qj0>
41. Schöpfer M, Ritter H, Heidemann G (2007) Acquisition and application of a tactile database. In: Proceedings of ICRA, pp 1517–1522
42. Meier M, Schöpfer M, Haschke R, Ritter H (2011) A probabilistic approach to tactile shape reconstruction. Trans Robot 27(3):630–635

## **Part II**

# **Motion Planning of Robotic Manipulators**

# Obstacle Avoidance with Industrial Robots

T. Petrič, A. Gams, N. Likar and L. Žlajpah

**Abstract** One of the important features that a robot must possess when working in an unstructured environment is the ability to deal with objects. Such objects can be a part of the task, e.g., in assembly operations, or they can represent an obstacle. In the case when contact with the objects is not desired, the main issue is how to perform the desired task without any risk of collisions with the objects in the workspace. A general strategy for obstacle avoidance is to reconfigure the robot so that it is not in the contact with the obstacle. However, a reconfiguration without changing the task motion is only feasible if the robot has sufficient redundant degrees of freedom (DOFs). In this chapter we present different approaches to the control methods of redundant robot manipulators performing multiple tasks with obstacle avoidance. The pros and cons of the presented methods and the differences between them are also discussed. The performance of the methods is also demonstrated by simulation and on real robots.

**Keywords** Redundant robots · Obstacle avoidance · Kinematic control · Prioritized task control · Dynamic movement primitives

---

T. Petrič · A. Gams · N. Likar · L. Žlajpah (✉)

Department for Automation, Biocybernetics and Robotics, Jožef Stefan Institute,  
Jamova Cesta 39, Ljubljana, Slovenia  
e-mail: leon.zlajpah@ijs.si

T. Petrič  
e-mail: tadej.petric@ijs.si

A. Gams  
e-mail: andrej.gams@ijs.si

N. Likar  
e-mail: nejc.likar@ijs.si

## 1 Introduction

In this chapter we give a brief overview of the most commonly applied obstacle-avoidance algorithms. In general, the algorithms can be divided into global and local. While the former rely on planning, the latter are control-based. We present different control-based approaches, that rely on kinematic algorithms to avoid the obstacles with the end-effector or with any other part of the body of the robot. We also discuss how to include obstacle-avoidance algorithms in novel trajectory-generation methods, such as dynamic movement primitives.

Just as with humans, robotic mechanisms have to act in environments with other objects and agents moving around, interacting with them, influencing the very same environment. The environment can be highly structured, like an industrial setting, or it can be very cluttered, like a kitchen or a workshop. Contact between the robot and an object is very likely to happen in any environment. The contacts can be part of the task, but they may very well also be an undesired event, and consequently, it is necessary to give the highest priority to avoiding them. Different obstacle algorithms have been proposed for this to ensure that tasks that demand no contact with objects, perceived as obstacles either at the end-effector or at any other point of the robot, can be successfully fulfilled.

A natural strategy of obstacle avoidance is to move the manipulator into a configuration where it is not in contact with the obstacle. In order to avoid interference with the motion of the end-effector, redundant degrees of freedom (DOFs) have to be utilized to achieve a collision-free configuration. The amount of flexibility depends on the degree of redundancy, i.e., on the number of redundant DOFs. The kinematic control of redundant mechanisms, where the redundancy is defined as the difference between the required and available DOFs, was thoroughly studied [1–4].

Two different strategy classes can be employed when solving the obstacle-avoidance problem, i.e., global and local. Global strategies rely on planning. They guarantee to find a collision-free path from the initial point to the goal point, if such a path exists. Typically, they are applied in the configuration space, which is also where the manipulator and all the obstacles are mapped. A collision-free path is found in the unoccupied portion of the configuration space [5–7]. One of the major drawbacks is that such methods rely on the assumption that the environment is not changing, as the computational complexity of the algorithms prevents any re-calculation within the typical response time of a manipulator. Despite efforts to reduce the computational complexity of such global algorithms [8–10], these methods cannot offer ability for real-time implementations. This limits their applicability to static and well-defined environments.

Local strategies, on the other hand, treat obstacle avoidance as a control problem. They exploit the capabilities of low-level control, e.g., they can use the sensor information to change the path if an obstacle appears or moves in the workspace. They are primarily suitable when the obstacle position is not known in advance, but is detected in real-time during the task's execution. In this sense, they are not meant to replace the global, higher-level path-planning methods. Local methods are also

computationally less demanding than global methods. However, local methods may cause suboptimal behavior or may even become stuck when a collision-free path cannot be found from the current configuration.

The collision avoidance of redundant manipulators was thoroughly studied [11–20]. The approach proposed by Maciejewski and Klein [17] is to assign to the critical point an avoiding task-space motion, with which the point is then moved away from the obstacle. Colbaugh et al. [12, 13] used configuration control and they defined the constraints representing the obstacle avoidance. On the other hand, Khatib [15] proposed to use potential fields where obstacles generate repulsive forces that prevent the robot to come too close to the obstacle. Similar approaches were used later by several authors proposed potential functions where a repulsive potential is assigned to obstacles and an attractive potential is assigned to the goal position [16, 18, 20–25]. Yet another approach uses the optimization of an objective function maximizing the distance between the manipulator and the obstacles [14].

Many of the methods are applied at the kinematic level of control, using null-space velocity control for the internal motion of a redundant manipulator. However, some of the control strategies are acceleration based or torque based, considering also the manipulator dynamics [11, 15, 26, 27]. It has been established that certain acceleration-based control schemes exhibit instabilities [28]. An alternative is the augmented Jacobian, as introduced in [2]. Here, a secondary task is added to the primary task to obtain a square and, therefore, an invertible Jacobian matrix. The drawback to this technique is the algorithmic singularities, which occur when the secondary task causes a conflict with the primary task. The use of the second-order inverse kinematic, either at the torque or acceleration level, was thoroughly explored by Khatib [29], resulting in the recent task-prioritized humanoid applications [30–32].

Most of the local obstacle-avoidance strategies at the kinematic level aim at assigning a motion component away from the obstacle for every point on the manipulator close to the obstacle [12–14, 16, 17, 19]. A similar situation applies to the presented proposed strategies. The emphasis of the presentation is on the definition of the avoiding motion. The latter is typically defined in Cartesian space, and this can be used to define the obstacle avoidance as a simple one-dimensional problem, with a one-dimensional operational space for each critical point. This avoids singularity issues when the redundancy level is locally too low. Alternatively, an approximative calculation can be used for the avoiding motion. In contrast to the exact avoiding motion as proposed in [17], the obtained velocity direction does not exactly coincide with the direction away from the obstacle [33]; however, the calculation is faster. In the case of multiple obstacles the situation is even more complex and more specific methods have to be applied, which also consider the relationship between the obstacles and the required avoidance movements. In the chapter we discuss strategies that consider multiple, simultaneously active obstacles in the neighborhood of the robot.

Control of a manipulator, that is redundant with respect to the task can be broken down to control subtasks with different priorities. The main, also called the primary, task is commonly associated with the end-effector pose (position and orientation). Other sub-tasks, such as obstacle avoidance, joint configuration, etc., are then given

lower priorities. Sometimes, this is not the case. For example, the safety of the robot or objects/people in its workspace could be more important, and should also be fulfilled if the end-effector motion is disturbed. In dynamical environments the priority of the tasks can also change with time. In general, task-priority algorithms do not provide a simple means of changing the priority of tasks or transitions between them [34]. In the chapter we present a formulation that makes the end-effector pose the secondary task and obstacle avoidance the primary one. The novelty is in making the primary task (the obstacle avoidance) active only when necessary, i.e., only when the robot crosses a predefined distance-to-the-obstacle threshold. In this aspect, while far from the obstacle, the algorithm allows undisturbed control of the secondary task (as if it were the primary task) [35–37]. Upon reaching the threshold distance, the primary task (obstacle-avoidance) smoothly takes over and only allows motion in the null-space of the primary task. A similar approach was proposed by Sugiura et al. [38], who proposed a blending solution for the end-effector motion, and by Mansard et al. [30], with a generic solution to build a smooth control law for any kind of unilateral constraints.

The last approach we present is solving the obstacle-avoidance problem with the use of novel methods of generating and encoding trajectories with dynamical systems. We show how DMPs offer the means for on-line modulation and adaption of the trajectory in order to take into account the dynamic events from the environment. Introducing a coupling term to the dynamical equations encoding the trajectory, we can modulate its spatial evolution to avoid an obstacle. The choice of the coupling term may be specialized for a given task. Various aspects and applications of the proposed dynamical systems approach are discussed and evaluated.

The computational efficiency of the proposed algorithms, both at the kinematic level using classic control, and using the dynamical systems, allows real-time application in cluttered and/or time-varying environments. We demonstrate the applicability with simulations of a highly redundant planar manipulator moving in an unstructured and time-varying environment and by experiments on a real robot manipulator.

## 2 Background

The robotic systems under study are redundant serial manipulators. We consider the robot as a redundant system when the dimension of the joint space  $n$  exceeds the dimension of the task space  $m$ . The difference between  $n$  and  $m$  is denoted as the degree of redundancy  $r = n - m$ . Note that this definition of the redundancy is not only a characteristic of the manipulator itself, but also of the task. This means that a nonredundant manipulator may also become a redundant manipulator for a specific task.

The relationship between the configuration variable  $\mathbf{q}$  and the task variable  $\mathbf{x}$  can be described by the following equation

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \quad (1)$$

where  $f$  is an  $m$ -dimensional vector function. The corresponding relationship between the joint velocities  $\dot{q}$  and the task velocities  $\dot{x}$  is obtained by differentiating (1)

$$\dot{x} = \mathbf{J}\dot{q} \quad (2)$$

where  $\mathbf{J}$  is the  $m \times n$  Jacobian matrix. The control problem is how to generate the motion in joints that will result in the desired task-space motion. At the velocity kinematic level this means calculating  $\dot{q}$  using the desired task-space velocities  $\dot{x}$ . For a non-redundant manipulator ( $n = m$ ) and when the robot is not in a singular configuration  $\dot{q}$  ( $\mathbf{J}$  has full rank,  $\text{rank}(\mathbf{J}) = n$ ) the joint velocities  $\dot{q}$  can be calculated from (2) as

$$\dot{q} = \mathbf{J}^{-1}\dot{x} \quad (3)$$

where  $\mathbf{J}^{-1}$  is the inverse of the Jacobian matrix  $\mathbf{J}$ . To avoid any drifts, a task-space controller is usually implemented for  $\dot{x}$ , namely

$$\dot{x} = \dot{x}_e + \mathbf{K}\mathbf{e} \quad (4)$$

where  $\dot{x}_e$  is the desired task-space velocity,  $\mathbf{e}$ ,  $\mathbf{e} = \mathbf{x}_d - \mathbf{x}$ , is the task-space error, and  $\mathbf{K}$  is a positive definite gain matrix.

In the case of a kinematically redundant manipulator, the manipulator possesses more DOFs than required to execute a task, i.e., the dimension of the joint space  $n$  exceeds the dimension of the task space  $m$ ,  $n > m$ . It is obvious that the Jacobian  $\mathbf{J}$  is no longer a square matrix, but an  $m \times n$  matrix, and hence the inverse  $\mathbf{J}^{-1}$  does not exist and (3) cannot be used. The classic general solution of (2) for a kinematically redundant manipulator is

$$\dot{q} = \mathbf{J}^\# \dot{x} + \mathbf{N}\dot{\phi} \quad (5)$$

where  $\mathbf{J}^\#$  is a generalized inverse of the Jacobian matrix  $\mathbf{J}$ ,  $\mathbf{N}$  is a matrix representing the projection into the null space of  $\mathbf{J}$ , and  $\dot{\phi}$  is an arbitrary  $n$ -dimensional joint-velocity vector. From (5) it is clear that  $\mathbf{N}$  projects the velocity  $\dot{q}_n$  into the null-space of  $\mathbf{J}$  and the corresponding motion does not affect the task motion. Remarkably, there is an infinite number of solutions  $\dot{q}$ . In most cases it is required to pursue a minimum-norm velocity leading, to the selection of the Moor-Penrose inverse  $\mathbf{J}^+$ ,  $\mathbf{J}^+ = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}$ , as the generalized inverse in (5)

$$\dot{q} = \mathbf{J}^+ \dot{x} + (\mathbf{I} - \mathbf{J}^+ \mathbf{J})\dot{\phi} \quad (6)$$

The first r.h.s. term in (6), i.e., the particular solution, provides the least-squares solution, i.e., it minimizes  $\|\dot{x} - \mathbf{J}\dot{q}\|$ , with a minimum joint-velocity norm. With the second r.h.s. term in (6) different joint velocities  $\dot{q}$  can be obtained that result in the same end-effector velocity  $\dot{x}$ . This additional joint motion can be exploited to

achieve some additional goals, i.e., some kind of optimization, obstacle avoidance, to fulfill some functional constraints or to execute additional constraint tasks. To perform this additional subtask, the velocity  $\dot{\phi}$  is used. Then the secondary task is defined by some motion  $x_t = f_t(\mathbf{q})$  like in the case of obstacle avoidance, the velocity  $\dot{\phi}$  can be defined as

$$\dot{\phi} = \mathbf{J}^+ \dot{x} \quad (7)$$

Another possibility is to define  $\dot{\phi}$  as

$$\dot{\phi} = \mathbf{K}_p \nabla p, \quad (8)$$

where,  $p$  is a function representing the desired performance criterion,  $\nabla p$  is the gradient of  $p$ , and  $\mathbf{K}_p$  is a gain. So, using (8) the optimization of  $p$  can be achieved.

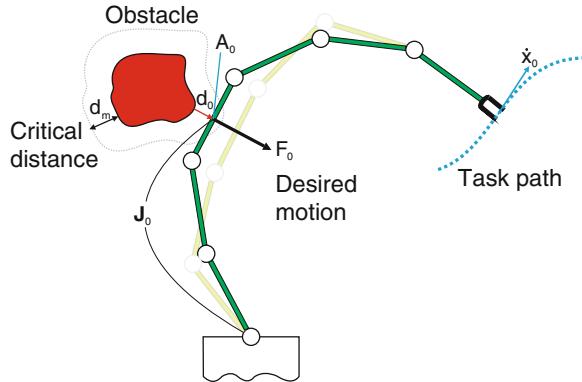
### 3 Obstacle-Avoidance Strategy

The obstacle-avoidance problem usually defines how to control the manipulator in order to track the desired end-effector trajectory while simultaneously ensuring that no part of the manipulator collides with any obstacle in the workspace of the manipulator. To avoid any possible obstacles the manipulator has to move away from them into a configuration where the distance between them becomes larger, as shown in Fig. 1. Reconfiguration of the manipulator without changing the motion of the end-effector is only possible if the manipulator has redundant DOFs. Note that in some cases it is possible that the redundant manipulator cannot avoid an obstacle, because it might be in a configuration where the avoiding motion in the desired direction is not feasible. Having a high degree of redundancy reduces the chance of getting into a such configuration, especially if the manipulator is working in an environment that has many potential collisions with obstacles.

Usually, the basic strategy for obstacle avoidance is to identify the points on the robotic arm that are near obstacles and then assign to them the motion component that moves those points away from the obstacle, as shown in Fig. 1. The robot motion (configuration) is changed if at least one part of the robot is at a critical distance from an obstacle. We denote the obstacles that are closer to the critical distance as the *active obstacles* and the corresponding closest points on the body of the manipulator as the *critical points*.

For industrial robots it is usually assumed that the motion of the end-effector is not disturbed by any obstacle. If such a situation occurs, either the task execution has to be interrupted and the higher-level path planning has to recalculate the desired motion of the end-effector or if the path-tracking accuracy is not important the control algorithms that move the end-effector around obstacles on-line can be used.

Since the position of the obstacle is usually not known in advance, the obstacle-avoidance algorithm must work in real-time. In order to ensure these requirements



**Fig. 1** Manipulator motion in the presence of some obstacles

some sensors have to be used to determine the position of the obstacles or to measure the distance between the obstacles and the body of the manipulator. There is a variety of sensor systems that can be used for such obstacle detection. In many cases a vision system is used to detect obstacles. Another possibility is offered by tactile sensors, like artificial skin, which can detect the obstacle only if they touch it, or by proximity sensors, which can sense the presence of an obstacle in the neighborhood.

## 4 Obstacle Avoidance Using Kinematic Control

The basic strategy for obstacle avoidance considers the obstacle-avoidance problem at the kinematic level. We denote  $\dot{x}_e$  as the desired velocity of the end-effector, and  $A_o$  as the critical point on the obstacle (see Fig. 1). To avoid a possible collision, one possibility is to assign a velocity to  $A_o$  such that it would move the manipulator away from the obstacle, as proposed in [17]. Here, the motion of the end-effector and the critical point can be defined as

$$\mathbf{J}\dot{\mathbf{q}} = \dot{x}_e \quad \mathbf{J}_o\dot{\mathbf{q}} = \dot{x}_o \quad (9)$$

where  $\mathbf{J}_o$  is a Jacobian matrix associated with the point  $A_o$ . In the following, different possibilities for finding the solution for both equations will be presented.

### 4.1 Exact Solution

Let  $\dot{x}$  in (5) be equal to  $\dot{x}_e$ . Then, by combining (5) and (9) we obtain

$$\dot{\phi} = (\mathbf{J}_o \mathbf{N})^\# (\dot{x}_o - \mathbf{J}_o \mathbf{J}^\# \dot{x}_e) \quad (10)$$

Using  $\dot{\phi}$  in (5) gives the final solution for  $\dot{q}$  in the form

$$\dot{q} = \mathbf{J}^\# \dot{x} + (\mathbf{J}_o \mathbf{N})^\# (\dot{x}_o - \mathbf{J}_o \mathbf{J}^\# \dot{x}_e) \quad (11)$$

Note that  $\mathbf{N}$  is both hermitian and idempotent [4, 17]. Here the first term  $\mathbf{J}^\# \dot{x}$  guarantees the tracking of the desired end-effector. Also,  $\dot{x}$  is used in (11) instead of  $\dot{x}_e$  to indicate that a task-space controller can be used to compensate for any task-space tracking errors

$$\dot{x} = \dot{x}_d + \mathbf{K}e. \quad (12)$$

where  $\dot{x}_d$  is the desired task-space velocity,  $\mathbf{K}$  is an  $m \times m$  positive-definite matrix and  $e$  is the task-position error, defined as

$$e = x_d - x. \quad (13)$$

Here,  $x_d$  is the desired task-space position. The second term in (11), i.e., the homogeneous solution  $\dot{q}_h$ , represents the part of the joint velocity causing the motion of the point  $A_o$ . The term  $\mathbf{J}_o \mathbf{J}^\# \dot{x}_e$  is the velocity in  $A_o$  due to the end-effector's motion. The matrix  $\mathbf{J}_o \mathbf{N}$  is used to transform the desired critical point velocity from the operational space of the critical point into the joint space. Note that the above solution guarantees that we achieve exactly the desired  $\dot{x}_o$  only if the degree of redundancy of the manipulator is sufficient.

## 4.2 Exact Solution with Reduced Operational Space

The system's ability to avoid obstacles is defined with the matrix  $\mathbf{J}_o \mathbf{N}$ , which combines the kinematics of the critical point  $A_o$  and the null-space matrix of the whole manipulator. Here, the properties of the matrix  $\mathbf{J}_o \mathbf{N}$  depend on the position of the point  $A_o$  and also on the definition of the operational space associated with the critical point  $A_o$ . Usually, all the critical points are defined in Cartesian space, which implies that the velocity  $\dot{x}_o$  is a 3-dimensional vector and the dimension of the matrix  $\mathbf{J}_o \mathbf{N}$  is  $3 \times n$ . This means that at least 3 DOFs are needed to move one point from an obstacle. Consequently, it might seem that a manipulator with two redundant DOFs is not capable of avoiding obstacles. However, we know from our experience that this is not true. For example, consider a planar 3 DOF manipulator that can move along a straight line and only the positions of the end-effector are important. In this case, the task space is 2-dimensional and the manipulator has one free degree of redundancy. Defining the velocity  $\dot{x}_o$  in the same space as the end-effector velocity, i.e., as a 2-dimensional vector, reveals the matrix  $\mathbf{J}_o \mathbf{N}$  to have the dimension  $2 \times 3$ . Furthermore, due to one degree of redundancy the components of the velocity vector  $\dot{x}_o$  are not independent. Hence, the rank of  $\mathbf{J}_o \mathbf{N}$  is one, and the pseudo-inverse  $(\mathbf{J}_o \mathbf{N})^\#$  does not give a feasible solution, at least the desired avoiding velocity  $\dot{x}_o$  cannot be achieved.

On the other hand, as the obstacle-avoidance strategy only requires motion in the direction of the line connecting the critical point with the closest point on the obstacle, this is a one-dimensional constraint for which only one degree of redundancy is needed. Therefore, we propose using a reduced operational space [39] for the obstacle avoidance and define the Jacobian  $\mathbf{J}_{d_o}$  as follows.

Let  $\mathbf{d}_o$  be the vector connecting the closest points on the obstacle and the manipulator (see Fig. 1) and let the operational space in  $A_o$  be defined as one-dimensional space in the direction of  $\mathbf{d}_o$ . Then, the Jacobian that relates the joint-space velocities  $\dot{\mathbf{q}}$  and the velocity in the direction of  $\mathbf{d}_o$  can be calculated as

$$\mathbf{J}_{d_o} = \mathbf{n}_o^T \mathbf{J}_o \quad (14)$$

where  $\mathbf{J}_o$  is the Jacobian defined in the Cartesian space and  $\mathbf{n}_o$  is the unit vector in the direction of  $\mathbf{d}_o$ ,  $\mathbf{n}_o = \frac{\mathbf{d}_o}{\|\mathbf{d}_o\|}$ . Now, the dimension of the matrix  $\mathbf{J}_{d_o}$  is  $1 \times n$ , and the velocities  $\dot{x}_o$  and  $\mathbf{J}_{d_o} \mathbf{J}^\# \dot{x}_e$  become scalars. Consequently, the computation of  $(\mathbf{J}_{d_o} \mathbf{N})^\#$  is also much faster [33, 35, 39]. Note that in this case we do not have to invert any matrix because the term  $(\mathbf{J}_{d_o} \mathbf{N} \mathbf{J}_{d_o}^T)$  is a scalar.

### 4.3 Selection of Avoiding Velocity

The performance of the obstacle-avoidance algorithm mainly depends on the selection of the desired critical point velocity  $\dot{x}_o$ . We propose changing  $\dot{x}_o$  with respect to the distance to the obstacle  $\|\mathbf{d}_o\|$

$$\dot{x}_o = \alpha_v v_o \quad (15)$$

where  $v_o$  is the nominal velocity and  $\alpha_v$  is the obstacle-avoidance gain defined as

$$\alpha_v = \begin{cases} \left( \frac{d_m}{\|\mathbf{d}_o\|} \right)^2 - 1 & \text{for } \|\mathbf{d}_o\| < d_m \\ 0 & \text{for } \|\mathbf{d}_o\| \geq d_m \end{cases} \quad (16)$$

where  $d_m$  is the critical distance to the obstacle. If the obstacle is too close ( $\|\mathbf{d}_o\| \leq d_b$ ) the main task should be stopped. The distance  $d_b$  is subjected to the dynamic properties of the manipulator and can also be a function of the relative velocity  $\dot{\mathbf{d}}_o$ . To ensure smooth transitions it is important that the magnitude of  $\dot{x}_o$  at  $d_m$  is zero. Special attention has to be given to the selection of the nominal velocity  $v_o$ . Large values of  $v_o$  would cause unnecessarily high velocities, which results in a rapid movement far from the obstacle. Such motion is undesirable and may cause problems, especially if there are more obstacles in close proximity. Namely, the manipulator may bounce between them. On the other hand, too small a value of  $v_o$  would not move the manipulator away from the critical point, which is undesirable as well. Selecting the right  $v_o$  is a trade-off between how quickly and how smoothly the robot avoids the obstacle.

For smoothing the motion Maciejewski et al. [17] proposed a factor  $\alpha_h$ , which changed the amount of homogenous solution to be included in the total solution

$$\dot{\mathbf{q}} = \mathbf{J}^\# \dot{\mathbf{x}} + \alpha_h (\mathbf{J}_{d_o} \mathbf{N})^\# (\dot{\mathbf{x}}_o - \mathbf{J}_{d_o} \mathbf{J}^\# \dot{\mathbf{x}}_e) \quad (17)$$

In our case we have selected  $\alpha_h$  as

$$\alpha_h = \begin{cases} 1 & \text{for } \|\mathbf{d}_o\| \leq d_m \\ \frac{1}{2} \left( 1 - \cos \left( \pi \frac{\|\mathbf{d}_o\| - d_m}{d_i - d_m} \right) \right) & \text{for } d_m < \|\mathbf{d}_o\| < d_i \\ 0 & \text{for } d_i \leq \|\mathbf{d}_o\| \end{cases} \quad (18)$$

where  $d_i$  is the distance at which the obstacle influences the motion. Note that in the region between  $d_b$  and  $d_m$  the complete homogenous solution is included in the motion specification and the avoidance velocity is inversely related to the distance. Between  $d_m$  and  $d_i$  the avoidance velocity is zero and only a part of the homogenous solution is included. As the homogenous solution compensates for the motion in the critical point due to the end-effector motion, the relative velocity between the obstacle and the critical point decreases when approaching from  $d_i$  to  $d_m$ , if the obstacle is not moving. With such a selection of  $\alpha_v$  and  $\alpha_h$ , smooth velocities can be obtained.

The control law given by (17) was derived for a single obstacle. When more than one obstacle is active at the same time, then the worst-case obstacle, which is the nearest, has to be used. This solution may result in discontinuous velocities and may cause oscillations in some cases. In particular when switching between active obstacles the particular homogenous solutions are not equal and a discontinuity in the joint velocities may occur. To improve this behavior we propose using a weighted sum of the homogenous solution of all the active obstacles

$$\dot{\mathbf{q}} = \mathbf{J}^\# \dot{\mathbf{x}} + \sum_{i=1}^{n_o} w_i \alpha_{h,i} \dot{\mathbf{q}}_{h,i} \quad (19)$$

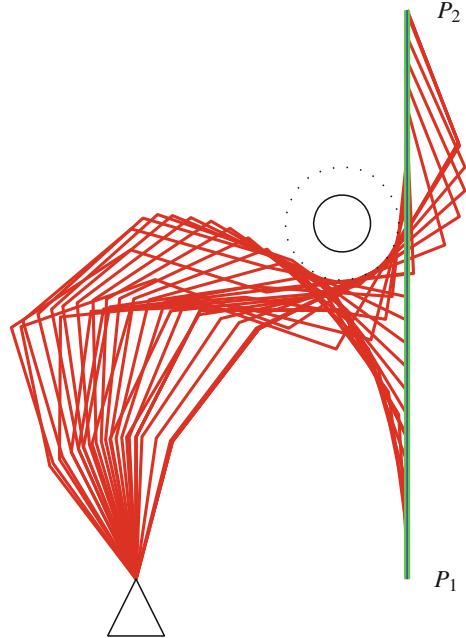
where  $n_o$  is the number of active obstacles, and  $w_i$ ,  $\alpha_{h,i}$  and  $\dot{\mathbf{q}}_{h,i}$  are the weighting factor, the gain and the homogenous solution for the  $i$ th active obstacle, respectively. The weighting factors  $w_i$  are calculated as

$$w_i = \frac{d_i - \|\mathbf{d}_{o,i}\|}{\sum_{i=1}^{n_o} (d_i - \|\mathbf{d}_{o,i}\|)} \quad (20)$$

Although the actual velocities in the critical points differ from the desired ones, using an exact solution significantly improves the performance.

As an illustration we present the simulation of a planar manipulator with five revolute joints. The primary task is to move along a straight line from point  $P_1$  to point  $P_2$ . The desired trajectory is shown by the green line in Fig. 2. The task

**Fig. 2** Planar 5 DOF manipulator: tracking of a line from point  $P_1$  to  $P_2$  and obstacle avoidance using an exact solution



trajectory has a trapezoid velocity profile with an acceleration of  $4 \text{ ms}^{-2}$  and a max. velocity of  $0.4 \text{ ms}^{-1}$ . We chose the critical distance  $d_m = 0.2 \text{ m}$  and the radius of the obstacle was  $r = 0.2 \text{ m}$ . The initial configuration of the manipulator was selected such that the motion was obstructed by an obstacle. The simulation results using the exact velocity controller EX (17) are presented in Figs. 2 and 3.

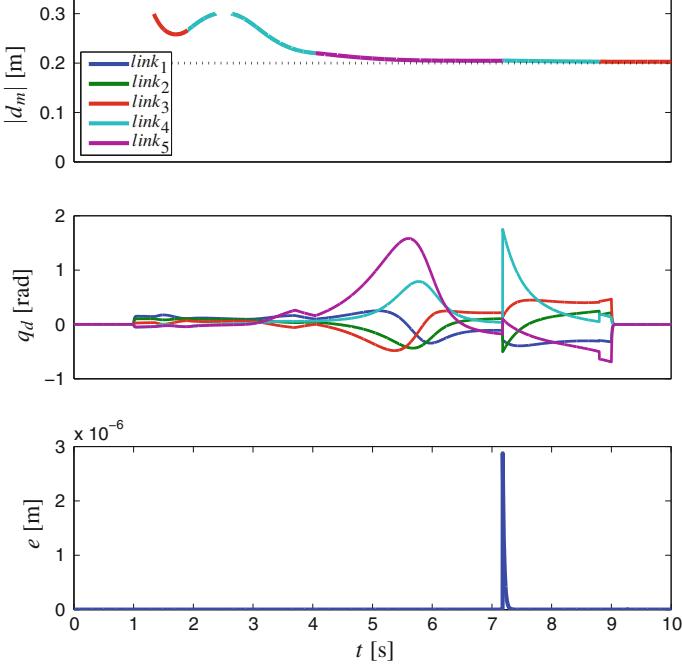
In the top plot in Fig. 3 we can see that the critical distance  $d_m$  is always above the predefined threshold  $d_0 = 0.2$ . However, in the middle plot we can see that with the exact method in some cases the joint velocities may not be smooth, which may also reflect in the tracking accuracy, as shown in the bottom plot. Even so, note that the tracking accuracy of the end-effector is in the range of  $10^{-6}$ .

#### 4.4 Approximate Solution

Another possible solution for  $\dot{\phi}$  is to calculate the joint velocities for the secondary goal as

$$\dot{\phi} = \mathbf{J}_{d_o}^{\#} \dot{x}_o \quad (21)$$

without compensating for the contribution of the end-effector motion and then substituting  $\dot{\phi}$  into (5) yields



**Fig. 3** *Top plot* shows the distance between the obstacle and the nearest link. *Middle plot* shows the joint velocities and the *bottom plot* shows the end-effector tracking error

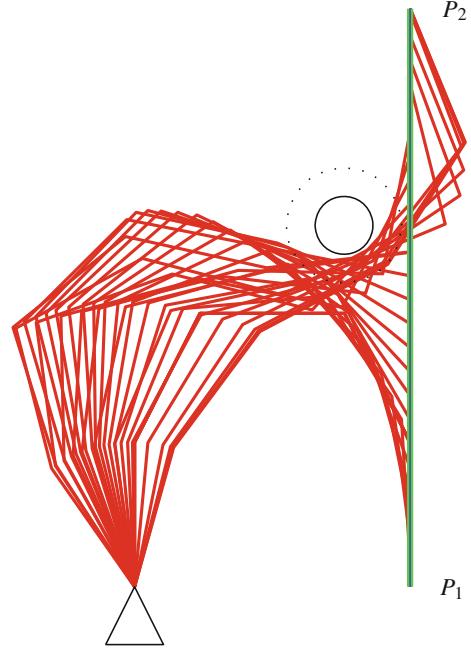
$$\dot{\mathbf{q}} = \mathbf{J}^\# \dot{\mathbf{x}} + \mathbf{N} \mathbf{J}_{d_o}^\# \dot{x}_o \quad (22)$$

This approach avoids the singularity problem of  $(\mathbf{J}_{d_o} \mathbf{N})$  [1]. The formulation (22), however, does not guarantee that the desired  $\dot{x}_o$  will be exactly achieved even if the degree of redundancy is sufficient. This is because in general  $\mathbf{J}_{d_o} \mathbf{N} \mathbf{J}_{d_o}^\# \dot{x}_o$  is not equal to  $\dot{x}_o$ .

To avoid the obstacle the goal velocity in  $A_o$  is represented by the vector  $\dot{x}_o$ . Using the original method (11) the velocity in  $A_o$  is exactly  $\dot{x}_o$ . The joint velocities in the exact solution ensure that the component of the velocity at point  $A_o$  (i.e.,  $\mathbf{J}_o \dot{\mathbf{q}}$ ) in the direction of  $\dot{x}_o$  is as required. The approximate solution gives, in most cases, a smaller magnitude of the velocity in the direction of  $\dot{x}_o$ . Therefore, the manipulator moves closer to the obstacle when an approximate solution is used. This is not so critical, because the minimum distance also depends on the nominal velocity  $v_o$ , which can be increased to achieve larger minimum distances, if needed. Additionally, the approximate solution possesses certain advantages when many active obstacles have to be considered. The joint velocities can be calculated as

$$\dot{\mathbf{q}} = \mathbf{J}^\# \dot{\mathbf{x}} + \mathbf{N} \sum_{i=1}^{n_o} \mathbf{J}_{d_o,i}^\# \dot{x}_{o,i} \quad (23)$$

**Fig. 4** Planar 5 DOF manipulator: tracking of a line from point  $P_1$  to  $P_2$  and obstacle avoidance using an approximate solution



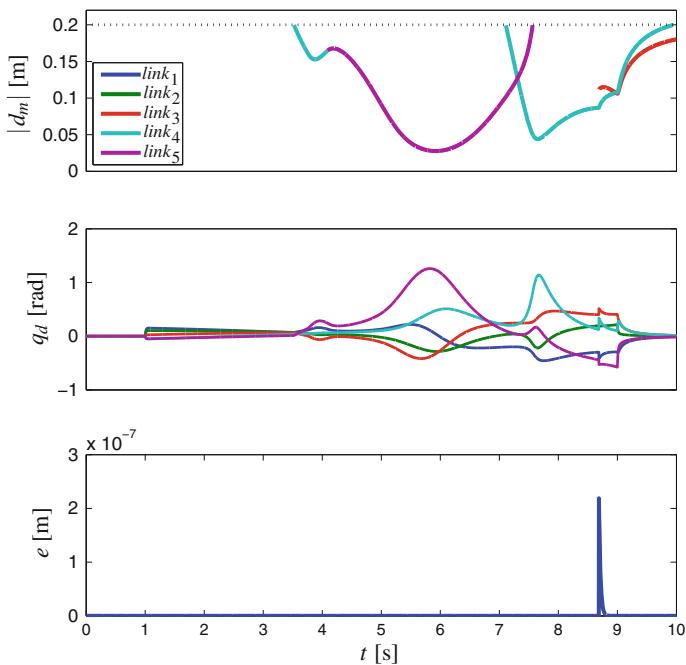
where  $n_o$  is the number of active obstacles and, therefore, the matrix  $\mathbf{N}$  has to be calculated only once. However, the pseudo-inverses  $\mathbf{J}_{o,i}^\#$  have to be calculated for each active obstacle.

We have implemented the approximate velocity controller AP (22) for the same system and the task as shown in Figs. 2 and 3. The results are presented in Figs. 4 and 5. We can see that the links are coming closer to the obstacle compared to the case of using the exact controller. Note that discontinuities in the joint velocities may also occur here, and that the tracking error of the end-effector is in the same range as in the case of the exact controller.

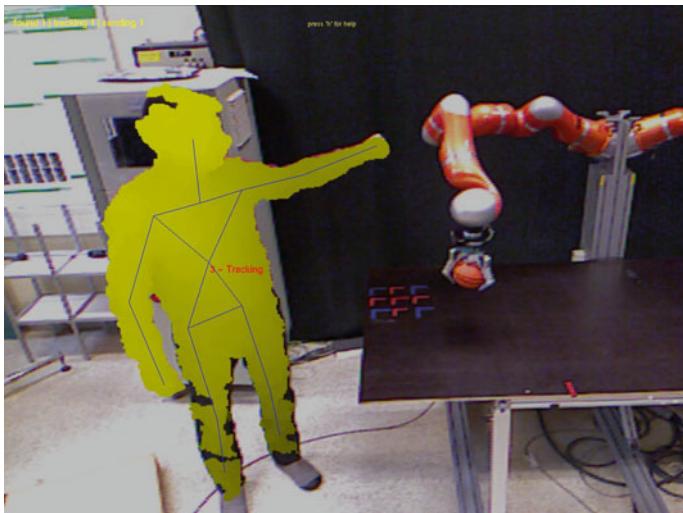
#### 4.5 Experimental Results

To support the simulation results we applied the obstacle-avoidance control using the approximated solution (23) to the 7 DOF Kuka LWR robot. The primary task for the robot was manipulating the ball in the Cartesian task space and the secondary task was avoiding human contact (a human was treated as an obstacle for the robot). The experimental setup is shown in Fig. 6.

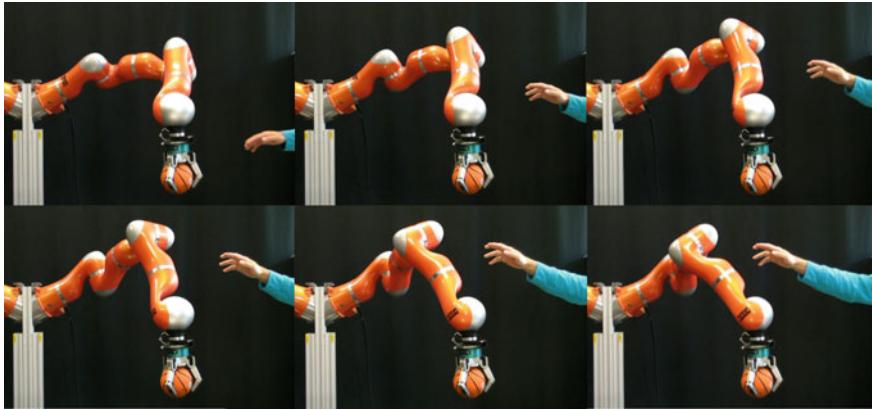
The human motion is captured using the Microsoft Kinect sensor. Microsoft Kinect is based on a range camera developed by PrimeSense, which interprets 3D scene information from a continuously projected infrared structured light.



**Fig. 5** Top plot shows the distance between the obstacle and the nearest link. Middle plot shows the joint velocities and the bottom plot shows the end-effector tracking error



**Fig. 6** Experimental setup for the manipulation task with the KUKA LWR robot, while avoiding the human in the robot workspace. The picture is taken with the Microsoft Kinect camera. Note that the picture from the Microsoft Kinect camera is mirrored



**Fig. 7** The image sequence shows obstacle avoidance using an approximate solution

By processing the depth image, the PrimeSense API enables the tracking of human limb movements in real time. To acquire the closest points (interpreted as point obstacles) between the human and the robot, we calibrated the Microsoft Kinect sensor to the robot base coordinate system. To obtain the proper transformation matrix, we recorded at least four pairs of points in both coordinate systems. During the calibration procedure the human placed his hand at the same locations as the robot end-effector and the position of the human hand and the position of end-effector were measured in the Kinect and robot base coordinate systems, respectively. The transformation matrix was calculated using least-squares fitting of two points set, as described in [40].

The results are shown as a sequence of photos in Fig. 7, where we can see a successful pose adaptation in order to prevent human contact, while maintaining the position of the end-effector.

## 5 Obstacle Avoidance as a Primary Task

The development of multi-arm robot mechanisms and humanoid robots emphasized the importance of being able to perform multiple tasks simultaneously [41–43], like controlling multiple points on the robot structure, stability, pose control or obstacle avoidance. Whether it is feasible that the robot can achieve all the goals at the same time depends on the one hand upon the robot's dexterity and its configuration, and on the other hand upon the goals themselves. Although highly redundant robot manipulators can perform multiple tasks, it is not likely that all the tasks can be fulfilled simultaneously or at least not all the time. For example, the robot may be able to perform all the tasks in one configuration, but when the robot moves to another configuration, some goals may become conflicting with the motion. In this case, it is

impossible to satisfy all the goals and the conflict can be handled in the framework of the task priority, where the tasks are arranged by their relevance. The priority indicates how important a task is compared to others and it can also imply some other things, like how important it is to execute the task accurately. Typically, the lower-priority tasks are less important and they are fulfilled completely only if not they are interfering with higher-priority tasks. The task with the highest priority is usually referred to as the primary task.

With multiple tasks it is important to know the relationship between the tasks. Assuming that each task can be executed per se, i.e., a feasible solution exists for all the tasks, this is not a guarantee that all tasks can be executed simultaneously. Namely, the motion necessary to perform one task can disturb the execution of other tasks and, hence, some tasks may become unfeasible with respect to others. The dependency between tasks can be determined by analyzing the range of the associated Jacobian inverse mappings [44–46]. It is important to know the relationship between the mapping, but it is not essential for the solution. When two tasks are disturbing each other, then it is necessary to ensure that the task with higher priority is fulfilled and then we should try to fulfil the lower-priority task as well as possible.

For a redundant robot one possible solution for obstacle avoidance is to consider the obstacle-avoidance task as a primary task  $T_a$ , and the end-effector tracking as a secondary task  $T_b$  defined by

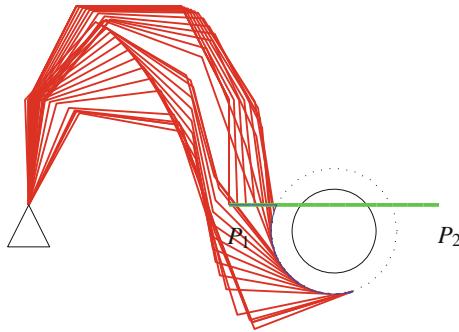
$$\mathbf{x}_a = \mathbf{f}_a(\mathbf{q}) \quad \mathbf{x}_b = \mathbf{f}_b(\mathbf{q}) \quad (24)$$

For each of the tasks, the corresponding Jacobian matrices can be defined as  $\mathbf{J}_a$  and  $\mathbf{J}_b$ , with the corresponding null-space projections denoted by  $\mathbf{N}_a$  and  $\mathbf{N}_b$ . Assuming that task  $T_a$  is the primary task, Eq. (5) can be rewritten as

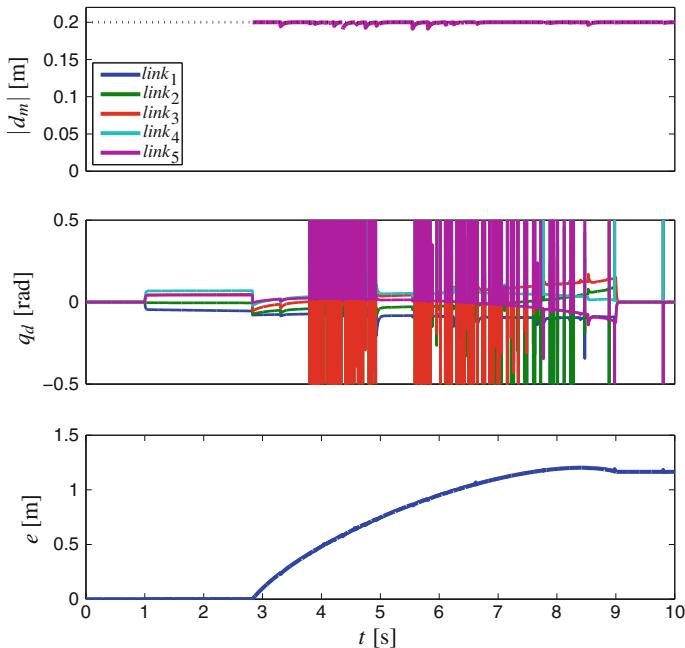
$$\dot{\mathbf{q}} = \mathbf{J}_a^\# \dot{\mathbf{x}}_a + \mathbf{N}_a \mathbf{J}_b^\# \dot{\mathbf{x}}_b \quad (25)$$

Previously, we have assumed that the end-effector motion is not disturbed by an obstacle. Now, it is assumed that the motion of the end-effector can be disturbed by any obstacle. If such a situation occurs, the task execution usually has to be interrupted and higher-level path planning has to be employed to recalculate the desired motion of the end-effector. However, if the end-effector path tracking is not essential, we can use the proposed control (25). Consequently, no end-effector path recalculation or higher-level path planning is needed.

Figure 8 shows an example of the prioritized control where we can see that in this case the robot can avoid obstacles even if they appear on the Cartesian task path. The same parameter set was used as in Sect. 4, except for the obstacle diameter, which was now set to  $r = 0.4$  m. In Fig. 9 we can also see that the critical distance  $d_m$  is exactly the same as the predefined  $d_0 = 0.2$ , which was expected since the obstacle-avoidance task is now the task with the highest priority. In contrast, in this particular example, we can see that such an approach has a disadvantage when compared to the global path search algorithms since the resulting motion may be suboptimal and as a result it may become stuck.



**Fig. 8** Planar 5 DOF manipulator: tracking of a line from point  $P_1$  to  $P_2$  is a secondary task and obstacle avoidance is the primary task



**Fig. 9** Top plot shows the distance between the obstacle and the nearest link. Middle plot shows the joint velocities and the bottom plot shows the end-effector tracking error

### 5.1 Smooth Transition Between Tasks

Another important aspect that should be considered with multiple tasks is the ability to change the task priority. When a robot is working in a changing environment, it may happen that the situation requires that one task becomes more important than before. A good example is obstacle avoidance, where the priority of the avoiding task

may depend on the type of obstacle and on the distance to the obstacle. Therefore, it is beneficial if the control method enables a smooth change of task priorities. Using formulation (25) this cannot be done in a smooth way. Therefore, we propose a new definition of the velocity  $\dot{\mathbf{q}}$  [35]. The velocity  $\dot{\mathbf{q}}$  is now defined as

$$\dot{\mathbf{q}} = \mathbf{J}_a^\# \dot{\mathbf{x}}_a + \mathbf{N}'_a \mathbf{J}_b^\# \dot{\mathbf{x}}_b, \quad (26)$$

where the matrix  $\mathbf{N}'_a$  is given as

$$\mathbf{N}'_a = \mathbf{I} - \lambda(\mathbf{x}_a) \mathbf{J}^\# \mathbf{J}, \quad (27)$$

where  $\lambda(\mathbf{x}_a)$  is a scalar measure of how “active” is the primary task  $T_a$ , scaling the vector  $\mathbf{x}_a$  to the interval  $[0, 1]$ . When the primary task  $T_a$  is active  $\lambda$  is  $\lambda(\mathbf{x}_a) = 1$ , and when the task  $T_a$  is not active, it is  $\lambda(\mathbf{x}_a) = 0$ .

The proposed algorithm allows a smooth transition in both ways, i.e., between observing the task  $T_a$  and the task  $T_b$  in the null-space of the task  $T_a$  or just the unconstrained movement of the task  $T_b$ . The proposed approach is general and can be used for different robotic tasks.

For obstacle avoidance using (26), we define the primary task  $T_a$  to be the motion in the direction  $\mathbf{d}_0$  and the motion of the end-effector to be the task  $T_b$ . Using the reduced operational space yields

$$\mathbf{J}_a = \mathbf{J}_{d_o}, \quad (28)$$

$$\mathbf{J}_b = \mathbf{J}. \quad (29)$$

Next, (26) can be rewritten in the form

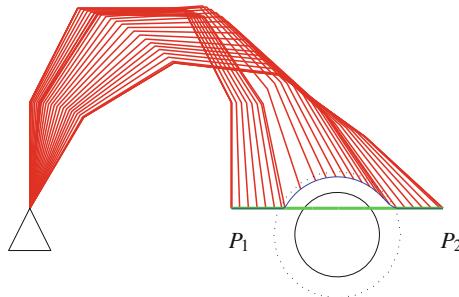
$$\dot{\mathbf{q}} = \mathbf{J}_{d_o}^\# \dot{\mathbf{x}}_o + \mathbf{N}'_0 \mathbf{J}^\# \dot{\mathbf{x}}. \quad (30)$$

Here,  $\dot{\mathbf{x}}$  is the task controller for the end-effector tracking and let  $\lambda(\mathbf{d}_0) = \alpha_h$ , then  $\mathbf{N}'_0$  is given by

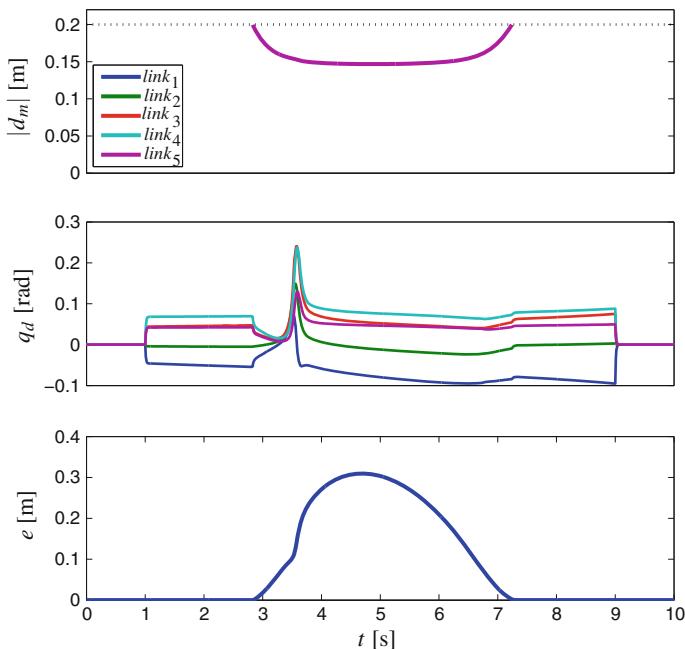
$$\mathbf{N}'_0 = \mathbf{I} - \alpha_h \mathbf{J}_o^\dagger \mathbf{J}_o. \quad (31)$$

Formulation (30) allows an unconstrained joint movement while  $\alpha_h$  is close to zero ( $\alpha_h \approx 0$ ). Thus, the robot can track the desired task-space path while it is away from the obstacle. On the other hand, when the robot is close to the obstacle ( $\alpha_h \approx 1$ ), the null space in (31) takes the form  $\mathbf{N}'_0 = \mathbf{N}_0$ , and only allows movement in the null space of the primary task, i.e., the obstacle-avoidance task. In this case, we can still move the end-effector, but the tracking error can increase due to the obstacle-avoiding motion.

Simulation results using the control algorithm (30) are presented in Figs. 10 and 11. We can see in Fig. 10 and in the top plot of Fig. 11 that in the case of a smooth transition between tasks the tracking error may become significant while



**Fig. 10** Planar 5 DOF manipulator: smooth transition between the primary task of obstacle avoidance and the secondary task of tracking a line from point  $P_1$  to point  $P_2$



**Fig. 11** *Top plot* shows the distance between the obstacle and the nearest link. *Middle plot* shows the joint velocities and the *bottom plot* shows the end-effector tracking error

the robot is close to the obstacle. The main reason for such behaviour is that in this case the obstacle-avoidance becomes primary and the end-effector tracking is the secondary task projected into the null space of the obstacle-avoidance task. Even though this may seem impractical, it is useful in situations when the obstacle is in the path of the end-effector. Since by using such control, the robot can avoid obstacles in real-time without using any additional path-planners if obstacles appear on the end-effector path during the motion.

An important observation is also that for this particular task and for the same configuration and parameter set as used in the example presented in Fig. 9, the robot does not become stuck in the local minimum. The main reason for such behaviour is that the transition to obstacle avoidance is now smooth and consistent. However, as we can see in the top plot in Fig. 11, as a consequence the robot comes closer to the obstacle. Note that this minimal distance to the obstacle could be increased by increasing the value of  $d_0$ .

## 5.2 Prioritized Damped Least-Squares Inverse

Another possibility for simultaneous end-effector tracking and obstacle-avoidance simultaneously is to treat them equally. Let us stack all the tasks the robot should perform  $x_i, i = 1, \dots, k$  into an extended task vector

$$\mathbf{x}_E = \left[ \mathbf{x}_1^T, \mathbf{x}_2^T, \dots, \mathbf{x}_k^T \right]^T \quad (32)$$

Then, the relation between the task space velocities and the joint velocities is given as

$$\dot{\mathbf{x}}_E = \mathbf{J}_E \dot{\mathbf{q}} \quad (33)$$

where the extended Jacobian is given in the form

$$\mathbf{J}_E = \left[ \mathbf{J}_1^T, \mathbf{J}_2^T, \dots, \mathbf{J}_k^T \right]^T \quad (34)$$

The solution to (33) (denoted later as E) is given in the form

$$\dot{\mathbf{q}} = \mathbf{J}_E^\# \dot{\mathbf{x}}_E \quad (35)$$

As all the tasks are included in  $\dot{\mathbf{x}}_E$  there is no need to consider the homogenous part of the solution, i.e., the null-space velocity, to solve these tasks. If the rank of  $\mathbf{J}_E$  equals at least the dimension of all the tasks,  $\text{rank}(\mathbf{J}_E) \geq m_t$ , then the solution to (35) results in  $\dot{\mathbf{q}}$ , which fulfill all the tasks.

Even though the approaches proposed by [2, 44, 47–49], for the calculation of joint velocities in the case of multiple prioritized tasks, solve the inverse kinematic problem when the system of equations is not ill-conditioned, it is likely that during the execution of multiple tasks the manipulator moves toward the configuration where one of the Jacobian matrices is near singularity and, consequently, the obtained joint velocities  $\dot{\mathbf{q}}$  become unfeasible. To overcome the problem of unfeasible velocities we could apply the damped least-squares (DLS) technique. Applying DLS to the extended Jacobian method gives feasible joint velocities. However, if the rank of the extended Jacobian  $\mathbf{J}_E$  is not sufficient with respect to the dimensions of all the tasks

$$\text{rank}(\mathbf{J}_E) < \sum_{i=1}^k m_i \quad (36)$$

then (35) results in a “best fit” (in a least-squares sense) solution. Since in (35) all the tasks are treated equally, it is not possible to prioritize some of the tasks in favor of others. To overcome this drawback we propose an approach in the framework of a DLS extended Jacobian [48, 50].

The basis of this method is a combination of the extended Jacobian approach (35) and the DLS inverse technique. The proposed solution is given in the form

$$\dot{\mathbf{q}} = \mathbf{J}_E^\# \dot{\mathbf{x}}_E \quad (37)$$

where

$$\mathbf{J}_E^\# = \mathbf{J}_E^T (\mathbf{J}_E \mathbf{J}_E^T + \lambda^2 \mathbf{P})^{-1} \quad (38)$$

and  $(P)$  is an  $m_t \times m_t$  diagonal matrix

$$\mathbf{P} = \begin{bmatrix} p_1 \mathbf{I}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & p_2 \mathbf{I}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & p_k \mathbf{I}_k \end{bmatrix} \quad (39)$$

where  $p_i$  are scalars depending on the desired priority of the task  $T_i$ , and  $\mathbf{I}_i$  are  $m_i \times m_i$  unit matrices. We denote this method as the *priority weighted damped least-squares* Jacobian method (denoted later as PWDLS). The proposed solution (38) with priority factors (39) minimizes

$$\lambda^2 \|\dot{\mathbf{q}}\| + \sum_{i=1}^k p_k \|\dot{\mathbf{x}}_k - \mathbf{J}_k \dot{\mathbf{q}}\| \quad (40)$$

The method is similar to the method proposed in [50] except that the weighting factors are defined by the priority of the tasks. For improving the performance it is essential to suitably select the factors in the damping term in (38). To focus on the priority issue of the problem, we assume that the optimal value for the damping factor  $\lambda$  has been selected using one of the well-known methods [48, 51–54]. To determine the optimal value of  $\lambda$  all the values  $p_i$  are set to 1, i.e.,  $\mathbf{P} = \mathbf{I}$ .

When dealing with the priority in the framework of redundancy resolution, the terms *primary task*, *secondary task*, and so on, imply that the control has fulfilled the primary task first, and next the secondary task, without disturbing the primary task. This philosophy is used by all redundancy-resolution schemes dealing with prioritized tasks. None of the redundancy schemes can deal with the information about “how much” one task is more important than the other. For example, even for

the obstacle-avoidance schemes, where the distance to the obstacle can be used as a measure of the importance of particular critical points, this information is actually used only to order the critical points. On the other hand, the parameters  $p_i$  can be used to quantify the relative importance of the tasks  $T_i$ . So, it is possible to quantify the priorities of the tasks [55]. It is obvious that the following relation must hold

$$\text{Priority}(T_i) > \text{Priority}(T_j) \Leftrightarrow p_i < p_j, \quad i, j \in \{1, \dots, k\} \quad (41)$$

To gain more insight into the relation between the tasks  $T_i$  one can compare the desired task velocities  $\dot{\mathbf{x}}$  and the task velocities  $\dot{\mathbf{x}}_a$  obtained as a solution of (42)

$$\dot{\mathbf{x}}_{Ea} = \mathbf{J}_E \dot{\mathbf{q}} = \mathbf{J}_E \mathbf{J}_E^\# \dot{\mathbf{x}}_e = \mathbf{A} \dot{\mathbf{x}} \quad (42)$$

The  $m_a \times m_a$  matrix  $\mathbf{A}$  represents the mapping between  $\dot{\mathbf{x}}$  and  $\dot{\mathbf{x}}_a$  and can be divided into several submatrices

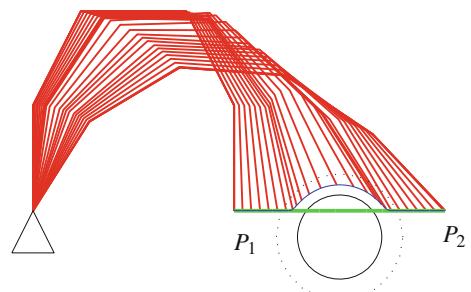
$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \dots & \mathbf{A}_{1,k} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \dots & \mathbf{A}_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{k,1} & \mathbf{A}_{k,2} & \dots & \mathbf{A}_{k,k} \end{bmatrix} \quad (43)$$

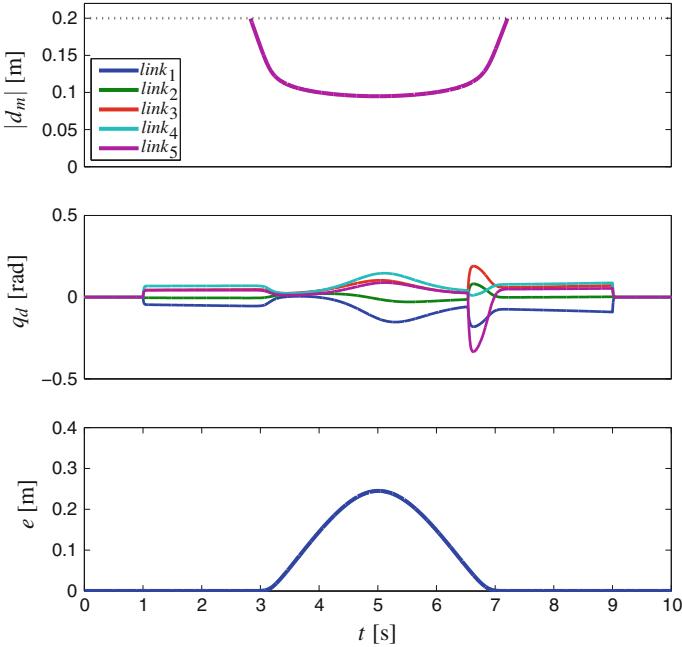
where  $\mathbf{A}_{i,j}$  are  $m_i \times m_j$  matrices. Remarkably, the diagonal matrices  $\mathbf{A}_{i,i}$  represent the transformation of the task velocity  $\dot{\mathbf{x}}_i$  in the space of the task  $T_i$ , and the off-diagonal submatrices represent the influence between the tasks. Note that as  $p_i$  are not equal,  $\mathbf{A}$  is a non-symmetric matrix. The explanation is apparent, the task with higher priority influences the task with lower priority more a vice versa.

An example of using the (42) algorithm is shown in Figs. 12 and 13. Here we can see similar behaviour as when using a smooth transition between tasks, e.g., Figs. 10 and 11. By comparing the results, the main difference between those two approaches while using the same parameter set is that in the case of PWDLs the robot comes closer to the obstacle.

In the following we present how the selection of  $p_i$  influences the solution of (42). For a better understanding we present a 4 DOF planar manipulator with revolute

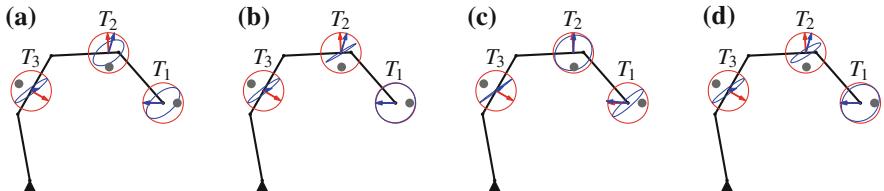
**Fig. 12** Planar 5 DOF manipulator: tracking of a line from point  $P_1$  to point  $P_2$  and obstacle avoidance using PWDLs Jacobi





**Fig. 13** Top plot shows the distance between the obstacle and the nearest link. Middle plot shows the joint velocities and the bottom plot shows the end-effector tracking error

joints where three control points have to be moved in different directions due to the obstacles near the robot. Note that in this example, the distance between each obstacle and the robot body is the same for all obstacles. Consequently, the desired avoiding motion is similar for all the critical points (except the direction, of course). We assume that only the positions of the control points are important and so the tasks are 2-dimensional,  $m_i = 2$ . Consequently,  $m_t = 6$  and  $n = 4$ . As  $\mathbf{J}_E$  has more rows than columns, the system is overdetermined and no exact solution exists. Figure 14 shows the situation for four different selections of  $P$ . The case a) presents the solution without prioritizing tasks (as a classic extended Jacobian approach). The



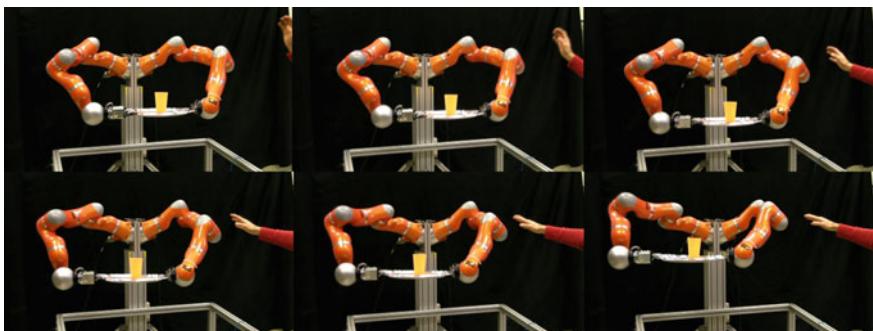
**Fig. 14** Influence of different priority factors in (42) for three tasks and for four priorities sets: **a**  $p = [1, 1, 1]$ , **b**  $p = [1, a, a^2]$ , **c**  $p = [a, 1, a^2]$ , **d**  $p = [a, a^2, 1]$ , where  $a = 5$  and  $\lambda = 10^{-8}$ . The circles represent the geometrical representation of submatrices  $\mathbf{A}_{i,i}$ : unit sphere (red)  $\rightarrow$  ellipsoid (blue). Red vectors are the desired task velocities  $\dot{\mathbf{x}}$  and blue vectors are the resulting task vectors  $\dot{\mathbf{x}}_{\text{r}}$

other three cases show the situation when each of the tasks becomes the main task. Note that the motion in a particular control point is not only due to the desired motion in that point but in other control points the desired motion contributes to. Actually, in case (d) most of the motion in control point 2 is due to the motion of the other two tasks. As one can see, with a suitable selection of  $p_i$  the proposed method makes it possible to achieve the desired behavior of the whole system.

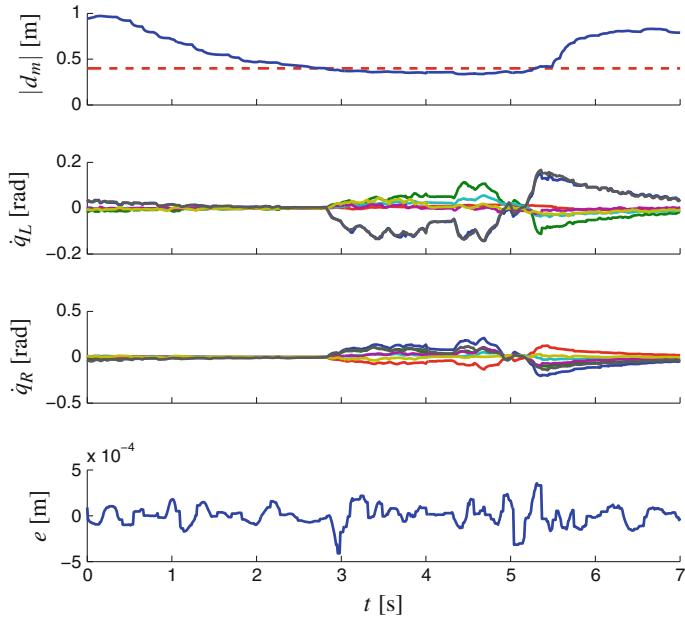
As the priority can be defined by changing the controller parameters rather than by changing the controller structure, the proposed method is also suitable when the priority has to change during the tasks' execution. Note that the priority change can be done continuously and no discontinuity in the joint-space solution  $\dot{q}$  is experienced. A method for determining the actual values of  $p_i$  is beyond the scope of this chapter. In general, it depends on the needs of all the tasks and the specific circumstances during the tasks' execution.

### 5.3 Experimental Results

To demonstrate the properties of the algorithm given with (42) we extended the task of the bimanual cooperation of two Kuka LWR robots equipped with Barret-Hand grippers holding a plate while balancing a bottle [56] with the task of preventing human contact. As in the case of the experiment in Sect. 4, the human motion was obtained using the Microsoft Kinect sensor. The results are shown in Fig. 16 and as a sequence of photographs in Fig. 15, where we can see that robots are able to successfully perform multiple tasks simultaneously, i.e., preventing human-robot contact and preserving the plate's orientation.



**Fig. 15** A sequence of still photographs shows the movement of two Kuka LWR robots, while they successfully avoid a human arm that is approaching the robot in the robot's work space. The detection and tracking of the human arm was done in real time using a Microsoft Kinect sensor



**Fig. 16** Results of a bimanual cooperation of two Kuka LWR robots equipped with Barret-Hand grippers holding a plate while balancing a bottle with the task of preventing human contact. The *top plot* shows the closest distance between human and nearest robot link. *Second and third plot* shows the joint velocities for obstacle avoidance for left and right robot respectively. *Bottom plot* shows the task error of balancing a bottle on a plate

## 6 Obstacle Avoidance Using Dynamical Systems

In this section we introduce dynamic movement primitives, which can be used to encode arbitrary trajectories, and are often associated with the learning-by-demonstration approach of controlling robots. We first provide the basics of the dynamic motor primitives, followed by obstacle-avoidance modulation. The obstacle avoidance in the DMP framework presented here is a modified approach of [57]. Simulated and real-world results are presented.

### 6.1 Dynamic Movement Primitives

The theoretical foundations of the dynamic movement primitives (DMPs) trajectory representation was developed by Ijspeert et al. [58]. Here the discussion is limited to discrete movement primitives, which can encode control policies for discrete point-to-point movements. See [59–61] for the discussion of rhythmic DMPs. The representation proposed by Ijspeert et al. is based on a set of nonlinear

differential equations with a well-defined attractor dynamics. We used the most current formulation as outlined in [57]. For a single degree of freedom denoted by  $y$ , which can either be one of the internal joint angles or one of the external task-space coordinates, the following system of linear differential equations with constant coefficients denotes a dynamic movement primitive

$$\tau \dot{z} = \alpha_z (\beta_z (g - y) - z) + f(x), \quad (44)$$

$$\tau \dot{y} = z. \quad (45)$$

$f(x)$  is defined as a linear combination of nonlinear radial basis functions

$$f(x) = \frac{\sum_{i=1}^N w_i \Psi_i(x)}{\sum_{i=1}^N \Psi_i(x)} x, \quad (46)$$

$$\Psi_i(x) = \exp\left(-h_i (x - c_i)^2\right), \quad (47)$$

where  $c_i$  are the centers of radial basis functions distributed along the trajectory and  $h_i > 0$  their widths. Provided that the parameters  $\alpha_z$ ,  $\beta_z$ ,  $\tau > 0$  and  $\alpha_z = 4\beta_z$ , the linear part of the system (44) and (45) is critically damped and has a unique attractor point at  $y = g$ ,  $z = 0$ . A phase variable  $x$  is used in (44), (46) and (47). It is utilized to avoid the direct dependency of  $f$  on time. Its dynamics is defined by

$$\tau \dot{x} = -\alpha_x x, \quad (48)$$

with the initial value  $x(0) = 1$ .  $\alpha_x$  is a positive constant.

The weight vector  $\mathbf{w}$ , composed of weights  $w_i$ , defines the shape of the encoded trajectory. [58, 62] describe the learning of the weight vector. Multiple DOFs are realized by maintaining separate sets of (44)–(47), while a single canonical system given by (48) is used to synchronize them.

## 6.2 Obstacle Avoidance

A control policy given by the DMP can encode either separate joint trajectories, or external task-space coordinates. Obstacle avoidance in Cartesian space is easier to implement since the trajectory is usually planned in Cartesian space as well. Let us assume a three degree-of-freedom DMP system that encodes point-to-point reaching in Cartesian space. The 3-D position vector of the 3 DOF discrete dynamical system is encoded by  $\mathbf{y} = [y_1, y_2, y_3]^T$ . The objective is to generate a reaching movement to a goal state  $\mathbf{g} = [g_1, g_2, g_3]^T$ . On the way to the goal state, an obstacle is positioned at  $\mathbf{o} = [o_1, o_2, o_3]^T$  and needs to be avoided. A suitable coupling term  $\mathbf{C}_t = [C_{t,1}, C_{t,2}, C_{t,3}]^T$  for the obstacle avoidance can be formulated as follows:

$$\mathbf{C}_t = \gamma \operatorname{sig}(\|\mathbf{o} - \mathbf{y}\|) \mathbf{R} \dot{\mathbf{y}} (\pi - \phi) \exp(-\beta\phi), \quad (49)$$

where

$$\phi = \arccos \left( \frac{(\mathbf{o} - \mathbf{y})^T \dot{\mathbf{y}}}{\|\mathbf{o} - \mathbf{y}\| \|\dot{\mathbf{y}}\|} \right), \quad (50)$$

$$\text{sig}(x) = \frac{1}{1 + e^{\eta(x-d)}}, \quad (51)$$

$$\mathbf{R} = \exp \left( \left( \frac{\pi}{2} - \phi \right) \mathbf{n} \right), \quad (52)$$

$$\mathbf{n} = \frac{(\mathbf{o} - \mathbf{y}) \times \dot{\mathbf{y}}}{\|\mathbf{o} - \mathbf{y}\| \|\dot{\mathbf{y}}\|}. \quad (53)$$

$\gamma$ ,  $\beta$ , and  $\eta$  are the scaling factors and  $d$  is the distance at which the obstacle should start affecting the robot's motion. The coupling term as defined above generates a velocity component that is in a plane defined by the vectors  $\mathbf{o} - \mathbf{y}$  and  $\dot{\mathbf{y}}$ . It is also orthogonal to the line  $\mathbf{o}-\mathbf{y}$ , which is connecting the tip of the robot and the obstacle.

We can ensure that the tip of the robot, i.e., the end-effector, avoids the obstacle by adding the coupling term  $\mathbf{C}_t$  to Eq. (45)

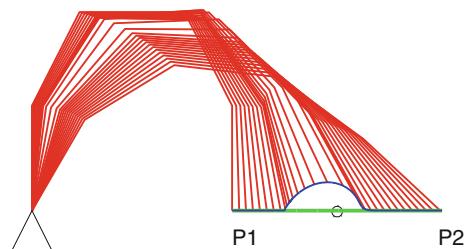
$$\tau \dot{\mathbf{z}} = \alpha_z (\beta_z (\mathbf{g} - \mathbf{y}) - \mathbf{z}) + \mathbf{f}(x) + \mathbf{C}_t \quad (54)$$

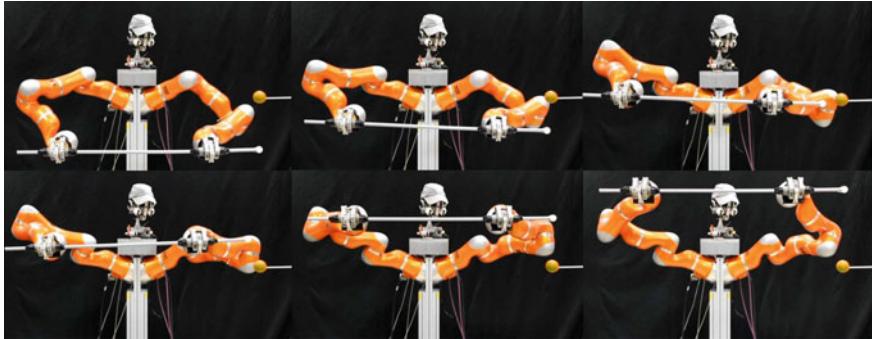
The resulting behavior is shown in Fig. 17. Note that in this way we can only ensure that the robot tip avoids the obstacle. However, the rest of the robot could still collide with it. Effectively, such an implementation of obstacle avoidance treats the problem of the end-effector collision as the primary task. Given that the DMP encodes a task-space trajectory, the actual joint trajectories are calculated using IK algorithms. Null-space obstacle avoidance such as discussed in Sect. 4 can be employed for the obstacle avoidance of separate segments of the robot.

### 6.3 Experimental Results

To show the applicability of the dynamic system for trajectory generation we applied it to two Kuka LWR robots. The task was a bimanual cooperative manipulation while avoiding obstacles. The obstacles in this example were detected using the

**Fig. 17** The obstacle is the black sphere, which is directly in the path of the robot, denoted by green. When the obstacle-avoidance term is introduced, the robot takes the blue trajectory





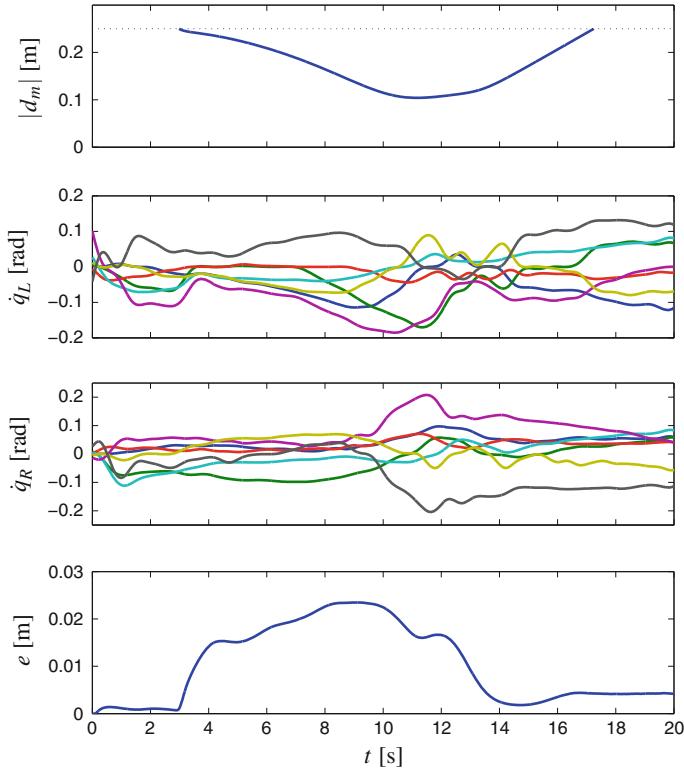
**Fig. 18** The image sequence shows a bimanual task, controlled with dynamical systems

stereo-vision cameras. The results are shown in Fig. 19 and as an image sequence in Fig. 18. As we can see one of the arms encounters an obstacle, given by the orange ball, and has to adapt its predefined trajectory (straight line) similar to that shown in the example given in Fig. 17. The control of the other arm is adapted as well in order to maintain a constant distance between them.

## 7 Conclusion

The presented approaches for on-line obstacle avoidance for redundant manipulators are based on redundancy resolution at the velocity level. For the first presented methods, the primary task is determined by the end-effector trajectories and for the obstacle avoidance the internal motion of the manipulator is used. The goal is to assign each point on the body of the manipulator, which is close to the obstacle, a velocity component in a direction that is away from the obstacle. We have shown that it is reasonable to define the avoiding motion in a one-dimensional operational space. In this way, some singularity problems can be avoided when not enough “redundancy” is available locally. Additionally, the calculation of the pseudo-inverse of the Jacobian matrix  $\mathbf{J}_o$  is simpler as it includes a scalar division instead of a matrix inversion. Using an approximate calculation of the avoiding velocities has its advantages computationally and it makes it easier to consider more obstacles simultaneously.

Next, the control algorithms are presented, where the tasks’ priorities can be altered during the execution of the motion. In the context of obstacle avoidance this means that the obstacle can also appear on the desired end-effector trajectory. For changing the priorities of the task we first show how to modify the prioritized task-control algorithm at the velocity level to implement smooth transitions between tasks with different priorities. The higher-priority task will only be active when the desired



**Fig. 19** *Top plot* shows the distance between the obstacle and the nearest link. Note that obstacle avoidance is active only under 0.25 m and that it only acts while the velocity is towards the obstacle. Once the robot is past the obstacle, the perturbation-rejection properties of DMPs ensure smooth return to the original trajectory. *Second and third plot* shows the joint velocities for left and right robot respectively, which are continuous and derivable. *Bottom plot* shows the task error

criterion is met and otherwise the higher-priority task is smoothly deactivated. This characteristic to separate tasks and to activate them only when necessary, improves the performance of the robot significantly. Furthermore, the presented method does this activation/deactivation of tasks in a smooth way. We also explain how to find the necessary motion of the robot for all the tasks simultaneously using the extended Jacobian. As such an approach does not always give a feasible solution we propose to use a priority weighted damped least-squares Jacobian for arranging the tasks by priority. In this way the best solution can be found for the particular situation. With some examples we show how the priority-based damping factors influence the motion generation for particular tasks. With a proper choice of these factors it is possible to get such joint velocities which ensure the desired behavior in the best possible way.

Finally, we show how a dynamical system for trajectory generation can be modified to be suitable for online control. Since the dynamical system can only avoid obstacles that appear in the trajectory path, it is necessary to use a control method that can modify the robot null-space configuration if needed. The combination of both dynamical systems for trajectory generation and control with obstacle avoidance is a powerful framework that can easily be used in different applications.

## References

- Chiaverini S (1997) Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Trans Robot Autom* 13(3):398–410. doi:[10.1109/70.585902](https://doi.org/10.1109/70.585902)
- Egeland O (1987) Task-space tracking with redundant manipulators. *IEEE J Robot Autom* 3(5):471–475. doi:[10.1109/JRA.1987.1087118](https://doi.org/10.1109/JRA.1987.1087118)
- Lenarcic J, Stanisic M (2003) A humanoid shoulder complex and the humeral pointing kinematics. *IEEE Trans Robot Autom* 19(3):499–506
- Nakamura Y, Hanafusa H, Yoshikawa T (1987) Task-priority based redundancy control of robot manipulators. *Int J Robot Res* 6(2):3–15
- Kuffner JJ, LaValle SM (2000) RRT-connect: an efficient approach to single-query path planning, April, pp 995–1001
- Lozano-Pérez T (1983) Spatial planning: a configuration space approach. *IEEE Trans Comput* 100(2):108–120
- Lozano-Pérez T, Wesley MA (1979) An algorithm for planning collision-free paths among polyhedral obstacles. *Commun ACM* 22:560–570
- Burns B (2005) Toward optimal configuration space sampling. In: Proceedings of robotics: science and systems, pp 1–6
- Diankov R, Kuffner J (2007) Randomized statistical path planning. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems. IEEE, pp 1–6. doi:[10.1109/IROS.2007.4399557](https://doi.org/10.1109/IROS.2007.4399557). <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4399557>
- Toussaint M (2009) Robot trajectory optimization using approximate inference. In: Proceedings of the 26th annual international conference on machine learning—ICML’09. ACM Press, New York, pp 1049–1056. doi:[10.1145/1553374.1553508](https://doi.org/10.1145/1553374.1553508). <http://portal.acm.org/citation.cfm?doid=1553374.1553508>
- Brock O, Khatib O, Viji S (2002) Task-consistent obstacle avoidance and motion behavior for mobile manipulation. In: Proceedings of IEEE international conference on robotics and automation, ICRA’02, vol 1, pp 388–393. doi:[10.1109/ROBOT.2002.1013391](https://doi.org/10.1109/ROBOT.2002.1013391)
- Colbaugh R, Seraji H, Glass K (1989) Obstacle avoidance for redundant robots using configuration control. *J Robot Syst* 6(6):721–744
- Glass K, Colbaugh R, Lim D, Seraji H (1995) Real-time collision avoidance for redundant manipulators. *IEEE Trans Robot Autom* 11(3):448–457
- Guo Z, Hsia T (1993) Joint trajectory generation for redundant robots in an environment with obstacles. *J Robot Syst* 10(2):199–215
- Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. *Int J Robot Res* 5(1):90–98. doi:[10.1177/027836498600500106](https://doi.org/10.1177/027836498600500106)
- Kim JO, Khosla PK (1992) Real-time obstacle avoidance using harmonic potential functions. *IEEE Trans Robot Autom* 8(3):338–349
- Maciejewski AA, Klein CA (1985) Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *Int J Robot Res* 4(3):109–117. doi:[10.1177/027836498500400308](https://doi.org/10.1177/027836498500400308)

18. McLean A, Cameron S (1996) The virtual springs method: path planning and collision avoidance for redundant manipulators. *Int J Robot Res* 15(4):300–319
19. Seraji H, Bon B (1999) Real-time collision avoidance for position-controlled manipulators. *IEEE Trans Robot Autom* 15(4):670–677
20. Volpe R, Khosla P (1993) A theoretical and experimental investigation of impact control for manipulators. *Int J Robot Res* 12(4):351–365
21. Feder HJS, Slotine JJE (1997) Real-time path planning using harmonic potentials in dynamic environments. In: Proceedings of IEEE international conference on robotics and automation, April, pp 874–881
22. Iossifidis I, Sch G (2006) Dynamical systems approach for the autonomous avoidance of obstacles and joint-limits for an redundant robot arm. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems, pp 580–585
23. Khansari-Zadeh SM, Billard A (2012) A dynamical system approach to realtime obstacle avoidance. *Auton Robot* 32(4):433–454. doi:[10.1007/s10514-012-9287-y](https://doi.org/10.1007/s10514-012-9287-y). <http://link.springer.com/10.1007/s10514-012-9287-y>
24. Park DHPDH, Hoffmann H, Pastor P, Schaal S (2008) Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In: Proceedings of 8th IEEE-RAS international conference on humanoid robots, humanoids 2008. IEEE, vol 121, pp 91–98. doi:[10.1109/ICHR.2008.4755937](https://doi.org/10.1109/ICHR.2008.4755937). <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4755937>
25. Sprunk C, Lau B, Pfaff P (2011) Online generation of kinodynamic trajectories for non-circular omnidirectional robots. In: Proceedings of IEEE international conference on robotics and automation, pp 72–77
26. Newman WS (1989) Automatic obstacle avoidance at high speeds via reflex control. In: Proceedings of IEEE international conference on robotics and automation. IEEE, pp 1104–1109
27. Xie F, Qu Z, Garfinkel A (1998) Dynamics of reentry around a circular obstacle in cardiac tissue. *Phys Rev E* 58(5):6355
28. O’Neil K (2002) Divergence of linear acceleration-based redundancy resolution schemes. *IEEE Trans Robot Autom* 18(4):625–631. doi:[10.1109/TRA.2002.801046](https://doi.org/10.1109/TRA.2002.801046)
29. Khatib O (1987) A unified approach for motion and force control of robot manipulators: the operational space formulation. *IEEE J Robot Autom* 3(1):43–53. doi:[10.1109/JRA.1987.1087068](https://doi.org/10.1109/JRA.1987.1087068)
30. Mansard N, Khatib O, Kheddar A (2009) A unified approach to integrate unilateral constraints in the stack of tasks. *IEEE Trans Robot* 25(3):670–685. doi:[10.1109/TRO.2009.2020345](https://doi.org/10.1109/TRO.2009.2020345)
31. Sentis L, Park J, Khatib O (2010) Compliant control of multicontact and center-of-mass behaviors in humanoid robots. *IEEE Trans Robot* 26(3):483–501. doi:[10.1109/TRO.2010.2043757](https://doi.org/10.1109/TRO.2010.2043757)
32. Stasse O, Escande A, Mansard N, Miossec S, Evrard P, Kheddar A (2008) Real-time (self)-collision avoidance task on a HRP-2 humanoid robot. In: IEEE international conference on robotics and automation, pp 3200–3205
33. Žlajpah L, Petrič T (2012) Serial and parallel robot manipulators—kinematics, dynamics, control and optimization, chap obstacle avoidance for redundant manipulators as control problem. InTech, pp 203–230
34. Sciavicco L, Siciliano B (2005) Modelling and control of robot manipulators, 2nd edn., Advanced textbooks in control and signal processing Springer, London
35. Petrič T, Žlajpah L (2013) Smooth continuous transition between tasks on a kinematic control level: obstacle avoidance as a control problem. *Robot Auton Syst* 61(9):948–959
36. Petrič T, Gams A, Babič J, Žlajpah L (2013) Reflexive stability control framework for humanoid robots. *Auton Robot* 34(4):347–361. doi:[10.1007/s10514-013-9329-0](https://doi.org/10.1007/s10514-013-9329-0)
37. Petrič T, Žlajpah L (2011) Smooth transition between tasks on a kinematic control level: application to self collision avoidance for two kuka lwr robots. In: 2011 IEEE international conference on robotics and biomimetics, pp 162–167
38. Sugiura H, Gienger M, Janssen H, Goericke C (2007) Real-time collision avoidance with whole body motion control for humanoid robots. In: IEEE/RSJ international conference on intelligent robots and systems, IROS 2007, pp 2053–2058. doi:[10.1109/IROS.2007.4399062](https://doi.org/10.1109/IROS.2007.4399062)

39. Žlajpah L, Nemeć B (2002) Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators. In: IEEE/RSJ international conference on intelligent robots and systems, vol 2, pp 1898–1903. doi:[10.1109/IRDS.2002.1044033](https://doi.org/10.1109/IRDS.2002.1044033)
40. Arun KS, Huang TS, Blostein SD (1987) Least-squares fitting of two 3-d point sets. *IEEE Trans Pattern Anal Mach Intell* 5:698–700
41. Khatib O, Brock O, Chang KS, Ruspini D, Sentis L, Viji S (2004) Human-centered robotics and interactive haptic simulation. *Int J Robot Res* 23(2):167–178
42. Konietzschke R, Hirzinger G (2009) Inverse kinematics with closed form solutions for highly redundant robotic systems. In: Proceedings of IEEE international conference on robotics and automation. IEEE, pp 2945–2950
43. Santis AD, Siciliano B (2008) Inverse kinematics of robot manipulators with multiple moving control points. In: Lenarčič J, Wenger P (eds) *Advances in robot kinematics: analysis and design*. Springer, New York, pp 429–438
44. Antonelli G (2009) Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *IEEE Trans Robot* 25(5):985–994. doi:[10.1109/TRO.2009.2017135](https://doi.org/10.1109/TRO.2009.2017135)
45. Chiaverini S, Oriolo G, Walker ID (2008) Kinematically redundant manipulators. In: Siciliano B, Khatib O (eds) *Springer handbook of robotics*, chap 11. Springer, Berlin, pp 245–268
46. Park J, Choi YJ, Chung WK, Youm Y (2001) Multiple tasks kinematics using weighted pseudo-inverse for kinematically redundant manipulators. In: Proceedings 2001 ICRA. IEEE international conference on robotics and automation (Cat. No.01CH37164), vol 4. IEEE, pp 4041–4047
47. Baerlocher P, Boulic R (1998) Task-priority formulations for the kinematic control of highly redundant articulated structures. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems, vol 1, October, pp 323–329
48. Chiaverini S, Siciliano B, Egeland O (1994) Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator. *IEEE Trans Control Syst Technol* 2(2):123–134
49. Sciaffico L, Siciliano B (1986) Solving the inverse kinematic problem for robotic manipulators. In: Morecki A, Bianchi G, Kdzior K (eds) *Proceedings of the 6th CISM-IFToMM symposium on theory and practice of robots and manipulators*. Springer, Krakow, pp 107–114
50. Egeland O, Sagli J, Spangelo I, Chiaverini S (1991) A damped least-squares solution to redundancy resolution. In: Proceedings 1991 IEEE international conference on robotics and automation. IEEE Computer Society Press, pp 945–950
51. Buss SR, Kim JS (2004) Selectively damped least squares for inverse kinematics. *J Graph Tools* 10:37–49
52. Deo A, Walker I (1992) Robot subtask performance with singularity robustness using optimal damped least-squares. In: Proceedings 1992 IEEE international conference on robotics and automation. IEEE Computer Society Press, pp 434–441
53. Maciejewski A, Klein C (1988) Numerical filtering for the operation of robotic manipulators through kinematically singular configurations. *J Robot Syst* 5(6):527–552
54. Nakamura Y, Hanafusa H (1986) Inverse kinematics solutions with singularity robustness for robot manipulator control. *Trans ASME J Dyn Syst Meas Control* 108(3):163–171
55. Žlajpah L (2013) Multi-task control for redundant robots using prioritized damped least-squares inverse kinematics. In: 22nd international workshop on robotics in Alpe-Adria-Danube region, Portorož, Slovenia, 11–13 September 2013
56. Likar N, Nemeć B, Žlajpah L (2012) Virtual mechanism approach for dual-arm manipulation. *Robotica* 1:1–16
57. Ijspeert A, Nakanishi J, Pastor P, Hoffmann H, Schaal S (2013) Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Comput* 25(2):328–373
58. Ijspeert A, Nakanishi J, Schaal S (2002) Movement imitation with nonlinear dynamical systems in humanoid robots. In: IEEE international conference on robotics and automation (ICRA), vol 2. Washington, DC, pp 1398–1403

59. Gams A, Ijspeert AJ, Schaal S, Lenarčič J (2009) On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Auton Robot* 27(1):3–23
60. Ijspeert AJ, Nakanishi J, Schaal S (2002) Learning rhythmic movements by demonstration using nonlinear oscillators. In: Proceedings of IEEE/RSJ international conference intelligent robots and systems. Lausanne, pp 958–963
61. Petrič T, Gams A, Ijspeert AJ, Žlajpah L (2011) On-line frequency adaptation and movement imitation for rhythmic robotic tasks. *Int J Robot Res* 30(14):1775–1788
62. Ude A, Gams A, Asfour T, Morimoto J (2010) Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Trans Robot* 26(5):800–815

# Path Planning Kinematics Simulation of CNC Machine Tools Based on Parallel Manipulators

Luc Rolland

**Abstract** Since the very successful application of parallel robots in material handling, many projects attempted to implement the Gough platforms as milling machine manipulators with limited success mainly achieving roughing. The displacement of the milling tool should meet surface finish requirements while increasing tool feedrate in order to improve productivity. This work introduces geometric formalization of surface finish which is more realistic than classic error calculations. This research work also proposes an off-line simulation tool analysing the milling task feasibility using a robot constituted by a general hexapod parallel manipulator, controlled by a typical CNC controller implementing classic position based algorithms where joint space polynomial interpolation is utilized. High and very high speed milling simulation results show the implementation of linear and third order interpolation between the actuator set-points calculated from the CAD/CAM computed end-effector or tool set-points. Linear interpolation is not sufficient for high speed milling and then third order interpolation reach the required surface finish at feasible CNC sampling rates.

**Keywords** Parallel manipulator · CNN · Kinematics simulation

Since the very successful application of parallel robots in material handling, many projects attempted to implement the Gough platforms as milling machine manipulators with limited success mainly achieving roughing.

The displacement of the milling tool should meet surface finish requirements. Users also wish to increase tool feedrate in order to improve productivity thereby reaching high speed milling levels. Even a constant high speed feedrate brings important challenges since they mean higher actuator accelerations even on straight lines. This work introduces geometric formalization of surface finish which is more realistic than classic error calculations.

---

L. Rolland (✉)

High Performance Robotics Laboratory, Memorial University of Newfoundland,  
St-John's Campus, St-John's, NL, Canada  
e-mail: lrolland@mun.ca

This research work proposes an off-line simulation tool analysing the milling task feasibility using a robot constituted by a general hexapod parallel manipulator, namely the Gough Platform, often referred as the Stewart Platform. Moreover, in order to meet the machine-tool standards, the parallel robot will be controlled by a typical CNC controller implementing classic position based algorithms adapted to the parallel robots with any kind of actuator polynomial interpolation. Control sampling rates are studied and their impact evaluated.

High and very high speed milling simulation results show the implementation of linear and third order interpolation between the actuator set-points calculated from the CAD/CAM computed end-effector or tool set-points. The results show that linear interpolation are not sufficient for high speed milling and then third order interpolation reach the required surface finish at fast and feasible CNC sampling rates.

## 1 Introduction

After the confirmed success of parallel robots as flight simulators followed by their more recent breakthroughs in material handling, they are actually implemented as machine-tools. Several commercialization attempts were made over the years. With the promise of increased productivity, we aim to achieve the two following goals:

1. To reach higher feedrates while keeping excellent surface finish quality
2. To obtain faster accelerations during path transfers between task trajectories.

The main advantages of these robotic manipulators compared to serial ones are simpler construction, more rigid structures, non-cumulative kinematics chain deflections, greater throughputs from higher accelerations and less energy consumption from smaller actuators. On the other hand, these manipulators feature drawbacks such as limited workspace and complex non-linear kinematics.

In material handling applications the ratio between actuator displacement travel and accuracy is around 1,000 over 1 mm, whereas in milling applications the ratio becomes 1,000 over 1 micron, meaning it 1,000 times larger.

Due to the highly non linear nature of parallel robots, their implementation still poses serious challenges.

Initial path planning investigations for parallel robots were trying to determine if any task would include their entire paths inside the robot workspace, where the notion of trajectory quality has been formulated in terms of distances from actuator limits [39]. Kinematics chain collision was added to the analysis [11]. Path planning involved singularity investigation to avoid instantaneous self-motion [46]. Singularities were extensively studied [3, 21, 23]. The problem evolved into multi-objective optimization finding the optimum path according to a certain number of criteria [9, 10]. Chablat and Wenger [41] introduced collision avoidance to singularity analysis to answer the question of moveability in the presence of obstacles. Planning time-minimal trajectories were introduced by [1, 27]. In [29], the authors minimize

electrical energy, kinetic energy, robot motion time separating two sampling periods, and maximize a measure of manipulability allowing singularity avoidance.

More specifically, implementing the Gough platform as a milling machine, often referred as the Stewart platform, path planning was studied where the contour error was used as a performance criteria to determine the effect of PID controls applied on each actuator [38]. The redundant sixth degree-of-freedom was utilized for optimization according to various criterias [44]. Path planning schemes also targeted the axial force minimization [65], where maximum constant cutting force along the contour were maximized [49]. Then, added objectives included stiffness maximization [52].

This research work addresses the feasibility of a successful machining task in terms of surface finish quality, the manipulator type, the sensor accuracy, the control strategy (position or velocity control), typical feedback servo loops, signal digitization, time digitization, inter-point polynomial interpolation, the related computer numerical control algorithms and even signal synchronization. This general framework allows to study any specific robot controlled by any typical Computer Numerical Controls (CNC). A novel formal approach to evaluate surface finish is proposed including a milling task description. A CNC module simulation block is introduced where the effect of time and signal digitization can be studied allowing to adjust sampling rates. The task is analyzed from a pure kinematics point of view, allowing to determine the best achievable result and eventually increase machining parameters such as feedrates.

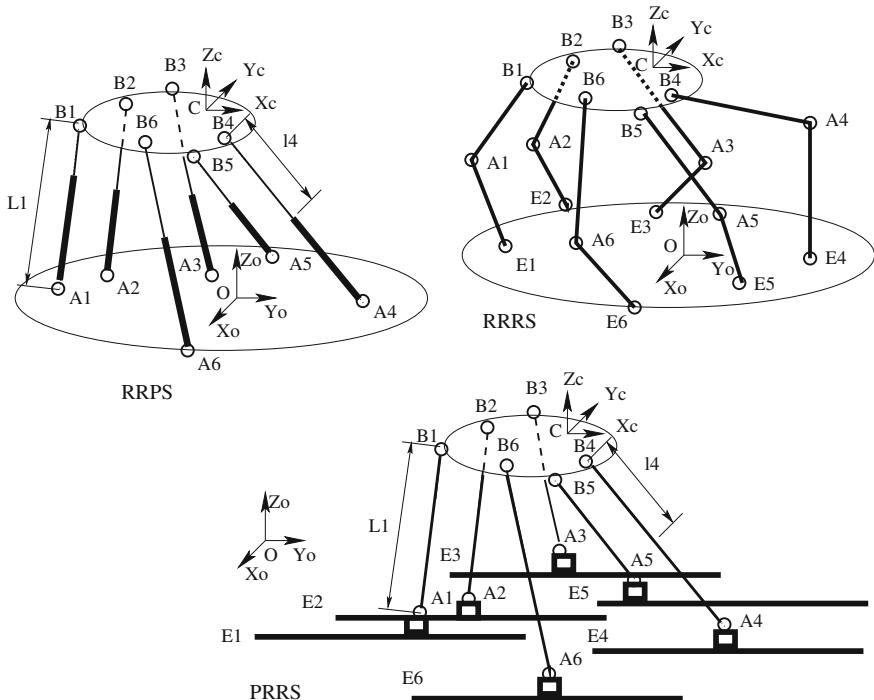
In the next section, the high speed milling problem and context are explained. It includes the theoretical background on parallel manipulator kinematics and CNC control. The third section reviews the machining Process. The fourth section covers the geometric formalization of surface finish. The fifth section presents the path planning simulation results with position control.

## 2 General Issues with Parallel Kinematic Machines

### 2.1 Problem Statement

To obtain five axis CNC machining at high speed feedrate levels, the Gough platform or hexapod has to be envisaged with six kinematics chains between the fixed base and the mobile platform where the tool is located, Fig. 1. Then, three possible cases can be derived. The 6UPS/6SPU configuration contains each kinematics chain with a free prismatic actuator (P) between one Universal joint (U) and one ball joint (S); the 6RUS/6RSU includes kinematics chains constituted by a revolute actuator (R) operating a crank moving a bar including one Universal joint (U) and one ball joint (S); and finally the 6PUS/6PSU replacing the crank by a tracked prismatic actuator (P).

In reality, any robotic system is never constructed identical to the ideally designed one. A significant difference can be often observed between the theoretical and



**Fig. 1** Typical 6-6 parallel robots: the 6UPS/6SPU, 6RUS/6RSU and 6PUS/6PSU

practical configurations translating into errors on the passive joint positions of the mobile platform and the fixed base. These configuration errors will without doubt have a significant impact on milling precision. These discrepancies will usually grow following various milling operations where unpredictable wear is occurring in the joints. These will also appear following maintenance where the manipulator was reassembled if not followed by an adequate calibration procedure [20].

In the literature, we can identify several procedures and software analysing the characteristics and performance of robotic manipulators [76]. These studies seek to evaluate the extremes of a certain number of criterions. More specifically, in parallel robotics, let's highlight some interesting packages proposing some level of verifications:

1. Localisation of robot trajectories inside the workspace [39, 43].
2. Singularities over nominal trajectories inside the workspace [21, 39, 46].
3. Power and torque of motors [61].
4. Positioning errors [38, 50].

These analyses concern the entire workspace where performance can be affected by large variations. In many scenarios, it may be possible to achieve the task over a large portion of the workspace and then the task quality may not reach the desirable

levels in certains specific areas of the workspace. The performance analysis shifted away from workspace studies towards the task trajectories themselves studying the following factors:

1. the joint travel in terms of the actuators and passive joints [39]
2. the kinematics chain and platform collisions [11, 39], extrapolated from serial robotics work [71]
3. maximum velocity [35]
4. dynamic rigidity [62]
5. servo modeling [38, 62]
6. robot control [38]
7. tool deformation in milling tasks [25]
8. sixth rotation angle optimization for milling tools [20, 44].

These research works do not include all the important criterias. The displacement of the milling tool should meet surface finish requirements and tool feedrate. The second criterion will be increased in order to improve productivity. Even a constant feedrate brings important challenges on trajectories such as arcs since they mean higher accelerations.

The goal of this work is to propose tools analysing the milling task feasibility using a robot constituted by a 6-6 hexapod parallel manipulator, namely the Gough Platform, often referred as the Stewart Platform. Moreover, in order to meet the standards of the machine-tool domain, the parallel robot will be controlled by a typical CNC controller implementing classic algorithms adapted to parallel robots.

The factors influencing robot trajectory following are the sub-space of task execution, tool feedrate, position sensor accuracy and the choice of control algorithms. The milling task is in turn described by several robot trajectories. For high speed milling, surface finish is required to obtain asperities not exceeding  $10\text{--}20 \mu$  over the entire trajectories constituting a milling task. To qualify as high speed milling (HSM), the feedrate should reach  $20 \text{ m/min}$  and the target is even  $60 \text{ m/min}$ , classified as ultra high speed milling (UHSM).

The simulation system will require solving the kinematics problems several times. To alleviate many problems related to usual numerical methods, an exact and certified method was derived and will be applied to perform end-effector position and orientation calculations [59, 60]. This method implements ideal based techniques utilizing Groebner bases and rational univariate representations (RUR) insuring that the produced equivalent system is exactly corresponding to the original system. The RUR system includes one univariate equation from which the real roots are calculated and proven in one-to-one objective correspondence with the original kinematics problem. Then, proven root isolation techniques will provide for all the exact real roots. The system applies the modular black-box approach where any user can replace the selected kinematics solver by any other, at the condition that it provides for sufficient accuracy to study milling tasks.

In practice, during design, construction, start-up or after robot maintenance, these simulation tools will allow to select the complete control approach including sensors and the path planning algorithms; The operator will be able to study the control

scheme, the path following algorithms, the joint interpolation functions, the axis servo controls, the response-time of the various control levels, the effect of time discretization, the effect of digital conversions and parameter fine-tuning. The proposed tools will allow to determine milling task feasibility.

## 2.2 Kinematics of the General 6-6 Parallel Manipulator

Any manipulator is characterized by its mechanical configuration parameters and the posture variables. The configuration parameters are thus  $\mathbf{OA}|_{R_f}$ , the base attachment point coordinates in  $R_f$  (the base reference frame, located at  $\mathbf{O}$ ), and  $\mathbf{CB}|_{R_m}$ , the mobile platform attachment point coordinates in  $R_m$  (the mobile platform reference frame, located at  $\mathbf{C}$ ). The kinematics model variables are the joint coordinates and end-effector generalized coordinates. The joint variables are described as  $l_i$ , the prismatic joint or linear actuator positions. The generalized coordinates are expressed as  $\vec{X}$  comprising the end-effector position and orientation.

The kinematics model is an implicit relation between the configuration parameters and the posture variables,  $F(\vec{X}, \vec{l}, \mathbf{OA}|_{R_f}, \mathbf{CB}|_{R_m}) = 0$  where  $\vec{l} = \{l_1, \dots, l_6\}$ . For the sake of clarity and simplicity,  $\mathbf{OA}|_{R_f}$  will be replaced by  $\mathbf{OA}_O$  and  $\mathbf{CB}|_{R_m}$  by  $\mathbf{CB}_O$ .

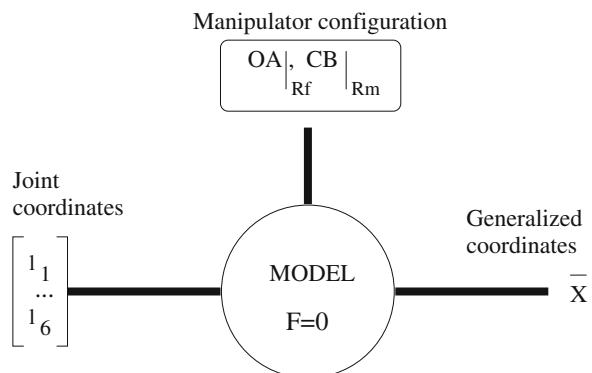
This simulator shall only require successive passages from the joint space to the task space and vice versa, Fig. 2. The Inverse Kinematics Problem (IKP) is defined as:

**Definition 1** Given the generalized coordinates of the manipulator end-effector, find the joint positions.

Accordingly, the Forward Kinematics Problem (FKP) is defined as:

**Definition 2** Given the joint positions, find the generalized coordinates of the manipulator end-effector.

**Fig. 2** Kinematics model



Usually the IKP is required to model the FKP. To solve the FKP, an exact method based on Groebner bases and rational univariate representations shall be applied [59, 60].

The forward kinematics problem (**FKP**), Fig. 2, has been identified as a difficult problem [57]. Usually the *inverse kinematics problem* is required to model the **FKP** and is defined as [56]: *given the generalized coordinates of the manipulator end-effector, find the joint positions.*

Accordingly, the *forward kinematics problem* is defined as [56]: *given the joint positions, find the generalized coordinates of the manipulator end-effector.*

The kinematics problem can be described that, contrarily to serial manipulators, the inverse kinematics problem yields a closed-form explicit solution and the forward kinematics involves the resolution of at least six non-linear equations. These kinematics models play an increasingly important role when robotic manipulator accuracy is decreased to the micron level.

### 2.3 Vectorial Formulation of the Implicit Kinematics Model

Containing as many equations as variables, vectorial formulation constructs an equation system for each kinematics chain [54], as a closed vector cycle between the  $A_i$  and  $B_i$  kinematics chain attachment points, the fixed base reference frame  $O$  and the mobile platform reference frame  $C$ . For each kinematics chain, an implicit function  $\overrightarrow{A_i B_i} = U_1(X)$  can be written between joint positions  $A_i$  and  $B_i$ . Each vector  $\overrightarrow{A_i B_i}$  is expressed knowing the joint coordinates  $\bar{L}$  and  $X$  giving function  $U_2(X, \bar{L})$ . The following equality has to be solved:  $U_1(X) = U_2(X, \bar{L})$ . The distance between  $A_i$  and  $B_i$  is set to  $l_i$ . Thus, the end-effector position  $X$  or  $C$  can be derived by one platform displacement  $\overrightarrow{OC}$  and then one platform general rotation expressed by the rotation matrix  $\mathcal{R}$ . For each distinct platform point  $\overrightarrow{B_i O}$  with  $i = 1, \dots, 6$ , see Fig. 3, the position can be calculated in terms of the base reference frame [53]:

$$\overrightarrow{OB_i O} = \overrightarrow{OC} + \mathcal{R}\overrightarrow{CB_i} \quad (1)$$

The vectorial formulation evolves as a displacement based equation system using the following relation:

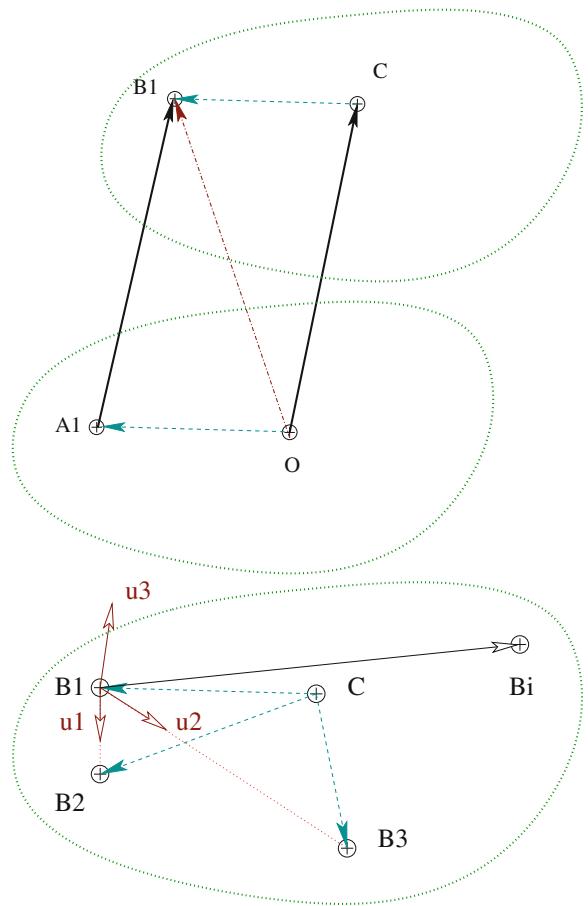
$$\overrightarrow{A_i B_i} = \overrightarrow{OC} + \mathcal{R}\overrightarrow{CB_i} - \overrightarrow{OA_i} \quad (2)$$

These six equations cannot be applied as such. Hence, each kinematics chain can be expressed using the distance norm constraint [53]:

$$l_i^2 = ||\overrightarrow{A_i B_i}||^2 \quad (3)$$

The rotation matrix  $\mathcal{R}$  can be written utilizing various orientation models with their specific rotation variable sets such as navigation angles (yaw, pitch and roll), Euler angles, quaternions or even taking the nine rotation matrix components as

**Fig. 3** Kinematics chain and mobile platform vectors



variables [60]. Implementing the Eq. (3) directly, various displacement based equation models can be derived depending on the selected orientation variables [60].

Another excellent approach is called the position based modeling and consists in considering any rigid object to be positioned into three dimensional space by three distinct points, Fig. 3. Any rigid body three points are actually characterized by three distinct distance constraints and a pointing axis which remain constant. This principle was then applied to the forward kinematics model of parallel manipulators by Lazard [55]. It is easy to choose three distinct points which are not collinear on most mobile platforms. These three points are usually selected to coincide with three joint centers connecting the mobile platform to the kinematics chains allowing to utilize the vectorial model, Fig. 3 and to rewrite of  $\overrightarrow{A_i B_i}$ , Eq. (2) as it is explained in details in [60].

Two reasons justify the choice of the position based model. Every variable yield the same units and their ranges are equivalent leading to the same weight in the

equation system. The rotation impact is included into the point parameters and made equivalent to the translation impact.

The coordinates of the three distinct joint center points become the nine variables from which constraints equation can be written. The three platform distinct points are usually selected as the three first joint centers, namely  $B_1$ ,  $B_2$  and  $B_3$ . Each coordinate of the selected joint centers becomes a variable. The nine end-effector variables are set to:  $\overrightarrow{OB_{i|O}} = [x_i, y_i, z_i]$  for  $i = 1 \dots 3$ . To simplify computations, we choose one non-Cartesian reference frame  $R_{b_1}$  to be located at  $B_1$  joint center. Then, we define  $u_1$ ,  $u_2$  and  $u_3$  as  $R_{b_1}$  reference frame axes which are calculated by:

$$u_1 = \frac{\overrightarrow{B_1 B_2}}{||\overrightarrow{B_1 B_2}||}, \quad u_2 = \frac{\overrightarrow{B_1 B_3}}{||\overrightarrow{B_1 B_3}||}, \quad u_3 = u_1 \wedge u_2 \quad (4)$$

This new reference frame  $R_{b_1}$  is applied instead of  $R_m$  as the mobile platform Cartesian reference frame and has its origin located at  $B_1$  and the reference frame axes  $u_1$  and  $u_2$  point towards  $B_2$  and  $B_3$  respectively. The third reference frame  $u_3$  points perpendicular to the plane determined by  $B_1$ ,  $B_2$  and  $B_3$ . It becomes the mobile platform pointing axis. This transformation is achieved to produce a simpler equation system.

Knowing that the mobile platform is supposed infinitely rigid, any platform point  $M$  can be expressed in the reference frame  $R_{b_1}$  by calculating the following linear composition:

$$\overrightarrow{B_1 M} = a_M u_1 + b_M u_2 + c_M u_3 \quad (5)$$

where  $a_M$ ,  $b_M$ ,  $c_M$  are constants in terms of these three points. Hence, in the case of the **IKP**, the constants are noted  $a_{B_i}$ ,  $b_{B_i}$ ,  $c_{B_i}$ ,  $i = i \dots 6$  and can explicitly be deduced from the mobile platform fixed distances  $\mathbf{CB}_{|C}$  by solving the following linear system of equations:

$$\overrightarrow{B_1 B_i}_{|R_{b_1}} = a_{B_i} u_1 + b_{B_i} u_2 + c_{B_i} u_3, \quad i = 1 \dots 6. \quad (6)$$

where  $\overrightarrow{B_1 B_i}_{|R_{b_1}} = \overrightarrow{B_1 B_i}_{|C}$ .

Note that the mobile platform fixed distances  $\mathbf{CB}_{|C}$  are given by the configuration which is obtained from the design values or deduced from a calibration procedure after the Gough platform manipulator construction. The configuration file is providing the position of all six joints of the mobile platform relative to the mobile platform reference frame and this ensure that the points belong to the same rigid body which is the mobile platform.

Equation 7 requires that we calculate the configuration distances with:

$$\overrightarrow{B_1 B_i}_{|C} = \overrightarrow{CB_i} - \overrightarrow{CB_1}, \quad i = 1 \dots 6. \quad (7)$$

Hence, the remaining three mobile platform joint centers  $B_4$ ,  $B_5$  and  $B_6$  are expressed in terms of the nine end-effector variables.

Using the relations Eq. (6), the distance constraint equations  $l_i^2 = ||\overrightarrow{A_i B_i}_O||^2$ ,  $i = 1 \dots 6$  can be expressed. Thus, for  $i = 1 \dots 6$ , the **IKP** is obtained by isolating the  $l_i$  actuator variables in the six following equations:

$$l_i^2 = (x_i - OA_{ix})^2 + (y_i - OA_{iy})^2 + (z_i - OA_{iz})^2, \quad i = 1 \dots 3 \quad (8)$$

$$l_i^2 = ||\overrightarrow{B_1 B_i}_{R_{b_1}} - \overrightarrow{OA_i}_O||^2, \quad i = 4 \dots 6 \quad (9)$$

## 2.4 The Inverse Kinematics Problem

The Eqs. (3) or (9) are actually the two general forms of the explicit IKP.

## 2.5 The Forward Kinematics Problem

For the general Gough platform parallel manipulator, it is actually not possible to express the FKP directly or explicitly [45]. We have to revert to the **IKP** expression which gives an algebraic system comprising six equations in terms of three point variables:  $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3$ , Eq. (9). This system contains algebraic (polynomial) functions which can be handled by the numerical solvers implemented in all genetic algorithms.

The usual method advocated for writing the FKP equation system starts by rewriting the IKP as functions. This produces an algebraic system of three leg equations and three functions in terms of the nine variables:  $x_i, y_i, z_i, i = 1, 2, 3$ .

$$F_i = (x_i - OA_{ix})^2 + (y_i - OA_{iy})^2 + (z_i - OA_{iz})^2 - l_i^2, \quad i = 1 \dots 3 \quad (10)$$

$$F_i = ||\overrightarrow{B_1 B_i}_{R_{b_1}} - \overrightarrow{OA_i}_O||^2 - l_i^2, \quad i = 4 \dots 6 \quad (11)$$

When solving the FKP with numeric or algebraic methods, it is necessary to provide a zero-dimensional system, meaning an equation system which contains as many equations as their are variables [59, 60]. In this case, this means that to the six equations provided by the IKP, three more shall be selected to close the system.

Moreover, the actual FKP is derived directly from the IKP model, Eq. (11), and it does not provide for any information to constrain the position of the mobile platform joint positions which are necessary to describe the FKP.

Hence, to complete the algebraic system and to constrain the mobile platform joint positions, three constraints are derived from the following three functions. Two functions can be written using two characteristic platform distances, expressed as norms between the  $B_1, B_2$  distinct points and the  $B_1, B_3$  ones. The computations will select the variables which are only at the right distance from the  $B_1$  reference joint point. These constraint equations require one last equation. The points are

known relative to each other in terms of distance but the mobile platform alignment is left undetermined. To alleviate this problem, the third constraint equation will determine where the mobile platform is pointing. The pointing vector is selected as the one perpendicular to the three points  $B_i$ ,  $i = 1, 2, 3$  by calculating the vectorial multiplication of the two vectors separating  $B_2$  and  $B_3$  from  $B_1$ :

$$F_7 = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 - \|\overrightarrow{B_2 B_1}_{|R_{B_1}}\|^2 \quad (12)$$

$$F_8 = (x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2 - \|\overrightarrow{B_3 B_1}_{|R_{B_1}}\|^2 \quad (13)$$

$$\begin{aligned} F_9 = & (x_3 - x_1)(x_2 - x_1) + (y_3 - y_1)(y_2 - y_1) + (z_3 - z_1)(z_2 - z_1) \\ & - \|\overrightarrow{B_3 B_1}_{|R_{B_1}}\| \wedge \|\overrightarrow{B_2 B_1}_{|R_{B_1}}\| \end{aligned} \quad (14)$$

The choice of  $F_9$ , the last function, provided an important mobile platform constraint related to the pointing axis. For  $F_9$ , it would be possible to write a function related to the distance between  $B_2$  and  $B_3$  but our experience shows us that it does lead to better results than the platform pointing function.

The result constitutes then an algebraic system with nine equations in the former nine unknowns.

## 2.6 Machine Tool Control

In a high speed milling machine, a typical Gough platform being a general **6-6** or hexapod robot is constituted by several parts driven by a controller connected to a remotely located CAD-CAM computer. As it is explained in [45], one CNC machine-tool is essentially considered identical to a robot achieving the predetermined continuous path following encountered in machine tool processes.

*The CNC is defined as the control system capable to manage the machine-tool and its control in order to follow a program achieving a milling task [45].*

Practically, the CNC handles a written program in a standard format constituted by G codes from the ISO standard [36, 37]. Note that the machine-tool industry considers this format mandatory for machine-tool controls. Any simulation package shall consider that CNC systems handle these codes and simulate their operations.

In typical CNC, the control unit is divided into three control stages or levels: the off-line CAD-CAM level providing the task set-points describing the nominal paths, the on-line nominal path following as the upper controller level and the motor servoing as the lower controller level, usually driving directly the actuators by implementing one **PID** feedback loop for each axis. Each stage operates in discrete time according its own cycle time or sampling rates:

- $T_c$ : The task trajectory set-point file sampling rate produced by the CAM program.
- $T_p$ : The path following cycle time corresponding to the time required to calculate the joint servo trajectory set-points.

- $T_s$ : The motor servo cycle time corresponding to the time dedicated to PID loop computation.
- $T_a$ : The motor amplifier sampling rate which gives the time at which their output is being refreshed.

The simulation module will allow to test and verify the three first cycle times. The amplifier sampling rates will not be included in the simulation work. The task follows one or several nominal functions from which discretization produces the task path file containing a large number of points being dependent on the sampling rates. The number of points will have an impact on surface finish and impact CNC's ability to follow the nominal path.

The machine-tool operates in a spatial continuous domain which is completely described by 6 dimensions (3 translations and 3 rotations),  $\lambda = 6$ , with parameters  $\in \Re$ . To execute a milling task, the path following algorithm may require from three to five axes control. The sixth axis corresponds to the tool spindle rotation axis and therefore does not participate to the trajectory pursuit. The CNC should then receive five analog inputs or encoder inputs for actuator axis positions and drive five analog or direct pulse-width-modulation outputs for actuator positioning. The simulation will not include the tool spindle axis angular speed control.

The CNC can either implement one of the two control types: position and speed control [15]:

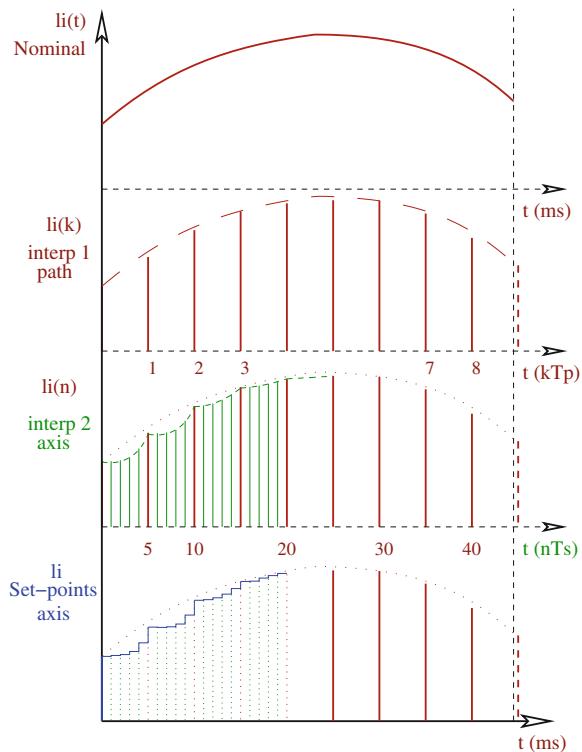
- Position control is preferred when you can calculate the IKP. Joint position control follows the trajectory profile at the axis level from interpolated point to interpolated point and does not control the velocities between these points leading to a discrepancy between the exact nominal trajectory and the achieved trajectory at the tool level. If the range of motion is important then the robot reaches its destination with larger inaccuracies. The traditional solution is to slowdown robots.
- Speed control is based on small displacements and implements the computation of the inverse Jacobian matrix. You will need to calculate the FKP.

## 2.7 Task Space Conversion to Joint Space

In principle, implemented in the off-line CAM, the trajectory planning algorithm calculates one inverse kinematics problem from the Cartesian-space set-point trajectory functions to determine the six actuator-space functions which are then called the joint set-point trajectories. The real continuous signals are computed from these functions. Then, the continuous signals are sampled according to the first level cycle time  $T_p$  corresponding to the time required to calculate these points and the signal magnitude discretized into a certain number of bits (Fig. 4).

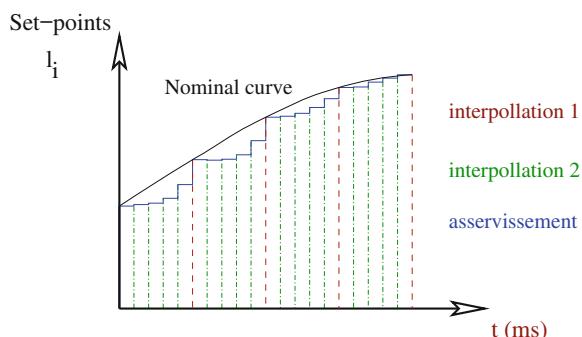
When planning and following any task path, the upper level controller calculates, in advance and in real time at each  $t = k T_p$  for  $k = 1 \dots n_p$  where  $n_p$  is the number of points provided by CAD/CAM, all interpolated points between joint set-points that will then serve as set-points to the six lower level servo controllers driving the

**Fig. 4** Example of signal and time digitization of nominal actuator function



actuators. It is interpolating these reference values using a polynomial interpolation function or blended polynomial function sets (Fig. 5). Since the majority of control algorithms calculate the instructions in the joint space and there are no sensors for performing a return position on the end-effector where the milling tool is located in task space, then the controller must perform the forward kinematics problems (FKP) calculations to return the tool Cartesian position and orientation.

**Fig. 5** Details of actuator signal digitization



## 3 CNC Handling of the Machining Process

### 3.1 Introduction on Milling

Tournassoud emphasizes that the robotic task is defined in terms of constraint verification for a set of measurements applied on the system [71]. All the performance of a robotic task is then reduced to trajectory tracking and is expressed as follows:

*Let  $q_0$  be an initial configuration and  $q_f$  a final configuration, both achievable, that is to say, within the robot workspace and non-singular; then one trajectory  $H(\lambda)$  with  $\lambda \in [0, 1]$  is calculated in the free space, such that  $H(0) = q_0$  and  $H(1) = q_f$ .*

Nilsson and Udupa proposed initial work on robotic tasks for specific robots [48, 73]. In [34], a first general approach included the first trajectory planning algorithm. In [6, 31], numerous work summary indicates mostly obstacles avoidance. Coiffet extends the application of constraints to the end-effector member maintained in a constant orientation, singularity avoidance and sampling rates [15]. Specifically, the milling goal is to produce a workpiece by material removal [45]. The end result is an object whose surfaces are characterized by a certain quality of surface finish. This quality is normally defined by a permissible error denoted by a tolerance in terms of the part's drawing and an index describing the surface quality. The part is thus represented like a geometric object drawn using one typical CAD software. The CAM functionality translates the virtual object shape into a certain number of paths spanning and scanning the part. These task paths are the CNC set-points in one machining file.

*The machining path is defined as the functional path that determines the contact position between the tool tip and the workpiece [11].*

### 3.2 Description

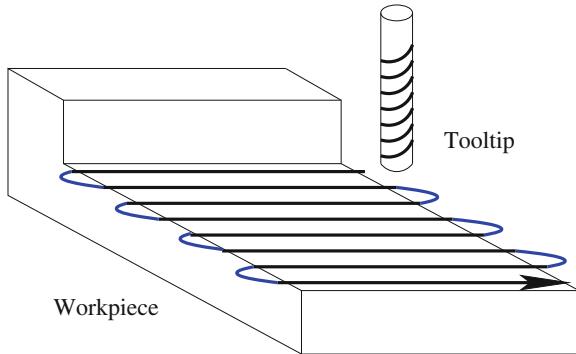
Several parameters are required to proceed with tool operation description: tool tip position, tool tip orientation, tool feedrate, nominal trajectory to follow during machining and tool rotational speed. These parameters, except the last one, have been integrated into the simulation tool since they are all specifically related to the robot operation. Machining consists of a set of task trajectories, Fig. 6.

Simulation proceeds with surfacing tasks which are easy to visualize and simple to represent. However, from the point-of-view of the robot control, they are not necessarily easier with a parallel manipulator featuring non-linear kinematics.

**Definition 3** Let  $H$  be a machining task, cut into a set of  $m$  paths,  $H = h_1, h_2, \dots, h_m$ . Let  $\tau_{tot}$  be called the total time to perform all machining and let  $\tau_i$  be path  $i$  duration.

The trajectory  $P_d$  departure point and the  $P_f$  arrival point or final point are respectively corresponding to time  $t = 0$  and  $t = \tau_{tot}$  [35, 69]. We know that the

**Fig. 6** Example of a typical milling task



end-effector is at rest at the point of departure and arrival, where the velocity and acceleration are then set to zero at these points. For each task path  $h_i$ , the start point and the end point are made to respectively correspond to times  $t_i = \sum_{k=1}^i \tau_{k-1}$  and  $t = \sum_{k=1}^i \tau_k$ .

The realization of the task is essentially reduced to the location of the tool tip in task space. According to Chedmail and Mery [11, 45] the majority of machining tasks consists of two types of paths, Fig. 6: Continuous machining path and transition paths between them when there is no contact between the tool and the part. The transitions can be described as robotics classical point-to-point motion which should last a minimum amount of time [45]. This is actually where parallel robots can also be of advantage compared to massive serial Cartesian machines.

**Definition 4** The functional paths are defined as continuous paths corresponding to the machining process of the workpiece [11].

These paths are usually made at a constant feedrate to ensure the quality of the finished surface. Each functional path is defined by two nominal functions: one function describing the tool Cartesian position, a second function describing orientations. For example, in Fig. 6, we observe that the straight line segments are the machining paths. A task is defined by a succession of displacements when the tool is actually in operation [45]:

$$X_i^{nom}|_{R_f} = (x_i(t), y_i(t), z_i(t), \theta_{1i}(t), \theta_{2i}(t), \theta_{3i}(t))^t \quad (15)$$

Typically, milling tasks generally consist of sets of arcs, straight lines, spirals and eventually splines. The robot moves the end-effector at constant feedrate. This translates by the following Cartesian constraint:  $\|\vec{V_c}(t)\| = F_r$  where  $F_r$  is a constant. Thus, the speed being the velocity magnitude is always constant. These tasks are usually defined on planes parallel to the XY plane of  $R_f$ , the robot reference frame, meaning that we must ensure that:  $P_d[z] = P_f[z] = P_i[z]$ .

### 3.3 Trajectory Position Nominal Function

A task is defined by a parameterized nominal function set where each function is defined as  $\vec{P}^{nom}(\lambda)$  with  $\lambda \in [0, 1]$  to exactly describe the task trajectory to follow. It covers the vast majority of machining work in the industry [45]. For each segment, we assign  $\lambda = 0$  to the start point  $P_d$  to and end point  $P_f$  with  $\lambda = 1$ . The task will seek to move the robot tool along a function whose general implicit form is defined as follows:

$$\vec{P}^{nom}(\lambda) = f(P_d, P_f, \lambda) \quad (16)$$

In the case of a constant feedrate,  $\tau$  represents the time to complete a path, one can express the parameter  $\lambda$  versus time  $t$  according to the following relationship:  $\lambda = \frac{t}{\tau}$ . The implicit function becomes:

$$\vec{P}^{nom}(t) = f(P_d, P_f, t, \tau). \quad (17)$$

Knowing that the traveled distance  $\delta S$  is the actual distance along the path between  $P_d$  the start point and  $P_f$  the final point and is calculated by  $\delta S = F_r \tau$  where  $F_r$  is the constant tool feedrate. Then, the implicit function is expressed by:

$$\vec{P}^{nom}(t) = f(P_d, P_f, t, F_r). \quad (18)$$

This form will be retained for the simulation since, in the machine-tool domain, it is customary to specify the machining tasks in terms of initial points, endpoints, path type and feedrate [45].

#### 3.3.1 Trajectory in a General Plane

For reasons of simplicity, the machining majority is arranged on planes parallel to the XY plane.

#### 3.3.2 Straight Line Segment Formulation

The straight line segment starts by calculating the trajectory time:

$$\tau = \frac{\|P_d - P_f\|}{F_r} \quad (19)$$

Then, the segment equation is determined by:

$$\vec{P}^{nom} = P_d + \frac{(P_f - P_d)}{\tau} t \quad (20)$$

### 3.3.3 Arc Formulation

It is therefore proposed several methods to evaluate an arc depending on the data entered:

- First case—start point:  $P_d$ , end point:  $P_f$ , feedrate:  $F_r$ , centre of rotation:  $CC$  and radius:  $r$ ;
- Second case—start point:  $P_d$  or end point:  $P_f$ , displacement angle:  $\delta\phi$ , feedrate:  $F_r$ , centre of rotation:  $CC$  and radius:  $r$ ;
- Third case: start angle:  $\phi$ , end angle:  $\Phi$ , feedrate:  $F_r$ , centre of rotation:  $CC$  and radius:  $r$ .

Two additional inputs are necessary. To calculate the path as such,  $P_f$  is not directly used and it will only be used to calculate the total time  $\tau$ .

Firstly, the angular velocity is calculated and then, the circular function is instantiated. Particular attention must be brought to the  $\phi$  angle calculation which corresponds to either the starting point or end point:

- to match the start time which is not always zero,
- to proceed with quadrant verification related to trigonometric function inversion.

The first case is selected being considered sufficient for simulation purposes and the following algorithm is implemented:

---

```

Arc(Input)       $\omega = \frac{F_r}{r}$ 
    if  $P_d[2] - CC[2] \geq 0$  then
         $\phi = \arccos(\frac{P_d[1]-CC[1]}{\|P_d-CC\|})$ 
    else
         $\phi = \pi - \arccos(\frac{P_d[1]-CC[1]}{\|P_d-CC\|})$ 
    if  $P_f[2] - CC[2] \geq 0$  then
         $\Phi = \arccos(\frac{P_f[1]-CC[1]}{\|P_f-CC\|})$ 
    else
         $\Phi = \pi - \arccos(\frac{P_f[1]-CC[1]}{\|P_f-CC\|})$ 
     $\vec{P}^{nom} = [CC[1] + r\cos(\omega t + \phi), CC[2] + r\sin(\omega t + \phi), P_d[3]]$ 
    tau =  $\frac{\Phi - \phi}{\omega}$ 
return( $\vec{P}^{nom}$ ,  $\tau$ )

```

---

### 3.4 Trajectory Orientation Nominal Function

The end-effector motion can be modeled to obtain decoupled translation and rotation displacements [15, 26]. Many methods exist for modeling orientations and their displacements: navigation angles (roll, pitch, yaw), two types of Euler angles, quaternions, Rodriguez parameters, the normal vector to the mobile platform, the pointing vector of the tool axis, etc. The constant orientation was selected for the proposed simulation.

The first set of encountered trajectories are the so-called 3 DOFs milling tasks or surfacing tasks. These are performed at constant orientation where the tool axis is kept perpendicular to the workpiece. To simplify calculations, the parallel robot is positioned to keep the tool axis parallel to the base reference frame z axis. Then, the rotation matrix is equal to the identity matrix. This means that the end-effector axis is set to  $N_c = [0, 0, 1]$  which is selected for orientation formulation, since many rotation formulations lead to singularities when  $R = I$  (Euler angles, Bryant angles, etc.) as shown in [15, 26]. Path planning can be simplified with the calculations avoiding rotation matrix transformations. This axis can be called pointing axis or normal axis since it is usually selected the mobile platform normal axis coinciding with the tool axis.

It is possible to apply the same formulation for any other constant pointing axis displacement. The normal vector becomes  $N_c = [n_x, n_y, n_z]$ . However, in this case, the normal rotation parameters are converted into a rotation matrix. This is used to calculate the IKP in trajectory analysis.

### 3.5 Milling Task Preparation

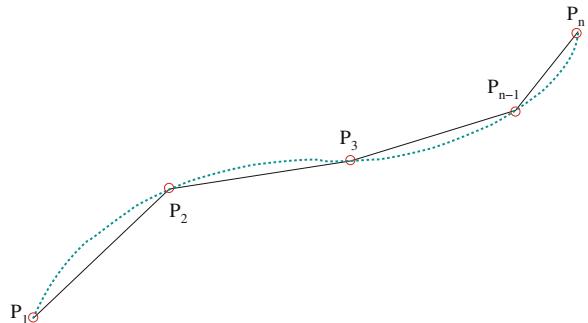
One mechanical workpiece is drawn on a CAD program as a virtual solid. The CAM machining module defines cutting planes on the workpiece. It proceeds by intersecting the cutting planes with the virtual solid to determine several parallel surfaces. It fills the surface with cutting paths resulting into a set of nominal Cartesian trajectory functions that are saved in a nominal Cartesian trajectory file. The CAM program further transforms the nominal Cartesian trajectory functions into sets of points that are saved in a theoretical Cartesian trajectory set-point file which can be uploaded to the CNC controller.

### 3.6 Initial Digitization of Milling Trajectories

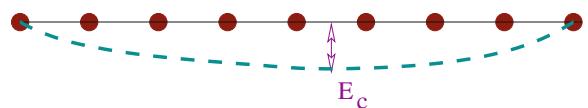
As input, a task definition file comprises a series of nominal functions; each function is of the form  $\vec{X}^{nom}(t)|_{r_F}$ . The points of departure and arrival  $P_D$  and  $P_f$  are known for each function. Theoretical positions are thus calculated from these nominal symbolic functions:  $\vec{X}^{th}(c)|_{r_F} = \vec{X}^{nom}|_{r_F}(t)$  at each  $T_c$  sampling cycle. Time  $T_c$  is assumed to remain constant throughout the process. The total time is therefore set to  $t = c T_c$  for  $c = 1, \dots, s$ . Firstly, a first time digitization occurs at the sampling rate, Fig. 7, which has the effect of transforming the paths in point series.

Finally, the CAM program considers that all theoretical points are connected by line segments in some kind of linear approximation, Fig. 7. Further point sampling is then performed by separating the points selected by a calculated distance in accordance with a chord error  $E_c$ , Fig. 8. Thus, as an arc is bent by a straight line rope, each

**Fig. 7** Digitization of task  
Cartesian theoretical path



**Fig. 8** Digitization of task  
theoretical section



pair of set points sees a line segment connecting them. This cord is at a maximum distance of  $E_c$  from the nominal trajectory. Let the arc be of radius R and length L, then  $E_c = R - R\cos(\frac{L}{2R})$ . In order to obtain a predetermined  $E_c$  cord error, the arc point distance is calculated by:  $L = \arccos(1 - \frac{E_c}{R})$ . Then, the cord distance is calculated by:  $D = 2\sqrt{E_c^2 - 2R E_c}$ . Knowing the constant feedrate and the cord distance, the sampling time  $T_p$  is then calculated. Each new point will then add to the original theoretical path file. The resulting file is called the complete theoretical Cartesian path. The CAM program linearization is typically already introducing an error, so that the accuracy of the robot can never be better than this  $E_c$  cord error value.

Then, the IKP is calculated on each theoretical Cartesian path. For each pose point comprising the position and orientation, the actuator positions are calculated. The result will be written in an actuator theoretical set-point file which is then uploaded to the CNC controller.

### 3.7 Second Digitization of Milling Trajectories

Running at a smaller cycle time, the six servo feedback loops traditionally implement a PID feedback loop on each linear axis position. During each  $T_p$  cycle time, the path following level interpolates a certain number points inside the interval determined by each point pair in the actuator theoretical set-point file. The number of points is determined by:  $N = \text{floor}(\frac{T_p}{T_s})$  where  $T_s$  is the servo feedback loop cycle time determined by the time to calculate the PID algorithm. Actuator point sampling is then performed by utilizing a polynomial interpolation function.

Typically, in many CNC controllers, it is observed that the servo sampling rate (second level) can be ten times the cycle time of the first level.

## 4 Verification Criteria for Machining

### 4.1 Machining Accuracy

The most important performance criterion is the machining surface finish. Since machining requires a trajectory following with high precision, we must ensure that the path is simulated within a given precision [45].

In classic robotics, the majority of path planning applications are classified as point-to-point and a marginal number are concerned by continuous paths such as in machining. However, even when implementing continuous, the robot control algorithms handles points. The main difference with point-to-point control is that the task is defined by several hundreds of points instead of a few points. Liege and Coiffet define four types of precision: static accuracy, dynamic accuracy, repeatability and resolution [15, 32]. Repeatability stands for the reproduction accuracy of the same movement and does not really apply for continuous trajectory tasks. The resolution is the smallest amount of change in the positions and orientations. It is determined by robot component choices.

**Definition 5** Static accuracy is defined as the ability of the robot to position and orient the end-mechanism in accordance with the programmed instructions.

This notion is applicable to a specific point and then cannot be extrapolated to one entire continuous trajectory.

**Definition 6** Dynamic accuracy is the ability of the robot to follow a path by the end-effector mechanism in accordance with the programmed path.

In principle, the error is calculated at all points along its theoretical path  $\overrightarrow{X(kT)}^{th}$  where  $k = 1, \dots, k_{max}$  where  $k_{max}$  is the number of discretized points. The error vector between the nominal path and the simulated path is then:

$$\overrightarrow{\varepsilon}(kT) = \overrightarrow{X(kT)}^{th} - \overrightarrow{X(kT)}^{nom} \quad (21)$$

The distance or error vector magnitude is also calculated:

$$\varepsilon(kT) = ||\overrightarrow{X(kT)}^{th} - \overrightarrow{X(kT)}^{nom}|| \quad (22)$$

After calculating the error vector or value of distance for a path, we determine the overall path accuracy for each error vector component and the error vector distance by choosing the largest value.

## 4.2 Error over the Cartesian Position

### 4.2.1 Calculation of the Absolute Error and the Error Vector Between the Points

In practice, the end-effector precision calculation is divided into two task space parts: Cartesian position and Cartesian orientation. For the error in the Cartesian position, we obtain the equation is calculated for each theoretical Fig. 8:

$$\varepsilon(kT) = \|\overrightarrow{X(kT)}^{sim} - \overrightarrow{X(kT)}^{th}\| \quad (23)$$

We also study the nature of the error vector.

$$\overrightarrow{\varepsilon(kT)} = \overrightarrow{X(kT)}^{sim} - \overrightarrow{X(kT)}^{th} \quad (24)$$

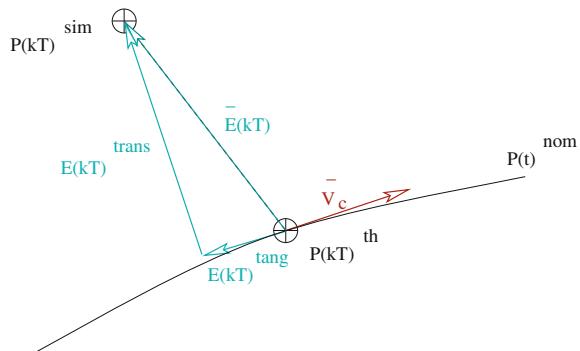
This calculation is also applicable on theoretical points of the *CAD* produced files.

Since the error along the trajectory is not as significant as the transversal path error, we calculate the tangential error and transversal error, Fig. 9. The transverse error can also be called cross-sectional, normal or perpendicular error.

The tangential error allows us to evaluate if the simulated path is ahead or behind the nominal planned route. A tangential error indicating that the real path is followed ahead of time is of course advantageous because it means that the trajectory can be continued in a shorter time than expected. In fact, Liegeois states that a robot can be late in the path set without the finished surface being affected [32]. A tangential error indicating that the real path is plagued by a slowdown may not necessarily affect the surface finish as such and therefore is not so considered important.

On the other hand, the transversal error will directly affect the surface finish. It corresponds to the difference between the simulated path and the nominal path at time  $t = kT$  where  $k = 1, \dots, m$  with  $m$  the number of points. Then, we try to determine if the simulated path is located within a given path tube with a predefined radius. The tube radius is determined by machining tolerances.

**Fig. 9** Error vector, tangentielle error and transverse error



To calculate the vector tangential error, we must determine the unit vector tangential to the nominal curve through the velocity vector:

$$\overrightarrow{u(t)} = \frac{\overrightarrow{V_c(t)}}{\|\overrightarrow{V_c(t)}\|} \quad (25)$$

The value of the tangential error is obtained by:

$$\varepsilon(kT)^{tang} = \overrightarrow{u(t)} \cdot \overrightarrow{\delta P(kT)} \quad (26)$$

Applying the Pythagorean theorem, we finally find the value of the transversal error:

$$\varepsilon(kT)^{trans} = [(\varepsilon(kT))^2 - (\varepsilon(kT)^{tang})^2]^{1/2} \quad (27)$$

The calculation of transversal error with respect to the nominal trajectory is not exact but an approximate value of the deviation sought because it is obtained from the digitized values and is not necessarily the perpendicular error defined as the minimal distance between the nominal and theoretical trajectories. It is necessary to nuance this comment. The perpendicular error may not be a direct measurement of surface finish. For example, during 3D milling, the robot is positioned so as to obtain the Z-axis of the terminal member perpendicular to the surface to be machined. Then, we seek to mill a planar surface that is positioned parallel to the XY plane and the finished surface will be evaluated by calculating  $\varepsilon(kT)_z$ . Upon reaching the portion of the part where a wall is reached, the wall perpendicular error will be determined.

### 4.3 Calculate the Actual Deviation from a Nominal Curve

To be meaningful, dynamic precision must be defined relative to the nominal path [32]. On the Fig. 9, we note that  $\overrightarrow{\varepsilon(kT)^{trans}}$  is not the actual deviation from the nominal curve. To achieve this, we must calculate the point  $\tilde{P}$  being the closest to  $\overrightarrow{P(kT)^{sim}}$  on the nominal curve. To do this, we determine the time  $t^{devi}$  which corresponds to the point  $\tilde{P}$  on the nominal curve, Fig. 9, and two methods can be derived.

The first method consists in determining the normal to the nominal curve which is performed by solving the following system:

$$(\overrightarrow{P(kT)^{sim}} - \overrightarrow{P(t)^{nom}}) \cdot \overrightarrow{V_c(t)} = 0 \quad (28)$$

The second method consists in searching the minimum distance between  $\overrightarrow{P(kT)^{sim}}$  and  $\overrightarrow{P(t)^{nom}}$  by calculating the minimum of the function:

$$G(t) = \|\overrightarrow{P(kT)}^{sim} - \overrightarrow{P(t)}^{nom}\| \quad (29)$$

which corresponds to determining the time at which the derivative of the function is zero, that is to say when  $G'(t) = 0$ .

Introducing  $t^{devi}$  time in the function, we obtain  $\tilde{P}$  and then the deviation is calculated:

$$\varepsilon(kT) = \|\overrightarrow{P(kT)}^{sim} - \tilde{P}\| \quad (30)$$

Deviation value is determined by calculating the maximum deviation of an entire trajectory. The second approach for calculating the deviation  $t^{devi}$  has the advantage of being less complex in terms of calculations and therefore will be preferred.

#### 4.4 Calculation of Deflection from a Straight Line Segment

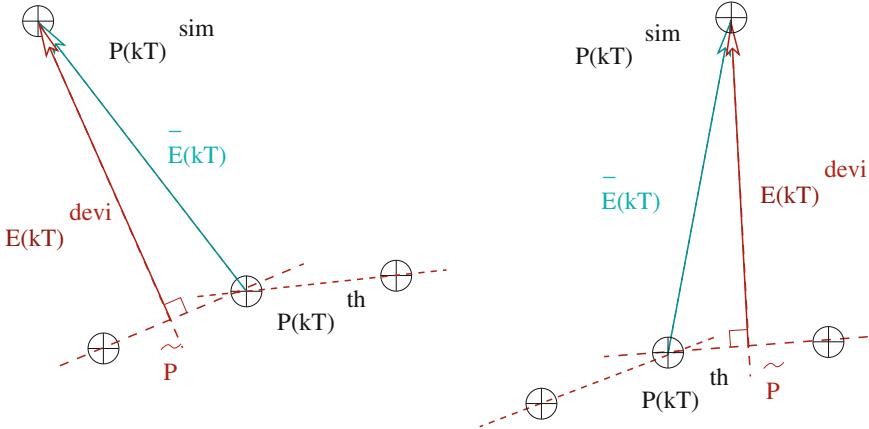
When the nominal paths are straight lines, it is not necessary to perform the calculation of the deviation to approach presented in the previous section. Determining the deviation  $\overrightarrow{P(kT)}$  directly by calculating the distance between the simulated  $\overrightarrow{P(kT)}$  and the line defined by the starting point  $\overrightarrow{P_1}$  and the arrival point  $\overrightarrow{P_2}$  of the section:

$$\varepsilon(kT) = (\|\overrightarrow{P_1 P(kT)}\|^2 - \|\overrightarrow{P_1 P(kT)} * \frac{\overrightarrow{P_1 P_2}}{\|\overrightarrow{P_1 P_2}\|}\|^2)^{1/2} \quad (31)$$

#### 4.5 Calculation of the Deviation from a Theoretical Curve

There are many cases where the nominal functions are not available and the curves are not necessarily straight lines. For example, as we have already explained, many *CAD* program produce files with an  $E_c$  cord error between selected points. Not knowing the curve profile between these points, the *CAM* module interpolates using a linear function, that is to say, we assume that the points are connected by line segments, being different from the exact shape having then an unknown curvature. The curvature was lost in the digitization process. The deviation calculation takes then Eq. (31). The question to be carefully addressed is the choice of the points  $P_1$  and  $P_2$ . We wish to determine the theoretical interval being closer to  $\overrightarrow{P_k^{sim}}$ , the point simulated, Fig. 10. The comparison is limited to adjacent intervals: the  $i - 1$  segment before and the segment  $i$  after the point  $\overrightarrow{P_k^{th}}$ .

There are two possible methods for interval selection. The first method is selecting the interval by the scalar products respectively for the interval  $i - 1$  and  $i$ :



**Fig. 10** Deviation vector from the theoretical points

$$v_{k-1} = (\overrightarrow{P_{k-1}^{th}} - \overrightarrow{P_k^{th}}) \cdot \overrightarrow{\varepsilon P(kT)} \quad (32)$$

$$v_k = (\overrightarrow{P_{k+1}^{th}} - \overrightarrow{P_k^{th}}) \cdot \overrightarrow{\varepsilon P(kT)} \quad (33)$$

The closest interval will be identified by selecting the positive result between  $v_{i-1}$  and  $v_i$ .

The second method involves the calculation of the time corresponding to the point on each straight line segment:

$$t_{k-1} = T_p \frac{\overline{P_x^{sim}} + \overline{P_y^{sim}} + \overline{P_z^{sim}}}{\|\overrightarrow{P_{k+1}^{th}} - \overrightarrow{P_k^{th}}\|^2} \text{ where } \overline{P^{sim}} = \overrightarrow{P_k^{sim}} - \overrightarrow{P_{k-1}^{th}} \quad (34)$$

$$t_k = T_p \frac{\overline{P_x^{sim}} + \overline{P_y^{sim}} + \overline{P_z^{sim}}}{\|\overrightarrow{P_{k+1}^{th}} - \overrightarrow{P_k^{th}}\|^2} \text{ where } \overline{P^{sim}} = \overrightarrow{P_k^{sim}} - \overrightarrow{P_k^{th}} \quad (35)$$

The two times are then compared with the cycle time  $T_p$  and the closest interval from the point is the one confirming  $0 \leq t \leq T_p$ .

The second approach is less complex to implement and has been chosen.

The distance is determined by replacing  $P_1$  and  $P_2$  by the extrema of the chosen interval in Eq. (31). This distance is not equal to the actual deviation since each interpolation corresponds to the straight line segment between two points. It is necessary to take the deviation vector and add the vector related to the  $E_c$  error being perpendicular to the straight line segment and included in the plane defined by the velocity vector at point i and the vector aligned with straight line segment.

Note that if the theoretical path is a straight line, then we can calculate the deviation directly with Eq. (31).

#### **4.6 Calculate the Actual Deviation from a Theoretical Curve with a Small Radius of Curvature**

In the case where the radius of curvature is high, this method is not guaranteed to calculate the minimum distance, since the theoretical  $\vec{P}_k^{th}$  is not necessarily the closest to the simulated  $\vec{P}_k^{sim}$  point. For example, such a situation is encountered when machining rectangles with corners with radii of curvature tending towards 0.

To remedy this problem, an added algorithm determines  $\vec{P}_n^{th}$ , the closest theoretical simulated point, by seeking the value of  $n$  such that  $(||\vec{P}_n^{th} - \vec{P}_k^{sim}||)$  is minimized by varying  $n$  from  $n - 20$  to  $n + 20$ . Indeed, it is not necessary to test all trajectory points. Then, the deviation is calculated with the aforementioned method.

#### **4.7 Orientation Errors**

There would as many methods to calculate errors over the orientations as there exists representation models. We chose to determine the orientation error by calculating the variations on the normal vector because it is more ergonomic to visualize the movement of a vector that characterizes the parallel robot mobile platform.

$$\overrightarrow{\delta N_c(kT)} = \overrightarrow{N_c(kT)} - \overrightarrow{N(kT)}_c^{th} \quad (36)$$

In addition, *CAD* programs represent orientations by expressing the pointing vector collinear with the tool axis which, in the case of parallel robots, is commonly corresponding to the mobile platform normal vector.

#### **4.8 Actuator Joint Errors**

The simulator also compares the theoretical and simulated actuator joint trajectories, thereby obtaining the actuator error for the six actuators. For  $i = 1, \dots, 6$ , we calculate  $\xi_i^{sim} = L_i^{sim} - L_i^{th}$ .

#### **4.9 Error Models**

In order to simulate a realistic trajectory pursuit, error models are introduced at different levels of the simulator. The majority of errors are introduced by adding a parameter to a function determined by randomly selecting a value in a specific interval  $[-max, +max]$ .

We have chosen to the modeling of all the following errors:

- CAD file precision,
- sensor accuracy,  $\delta l_i$ ,
- configuration precision,  $\delta OA_i$  and  $\delta CB_i$ ,
- precision on the calculation of the **FKP**,
- resolution of time measurement  $\Delta t$  and temporal digitization,
- the resolution of signal digitization,
- the asynchronous nature of joint signal updates.

We can thus simulate a trajectory introducing all errors, any combination of these or even only one. The simulation can be tailored to the actual study and it is possible to isolate errors and investigate their impact on surface finish. We propose two alternative calculation errors:

- the relative error between two steps,
- the absolute error giving the end-effector accuracy.

## 5 Results of Path Simulation

In this section, as part of the path planning related to milling and by extension to all high accuracy applications, kinematics simulation results calculates end-effector surface finish impact integrating configuration inaccuracies and position based CNC control strategies. The results are compiled, presented, analyzed and compared.

### 5.1 Parallel Robot Configuration

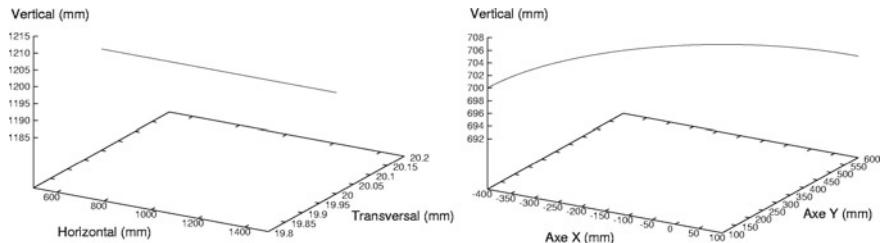
We try one difficult FKP example on a typical 6-6 hexapod with 40 complex solutions out of which 16 real solutions can be extracted. Let us take a typical **6-6** configuration example written in a configuration text file which includes the manipulator essential parameters: the coordinates of the joint center positions  $OA_i$  and the coordinates of the joint center positions  $CB_i$ . The unit is the millimeter. These values were determined by a calibration procedure from a real robot and are shown on Table 1.

### 5.2 Typical Trajectory and Realistic Milling Configuration

We have implemented various control strategies in position by interpolating points by polynomial functions of the first degree and third degree. In the first case, one has to calculate the acceleration as a function of the end conditions.

**Table 1** Parallel manipulator configuration table

Joint coordinates	Respective values
$OA_1(x)$ $OA_1(y)$ $OA_1(z)$	464.141 389.512 -178.804
$OA_2(x)$ $OA_2(y)$ $OA_2(z)$	569.471 207.131 -178.791
$OA_3(x)$ $OA_3(y)$ $OA_3(z)$	529.050 -597.151 -178.741
$CB_1(x)$ $CB_1(y)$ $CB_1(z)$	68.410 393.588 236.459
$CB_2(x)$ $CB_2(y)$ $CB_2(z)$	375.094 -137.623 236.456
$CB_3(x)$ $CB_3(y)$ $CB_3(z)$	306.664 -256.012 236.461

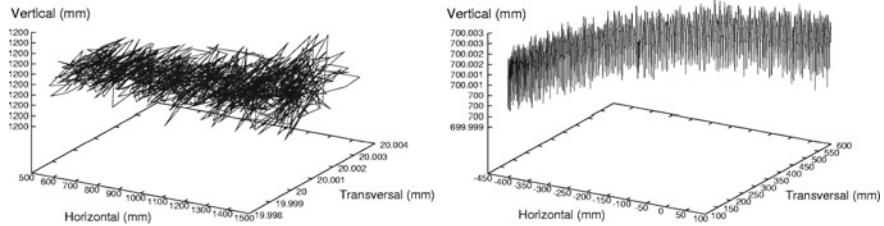
**Fig. 11** Selected nominal paths

We chose two nominal paths located on planes parallel to the XY plane. These path nominal functions are respectively determined by the following configurations, Fig. 11:

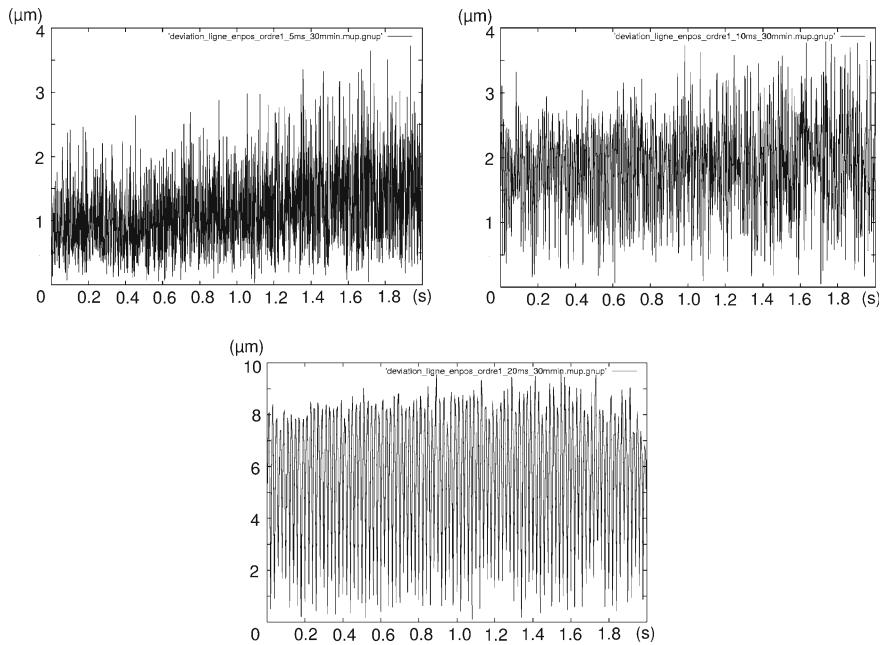
- a line segment starting at point [500, 20, 1,200] and ending at point [1,500, 20, 1, 200] traveled at three constant feed forward speeds: 30, 45 and 60 m/min.
- an arc of radius 500 mm from the point [100, 600, 700] to reach point [-400, 100, 700] using the same three feed rates. The center of the arc is point [100, 100, 700].

Note that the selected tasks are simulated trying to reproduce realistic milling conditions. We will study the trajectories at different feed rates which are set to 30, 45 and 60 m/min. The feedrates of 30 and 60 m/min speeds correspond to the speeds of high speed milling *HSM* and ultra high speed milling *UHSM* respectively. A study is also conducted on the impact of path following cycle times which will be set at 5, 10 and 20 ms.

The simulator computes and sketches the two resulting Cartesian tool paths utilizing a controller with a cycle time of 10 ms and a feed rate of 30 m/min, Fig. 12 where the one dimension is exaggerated to visualize the path errors. There is a complex high-frequency noise on every simulated patterns which highlight sudden and unpredictable changes in the continued trajectory.



**Fig. 12** Simulated path pursuits: straight line segment and arc



**Fig. 13** Simulated path deviation for a straight line segment: cycle times of 5, 10 and 20 ms, linear joint interpolation

### 5.3 Control with Linear Interpolation

#### 5.3.1 Straight Line Segment with Linear Interpolation

The first tests with the simulator implements the first level control proceeding with actuator joint set-point interpolation utilizing a linear interpolation. In the first analysis, we calculate the deviation of a typical path segment simulated over a nominal path. We therefore study the straight line segment path by first varying the cycle time of the order and the results are shown in Fig. 13.

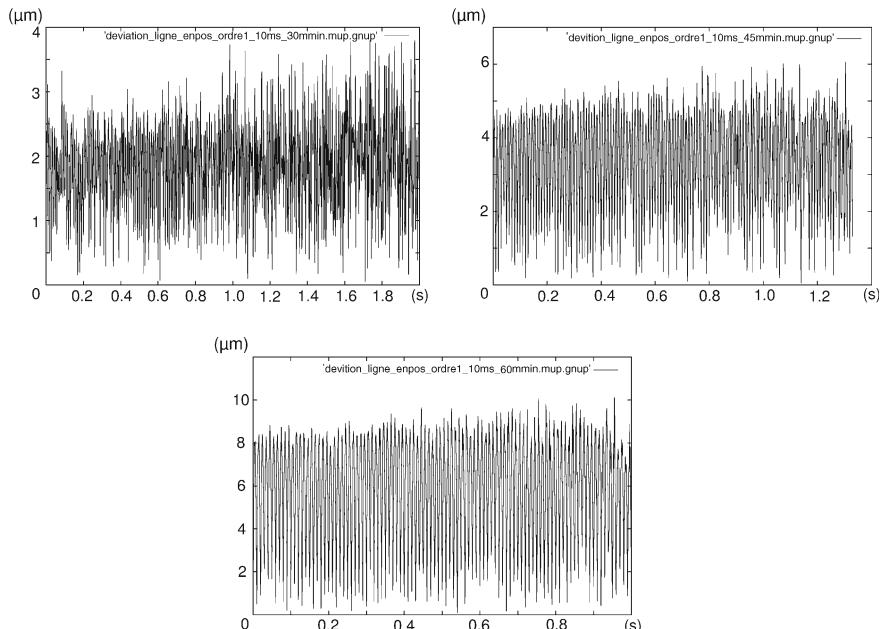
At 5 ms, the deviation is a high frequency signal oscillating around a straight line function  $f(t) = t/4 + 1$  in microns. The amplitude increases and has peaks reaching  $3.6 \mu$ . At 10 ms, the signal oscillates around a constant straight line at  $1.8 \mu$  and oscillations then increase very slowly. At 20 ms, the average rose to  $4.75 \mu$  and the extrema of the oscillations are 1 and  $8.5 \mu$  with peaks at  $9.5 \mu$ .

In the second analysis, we study the same trajectory by now varying the feed-rates and the results are shown in Fig. 14.

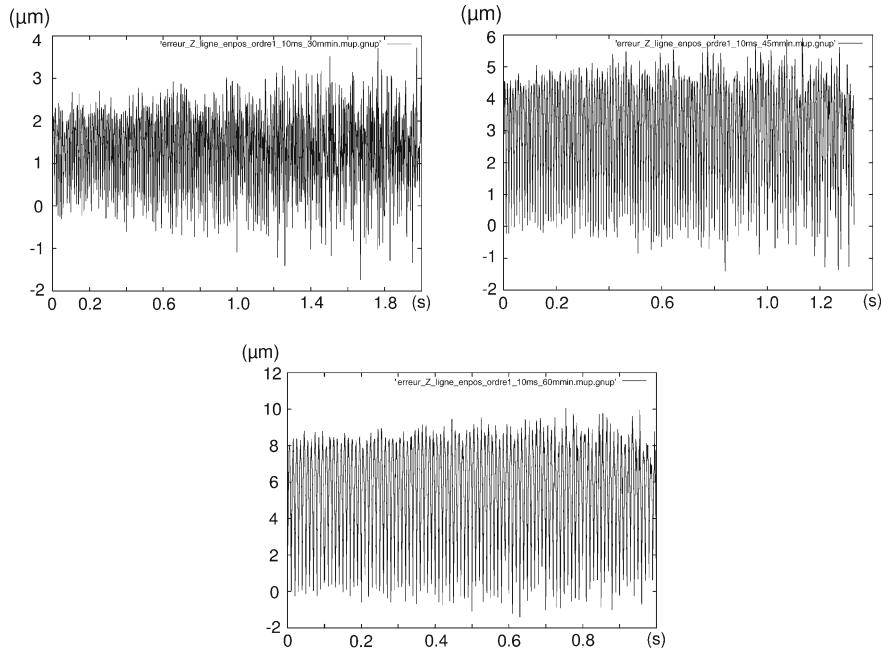
At feedrates of 30 m/min, the average is near  $2 \mu$  and the high frequency oscillations feature peaks from 0 to  $3.5 \mu$ . At 45 m/min, a similar signal is obtained where the average rises to  $3.5 \mu$  and peaks reach  $6 \mu$ . A 60 m/min, the oscillation average reaches  $5 \mu$  with  $10 \mu$  peaks.

We continue the analysis by showing graphs of vertical errors that are perpendicular to the machined surface errors since they provide with an excellent account of surface finish. The first graph shows the results at the selected feedrates, Fig. 15.

At feedrates of 30 m/min, the signal shows a high frequency oscillation with an average of approximately  $1.25 \mu$  with peaks as low as  $-1 \mu$  and as high as  $3 \mu$ . At 45 m/min, there is an oscillation between  $-1$  and  $5 \mu$  with an average of just over  $2.5 \mu$ . A 60 m/min, it is observed that the oscillation evolves mainly around  $4.7 \mu$  between  $-1$  and  $8.5 \mu$  with some peaks at  $10$  and  $-1.5 \mu$  (Fig. 16).



**Fig. 14** Simulated path deviation for a straight line segment: feedrates of 30, 45 et 60 m/min, linear joint interpolation



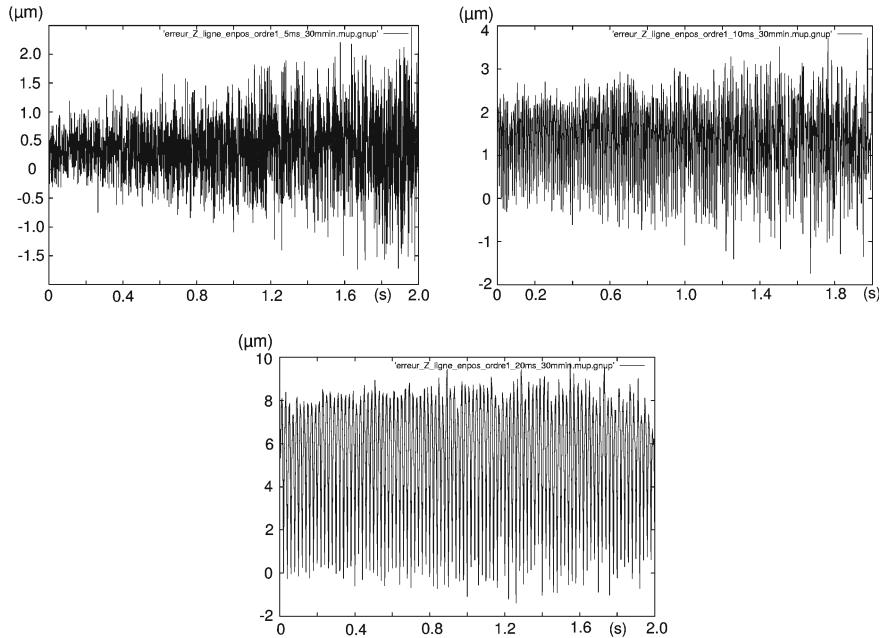
**Fig. 15** Simulated vertical error for a straight line segment: feedrates of 30, 45 et 60 m/min, linear joint interpolation

We close this simulation cycle with vertical errors at the selected cycle times, Fig. 16.

At 5 ms, the signal is oscillating at a high frequency of around  $0.3 \mu\text{m}$ . The amplitude of oscillation increases significantly. Peaks reached  $2.4 \mu\text{m}$  and  $-1.6 \mu\text{m}$  causing surface finish error to become  $4 \mu\text{m}$ . At 10 ms, the signal oscillates around  $1.6 \mu\text{m}$  with an amplitude increasing less rapidly where extrema of  $3.75 \mu\text{m}$  and  $-1.6 \mu\text{m}$  are extracted giving surface finish variations of  $5.35 \mu\text{m}$ . At 20 ms, the oscillation is constant between  $8.5 \mu\text{m}$  and  $-0.5 \mu\text{m}$  with an average of  $4.2 \mu\text{m}$ . Peaks reach  $-1.4 \mu\text{m}$  and  $9.5 \mu\text{m}$  leading to vertical variations of almost  $11 \mu\text{m}$ .

Simulation results are collected in Table 2. On the table, the order of interpolation functions, the  $T_p$  cycle time in ms, the  $F_r$  feed rate in m/min, then the minimum and maximum extrema for the  $\varepsilon$  vector error magnitude in microns, the  $\varepsilon_Z$  vertical error in microns and  $||\delta||$  deviation in microns.

As might be suspected by intuition, we get better results by reducing the path controller (first level) cycle time and also the feedforward velocity. At very high speeds or with long cycle times, we met and exceeded the threshold of  $10 \mu\text{m}$ . At feedrates below or equal to 30 m/min and cycle times equal or less than 10 ms, the kinematics surface finish or the best feasible surface finish would reach  $5 \mu\text{m}$ .



**Fig. 16** Simulated vertical error for a straight line segment: cycle times of 5, 10 et 20 ms, linear joint interpolation

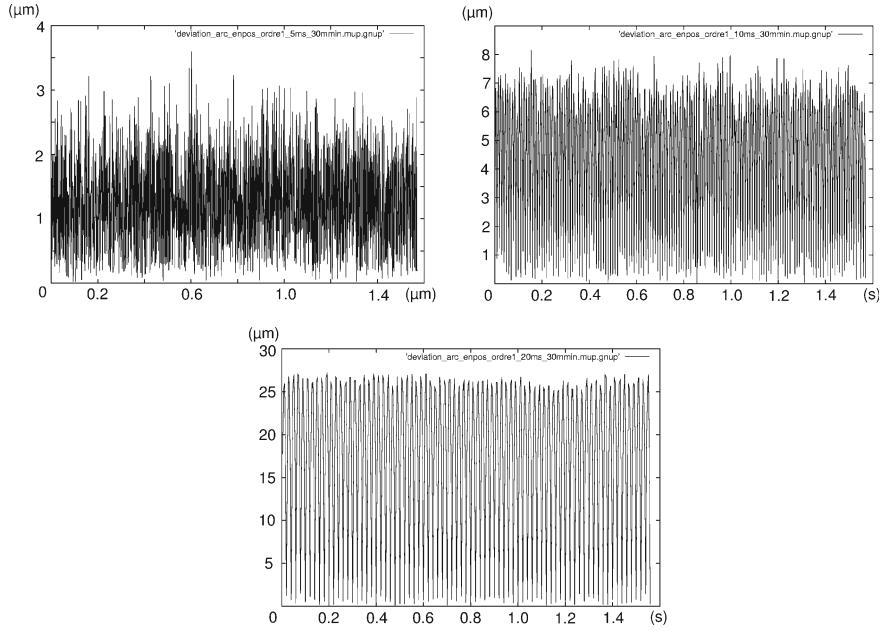
**Table 2** Simulated errors and deviations for a straight line segment: position control with linear joint interpolation

Order	$T_p$ ms	$F_r$ m/min	$\varepsilon^{\max}$ $\mu$	$\varepsilon^{\min}$ $\mu$	$\varepsilon_Z^{\max}$ $\mu$	$\varepsilon_Z^{\min}$ $\mu$	$  \delta  ^{\max}$ $\mu$	$  \delta  ^{\min}$ $\mu$
1	10	30	4.900	0.142	3.716	-1.738	3.796	0.047
1	10	45	7.198	0.219	5.918	-1.403	6.055	0.047
1	10	60	11.872	0.026	10.067	-1.403	10.106	0.096
1	5	30	3.783	0.142	2.470	-1.738	3.726	0.021
1	10	30	4.900	0.142	3.716	-1.738	3.796	0.047
1	20	30	11.487	0.258	9.734	-1.403	9.747	0.096

### 5.3.2 Arc with Linear Joint Interpolation

In the second analysis, the same simulation process is repeated for a typical arc path by first varying the cycle time of the order and the results are shown in Fig. 17.

On Fig. 17, the signals are high frequency oscillations around a constant value. At 5 ms, the signal oscillates around an average of 1.5  $\mu$  with peaks evolving from 0 to 3  $\mu$ . At 10 ms, the signal oscillates around the value of 4  $\mu$  between extremes of



**Fig. 17** Simulated path deviation for an arc: cycle times of 5, 10 et 20 ms, linear joint interpolation

0.5 and 7.5  $\mu$  with peaks near 0 and 8  $\mu$ . Increasing to 20 ms, the average increases to 14  $\mu$ . The signal resembles a very regular high frequency sinusoidal curve ranging from near 0 to 27  $\mu$ .

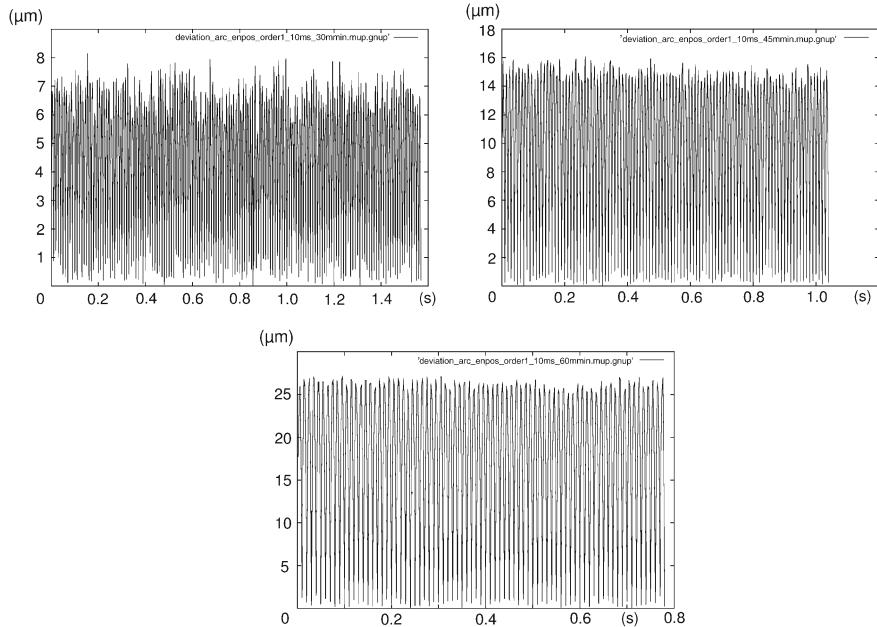
In the second analysis, we then continue the arc path analysis by plotting vertical errors at the usual different feed-rates and the results are shown in Fig. 18.

The feedrate change from 30 m/min speed to 45 m/min doubles the signal average and its oscillation amplitude (from [0, 8] to [0, 16]). Similarly, The feedrate change from 30 m/min speed to 60 m/min triples the signal average and its oscillation amplitude (from [0, 8] to [0, 27]). In the later, the signal average is 15  $\mu$  (Fig. 19).

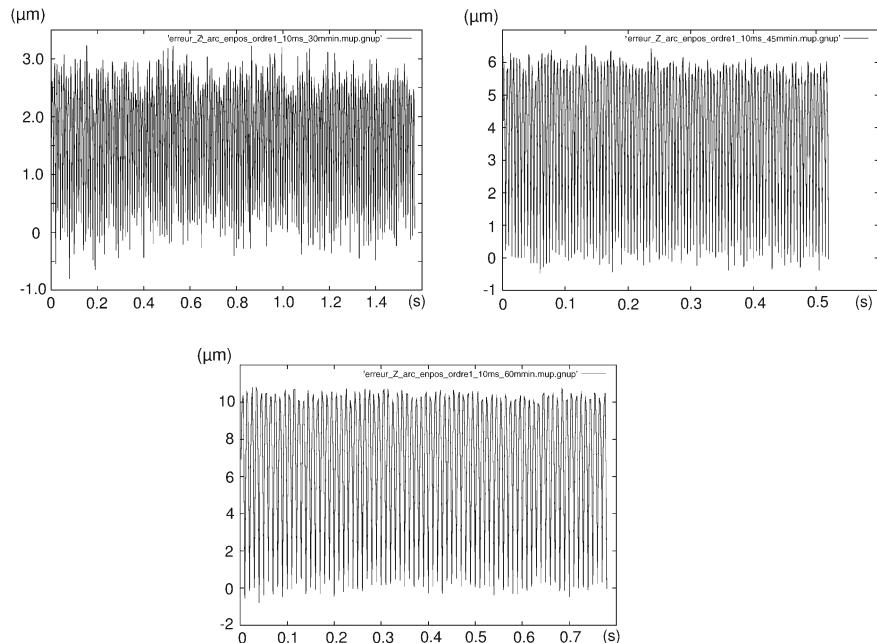
The feedrate change from 30 m/min speed to 45 m/min doubles the signal average and its oscillation amplitude (from [-0.5, 3] to [-0.5, 6]). Similarly, The feedrate change from 30 m/min speed to 60 m/min triples the signal average and its oscillation amplitude (from [-0.5, 3] to [-0.5, 10.5]). In the later, the signal average nears 5.5  $\mu$ .

To end this simulation cycle, vertical errors are computed at the selected cycle times, Fig. 20.

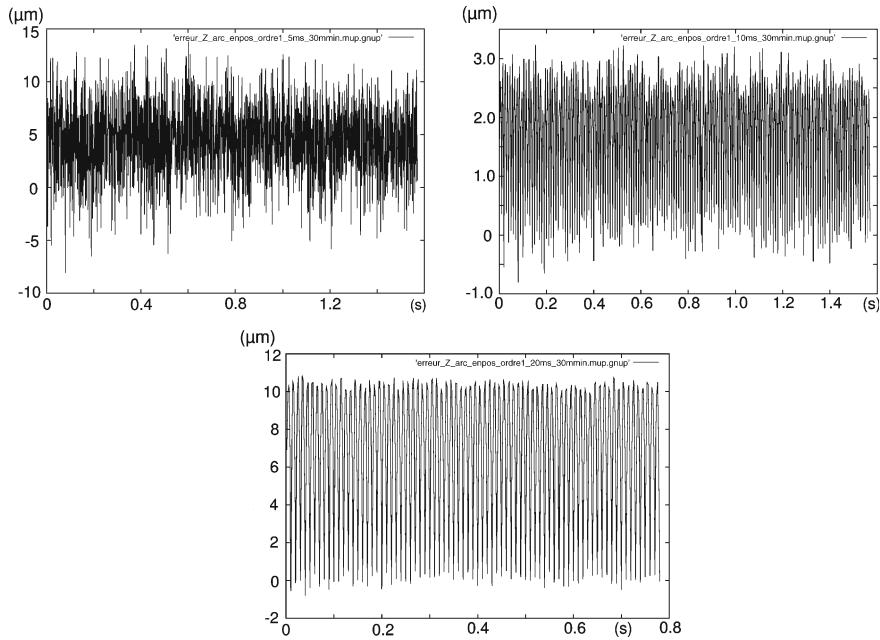
The oscillating signals are similar to the high frequency previous ones. At 5 ms, the oscillation ranges from -0.25 and 1.25  $\mu$  with an average at around 0.5  $\mu$ . At 10 ms, the oscillation extremes reach -0.2 and 2.8  $\mu$  with an average at 1.5  $\mu$ . At 20 ms, the signal is a high frequency composite oscillation with extrema at 0 and 10.5  $\mu$  and peaks at -1 and 11  $\mu$ .



**Fig. 18** Simulated path deviation for an arc: feed-rates of 30, 45 et 60 m/min, linear joint interpolation



**Fig. 19** Simulated vertical error for an arc: feed-rates of 30, 45 et 60 m/min, linear joint interpolation



**Fig. 20** Simulated vertical error for an arc: cycle times of 5, 10 et 20 ms, linear joint interpolation

Table 3 compiles the results of kinematics simulations for the arc path tests.

The results confirm the former results obtained with straight line segments. The simulator can provide surface finish of  $10\mu$ , only in the case of high speed milling ( $<30$  m/min).

### 5.3.3 Discussion on the Linear Joint Interpolation

From the kinematics analysis simulation, providing the lowest performance bounds, the CNC robot controller with linearly approximated trajectories can reach the surface

**Table 3** Simulated errors and deviations for an arc: position control with linear joint interpolation

Order	$T_p$ ms	$F_r$ m/min	$\varepsilon^{max}$ $\mu$	$\varepsilon^{min}$ $\mu$	$\varepsilon_Z^{max}$ $\mu$	$\varepsilon_Z^{min}$ $\mu$	$  \delta  ^{max}$ $\mu$	$  \delta  ^{min}$ $\mu$
1	10	30	8.164	$3e^{-5}$	3.2325	-0.806	8.157	0.016
1	10	45	16.128	0.023	6.533	-0.481	6.533	-0.481
1	10	60	27.178	0.129	10.803	-0.806	27.093	0.043
1	5	30	3.602	0.023	1.378	-0.806	3.598	0.023
1	10	30	8.164	$3e^{-5}$	3.2325	-0.806	8.157	0.016
1	20	30	27.178	0.129	10.803	-0.806	27.093	0.037

finish if the feed-rate and path following cycle time are set properly. The controller can provide surface finish of  $10\mu$ , only in cases up to high speed milling ( $<30\text{ m/min}$ ). The surface finish is not met at faster feedrates. Linear interpolators should keep control cycle times relatively short ( $<10\text{ ms}$ ) in order to reach the required surface finish. The surface finish is not met at longer cycle times.

To achieve an accuracy of less than  $10\mu$ , one should set the response time at  $10\text{ msec}$  or less and maintain the feed-rates below  $45\text{ m/min}$ . This also means that **UHSM** is not feasible.

Those linear displacements are performed by a robotic system which is not linear. The interpolators try to transform curved path segments into linear path segments leading to interrupted segments. The linear interpolation is only matching the positions at the ends of the intervals, Fig. 7. As an advantage, the linear interpolation algorithm implementation is easy and does not require difficult computations leading to smaller cycle times. As disadvantage, with the application of parallel robots, the control system will not be able to reach  $10\mu$  surface finish without very fast controllers featuring small cycle times. Moreover, rapid feedrates are not practical.

Let us add that the Cartesian velocity vector undergoes abrupt changes when passing from one linear segment to another which will result in dynamics overshoot. In fact, the continuation of this type of movement by an effective robot is impossible without stopping at each interval change which would mean slowing down the milling process.

This type of interpolation is only recommended for roughing milling.

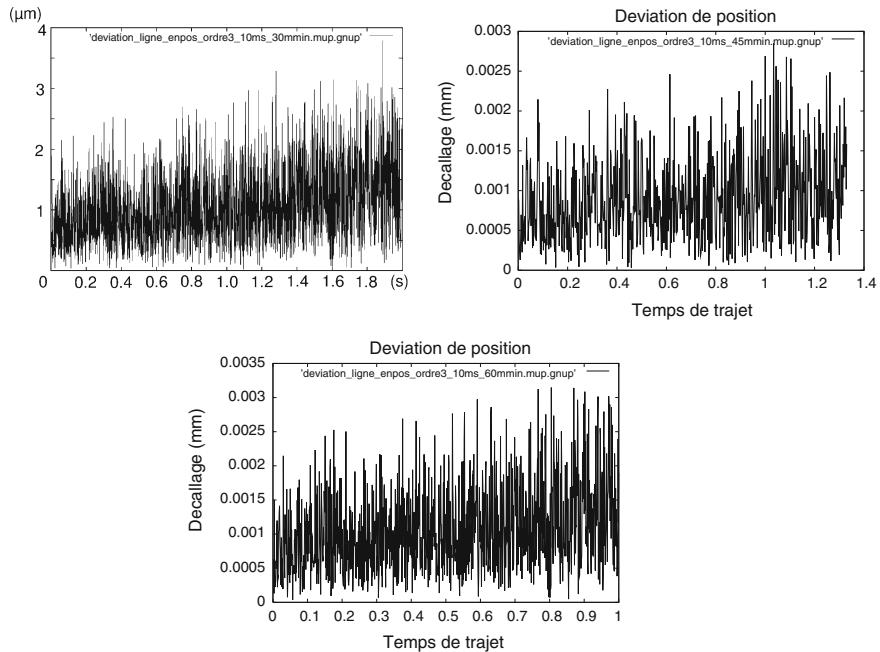
## 5.4 Control with Third Order Interpolation

### 5.4.1 Straight Line Segment with Third Order Interpolation

The second simulation test series implement the first level control proceeding with actuator joint set-point interpolation utilizing a third order polynomial interpolation. In order to ensure the continuity of movement, it is then found to match the positions and joint velocities at the ends of intervals. As it was done for linear interpolation, tests begin with an analysis of deviation with the different selected feed-rates, Fig. 21.

The three signals are featuring growing high frequency oscillations until the trajectory ends. At  $30\text{ m/min}$ , the signal oscillates around a straight line described by equation  $f(t) = 0.375t + 0.75$  and it peaks at  $3.75\mu$ . At  $45\text{ m/min}$ , the curve deviation appears more advantageous since it oscillates about the same straight line axis as with  $30\text{ m/min}$  and the signal peaks do reach just under  $3\mu$ . At  $60\text{ m/min}$ , the same conclusions can be deduced and the peaks exceed  $3\mu$  slightly.

The test are repeated by varying the cycle time of the CNC controller. The results are shown in Fig. 22.



**Fig. 21** Simulated path deviation for a straight line segment: feedrates of 30, 45 et 60 m/min, cubic joint interpolation

The three signals are actually very similar and are featuring growing high frequency oscillations until the trajectory ends. The signals oscillate around a line described by equation  $f(t) = 0.375 + 0.75t$ . The peaks reach  $3.75 \mu$ . Note that the feed-rate does not seem to impact deviation significantly. The difference between the error vector  $\|\varepsilon\|$  and deviation  $\|\delta\|$  is less than one micron. This result means that the third order interpolation allows accurate theoretical trajectory following. This also means that the path following will take place without undue delay or advance (Fig. 23).

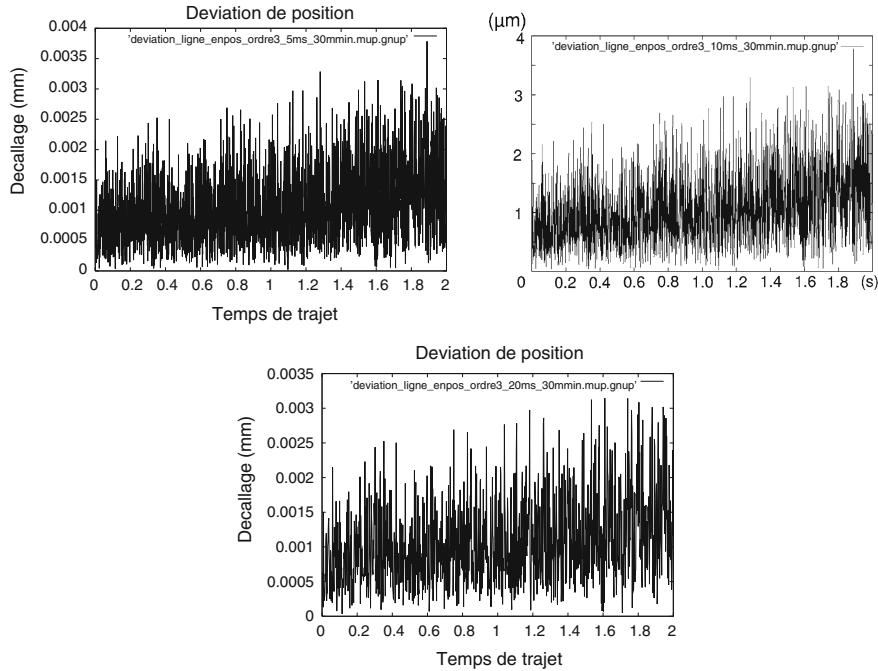
The three signals are actually very similar and are featuring growing high frequency oscillations until the trajectory ends. The graphs show signals which are centered on  $0.2 \mu$  with increasing oscillation with peaks getting close to 2 and  $-2 \mu$ .

The vertical error is simulated at various feed-rates, Fig. 24.

The three signals are actually very similar and are featuring growing high frequency oscillations until the trajectory ends. The signals are around the constant value  $0.1 \mu$  with peaks from  $-1.8$  to  $2 \mu$ . At 45 m/min, the peaks are  $\pm 1.4 \mu$ .

Table 4 shows a compilation of results.

All error and deviation values remain below or equal to  $4 \mu$ .



**Fig. 22** Simulated path deviation for a straight line segment: cycle times of 5, 10 et 20 ms, cubic joint interpolation

#### 5.4.2 Arc with Third Order Interpolation

The same simulation process is repeated for a typical arc path by first varying the feed-rate and the results are shown on Fig. 25.

The three signals are actually very similar and are featuring irregular oscillation signals with averages at approximately  $0.5 \mu$  with peaks at  $0,02$  and  $1.85 \mu$ . As the feedrate increases, the deviation signal becomes less dense indicating a reduction of oscillation frequencies.

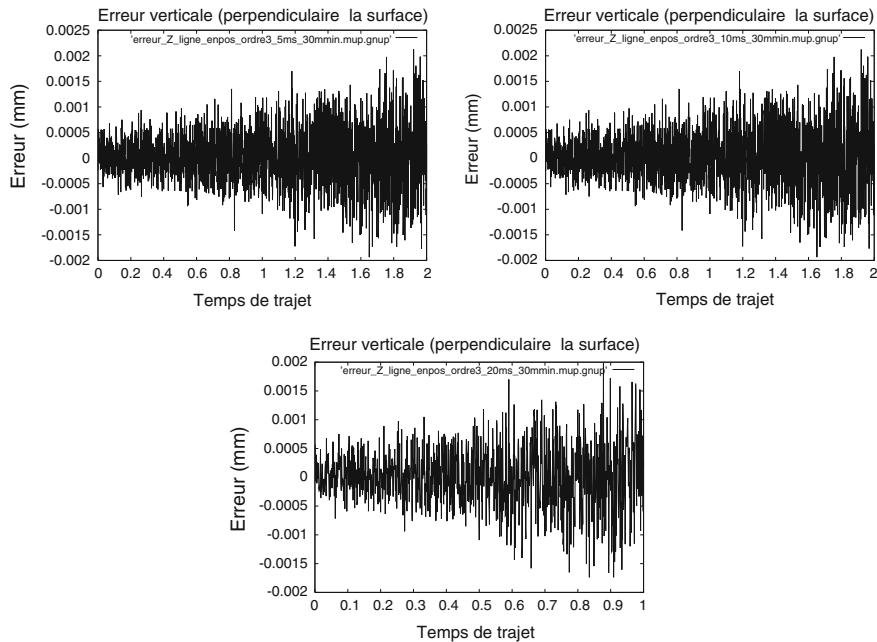
The simulation is repeated by varying the cycle time of the CNC controller and the results are shown in Fig. 26.

The three signals are actually very similar and are featuring irregular oscillation signals with averages at approximately  $0.5 \mu$ . Moreover, the deviation remains below  $2 \mu$  regardless of the case (Fig. 27).

In Fig. 24, the simulation results are shown for the selected feedrates.

The three signals are actually very similar and are featuring irregular oscillation signals with averages at approximately  $0,1 \mu$  with extrema at  $-0.4$  and  $0.6 \mu$  and peaks at  $\pm 0.8 \mu$ .

We then study the vertical error where the controller cycle times are varied, Fig. 28.



**Fig. 23** Simulated vertical error for a straight line segment: cycle times of 5, 10 et 20ms, cubic joint interpolation

As it was observed for the former tests, the vertical error signals are very similar and their density is inversely proportional to the controller cycle time.

Tests ends by collecting the results onto the following Table 5.

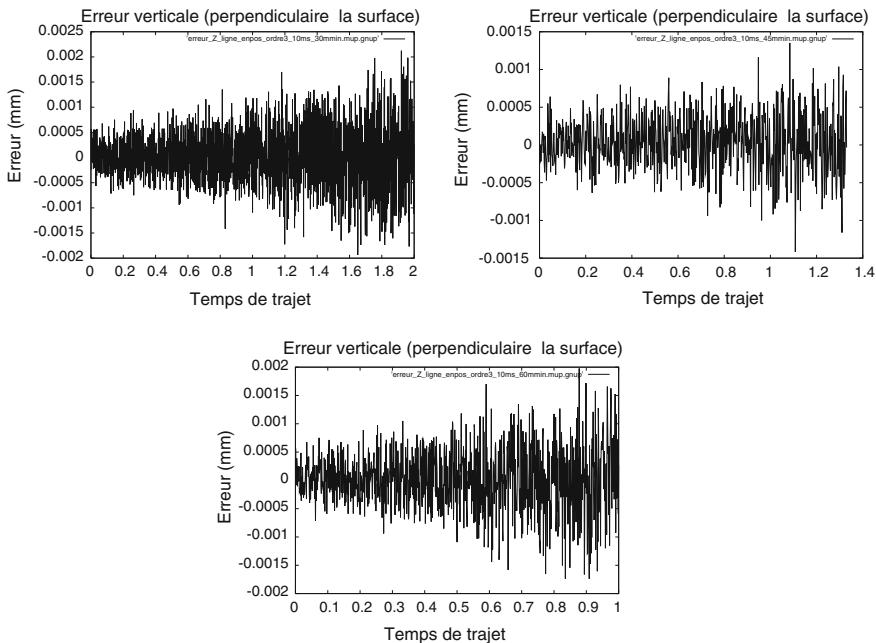
The results barely exceed the value of  $2 \mu$  whatever the speed and response time.

#### 5.4.3 Discussion of the Third Order Joint Interpolation

A trajectory tracking using third order interpolators gives very satisfactory results. In all instances, deviation of less than  $2 \mu$  are obtained.

It is notable that the arc path results are better than for straight line segment. The difference between the error vector and deviation is at most  $0.2 \mu$ . As a consequence, the simulated path is not significantly delayed or ahead of the nominal path. It is observed that the curve is simulated even closer to the theoretical curve for the case of the line segment.

The results of surface finish indicate milling quality within 5 and  $2 \mu$  respectively for the line segments and circular arcs. Indeed, the results of the third order interpolation show that hexapod performance should be sufficient for UHSM (feedrate of 60 m/min or higher). Position control with cubic interpolators are highly recommended.



**Fig. 24** Simulated vertical error for a straight line segment: feedrates of 30, 45 et 60 m/min, cubic joint interpolation

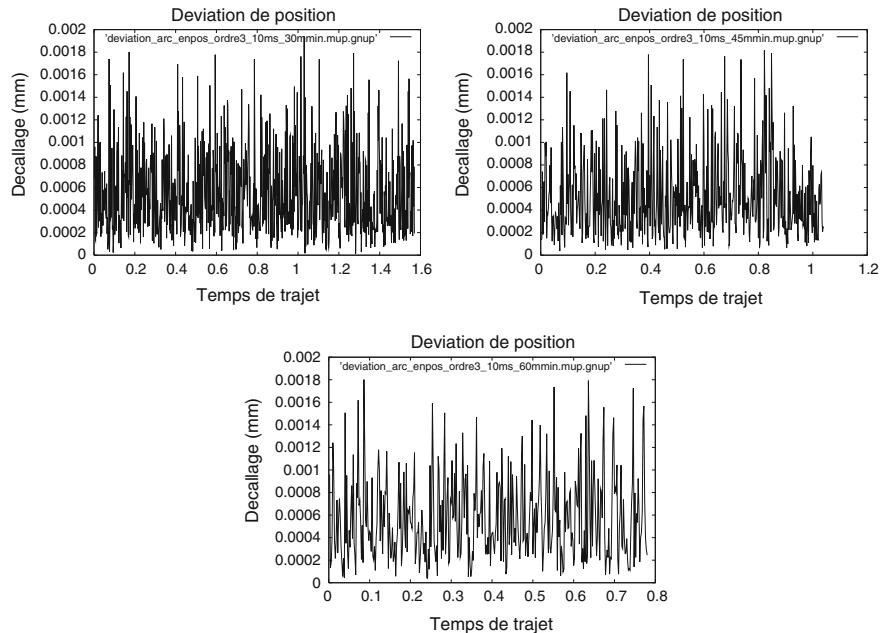
**Table 4** Simulated errors and deviations for a straight line segment: position control with cubic joint interpolation

Order	Trajet ms	$F_r$ m/min	$\varepsilon^{max}$ $\mu$	$\varepsilon^{min}$ $\mu$	$\varepsilon_Z^{max}$ $\mu$	$\varepsilon_Z^{min}$ $\mu$	$  \delta  ^{max}$ $\mu$	$  \delta  ^{min}$ $\mu$
3	10	30	3.787	0.069	2.124	-1.936	3.786	0.021
3	10	45	4.053	0.069	2.351	-2.059	3.285	0.032
3	10	60	3.692	0.069	1.978	-1.738	3.146	0.035
3	5	30	3.787	0.069	2.124	-1.936	3.786	0.021
3	10	30	3.787	0.069	2.124	-1.936	3.786	0.021
3	20	30	3.692	0.069	1.978	-1.738	3.146	0.035

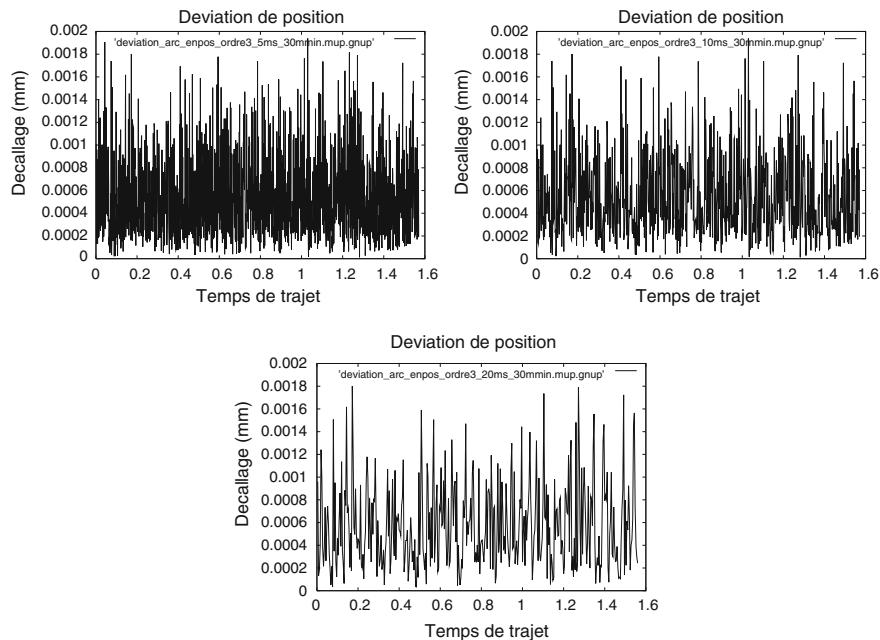
Furthermore, algorithms can be implemented in a conventional CNC adjusted with relatively slow response time.

Among other advantages, the following can be observed:

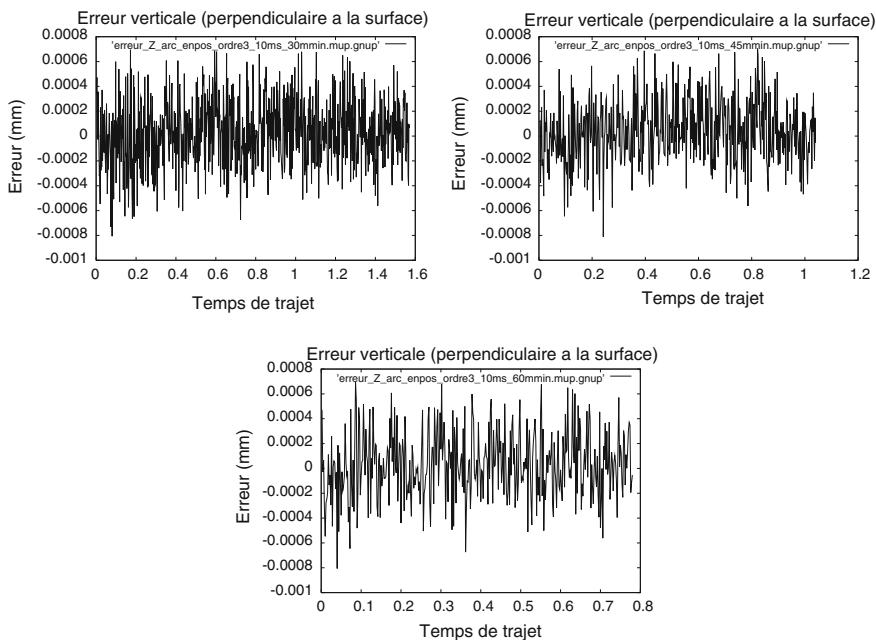
- The relative ease for calculating joint speeds at the beginning and the end of a trajectory interval.
- The continuity of movement is ensured.



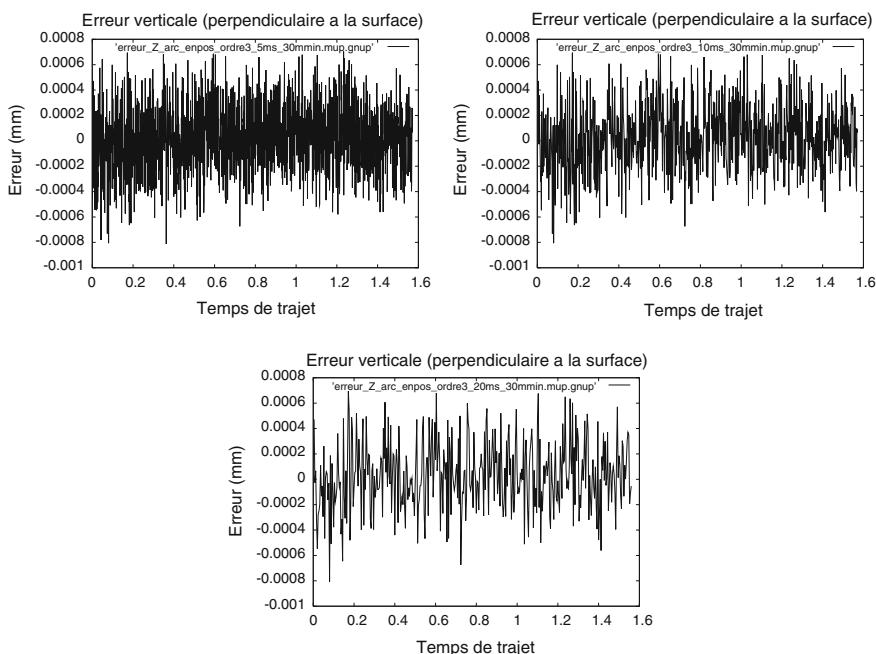
**Fig. 25** Simulated path deviation for an arc: feedrates of 30, 45 et 60 m/min, cubic joint interpolation



**Fig. 26** Simulated path deviation for an arc: cycle times of 5, 10 et 20 ms, cubic joint interpolation



**Fig. 27** Simulated vertical error for an arc: feedrates of 30, 45 et 60 m/min, cubic joint interpolation



**Fig. 28** Simulated vertical error for an arc: cycle times of 5, 10 et 20 ms, cubic joint interpolation

**Table 5** Simulated errors and deviations for an arc: position control with cubic joint interpolation

Order	$T_p$ ms	$F_r$ m/min	$\varepsilon^{max}$ $\mu$	$\varepsilon^{min}$ $\mu$	$\varepsilon_Z^{max}$ $\mu$	$\varepsilon_Z^{min}$ $\mu$	$  \delta  ^{max}$ $\mu$	$  \delta  ^{min}$ $\mu$
3	10	30	2.034	0.042	0.695	-0.806	1.939	0.012
3	10	45	1.991	0.023	0.704	-0.812	1.816	0.037
3	10	60	1.823	0.058	0.695	-0.806	1.802	0.040
3	5	30	2.196	0.023	0.704	-0.812	1.939	0.012
3	10	30	2.034	0.042	0.695	-0.806	1.939	0.012
3	20	30	1.823	0.058	0.695	-0.806	1.802	0.033

The only drawback is that the acceleration continuity will not be ensured. Indeed, nothing prevents large acceleration variations to be applied on the motors.

### 5.5 Discussion on the Results of Position Control

Linear orders are not recommended despite their simplicity because you can not perform high-speed machining. The third order gives the best results because the accuracy is always ensured to remain under 4 and 2  $\mu$  respectively for straight line segments and arcs. Order 5 provides slightly less favorable results and it is more complex to implement.

The implementation of high order interpolators becomes difficult because you have to compute interval transition conditions that are not easy to calculate.

All interpolators allow to follow trajectories with feed-rates up to 30 m/min corresponding to **HSM** at rapid cycle times of 5 ms or less. Note that trying to verify **UHSM** with feedrates up to 60 m/min, the results indicate the application of order three or five. Any case is feasible with third order interpolations.

## 6 Conclusion

The existence of an exact method for solving the **FKP** of the general **6-6** hexapod allows the design of a complete kinematics simulator to study milling processes. A certified calculation method of the robot end-effector position has been implemented in the analysis of milling tasks. It consists of a trajectory following algorithm required for task planning applications, simulation and control. Several modeling modules can simulate various essential elements: parallel manipulator configuration, kinematics modeler and solver, CNC control algorithms, set-point interpolators and performance calculations. For performance evaluation, new metrics were proposed to evaluate surface finish more accurately.

This simulation package provides a kinematics result in the form of the trajectory deviation and vertical error as a lower bound on the estimation of the surface finish of any milling task.

We studied the performance of the classic CNC position control scheme applied to the general **6-6** parallel robots and compared it with an existing hexapod. Modeling of various interpolation strategies at various feedrates and cycle times allowed us to determine that milling quality surface finish can be obtained for **HSM** if third-order interpolations are implemented. We can also implement functions interpolations of the fifth order, but we must implement control cycle time less than or equal to 10 ms but they remain more mathematically involved to prepare. Linear interpolations will not allow for **HSM** and will only be limited to roughing at feedrates slower than 20 m/min.

With position control, **UHSM** becomes only feasible if third order interpolations are established. Results are slightly better for arcs then for straight line segments.

This work has allowed the design and programming of a complete robotic simulation package served as the backbone for the complete high speed milling simulation program prepared as a collaboration of the INRIA in Nancy and Paris VI University to fine-tune general Gough platforms and their position-based CNCs.

**Acknowledgments** This research work was produced by the author during his PhD and with special funding from the Lorraine Region, the INRIA and CMW-Marioni. It has helped French hexapod manufacturers to fine-tune their milling machines.

## References

1. Abdellatif H, Heimann B (2005) Adapted time-optimal trajectory planning for parallel manipulators with full dynamic modelling. In: IEEE international conference on robotics and automation. Barcelona, 19–22 April 2005, pp 413–418
2. Bayaziz OB, Xie D, Anamato NM (2005) Iterative relaxation of constraints: a framework for improving automated motion planning. In: IEEE international conference on robotics and automation. Barcelona, 19–22 April 2005 pp 3433–3440
3. Bhattacharya S, Hatwal H, Ghosh A (1998) Comparison of an exact and an approximate method of singularity avoidance in platform type parallel manipulators. *Mech Mach Theory* 33(7):965–974
4. Bohigas O et al (2012) A singularity-free path planner for closed-chain manipulators. In: IEEE international conference on robotics and automation, Saint Paul, 14–18 May 2012 pp 2128–2134
5. Bohigas O, Manubens M, Ros L (2012) Planning singularity-free force-feasible paths on the stewart platform. In: ARK. Innsbruck, 25–28 June 2012 pp 245–253
6. Brady M et al (1982) Robot motion: planning and control. MIT Press, Cambridge
7. Briot S, Arakelian V (2008) Optimal force generation in parallel manipulators for passing through the singular positions. *Int J Robot Res* 27(2):967–983
8. Carbone G, Gmez-Bravo F, Selvi O (2012) An experimental validation of collision-free trajectories for parallel manipulators. *Mech Based Des Struct Mach* 40(4):414–433
9. Carbone G et al (1997) An optimum path planning for Cassino parallel manipulator by using inverse dynamics. *Robotica* 26(02):229–239

10. Chablat D, Wenger P (1998) Moveability and collision analysis for fully-parallel manipulators. In: 12th RoManSy, Paris, 6–9 July 1998 pp 61–68
11. Chedmail P, Hascoet JY, Guerin F (1994) Collision detection analysis for milling. *Adv Manuf Syst* 1:247–252
12. Chen C-T, Chi H-W (2008) Singularity-free trajectory planning of platform-type parallel manipulators for minimum actuating efforts and reactions. *Robotica* 26(3):371–384
13. Chen C-T, Liao TT (2008) Optimal path programming of the Stewart platform manipulator using the Boltzmann–Hamel-d’Alembert dynamics formulation model. *Adv Robot* 22(6–7):705–730
14. Chen Y, McInroy JE, Yi Y (2003) Optimal, fault-tolerant mappings to achieve secondary goals without compromising primary performance. *IEEE trans robot autom*, vol 19(4). University Park, pp 681–691
15. Coiffet P (1986) Les robots, tome 1, modélisation et commande. Hermès, Paris
16. Corts J (2003) Motion planning algorithms for general closed-chain mechanisms. Ph.D. Thesis, Institut National Polytechnique de Toulouse, Toulouse, 16 December 2003
17. Corts J, Simon T, Laumond J-P (2002) A random loop generator for planning the motions of closed kinematic chains using PRM methods. In: IEEE international conference on robotics and automation. Washington, 11–15 May 2002 pp 2141–2146
18. Corts J, Simon T (2003) Probabilistic motion planning for parallel mechanisms. In: IEEE international conference on robotics and automation Taipei, 14–19 September 2003 pp 4354–4359
19. Dallefrate D et al (2002) A feed rate optimization technique for high-speed CNC machining with parallel manipulators. In: 3rd Chemnitzer Parallelkinematik Seminar, Chemnitz, 23–25 April 2002 pp 371–388
20. Daney D (2000) Etalonnage géométrique des robots parallèles. Ph.D. thesis, Université de Nice-Sophia Antipolis
21. Dasgupta B, Mruhyunjaya TS (1998) Singularity-free path planning for the Stewart platform manipulator. *Mech Mach Theory* 33(6):711–725
22. Dash AK et al (2002) Workspace analysis and singularity-free path planning of parallel manipulators. In: International conference on mechatronics technology (ICMT), Fukuoka, 29 September–3 October 2002 pp 457–462
23. Dash AK et al (2003) Singularity-free path planning of parallel manipulators using clustering algorithm and line geometrie. In: IEEE international conference on robotics and automation, Taipei, 14–19 September 2003 pp 761–766
24. Dash AK et al (2005) Workspace generation and planning singularity-free path for parallel manipulators. *Mech Mach Theory* 40(7):778–805
25. Depince P, Hascoet JY, Furet B (1997) Compensation de trajectoire d’usinage: simulation et experimentation. In: Proceedings of 13e Congrès français de mécanique, vol 3. pp 293–296
26. Dombre E, Khalil W (1999) Modélisation, identification et commande des robots, seconde édition. Robotique. Hermès, traité des nouvelles technologies édition
27. Huang T et al (2007) Time minimum trajectory planning of a 2-DoF translation parallel robot for pick-and-place operations. *Ann CIRP* 56/1/2007:365–368
28. Jui CKK, Sun Q (2003) Path trackability and verification for parallel manipulators. In: IEEE international conference on robotics and automation. Taipei, 14–19 September 2003 pp 4336–4341
29. Khoukhi A, Baron L, Balazinski M (2009) Constrained multi-objective trajectory planning of parallel kinematic machines. *Robot Comput-Integr Manuf* 25(4–5):756–769
30. Lahouar S, Zeghloul S, Romdhane L (2008) Singularity free path planning for parallel robots. Analysis and design: advances in robot kinematics, pp 235–242
31. Latombe JC (1991) Robot motion planning. Kluwer Academic Publisher, Boston
32. Liegeois A (1984) Les robots, tome 7, analyse des performances et CAO. Hermès, Paris
33. Liu G, Trinkle JC, Shvalb N (2006) Motion planning for a class of planar closed-chain manipulators. In: IEEE international conference on robotics and automation, Orlando, 16–18 May 2006 pp 133–138

34. Lozano-Prez T, Wesley M (1979) An algorithm for planning collision-free paths among polyhedral obstacles. In: Communications of ACM, vol 22, pp 560–570
35. Luh JYS, Lin CS (1981) Optimum path planning for mechanical manipulators. Trans ASME 142–151
36. Magnin R, et Urso JP (1991) Commande numerique, programmation. Memotech
37. Marty C, Cassagnes C, La Martin P (1993) pratique de la commande numerique des machines-outils. Technique et documentation. Lavoisier, Paris
38. Masory O, Xiu D (1998) Contour errors in a new class of CNC machine tools. In: Proceedings of WAC98, vol 1, pp 791–798
39. Merlet JP (1993) Manipulateurs paralleles, septieme partie: Verification et planification de trajectoire dans l'espace de travail. Technical Report 1940, INRIA, Sophia-Antipolis, June 1993
40. Merlet J-P (2000) An efficient trajectory verifier for motion planning of parallel machine. In: Parallel kinematic machines international conference, Ann Arbor, 14–15 September 2000
41. Merlet J-P (2001) A generic trajectory verifier for the motion planning of parallel robots. J Mech Des 123(4):510–515
42. Merlet J-P (2007) A local motion planner for closed-loop robots. In: IEEE international conference on intelligent robots and systems (IROS), San Diego, 22–26 September 2007 pp 3088–3093
43. Merlet J-P, Mouly N (1994) Espace de travail et planification de trajectoire des robots paralleles plans. Technical Report 2291, INRIA, Sophia-Antipolis, February 1994
44. Merlet JP, Perng MW, Daney D (2000) Optimal trajectory planning of 5-axis machine-tool based on a 6-axis parallel manipulator. Adv Robot Kinemat 1(1):315–322
45. Mery B (1997) Machines a commande numerique. Hermes, Paris
46. Nenchev DN, Uchiyama M (1996) Singularity-consistent path planning and control of parallel robot motion through instantaneous-self-motion type. In: IEEE international conference on robotics and automation. Minneapolis, 24–26 April 1996 pp 1864–1870
47. Nguyen CC et al (1992) Trajectory planning and control of a Stewart platform-based end-effector with passive compliance for part assembly. J Intell Robot Syst 6(2–3):263–281
48. Nilsson N (1969) A mobile automaton: an application of artificial intelligence. In: Proceedings of the international joint conference on artificial intelligence, pp 509–520
49. Oen K-T, Wang L-CT (2007) Optimal dynamic trajectory planning for linearly actuated platform type parallel manipulators having task space redundant degree of freedom. Mech Mach Theory 42(7):727–750
50. Patel A, Ehmam K (1997) Volumetric error analysis of a Stewart platform based machine tool. In: Annals of the CIRP, vol 46, pp 287–290
51. Pouyan A et al (2010) Eliminating redundancy and singularity in robot path planning based on masking. Expert Syst Appl 37(9):6213–6217
52. Pugazhenthi S, Nagarajan T, Singaperumal M (2002) Optimal trajectory planning for a hexapod machine tool during contour machining. Proceed Inst Mech Eng Part C, J Mech Eng Sci A 216(12):1247–1257
53. Merlet JP (1997) Les Robots Parallel, 2nd edn. Herms, Paris
54. Dieudonne JE, Parrish RV, Bardusch RE (1972) An actuator extension transformation for a motion simulator and an inverse transformation applying Newton-Raphson's method, Technical Report D-7067. NASA, Washington
55. Lazard D (1993) On the representation of rigid-body motions and its application to generalized platform manipulators. J Comput Kinemat 1:175–182
56. Raghavan M (1993) The Stewart platform of general geometry has 40 configurations. ASME J Mech Des 115:277–282
57. Raghavan M, Roth B (1995) Solving polynomial systems for the kinematic analysis and synthesis of mechanisms and robot manipulators. Trans ASME 117:71–79
58. Rolland L (2001) Introduction to algebraic methods for solving the forward kinematics problem of parallel robots applied to high throughput and high accuracy. In: 3rd European-Asian congress on mechatronics, Besancon, 9–11 October 2001

59. Rolland L (2005) Certified solving of the forward kinematics problem with an exact algebraic method for the general parallel manipulator. *Adv Robot* 19(9):995–1025
60. Rolland L (2008) Synthesis on modeling and certified solving of the kinematics problems of Gough-type parallel manipulator with an exact algebraic method. In: Wu H (ed) Parallel manipulators, towards new applications. I-Tech Education and Publishing, Vienna, pp 175–206
61. Salerni G (1995) The linear delta. Technical report, University of Pisa
62. Shulz et al (1999) Dynamic stiffness and contouring accuracy of a HSC linear motor machine. In: Proceedings of the 2nd international conference on high speed machining, vol 1. Darmstadt, pp 75–83
63. Shulz H, Gao H, Stanik B (1999) Analysis and optimization of the dynamic contouring accuracy using the example of a linear motor machine tool. In: Proceedings of the 2nd international conference on high speed machining, vol 1. Darmstadt, pp 107–115
64. Sen S, Dasgupta B, Mallik AK (2003) Variational approach for singularity-path planning of parallel manipulators. *Mech Mach Theory* 38(11):1165–1183
65. Shaw D, Chen Y-S (2001) Cutting path generation of the Stewart platform-based milling machine using an end-mill. *Int J Prod Res* 39(7):1367–1383
66. Soni AH, Tanasi GC, Varanasi S (1995) Closed-loop multi-degree freedom mechanisms for surface generation and patching in machining 3d surfaces. In: 9th IFToMM world congress on the theory of machines and mechanisms. Milan, 30 August–2 September pp 2668–2674
67. Su H-J, Dietmaier P, McCarthy JM (2003) Trajectory planning for constrained parallel manipulators. *ASME J Mech Des* 125(4):709–716
68. Takeda Y (2005) Kinematic analysis of parallel mechanisms at singular points at which a connecting chain has local mobility. In: Computational kinematics, Cassino, 4–6 May 2005
69. Taylor R (1979) Planning and execution of straight line manipulator. *IBM J Res Dev* 23(4):424–436
70. Tchon K et al (2012) Motion planning for parallel robots with non-holonomic joints. In: ARK, Innsbruck, 25–28 June 2012 pp 115–122
71. Tournassoud P (1992) Planification et contrôle en robotique, application aux robots mobiles et manipulateurs. Robotique. Hermes, Paris, Traité des nouvelles technologies edition
72. Trinkle JC, Milgram RJ (2002) Complete path planning for closed kinematic chains with spherical joints. *Int J Robot Res* 21(9):773–789
73. Udupa SM (1977) Collision detection and avoidance in computer controlled manipulators. In: Proceedings of the international joint conference on artificial intelligence, pp 737–748
74. Ur-Rehman R, Caro S, Chablat D, Wenger P (2010) Multi-objective path placement of parallel kinematics machines based on energy consumption, shaking forces and maximum actuator torques: application to the Orthoglide. *Mech Mach Theory* 45(8):1125–1141
75. Vaca R, Aranda J, Thomas F (2012) Simplified Voronoi diagrams for motion planning of quadratically-solvable Gough-Stewart platforms. In: ARK, Innsbruck, 25–28 June 2012 pp 157–164
76. Vaishnav RN, Magrab EB (1987) A general procedure to evaluate robot positioning error. *Int J Robot Res* 6(1):59–74
77. Yakey JH et al (2001) Randomized path planning for linkages with closed kinematic chains. *IEEE Trans Robot Autom* 17(6):951–958

# Planning Automatic Surgical Tasks for a Robot Assistant

**Enrique Bauzano Nuñez, Belen Estebanez Campos,  
Isabel Garcia Morales and Victor F. Muñoz Martinez**

**Abstract** One of the main goals of surgical robotics has always relied on developing a robotized platform to allow the surgeon make an intervention alone, which is also known as the co-worker concept. These robotic systems have evolved over the last years depending on their tasks and interfaces with the surgeon. This evolution led to the teleoperated systems, which have the main drawbacks of a high complexity and economic costs. Many researchers have focused their efforts in minimizing these problems by automating certain actuations on the surgical environment. In this way, this chapter focuses on the design and implementation of a robotic surgical motion controller, which has been designed for performing autonomous tasks to assist the surgeon with an additional instrument. For this purpose, a hierarchical architecture has been implemented which includes an auto-guide velocity planner connected to a force controller. The first one, a trajectory planner based on a behavior approach, is devoted to find a collision-free trajectory of the surgical instrument tip held by the robot, with the final aim of reaching a target location inside of the abdominal cavity. However, the surgical tasks may also require pressing the tissue or stretching the thread for needle suturing. In this way, the force controller grants the exertion of these required forces. The performance of both, the trajectory planner and the force controller, have been tested by means of in vitro trials.

**Keywords** Laparoscopic surgery · Real-time automatic movements · Force feedback control · Surgical robot assistant

## 1 Introduction

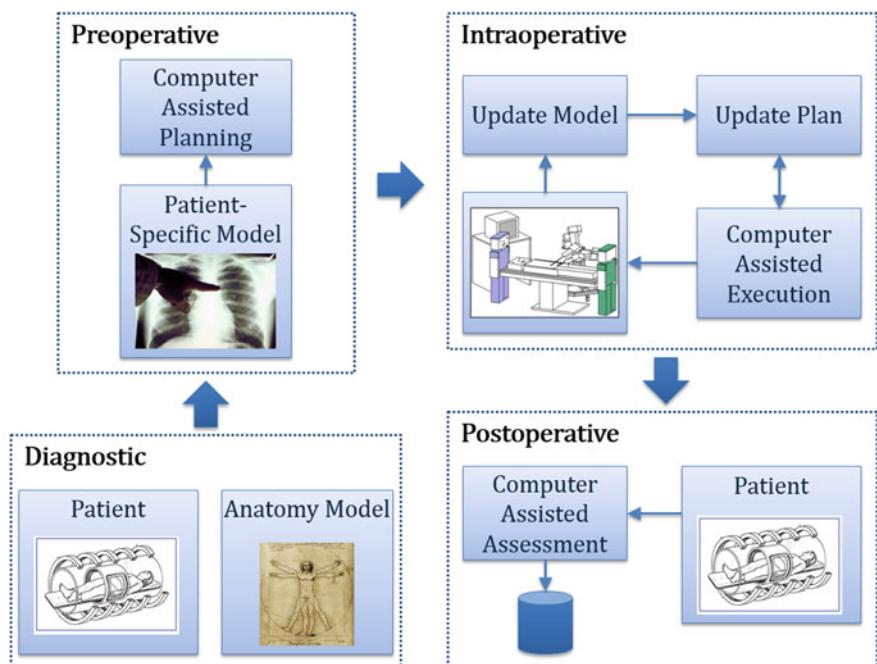
Over the last years, robotic systems have been introduced in several surgical techniques as an additional and very valuable tool for surgeons. Some of their benefits include a higher precision, improved security and freedom on the movement of the

---

E. Bauzano Nuñez (✉) · B. Estebanez Campos · I. Garcia Morales · V.F. Muñoz Martinez  
University of Malaga, Severo Ochoa 4, 29590 Malaga, Spain  
e-mail: ebauzano@uma.es; vfmm@uma.es

surgical instruments. Almost all commercial surgical robots have been designed for two main surgical procedures: orthopedics or neurosurgery interventions, and minimally invasive surgery (MIS) techniques.

The orthopedics procedures are covered by the computer-aided design and manufacturing methodologies (CAD/CAM), which lead to the concept of Computer Integrated Systems (CIS) [1, 2]. The CAD/CAM systems are mainly focused on orthopedic surgery because the work on the bones is very similar to the mechanized on a piece of raw material. Systems based on CAD/CAM aid the surgeon during the standard procedure followed on any intervention as described on the scheme of Fig. 1 [3]. After the medical diagnostic, firstly all the relevant information about the patient is gathered with non-invasive CAD techniques on the preoperative phase. This information of the anatomical model is used for planning the intervention with the use of surgical navigator systems [4, 5]. Once the planning is generated, secondly the intervention is performed during the intraoperative phase with a CAM integrated on the robot and a register procedure [6]. The virtual coordinates obtained from the model of the patient are updated in this phase with the real ones in order to establish a correspondence to guide the surgeon correctly [7]. Finally, the patient recovery is followed on the postoperative phase.



**Fig. 1** Architecture of a surgical CAD/CAM system, where the preoperative phase is CAD and the intra-operative phase is CAM



**Fig. 2** Robodoc on the *left* and SpineAssist on the *right*

Some of the first commercial CAD/CAM robot systems are the Robodoc by Integrated Surgical Systems [8] or Caspar by Ortho Maquet [9], which are both devoted to the hip and knee orthopedic surgery. One of the latest releases, the SpineAssist [10], consists of a specialized robot for prosthetic interventions on the spinal column (see Fig. 2).

The second kind of procedures (MIS) covers the surgical robots specialized in abdominal interventions. MIS has become one of the most important surgical techniques over the last years due to its capability of reducing the postoperative convalescence and diminishing any other complications. As a consequence, MIS offers several social and economic consequences like minimizing the stay of patients at the hospital. As opposite to the orthopedic systems, MIS robots assist the surgeon during the surgical procedure and are not programmed on a previous phase of the intervention.

In general terms, MIS robots are designed to handle instrumentation on laparoscopic procedures with automated movements generated by direct orders of the surgeon or by teleoperation with a master-slave system. Commercial systems for the first group are mainly focused on positioning the laparoscopic camera, for example the AESOP by Computer Motion shown on the left side of Fig. 3 [11], the EndoAssist by Armstrong Healthcare [12] or the Lapman by Medsys [13]. On the other hand, most relevant teloperated systems includes several robotic arms like ZEUS by Computer Motion [14] and Da Vinci by Intuitive Surgical (see right image of Fig. 3) [15]. Both systems have three slave manipulators so one of them handles the laparoscopic



**Fig. 3** The AESOP robot laparoscopic assistant on the *left* and the Da Vinci system on the *right*

camera whereas the others manage the surgical instruments. More specifically, the Da Vinci system has been used on an intercontinental tele-interventions from San Francisco to Boston and from Manhattan to Strasbourg.

There are several automation techniques for MIS robots that have been already covered on the literature [16]. One of them is the *visual servoing*, which consists of automatically guiding the laparoscopic camera. This technique has been used for the safe movements of the endoscope on cardiac surgery [17] or the prediction of the end effector location [18, 19]. Other works are based on the automation of surgical maneuvers without the direct intervention of the surgeon on in vitro experiments, so they are not designed for a collaborative assistance. For example, the recognition of surgical tasks that initially focused on the evaluation of the surgeon's skills [20], has been adapted in some developments to transfer those skills to a robot assistant. Moreover, some of the main surgical tasks have been fully automated like the automatic stitching [21] and knot tying [22] on suture procedures, as well as the grasping and lifting on tissue retraction [23].

In this way, the contents of this chapter focuses on the control movements of the surgical assistants, as well as the most relevant planning methodologies to displace the surgical tools handled by the robot. For this purpose, firstly the main kinematic structures to spherically navigate the laparoscopic instruments are presented in Sect. 2 with their particular problems and the general control schemes for each of them. Secondly, the chapter makes a classification of surgical robots in groups depending on their degree of autonomy in Sect. 3. Each of these groups have a general planning method to generate the appropriate trajectories for the robot instrumentation. One of

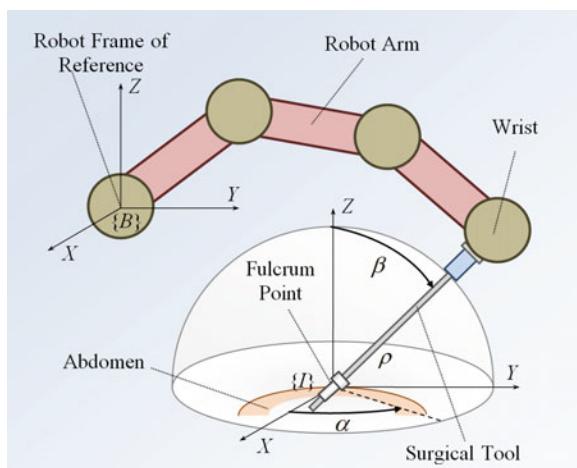
these methods, and a current trend of surgical robotics, consists of the collaborative planning with the surgeon's tasks. Thus, the following Sect. 4 is focused on this kind of robots with the auto-guide planner a special system capable of finding trajectories related to a desired location or force exerted by the robot instrument. Finally, the chapter will present the platform on Sect. 5, where this collaborative system has been implemented and an experimental result to show the behavior of the proposed auto-guide planner.

## 2 Laparoscopic Navigation

All robot assistants require the development of kinematic structures for the robotic arms that handle the surgical instruments. Moreover, these mechanisms must be controlled by specific algorithms for tool movements that compute the trajectories with precision. In this way, Fig. 4 shows a scheme of the laparoscopic navigation, which consists of a controlled movement of the laparoscopic tool in order to locate it at certain coordinates inside the abdominal cavity. It can be seen that the Robot Arm holds the laparoscopic tool and describes spherical trajectories around the place where it is inserted into the Abdomen of the patient (Fulcrum Point). Thus, actual localization of the instrument can be established through the spherical coordinates  $\alpha$  (orientation),  $\beta$  (altitude) and  $\rho$  (external distance from the robot Wrist to the fulcrum). These parameters are related to the fulcrum frame of reference  $\{I\}$  located at the fulcrum point, which is also referred to the base Robot Frame of Reference  $\{B\}$  for planning the movements of the laparoscopic tool.

The main problem of the laparoscopic navigation relies on the uncertainty of the fulcrum point from the robot frame of reference  $\{B\}$  point of view. More specifically,

**Fig. 4** Navigation of the surgical tool



it is quite difficult to locate the patient over the operating table and perform the incision on its abdominal wall, in such a way that the location of  $\{I\}$  related to  $\{B\}$  is known with accuracy. This uncertainty leads to an imprecise positioning of the instrument at the specified spherical location. From a kinematics point of view, this problem can be solved with the exact knowledge of the three spherical coordinates  $\alpha$ ,  $\beta$  and  $\rho$  where the instrument is currently located respect to frame  $\{I\}$ . However, while coordinates  $\alpha$  and  $\beta$  can be obtained from the kinematics model of the robot arm and the measure of its inner sensors, it is not possible to obtain the distance  $\rho$  without the use of external 3D trackers.

Following subsections state the current commercial solutions which are able to solve this problem of the spherical laparoscopic navigation through different approaches: the actuated and passive wrists.

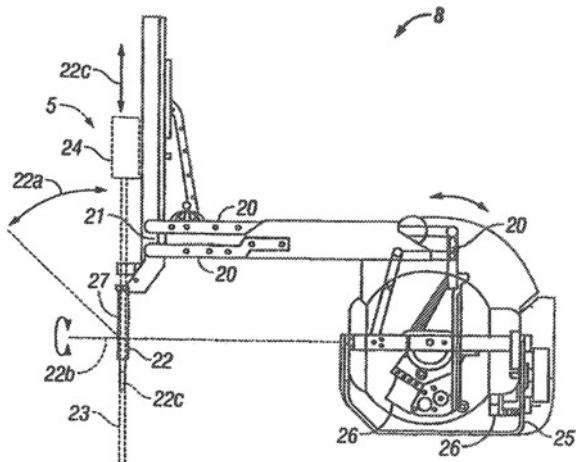
## ***2.1 Remote Center of Rotation***

This kind of wrists has its degrees of freedom directly actuated by motors, and thus the position and orientation of the laparoscopic tool can be established with the corresponding joint vector. These wrists avoid the inconvenience of the location accuracy regard to special mechanical structures. These devices are designed to rotate the tool around a remote center of rotation that must be calibrated accordingly. With this system, the precision of the spherical movements is guaranteed as long as the fulcrum point remains at the same location. Figure 5 shows on the up side the four-link scheme used on the Da Vinci robotic assistant, whereas the down side presents the implantation of this wrist as it is described on the corresponding patent [24]. The displacement of the four-link system labeled as 20 orientates the tool at 22a that turns around point 22 where the fulcrum shall be located. The main problem of these kind of wrists relies on the precise localization of the fulcrum at the remote center of the wrist, because otherwise undesired forces can be exerted over the abdominal wall of the patient.

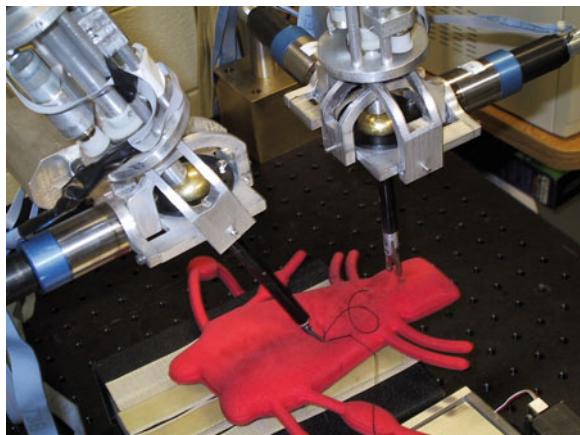
There are other schemes for remote center of rotation systems based on four-link mechanisms, for example the Black Falcon [25], the Blue Dragon [26] or the robotic extender configuration [27]. Other works use semicircular elements for this purpose as the Endobot system, shown in Fig. 6, which is composed by two robotic arms [28].

This wrist configuration is commonly used on master-slave systems due to the precision on the tools movements. Thus, control schemes that manage these mechanisms are based on the control of the Cartesian velocity of the laparoscopic tool as it is expressed on Fig. 7. It can be seen that the movement order obtained from a master device is computed by a planner which sends the control inputs for the robot and the remote center of rotation wrist. The feedback signal is the inverse Jacobian of the corresponding wrist used to control the velocities of the elements of its structure.

**Fig. 5** Remote center of rotation scheme used on Da Vinci robotic assistant

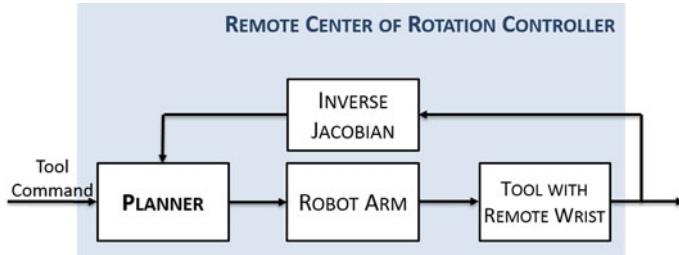


**Fig. 6** Two-arm Endobot system with remote center of rotation wrists

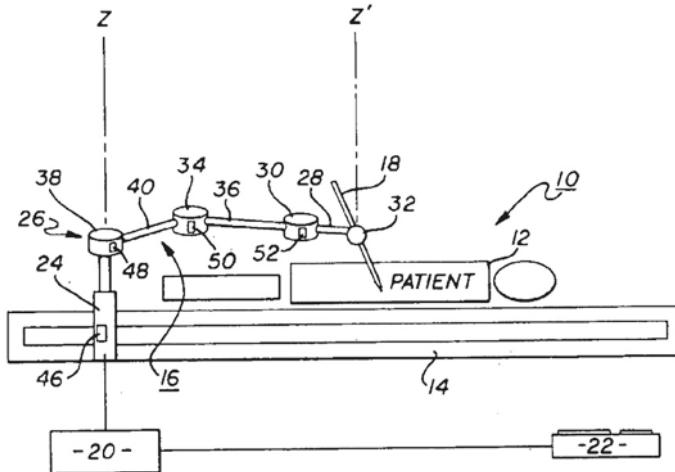


## 2.2 Passive Wrists

A robotic assistant based on a passive wrist has one or more non-actuated degrees of freedom. Although it is possible to establish the Cartesian position of the tool, its orientation depends on the relative position between the wrist center of rotation and the fulcrum point. This configuration guarantees that no forces are exerted over the fulcrum point, since its mechanical structures allows the accommodation of the surgical instrument. However, any uncertainty on the relative position between the assistant and the fulcrum point reduces the precision on the robot tool tip positioning. The problem is that the insertion cannot be computed with the inner sensors of the robot arm, and must be estimated with the use of geometrical techniques [29]. Passive wrist can be found on systems like AESOP [11] or ZEUS [14].



**Fig. 7** Control basic scheme for remote center of rotation wrists

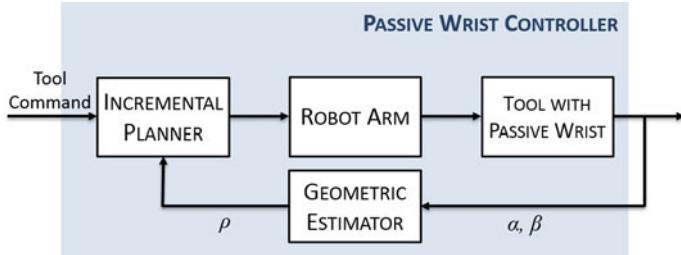


**Fig. 8** Endobot robot assistant with a passive wrist

Figure 8 shows the scheme of AESOP system as it is described on its patent [30]. The arm joints labeled as 24, 38 and 34 are actuated. However, the two degrees of freedom of the wrist (30 and 32) are not actuated.

Robotic assistants with passive wrists are mainly used for the management of the laparoscopic tool, which requires less precision on its localization than a standard instrument used for interacting with the patient's organs. Thus, a lack of precision on the location of the laparoscopic tool just leads to an error when targeting the region of interest, which is a non-critical task. The control loop was closed by the own surgeon during the first versions of AESOP. In this way, the surgeon commanded the robot to make an increment of the rotation or altitude angle for the camera until the image reached the anatomical region of interest. However, to achieve a more precise positioning, it is necessary to estimate the coordinate  $\rho$  during the movement with geometrical techniques.

Figure 9 proposes the control scheme for robot assistants with passive wrists. The planner transforms the movement order of the surgeon to a Cartesian position of the

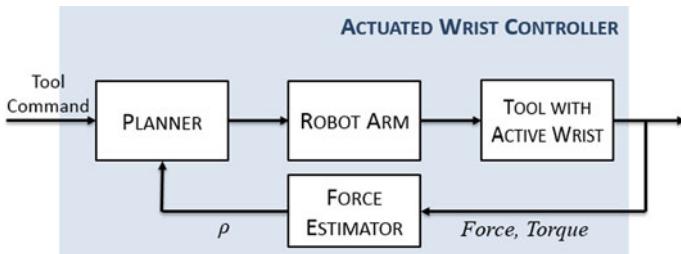


**Fig. 9** Control scheme for passive wrists

rotation center of the wrist, whereas the orientations  $\alpha$  and  $\beta$  are computed with the inner sensors of the robot. For each measure of these angles, the geometric estimator computes the axis of the laparoscopic tool. In this way, the estimated axes obtained each two consecutive positions of the tool intersect in one point that can be used for the estimation of the distance  $\rho$ . With the use of this scheme, the estimation of distance  $\rho$  can be improved over the movement, but it does not include any method to solve a positioning error in case the system does not achieve a realist estimation of the  $\rho$  parameter.

### 2.3 Actuated Wrists

The third category of wrists does not use any kind of additional mechanical structure, but it assembles the surgical instrument right to the end effector of the manipulator. However, these wrists require a force feedback controller like shown in Fig. 10 for computing the insertion of the instrument and positioning it into the abdominal cavity with accuracy [31, 32]. The geometric estimator of the passive wrists is replaced by a Force Estimator that obtains the external distance estimation from the force and torque measurements.



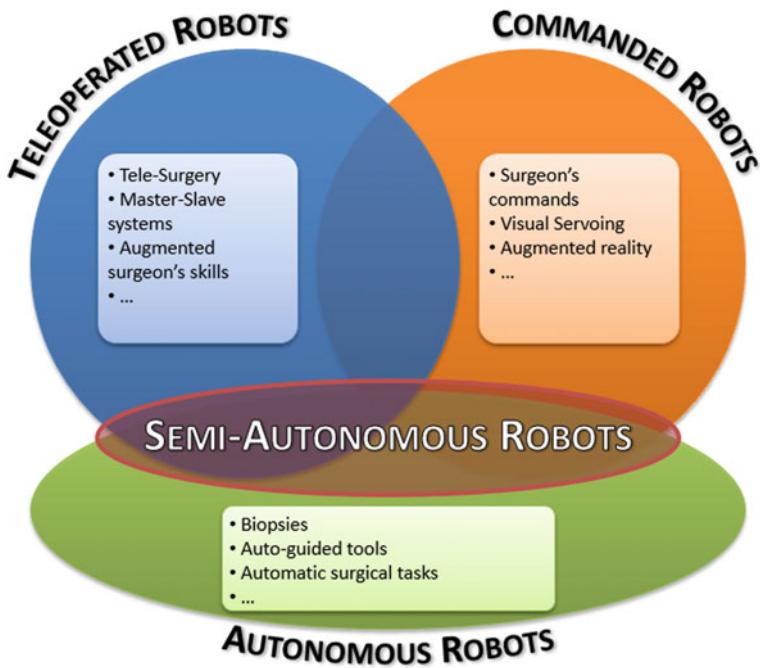
**Fig. 10** Control scheme for direct actuated wrists

This methodology presents two main problems. On one hand, the forces exerted over the tool tip must be separated from those applied by the abdominal wall. On the other hand, laparoscopic tools are used to be flexible, so their deformation affects the positioning accuracy. These problems are usually solved by integrating a force sensor on the trocar, as well as by using special rigid instruments.

### 3 Planning Methods on Laparoscopic Surgery

All the kinematic structures commented on previous section require a planner element to navigate the laparoscopic tool. The trajectory generated by this planner mainly depends on the degree of automation for the robot to complete its tasks. In this way, Fig. 11 establishes a classification of the robot assistants attending to their degree of automation.

The right side presents the commanded robots, usually cameramen assistants specialized on controlling the laparoscopic camera movements. Such movements can be directly managed by the surgeon through a communication interface, as well as automatically performed for the tracking of the surgeon's tools or a point of interest.



**Fig. 11** Classification of surgical robots depending on their autonomy degree

The upper-left side of the graph shows the teleoperated robots, which directly replicate the surgeon's movements on the surgical tools managed by the robot arms. These systems may also improve surgeon's skills, for example by suppressing the constraints on the movements imposed by the laparoscopic surgery (fulcrum point, additional degrees of freedom on the tool tip...).

The down side of the diagram represents the group of robot assistants that can work without any intervention on the surgeon's hand. Autonomous robots are systems capable of make biopsies or the automatic guidance of a surgical tool on the region of interest. Meanwhile, the surgeon focuses on the main tasks of the intervention while verifying that the overall procedure is followed as expected.

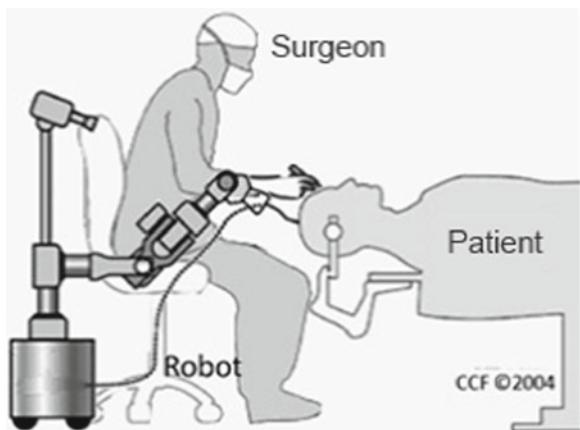
The intersection of these groups of robot assistants represent the semi-autonomous robots. This subgroup consists of those systems that have certain degree of autonomy, in such a way that they always collaborate with the surgeon's tasks. These robots are very versatile, since they combine different capabilities as the tracking of the laparoscopic camera in combination with the management of a surgical instrument for assisting the surgeon in a collaborative way.

Firstly, the following Sects. 3.1–3.3 describe different planning strategies for each of the groups of robot assistants explained with Fig. 11: commanded, teleoperation and autonomous planning. After this explanation, Sect. 4 introduces a planning strategy proposed by the authors for a more advanced semi-autonomous robot assistant.

### 3.1 Commanded Planning

One of the challenges of surgical robot focuses on the substitution of the human cameraman on the MIS procedures as represented on Fig. 12. This surgical techniques require that the assistant centers the camera on the region of interest where the surgeon

**Fig. 12** Surgical robot working in commanded mode with the surgeon



makes the intervention. Fatigue or stress of this assistant may affect the image quality, in such a way that the endoscope may touch tissue, center a wrong area or the image is unstable due to the trembling of his hand.

The use of arm manipulators for managing the endoscope avoids the problems stated above, and provide more accurate movements and a steady hand along all the intervention. These robots need a communication interface with the surgeon to perform the corresponding orders accordingly. Most elemental systems use a joystick, others use a gyroscope attached to the surgeon's head in order to drive the image with head's movements [33], or recognize voice commands [34]. There are also works where the robot movements are guided by interpreting the head's movements with vision algorithms [35], and others with a remote controller attached to the surgeon's hand [13].

The first laparoscopic camera positioners were the ones based on the classic works of R.H. Taylor and J. Funda with their LARS [36] and HISAR [37] systems, respectively. Other commercial robot assistants followed these ones like the already mentioned AESOP [11], ENDOASSIST [12] or LAPMAN [13]. These systems are considered the first step on the co-worker concept, where the surgeon may perform a full intervention without any human assistance. In this way, the University of Malaga developed the ERM system, a laparoscopic camera positioner commanded by the surgeon's voice or directly with a joystick, as shown on Fig. 13 [38]. This robot uses a passive wrist for navigating the laparoscopic tool, and as additional features it is able to work without wires and has wheels to place it wherever the surgeon wants.

### ***3.2 Tele-Operation Planning***

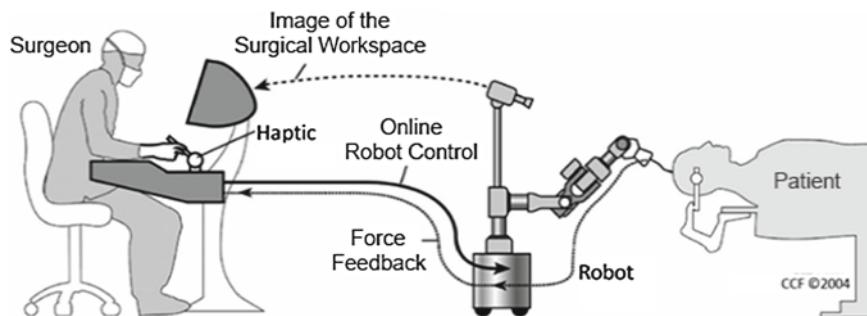
On many surgical procedures the assistant does not only locate the endoscope, but also handles other instruments. In this way, robot assistants would require a more sophisticated systems that manage specific tools in addition to the laparoscopic camera. This feature can be considered as another step towards the co-worker concept.

One of the most addressed focuses on the scientific literature for achieving this co-worker system consists of the use of teleoperation and telepresence techniques. Figure 14 shows the fundamentals of a teleoperation station, where the surgeon remotely controls the surgical instruments attached to the arm manipulators.

Thus, several works appeared on the middle nineties which defined the limitations of the movement controller systems to teleoperate surgical instruments into a restricted environment as the abdominal cavity [39]. Moreover, some developers started the design of special mechanical structures to handle the instruments which also allowed the planning of accurate trajectories [27]. Some telepresence features were added, for example the force feedback through haptic devices that provided some kind of tactile sensing to the surgeon [25, 29, 40], and even to distinguish among different tissue textures [41]. There are also teleoperated systems that improve the surgeon's skills in order to perform sub-millimeter tasks [42, 43].

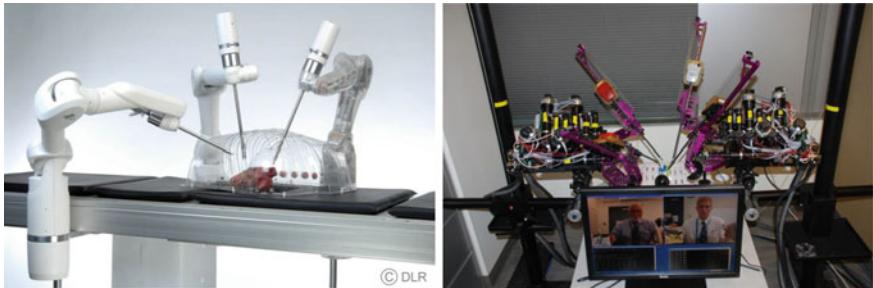


**Fig. 13** ERM system for holding the laparoscopic camera (*left*) and its user interface based on the surgeon's voice or joystick device (*right*)



**Fig. 14** Surgical robot remotely teleoperated by the surgeon

There are more complex teleoperated systems with two arm manipulators to handle surgical instruments and an additional one to manage a stereo vision system [36]. The most relevant commercial system of this class is the Da Vinci platform [15],



**Fig. 15** The MiroSurge by DLR on the *left* and RAVEN-II by the University of Washington on the *right*

which has been already commented in the introduction and shown on Fig. 3 and has been successfully tested with different MIS techniques of several hospitals around the world.

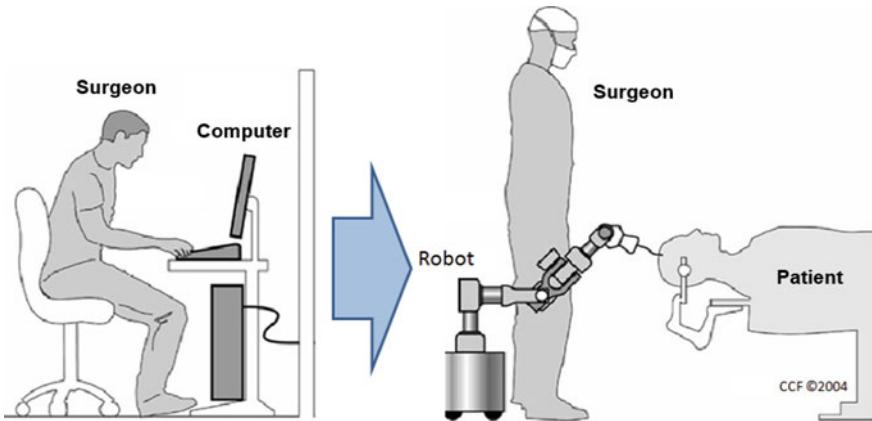
Although Da Vinci is the most known and used teleoperated system, there are other robots of this kind currently in development. Two examples of these works on this field are MiroSurge from the DLR [44] and the Raven-II from the University of Washington [45], which can be shown in Fig. 15.

### 3.3 Autonomous Planning

The enormous complexity and costs of the teleoperated systems limit their clinic impact. A simpler system designed to make more specific tasks, without special installations and no additional training can be more reliable for improving certain laparoscopic procedures. In this way, there are several lines of research which propose systems with two robotic arms, one to handle the endoscope and the other for an extra instrument [46]. As an illustrative scenario, Fig. 16 shows how the surgeon firstly programs the procedure on the computer. This information is used during the intervention for the robot to run the corresponding sequence of tasks that completes the planned procedure.

Despite these efforts and their proven accuracy, some authors still defend that the use of robots in surgery may lengthen the intervention time [47]. Thus, other alternatives must be proposed in order to increase the reliability of robotic assistants. One solution consists of the automation of certain surgical tasks. In this way, there are works that automate the movement of the camera [48] depending on the current state of the intervention. Others are based on dividing complex tasks like the suture in more elemental actions feasible by the robot assistant [28].

One of the most commons techniques for performing automatic tasks is the visual servoing, which consists of dynamically following with the endoscope a surgeon's tool or a target marked on the tissue [49]. The control movement of the surgical

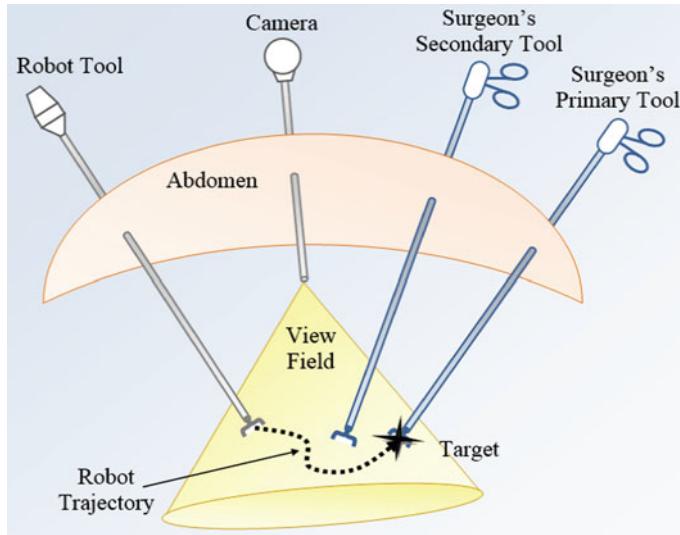


**Fig. 16** Surgeon plans the intervention on the *left*, and then the robot performs the autonomous tasks on the *right*

instruments implies the computation of their linear and angular velocity references at each time period. These references are obtained regard to the analysis of the images received by a calibrated stereo vision system [50], or also with the image acquisition through a traditional laparoscopic camera [51]. This last situation requires the use of a modified instrument that emits a laser in order to measure the distance to the closest organ. This technique allows a safe movement of the laparoscopic tool towards a target location by following the surface of an organ at a certain distance. A variant of this work uses the robotic arms for cardiac surgery. More specifically, with a visual guidance the instruments movements can be synchronized with the heart beats [52], in such a way that the organ remains static from the surgeon's point of view and can make teleoperated cardiac procedures that otherwise would be very difficult. Other works introduce the use of ecographs as feedback [53]. Therefore, despite the increase of automation on robot assistants, they may complement the capabilities of the teleoperated systems.

#### 4 Collaborative Planning: Auto-Guided Movements

The environment where the robot interacts with the patient as well as the surgeon consists of a closed space, the abdominal cavity, as it is shown in Fig. 17. Both tools handled by the surgeon, the Camera and the instrument of the Robot Tool, are inserted through their respective fulcrum points over the abdomen. Moreover this environment also includes the surgeon's tools used for surgery procedures. The Surgeon's Primary Tool is considered the target for the robot, whereas the Surgeon's Secondary Tool is the obstacle. The surgeon is able to displace his tools during any robot movement.



**Fig. 17** The camera focuses over the surgical workspace, whereas the robot tool goes where the surgeon's target tool is located

The abdominal cavity where the robot instrument may move is defined by a cone-shaped view field, which contains the scene seen in the screen by the surgeon. This work considers the displacement of the robot tool to the surgeon's primary tool location as the proposed auto-guided movement. The secondary tool, as well as the tissue and organs inside the abdominal cavity focused by the camera, are defined as obstacles to be avoided during an automatic task. The trajectory has to be computed on-line, because the surgeon's tools are continuously being displaced during the intervention.

There are two main collaborative actions that the robot should develop in order to interact with the surgeon:

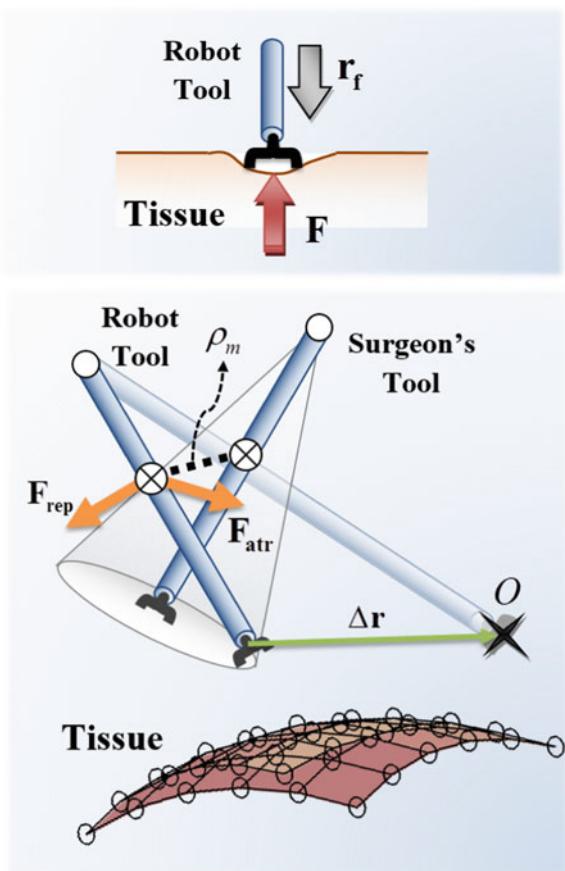
**Pressure over the tissue** (top of Fig. 18). The goal of this actuation consists of exerting a force  $\mathbf{F}$  by means of a movement along the direction defined by the vector  $\mathbf{r}_f$ .

**Navigation to a target location** (bottom of Fig. 18). The purpose of this actuation relies on reaching the target location  $O$  with the Robot Tool tip while it avoids the contact with the Surgeon's Tool and the Tissue on the inner abdominal cavity.

The motion planner has been designed as shown in Fig. 19 to manage these actuations. The required location  $\mathbf{r}_c$  for the Robot is computed by means of a hybrid force-position controller, where  $\mathbf{r}_p$  and  $\mathbf{r}_f$  are the respective contributions from the position and force control.

The position and force controllers receive as inputs a desired location  $\mathbf{r}_d$  and force exerted  $\mathbf{F}_d$ . The position and force vectors  $\mathbf{r}_p$  and  $\mathbf{r}_f$  are expressed in a reference frame attached to the endoscope tip. The hybrid position-force controller of Fig. 19 includes

**Fig. 18** Possible robot actions for a collaborative planning: pressure over tissue (*top*) and navigation to target location (*bottom*)



a force-accommodative controller which feedbacks the robot-tissue interaction force  $\mathbf{F}$  and outputs the vector force  $\mathbf{F}_r$ , which is substracted to  $\mathbf{F}_d$  afterwards and transformed to  $\mathbf{r}_f$  by means of a gain  $K$ , which models the robot-environment relative stiffness (1) with:

$$\mathbf{r}_f = K^{-1} (\mathbf{F}_d - \mathbf{F}_r) \quad (1)$$

where the stiffness  $K$  is adjusted by in vitro experiments with similar materials to the abdominal cavity tissues.

Secondly, the position control consists of an Auto-Guide Planner which focuses on finding a free-obstacle trajectory towards  $\mathbf{r}_d$  inside the abdominal cavity. For this purpose, this element has been designed as shown in Fig. 20 to combine the behavior of three modules: the *Local Planner*, which computes the trajectory towards the target location by using the Artificial Potential Fields (APF) algorithm; the *Velocity Correction*, which adapts the robot tool velocity depending on the trajectory of the

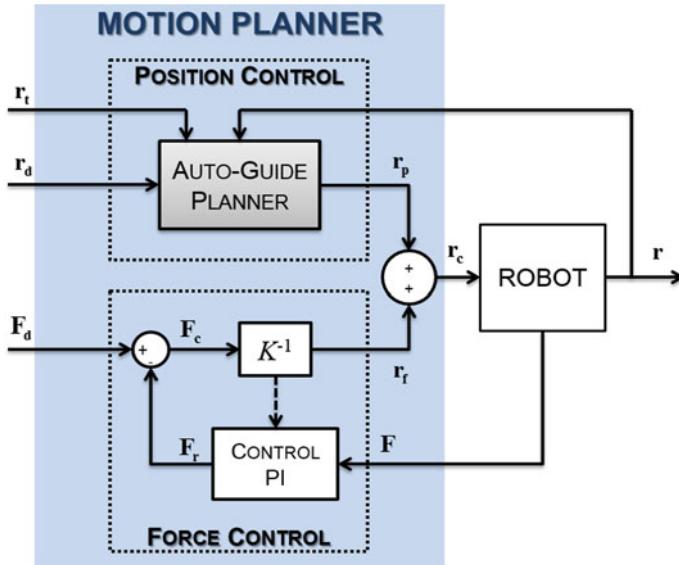


Fig. 19 Motion planner system for robot autonomous actuations

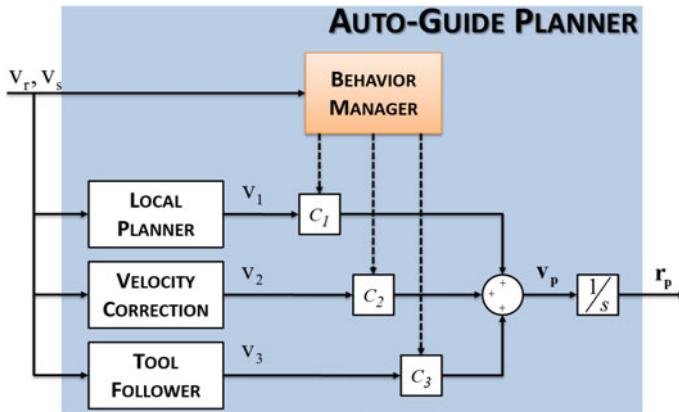


Fig. 20 The proposed auto-guide planner system for finding free-obstacle paths

surgeon's tool; and the *Tool Follower*, which considers an intentional contact of the surgeon's tool to modify the robot tool location.

The outputs of these behaviors  $v_1$ ,  $v_2$ ,  $v_3$  are weighted by their respective gains  $c_1$ ,  $c_2$ ,  $c_3$ , which are computed by the *Behavior Manager*. This element receives both vectors of the tools velocities: the robot  $v_r$  and the surgeon  $v_s$ . The weights  $c_1$ ,  $c_2$  and  $c_3$  are processed by a fuzzy logic algorithm. Thus, the contribution of the Position Control  $r_p$  already described in Fig. 18 is obtained by integrating the planned velocity  $v_p$  (2):

**Table 1** Fuzzy rules of the behavior manager

Robot velocity	Surgeon's velocity							
	1	2	3	4	5	6	7	8
1	A	MA	FA	MA	MN	MA	FA	MA
2	MA	A	MA	FA	MA	MN	MA	FA
3	MN	MC	A	MA	MN	MC	MN	MA
4	MC	MN	MA	A	MC	MN	MA	MN
5	MN	MA	FA	MA	A	MA	FA	MA
6	MA	MN	MA	FA	MA	A	MA	FA
7	MN	MC	MN	MA	MN	MC	A	MA
8	MC	MN	MA	MN	MC	MN	MA	A

FA moving far away, MA moving away, A advancing, MN moving nearby, MC moving closer

$$\mathbf{r}_p = \int_{(k-1)T}^{kT} \mathbf{v}_p dt = \int_{(k-1)T}^{kT} (c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 + c_3 \mathbf{v}_3) dt \quad (2)$$

The fuzzy-logic algorithm for the *Behavior Manager* uses a Mandani fuzzification and a center of gravity defuzzification. The velocities  $\mathbf{v}_r$  and  $\mathbf{v}_s$  are the antecedent data, which are classified by dividing the space into eight quadrants around the reference frame of the camera. Firstly, Table 1 computes the quadrant location of velocities  $\mathbf{v}_r$  and  $\mathbf{v}_s$  to establish the relative direction of the movement between both tools.

On the other hand, Table 2 represents the membership functions of each weight  $c_1, c_2, c_3$ . For example, if the surgical tools are Moving Away, then the Local Planner behavior has a very high relevance ( $c_1$ ), there is almost no need of a velocity correction ( $c_2$ ) and a null effect for the tool follower ( $c_3$ ).

Once the behavior manager processes the weights  $c_1, c_2, c_3$ , expression (2) requires the velocity vectors  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$  of each behavior module shown on Fig. 19. First behavior is the **Local Planner**, which is devoted to find free-obstacle trajectories and has been designed with an APF algorithm. The APF associates a virtual repulsion field to each obstacle and a virtual attraction field to the target location. The obstacles considered in this work consist of the surgeon's tool and the organs/tissues inside the abdominal cavity. The surface of the inner tissue is represented by a grid of  $M - 1$  vertices generated through a monocular SLAM technique with reallocation

**Table 2** Output weights of the behavior manager

	$c_1$	$c_2$	$c_3$
Far away	Very high	Very low	Null
Away	Very high	Very low	Null
Advancing	High	Low	Very low
Nearby	Medium	Very high	Very low
Closer	Very low	Medium	Very high

for laparoscopic sequences [54]. This work has considered the virtual forces vectors of attraction  $\mathbf{F}^{\text{att}}$  and repulsion  $\mathbf{F}^{\text{rep}}$  suggested by [55], because the extra unitary vector  $\mathbf{n}$  on  $\mathbf{F}^{\text{rep}}$  prevents the trajectory to find local minima position (3):

$$\begin{aligned}\mathbf{F}^{\text{att}} &= 2K^{\text{att}}\Delta\mathbf{r} \\ \mathbf{F}_m^{\text{rep}} &= \begin{cases} K^{\text{rep}} \left( \frac{1}{\rho_m} - \frac{1}{\rho_m^0} \right) \left( \frac{\Delta r_k \boldsymbol{\rho}_m}{\rho_m^2} + \frac{\mathbf{n}_m}{2} \right) & \rho_m \leq \rho_m^0 \\ 0 & \rho_m > \rho_m^0 \end{cases} \quad (3)\end{aligned}$$

The gains  $K^{\text{att}}, K^{\text{rep}}$  on (3) represent the relevance of that virtual force relative to the others. Parameter  $\rho_m$  defines the minimal distance between the robot tool and that  $m$ th obstacle,  $\boldsymbol{\rho}_m$  is the unitary vector with that direction and  $\rho_m^0$  is the maximum distance where the virtual field may affect the robot trajectory (see Fig. 18). The distance between the robot's tool tip and the target location is denoted by  $\Delta\mathbf{r}$ , and modifies the contribution of the repulse force nearby the target location. The velocity  $\mathbf{v}_1$  computed by the Local Planner on (4) is the integration of the total virtual force, where virtual mass has been considered to be one.

$$\mathbf{v}_1 = \int \mathbf{F}^{\text{att}} + \sum_{m=1}^M \mathbf{F}_m^{\text{rep}} dt \quad (4)$$

Second behavior is the **Velocity Correction**. This module adjusts the current velocity of the robot tool  $\mathbf{v}_r$  depending on the relative minimal distance  $\rho_m$  and the expected time of collision with the surgeon's tool. Those parameters are used as input variables of a fuzzy-logic algorithm similar to the one proposed in [56]. The consequent parameter denoted as  $K_2$  is limited a value between [0, 1] to guarantee the reduction of the robot tool velocity, thus the velocity  $\mathbf{v}_2$  computed by the velocity correction is (5):

$$\mathbf{v}_2 = K_2 \mathbf{v}_r \quad (5)$$

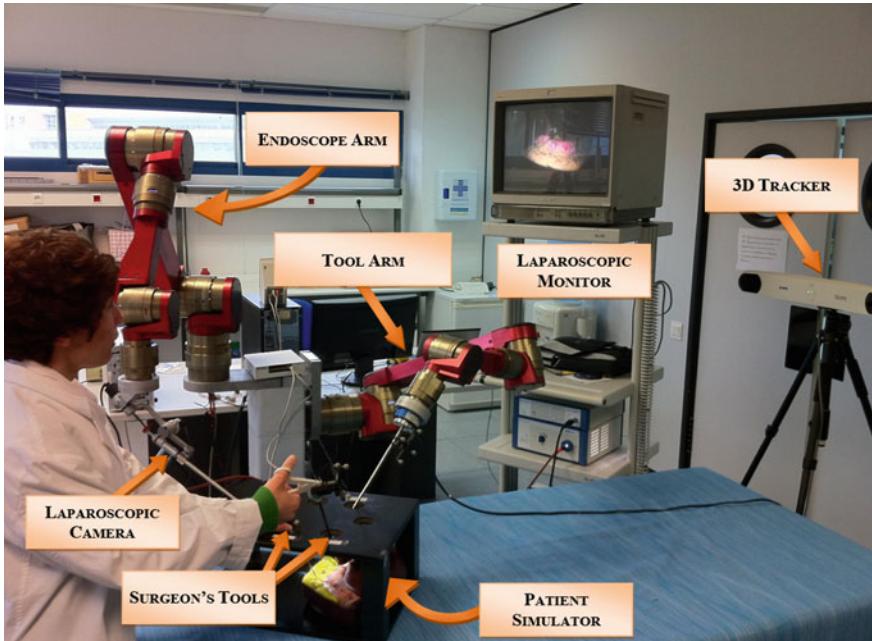
The third and last behavior is the **Tool Follower**. This module considers those situations where the surgeon wants to displace the robot tool by using his own tool (for example, if the robot tool is in front of the camera or blocks the surgeon's tool to perform a surgical task). To solve these problems, the tool follower module models the virtual collision between the surgeon and robot tools with a spring bumper. Whenever both tools are very close (that decision is made by the Behavior Manager, as it has been previously explained), this spring bumper exerts a virtual force of repulsion to the robot tool which can be integrated and leads to the velocity computed by the tool follower (6):

$$\mathbf{v}_3 = \int B_3 \mathbf{v}_s + K_3 (\boldsymbol{\rho} - \boldsymbol{\rho}_e) dt \quad (6)$$

where  $B_3$  and  $K_3$  are the constants for the spring bumper modeled, and  $\rho_e$  is the minimal distance of equilibrium between the surgeon and the robot tools.

## 5 Case of Study: CISOBOT Platform

This section is devoted to describe the experiments and the results for the auto-guided system evaluation. For this purpose, it has been used the CISOBOT system, designed and developed in the University of Malaga (see Fig. 21). This system consists of a two-arm robotic system for holding both, an endoscope and a surgical instrument. On the left side of this picture, it is showed the *Endoscope Arm* equipped with a non-actuated two degrees of freedom wrist for endoscope movements [38]. On the other hand, the *Tool Arm* (on the right side of the endoscope arm) has an actuated wrist with a force sensor in order to feedback information about the forces exerted in the abdominal wall [32]. In this way, the *Endoscope Arm* is commanded by the surgeon's voice for performing left, right, up, down, inside and outside basic endoscope movements, and the *Robot Tool* held by the *Tool Arm* is controlled by means of the auto-guided planning proposed on Sect. 4. Therefore, the *Tool Arm* will accomplish the experiments described in the following subsections.



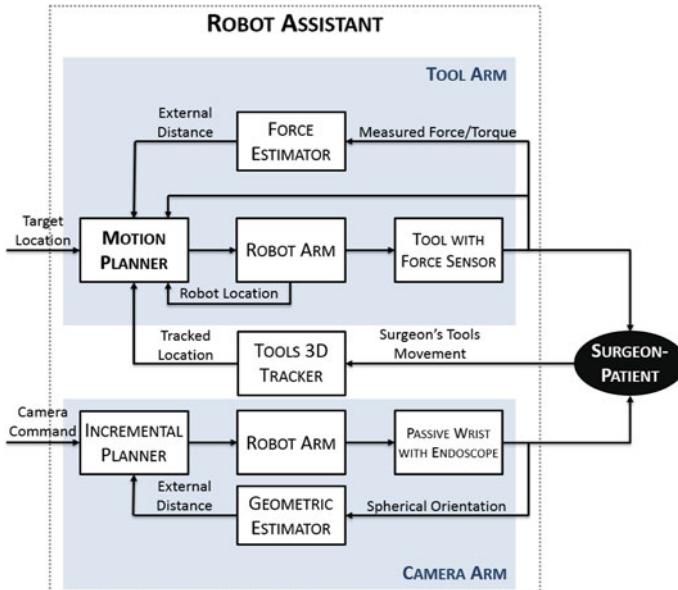
**Fig. 21** Experimental setup for the auto-guided system method proposed with CISOBOT platform

In order to complete the experimental set-up, an *Optical 3D Tracker* (right side on the picture) gives information on the location thanks to the passive marks attached over the *Surgeon's Tool*. On the auto-guiding experiment, the surgeon will hold an additional *Guide Tool* in order to mark the target location where the robot should move the needle. This sensor tracks data of the surgeon's position and orientation for both tools simultaneously. Finally, a standard *Patient Simulator* is used in order to emulate the abdominal cavity.

Firstly, this section explains the robot architecture that controls CISOBOT platform previously described. Next, an experiment illustrates the functioning of the auto-guide planner system already explained in Sect. 4.

## 5.1 Robot Architecture

The architecture scheme of Fig. 22 resumes the features of CISOBOT system. On one hand, the *Camera Arm* box includes a passive wrist controller which is similar to the one explained on Sect. 2.2. As an input, the scheme receives the Camera Command given by the surgeon and is processed by an Incremental Planner that generates the corresponding trajectory for the camera. The feedback of the position and orientation of the endoscope is processed by the Geometric Estimator to give an approach of the real external distance required for the spherical navigation.



**Fig. 22** Architecture scheme for CISOBOT robot assistant

On the other hand, the *Tool Arm* box is devoted to the control of the actuated wrist of the arm manipulator with the laparoscopic instrument. In this case, the controller is based on the actuated wrist controller explained in Sect. 2.3, so the external distance is computed by means of a Force Estimator which receives the forces and torques measured by a force sensor attached to the end effector of the robot. As input, the Motion Planner receives the following data: the target location; forces and torques exerted by the tool; the robot location; and the Tracked Location measured by a 3D Tracker that scans the location of the surgeon's tools in real time. This Motion Planner works in the same manner as the collaborative planning explained in Sect. 4 and shown on Fig. 19.

## 5.2 Experimental Results

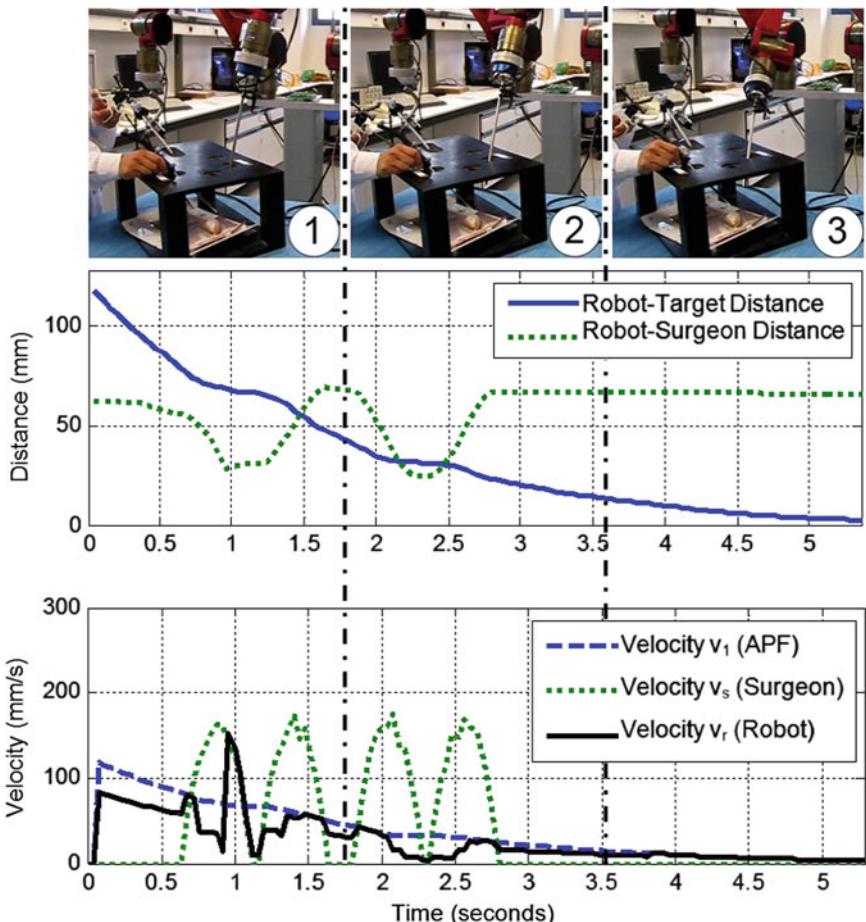
This experiment proposes to take a needle to a target location for a suture procedure. Therefore, the goal is to compute a free obstacle trajectory for the robot tool in real time from the initial tool location to the target by avoiding the surgeon's tool as well as the patient inner tissue. The robot tool is already located inside the abdominal cavity with the needle on its tool tip and the target location is defined by a second surgical tool whose location is read by optical 3D tracker. The robot will stop once the target location is achieved.

Figure 23 shows some snapshots of the resulting trajectory of an auto-guided movement with the robot tool, where the surgeon's tool has freedom of movement. This figure also shows one graph which plots the distance between both tools (dotted line) and the distance towards the target (solid line), whereas another graph represent the velocities of the robot and the surgeon tools. In order to demonstrate the behavior of the auto-guided planning methodology, this figure is divided into three time frames identified by the top labeled picture and separated by a vertical thick dotted line.

This experiment is focused on showing the actuation of all behaviors: the *Local Planner* with its APF algorithm, the *Velocity Correction* and the *Surgeon's Tool Follower*. For this purpose, the movement of the surgeon's tool consists of two oscillations and each one corresponds to two peaks in the Robot-Surgeon distance graph (dotted line), one caused by a surgeon forward movement and the other one caused by a backward displacement. In this way, the surgeon tool interferes the robot trajectory in the time frames labeled 1 and 2 in Fig. 23.

The first oscillation happens on the zone labeled as 1, where the surgeon's tool sweeps the robot tool. As a result, the auto-guide planner changes the robot velocity to reach the surgeon's tool velocity, as it has been described on the *Surgeon's Tool Follower* behavior. When the surgeon's tool moves backward the robot may increase its velocity up to the APF planned one.

The second oscillation can be seen on the time frame 2 of the velocity graph of Fig. 23. This time, the surgeon's tool blocks the robot trajectory, so the robot velocity must shrink its speed to avoid the collision with the surgeon's tool. Similarly to the



**Fig. 23** Auto-Guided trajectory with the surgeon's tool as a dynamic obstacle

first oscillation, when the surgeon's tool moves away then the robot tool may follow the APF planned trajectory for reaching the target location.

## 6 Discussion

This chapter has resumed the current state of the development on the main controllers for navigating the laparoscopic tools for the robot assistants, as well as the different planning systems classification depending on their degree of autonomy. More specifically, the authors have presented their contribution of an auto-guided planning for collaborative surgical robots. This motion planner can generate trajectories

either with a target force or location while it avoids possible obstacles on the robot tool trajectory. These two issues together are not implemented in the most relevant commercial robotic assistants.

The global control architecture of CISOBOT uses an actuated wrist instead of a remote center of rotation one for handling the tool. Although the second solution is the most frequent mechanism for surgical instrument movements, it only provides an initial robot calibration and it is not able to react for unexpected changes in the fulcrum point position. This question has a strong influence on the patient's safety, since unwanted forces can be exerted in his abdominal wall. On the other hand, the actuated wrist controller estimates the fulcrum location online so it can adapt the tool navigation and minimize a potential damage on the abdomen.

Furthermore, a valid strategy focused in the robot co-worker concept has been developed. This is based on real time free-obstacle trajectory computation in order to assist the surgeon in tasks like needle delivery. Collisions with both, inner tissue and surgeon tool are avoided by means of 3D tracker surgical tool position estimation and a 3D map generator of the inner organs and tissue of the patient. In order to validate the methodology, a two arm robotic system has been used for implementation purposes. An experiment where the robot takes needle to the surgeon's tool has been developed with success.

Related to the future of auto-guiding tasks, the authors believe that a deeper interaction between the camera movements and the robot tool would improve auto-guiding tasks. Auto-guiding tasks are also very useful not only for searching trajectories, but also for other more complex autonomous maneuvers like knot tying, suture or hold tissue where the robot interacts with the surgeon and the patient.

The final goal of this work consists of developing an autonomous platform, which aids the surgeon during the surgical procedure without direct orders for the robot to perform a certain task. In this way, the robot would develop the expected tasks at each step of the surgical procedure (if it can be divided as a protocol), but if the surgeon wants to change its behavior he or she always could do it by sending easy commands through gesture or voice.

## References

1. Fichtinger G, Stoianovici D, Taylor RH (2001) The surgical CAD/CAM paradigm and an implementation for robotically-assisted percutaneous local therapy. Paper presented at the 30th applied imagery pattern recognition workshop, 10–12 October 2001, pp 3–8
2. Taylor RH, Stoianovici D (2003) Medical robotics in computer-integrated surgery. IEEE Trans Robot Autom 19(5):765–781
3. Kazanzides P, Fichtinger G, Hager GD, Okamura AM, Whitcomb LL, Taylor RH (2008) Surgical and interventional robotics—core concepts, technology, and design. IEEE Robot Autom Mag 15(2):122–130
4. Chapuis J, Schramm A, Pappas I, Hallermann W, Schwenzer-Zimmerer K, Langlotz F, Caversaccio M (2007) A new system for computer-aided preoperative planning and intraoperative navigation during corrective jaw surgery. IEEE Trans Inf Technol Biomed 11(3):274–287

5. Fichtinger G, Kazanzides P, Okamura A, Hager G, Whitcomb L, Taylor R (2008) Surgical and interventional robotics: part II. *IEEE Robot Autom Mag* 15(3):94–102
6. Bootsma GJ, Siewerdsen JH, Daly MJ, Jaffray DA (2008) Initial investigation of an automatic registration algorithm for surgical navigation. Paper presented at 30th annual international conference of the IEEE engineering in medicine and biology society, 20–25 August 2008, pp 3638–3642
7. Hager G, Okamura A, Kazanzides P, Whitcomb L, Fichtinger G, Taylor R (2008) Surgical and interventional robotics: part III. *IEEE Robot Autom Mag* 15(4):84–93
8. Sungchoon L, Medi NT, Jeonghoon L, Kyunghwan K (2010) Control performance of a motion controller for robot-assisted surgery. Paper presented at IEEE workshop on advanced robotics and its social impacts (ARSO), 26–28 October 2010
9. Meister D, Pokrandt P, Both A (1998) Milling accuracy in robot assisted orthopaedic surgery. Paper presented at proceedings of the 24th annual conference of the IEEE industrial electronics society, vol 4, 31 August–4 September 1998, pp 2502–2505
10. Pechlivanis I, Kiriyathan G, Engelhardt M, Scholz M, Lucke S, Harders A, Schmieder K (2009) Percutaneous placement of pedicle screws in the lumbar spine using a bone mounted miniature robotic system, first experiences and accuracy of screw placement. *Spine J* 34(4):392–398
11. Wang Y, Laby K (1998) Automated endoscope system for optimal positioning. USA patent US5815640
12. Dowler N, Holland S (1996) The evolutionary design of an endoscopic telemanipulator. *IEEE Robot Autom Mag* 3(4):38–45
13. Polet R, Donne J (2008) Using a laparoscope manipulator (LAPMAN) in laparoscopic gynaecological surgery. *Surg Technol Int* 17:187–191
14. Marescaux J, Rubino F (2003) The ZEUS robotic system: experimental and clinical applications. *Surg Clin N Am* 83:1305–1315
15. Guthart GS, Salisbury JK (2000) The intuitive telesurgery system: overview and application. Paper presented at proceedings of the IEEE international conference on robotics and automation. San Francisco, 24–28 April 2000, pp 618–621
16. Kranzfelder M, Staub C, Fiolka A, Schneider A, Gillen S, Wilhelm D, Friess H, Knoll A, Feussner H (2012) Toward increased autonomy in the surgical OR: needs, requests, and expectations, *Surg Endosc*. doi:[10.1007/s00464-012-2656-y](https://doi.org/10.1007/s00464-012-2656-y)
17. Elshawary H, Popovic A (2011) Robust feature tracking in the beating heart for a robotic-guided endoscope. *Int J Med Robot Comput Assist Surg* 7:459–468
18. Weede O, Mönnich H, Müller B, Wörn H (2010) An intelligent and autonomous endoscopic guidance system for minimally invasive surgery. Paper presented at IEEE international conference on robotics and automation. Shanghai, 9–13 April 2010, pp 5762–5768
19. Staub C, Osa T, Knoll A, Bauernschmitt R (2010) Automation of tissue piercing using circular needles and vision guidance for computer aided laparoscopic surgery. Paper presented at IEEE international conference on robotics and automation. Alaska, 3–8 May 2010, pp 4585–4590
20. Reiley CE, Hager GD (2009) Task versus subtask surgical skill evaluation of robotic minimally invasive surgery. Paper presented at medical image computing and computer assisted intervention, pp 435–442
21. Nageotte F, Zanne P, Doignon C, Mathelin M (2009) Stitching planning in laparoscopic surgery: towards robot-assisted suturing. *Int J Robot Res* 28(10):1303–1321
22. Fuhan H, Payandeh S (2007) Real-time knotting and unknotting. Paper presented at IEEE international conference on robotics and automation. Roma, 10–14 April 2007, pp 2570–2575
23. Patil S, Alterovitz R (2010) Toward automated tissue retraction in robot-assisted surgery. Paper presented at IEEE international conference on robotics and automation. Alaska, 3–8 May 2010, pp 2088–2094
24. Cooper T, Blumenkranz SJ, Guthart GS, Rosa D (2006) Modular manipulator support for robotic surgery. Intuitive Surgical Inc, Patent no WO 2006/079108 A1

25. Madhani A, Niemeyer G, Salisbury K (1998) The black falcon: a teleoperated surgical instrument for minimally invasive surgery. Paper presented at proceedings of the IEEE/RSJ international conference on intelligent robots and systems. Victoria BC, Canada, 13–17 October 1998, pp 936–944
26. Rosen J, Brown JD, Chang L, Barreca M, Sinanan M, Hannaford B (2002) The blueDRAGON—a system for measuring the kinematics and the dynamics of minimally invasive surgical tools in-vivo. Paper presented at proceedings of the IEEE international conference on robotics and automation. Washington DC, May 2002, pp 1876–1881
27. Faraz A, Payandeh S (1999) On inverse kinematic and trajectory planning for tele-laparoscopic manipulator. Paper presented at proceedings of IEEE international conference on robotics and automation. Detroit, Michigan, May 1999, pp 1734–1739
28. Kang H, Wen JT (2001) EndoBot: a robotic assistant in minimally invasive surgeries. Paper presented at proceedings of the IEEE international conference on robotics and automation. Seoul, Korea, 21–26 May 2001, pp 2031–2036
29. Ortmaier T, Hirzinger G (2000) Cartesian control issues for minimally invasive robot surgery. Paper presented at proceedings of IEEE/RSJ international conference on intelligent robots and systems, 31 October–5 November 2000, pp 565–571
30. Wang Y, Laby KP (1998) Automated endoscope system for optimal positioning. Computer Motion. Patent no US5815640
31. Zemiti N, Ortmaier T, Morel G (2004) A new robot for force control in minimally invasive surgery. Paper presented at proceedings IEEE international conference on intelligent robots and systems. Sendai, Japan, 28 September–2 October 2004, pp 3643–3648
32. Bauzano E, Muñoz VF, Garcia-Morales I, Estebanez B (2009) Three-layer control for active wrists in robotized laparoscopic surgery. Paper presented at IEEE international conference on intelligent robots and systems. St. Louis, 11–15 October 2009, pp 2653–2658
33. Stolzenburg JU, Franz T, Kallidonis P, Minh D, Dietel A, Hicks J, Nicolaus M, Al-Aown A, Liatsikos E (2010) Comparison of the FreeHand robotic camera holder with human assistants during endoscopic extraperitoneal radical prostatectomy. *BJU Int* 107(6):970–974. doi:[10.1111/j.1464-410X.2010.09656.x](https://doi.org/10.1111/j.1464-410X.2010.09656.x)
34. Sackier J, Wooters C, Jacob L, Halverson A, Uecker D, Wang Y (1997) Voice activation of a surgical robotic assistant. George Washington University, Washington
35. Nishikawa A (2003) Face mouse: a novel human machine interface for controlling the position of a laparoscope. *IEEE Trans Robot Autom* 19(5):825–841
36. Taylor R (1995) A telerobotic assistant for laparoscopic surgery. *IEEE Eng Med Biol Mag* 14(3):279–288
37. Funda J, Gruben K, Eldridge B, Gomory S, Taylor R (1995) Control and evaluation of a 7-axis surgical robot for laparoscopy. Paper presented at proceedings of IEEE international conference on robotics and automation, pp 1477–1484
38. Muñoz VF, Garcia-Morales I, Perez-DelPulgar C, Gomez-DeGabriel JM, Fernandez-Lozano J, Garcia-Cerezo A, Vara-Thorbeck C, Toscano R (2006) Control movement scheme based on manipulability concept for a surgical robotic assistant. Paper presented at IEEE international conference on robotics and automation. Florida, May 2006, pp 245–250
39. Funda J, Taylor R, Eldridge S, Gruben K (1996) Constrained Cartesian motion control for teleoperated surgical robots. *IEEE Trans Robot Autom* 12(3):453–465
40. Tavakoli M, Patel R, Moallem M (2003) A force reflective master-slave system for minimally invasive surgery. Paper presented at proceedings of the IEEE/RSJ international conference on intelligent robots and systems. Las Vegas, Nevada, pp 3077–3082
41. Rosen J, Hannaford B, MacFarlane M, Sinanan M (1999) Force controlled and teleoperated endoscopic grasper for minimally invasive surgery: experimental performance evaluation. *IEEE Trans Biomed Eng* 46(10):1876–1881
42. Dario P, Hannaford B, Menciassi A (2003) Smart surgical tools and augmenting devices. *IEEE Trans Robot Autom* 19(5):782–792
43. Wolf A, Shoham M (2009) Medical automation and robotics. Springer handbook of automation. Springer, Berlin, pp 1397–1407

44. Konietzschke R et al (2009) The DLR mirosurge—a robotic system for surgery. Paper presented at IEEE international conference on robotics and automation. Kobe, pp 1589–1590
45. Hannaford B et al (2013) Raven-II: an open platform for surgical robotics research. *IEEE Trans Biomed Eng* 60(4):954–959
46. Schurr M, Arezzo A, Buess G (1999) Robotics and systems technology for advanced endoscopic procedures: experiences in general surgery. *Eur J Cardio-Thorac Surg* 16(2)
47. Mayer H, Nagy I, Knoll A, Schirmbeck E, Bauemschmitt R (2004) The endo[pa]r system for minimally invasive robotic surgery. Paper presented at proceedings of IEEE/RSJ international conference on intelligent robots and systems. Sendai, Japan, 28 September–2 October 2004, pp 3637–3642
48. Ko S, Kim K, Kwon D, Lee W (2005) Intelligent interaction between surgeon and laparoscopic assistant robot system. Paper presented at proceedings of IEEE international workshop on robots and human interactive communication, pp 60–65
49. Casals A, Amat J, Prats D, Laporte E (1995) Vision guided robotic system for laparoscopic surgery. Paper presented at IFAC international congress on advanced robotics. Barcelona, Spain
50. Hynes P, Dodds GI, Wilkinson AJ (2005) Uncalibrated visual-servoing of a dual arm robot for surgical tasks. Paper presented at proceedings of IEEE international symposium on computational intelligence in robotics and automation, 27–30 June 2005, pp 151–156
51. Krupa A et al (2003) Autonomous 3-D positioning of surgical instruments in robotized laparoscopic surgery using visual servoing. *IEEE Trans on Robot Autom* 19(5):842–853
52. Gangloff J, Ginhoux R, Mathelin M, Soler L, Marescaux J (2006) Model predictive control for compensation of cyclic organ motions in teleoperated laparoscopic surgery. *IEEE Trans Control Syst Technol* 14(2):235–246
53. Vitrani MA, Morel G, Bonnet N, Karouia M (2006) A robust ultrasound-based visual servoing approach for automatic guidance of a surgical instrument with in vivo experiments. Paper presented at the 1st IEEE/RAS-EMBS international conference on biomedical robotics and biomechatronics, 20–22 February 2006, pp 35–40
54. Grasa OG, Civera J, Montiel JMM (2011) EKF monocular SLAM with relocalization for laparoscopic sequences. Paper presented at IEEE international conference on robotics and automation. Shanghai, 9–13 May 2011, pp 4816–4821
55. Enxiu S, Tao C, Changlin H, Enxiu S, Junjie G (2007) Study of the new method for improving artificial potential field in mobile robot obstacle avoidance. Paper presented at IEEE international conference on automation and logistics. Jinan, 18–21 August 2007, pp 282–286
56. Fernandez R, Mandow A, Muñoz VF, Garcia-Cerezo A (1998) Real-time motion control for safe navigation. Paper presented at IFAC symposium on intelligent autonomous vehicles

**Part III**

**Motion and Operation Planning**

**for Wheeled Robots**

# Motion Planning Using Fast Marching Squared Method

S. Garrido, L. Moreno and Javier V. Gómez

**Abstract** Robotic motion planning have been, and still is, a very intense research field. Many problems have been already solved and even real-time, optimal motion planning algorithms have been proposed and successfully tested in real-world scenarios. However, other problems are not satisfactory solved yet and also new motion planning subproblems are appearing. In this chapter we detail our proposed solution for two of these problems with the same underlying method: non-holonomic planning and outdoor motion planning. The first is characterized by the fact that many vehicles cannot move in any direction at any time (car-like robots). Therefore, kinematic constrains need to be taken into account when planning a new path. Outdoor motion planning focuses on the problem that has to be faced when a robot is going to work in scenarios with non-flat ground, with different floor types (grass, sand, etc.). In this case the path computed should take into account the capabilities of the robot to properly model the environment. In order to solve these problems we are using the Fast Marching Square method, which has proved to be robust and efficient in the recent past when applied to other robot motion planning subproblems.

**Keywords** Fast marching · Fast marching square · Outdoor path planning · Non-holonomic path planning

## 1 Introduction

In nature, there are many fields that can be used as attractive fields. For example, the electromagnetic fields. We can guide us following the gradient of the field produced by an antenna and reach its position. In nature, there are also many fields that can be

---

S. Garrido (✉) · L. Moreno · J.V. Gómez  
Carlos III University of Madrid, Madrid, Spain  
e-mail: sgarrido@ing.uc3m.es

L. Moreno  
e-mail: moreno@ing.uc3m.es

J.V. Gómez  
e-mail: jvgomez@ing.uc3m.es

used as repulsive fields, for example the field produced by a system of particles with the same electrical charge of the charge placed in the position of interest.

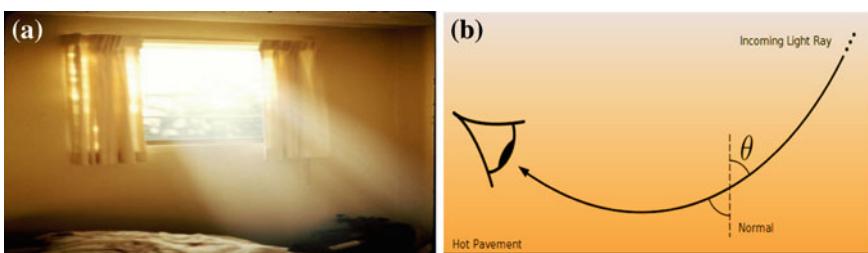
The problem with the potential fields is how to joint the two fields. Historically, different researchers have tried with different mathematical operations, but in this way, the total field has local minima that make the field unusable to find a path from the initial point to the goal.

Instead of using mathematical operations to joint the two fields, the Fast Marching Squared (FM<sup>2</sup>) method propose to joint the fields as Nature does. The bees and other insects that use the light to guide themselves can go out from a semidarkness room to the sunny exterior guiding themselves to the lighter zones as in Fig. 1a. In the same way, if we consider the light ray trough a system of lenses, it goes by the path that consumes a minimum time.

These considerations lead us to think that the solution is to use the repulsive field as the refraction index of space in which a light wave is propagated. In this way, the intuition about the propagation of the light in a non homogeneous media can guide us. In the typical road mirage, as you drive down the roadway, there appears to be a puddle of water on the road several metres in front of the car. The light rays coming from the sun are twisted in the vicinity of the hot road, due to the different refractive indices of the different layers of air parallels to the road, and make the beam reaches the driver's eyes as in Fig. 1b.

Mathematically, the propagation of the light is given by the Eikonal equation that is equivalent to the Fermat's principle: Light traveling through some substance has a speed which is determined by the substance. The actual path taken by light between any two points, in any combination of substances, is always the path of least time that can be traveled at the required speeds.

The Fermat's principle is especially interesting in our application, because if we have only a source of light waves, each point is connected with the source with a light path that it is parameterised by the time. The set of all the points of the domain with the time as last coordinate in the case of two spatial coordinates, gives us a Lyapunov surface in which the level curves are isochronals and the Fermats paths are orthogonal to them. It is impossible for the method to have local minima, because if there exists a local minimum  $x$ , and the time between other point  $y$  and  $x$  is minimum, as the time



**Fig. 1** **a** In a semidarkness room a bee goes to the lighter zones to go outside. **b** In the pavement mirage the sun rays bend near the hot road and go to the driver's eyes

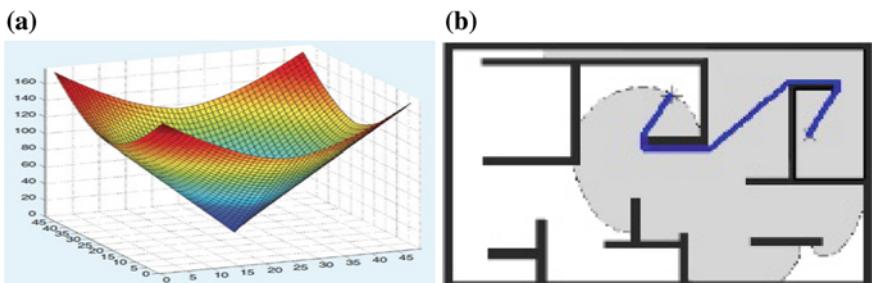
between  $x$  and the origin is minimum then the minimum time trajectory between  $y$  and the origin passes through  $x$ , i.e. the point  $x$  is not a local minimum.

In summary, the proposed method consists in the construction of a repulsive field by propagating a wave from the obstacles and walls. This gives us a refraction index or a slowness potential proportional to the inverse of the propagation velocity of the wave in the medium. Using this first potential as refraction index, a wave is propagated from the goal point. This results in the Lyapunov surface of the second potential. Applying gradient descent, maximum slope paths are obtained. Therefore these are the minimum time trajectories. The level sets of this second potential are, by definition isochronals, i.e. its points are at the same time from the origin.

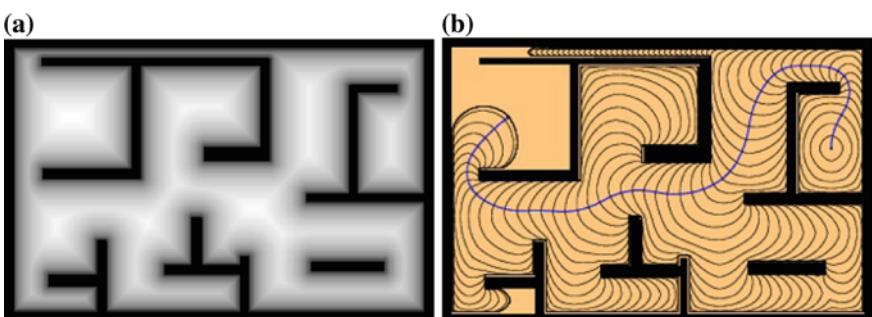
In Fig. 2 are shown the funnel potential of the light wave propagation with a constant refraction index and a path obtained in that way.

In Fig. 3a is shown the repulsive potential built by propagating the wave from obstacles and walls (first potential). It is similar to the distance transform, but in this case is continuous, not discrete. In Fig. 3b is shown the fronts of the propagation of the second wave, and the corresponding path.

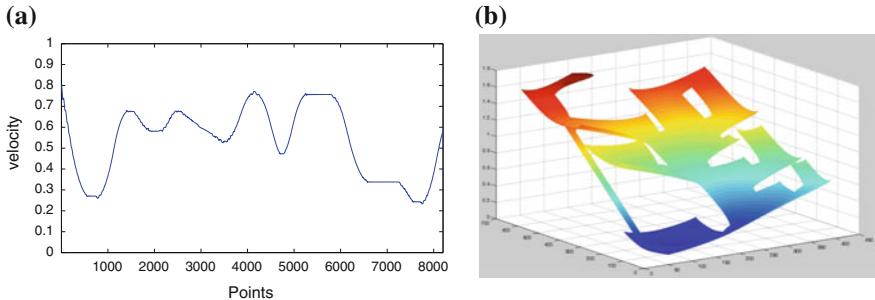
As the first potential can be interpreted as a difficulty map, and each point of the previous trajectory has associated a value of grey or difficulty, we can use it as



**Fig. 2** Lyapunov surface when the propagation wave starts in a point and the refraction index is constant (a) and Fast Marching Path when the refraction index is constant (b)



**Fig. 3** a Repulsive field by propagating a wave from the obstacles and walls (first potential) and b the corresponding wave fronts and its fast marching path when the refraction index is given by a

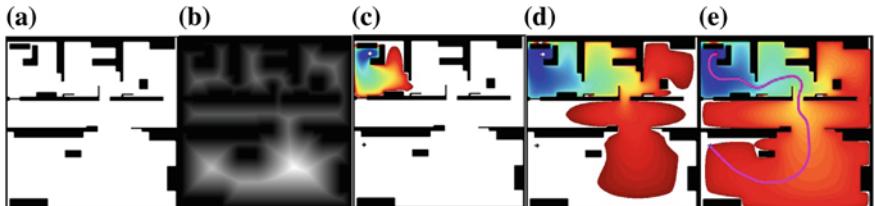


**Fig. 4** **a** Relative velocity profile of the path obtained in the previous figure, where 1 is the maximum velocity and **b** its Lyapunov surface

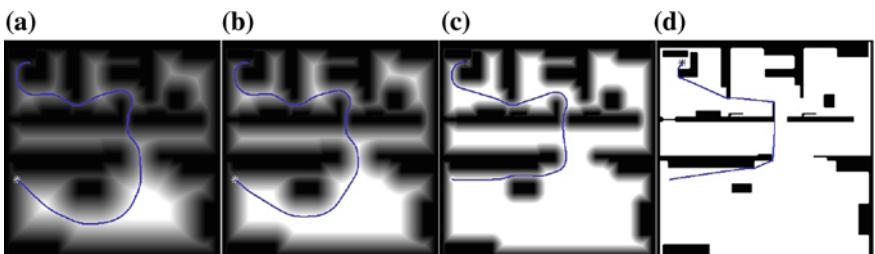
velocity profile with a maximum velocity given by the white color and zero velocity given by the black color, as shown in Fig. 3a. In Fig. 4b is shown the second potential (or funnel potential) of Fig. 3, in which the third axis represents time.

In Fig. 5 is shown an example of all the process: map of the environment, first potential and three moments of the wave expansion Fig. 5c–e. In the last is shown the corresponding path.

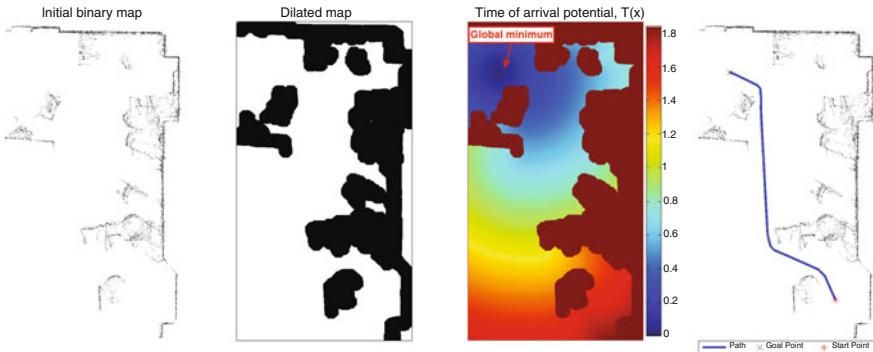
It is possible to stop the calculation of the first potential at different distances, or different levels of saturation. In this way, it is possible to have different kinds of shapes of the path: more or less close to the walls and obstacles. In Fig. 6 are



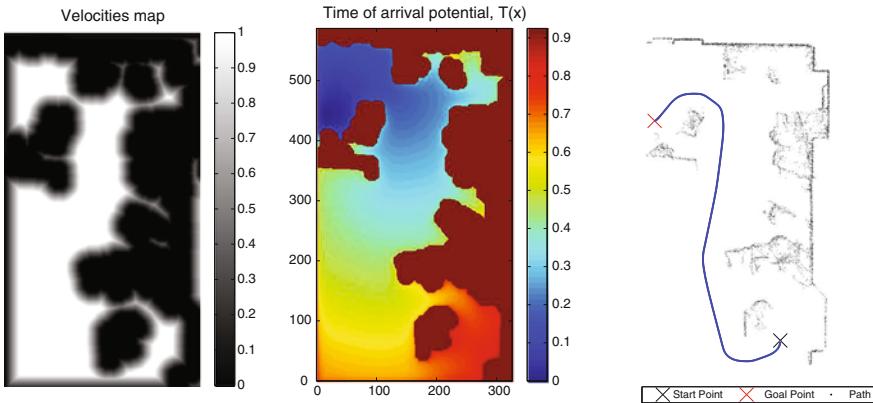
**Fig. 5** Example of environment **a**, first potential **b** and tree moments of the wave expansion **c–e**. In the last is shown the corresponding path



**Fig. 6** First potential and the paths that correspond to different levels of saturation. **a** Saturation: 0.75. **b** Saturation: 0.5. **c** Saturation: 0.25. **d** Saturation: 0



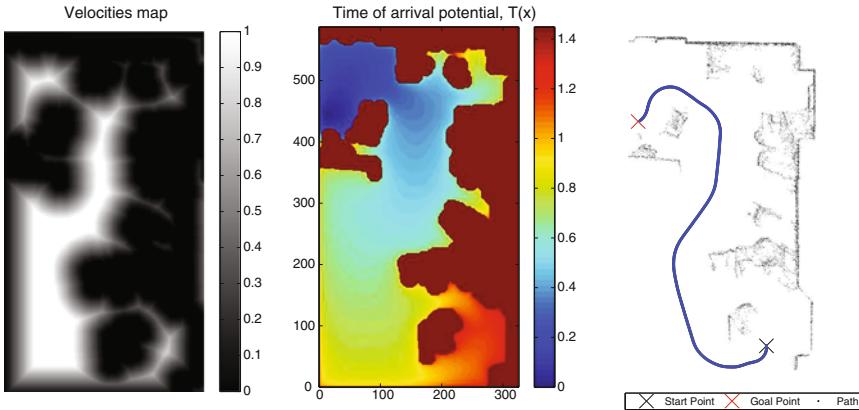
**Fig. 7** Laser data of our Lab (original map), First potential, second potential and the path corresponding to a high level of saturation



**Fig. 8** First potential, second potential and the path corresponding to a medium level of saturation

shown the paths that correspond to different levels of saturation. In Figs. 7, 8 and 9 are shown the first potential, second potential and the path corresponding to a three different levels of saturation for an environment data taken by the robots laser.

Fortunately, there exist a good method to solve the light propagation (Eikonal) equation numerically. This method is the Fast Marching (FM) method and was developed by Sethian [1]. It is an efficient computational numerical algorithm for tracking and modelling the motion of a physical wave interface (front) without reflexions. This method has been applied to different research fields, including computer graphics, medical imaging, computational fluid dynamics, image processing, computation of trajectories, etc. [2–4].



**Fig. 9** First potential, second potential and the path corresponding to a low level of saturation

The computational efficiency of the method allows the planner to operate at high rate sensor frequencies [5, 6]. For small and medium scale environments, the proposed method avoids the need for a collision avoidance algorithms plus a global motion planner. This enables simplification of the mobile robot or mobile manipulator architectures, while maintaining good time response, smooth and safe planned trajectories with continuous curvature. The trajectory generated by the planner is the fastest possible to achieve the goal position, by implication the best path according to the maximum acceptable velocity at each point in the trajectory (path plus velocity).

## 2 The Eikonal Equation and the Fast Marching Planning Method

One way to characterise the position of a front in expansion is to compute the time of arrival  $T$ , in which the front reaches each point of the underlying mathematical space of the interface. It is evident that for one dimension we can obtain the equation for the arrival function  $T$  in an easy way, simply considering the fact that the distance  $\theta$  is the product of the speed  $F$  and the time  $T$ .

$$\theta = F \cdot T \quad (1)$$

The spatial derivative of the solution function becomes the gradient

$$1 = F \frac{dT}{d\theta} \quad (2)$$

and therefore the magnitude of the gradient of the arrival function  $T(\theta)$  is inversely proportional to the speed.

$$\frac{1}{F} = |\nabla T| \quad (3)$$

For multiple dimensions, the same concept is valid because the gradient is orthogonal to the level sets of the arrival function  $T(\theta)$ . In this way, we can characterise the movement of the front as the solution of a boundary conditions problem. If speed  $F$  depends only on the position, then the Eq.(3) can be reformulated as the eikonal equation:

$$|\nabla T| F = 1. \quad (4)$$

The Fast Marching Method is a numerical algorithm for solving the Eikonal equation, originally, on a rectangular orthogonal mesh introduced by Sethian in 1996 [1]. The Fast Marching Method is an  $O(n)$  algorithm as has been demonstrated by [7], where  $n$  is the total number of grid points. The scheme relies on an upwind finite difference approximation to the gradient and a resulting causality relationship that lends itself to a Dijkstra-like programming approach.

Fast Marching Methods are designed for problems in which the speed function never changes sign, so that the front is always moving forward or backward (there are no reflections, interferences or diffractions). This allows us to convert the problem to a stationary formulation, because the front crosses each grid point only once. This conversion to a stationary formulation, in addition to a whole set of numerical tricks, gives it its tremendous speed.

Since its introduction, the Fast Marching Method approach has been successfully applied to a wide array of problems that arise in geometry, mechanics, computer vision, and manufacturing processes, see [5] for details. Numerous advances have been made to the original technique, including the adaptive narrow band methodology [8] and the Fast Marching Method for solving the static eikonal equation [5]. For further details and summaries of level set and fast marching techniques for numerical purposes, see [5].

## 2.1 Properties

The proposed FM<sup>2</sup> algorithm [6, 9–12] has the following key properties:

- *Fast response.* The planner needs to be fast enough to be used reactively in case unexpected obstacles make it necessary to plan a new trajectory. To obtain this fast response, a fast planning algorithm and fast and simple treatment of the sensor information is necessary. This requires a low complexity order algorithm for a real time response to unexpected situations.
- *Smooth trajectories.* The planner must be able to provide a smooth motion plan which can be executed by the robot motion controller. In other words, the plan does not need to be refined, avoiding the need for a local refinement of the trajectory.

The solution of the eikonal equation used in the proposed method is given by the solution of the wave equation:

$$\phi = \phi_0 e^{ik_0(\eta x - c_0 t)}$$

As this solution is an exponential, if the potential  $\eta(x)$  is  $\mathcal{C}^\infty$  then the potential  $\phi$  is also  $\mathcal{C}^\infty$  and therefore the trajectories calculated by the gradient method over this potential would be of the same class. At least from a theoretical point of view, because the equation is solved numerically and the result is an approximation of that trajectory.

This smoothness property can be observed in Fig. 3, where trajectory is clearly good, safe and smooth. One advantage of the method is that it not only generates the optimum path, but also the velocity of the robot at each point of the path. The velocity reaches its highest values in the light areas and minimum values in the greyer zones. The FM<sup>2</sup> Method simultaneously provides the path and maximum allowable velocity for a mobile robot between the current location and the goal.

- *Reliable trajectories.* The proposed planner provides a safe (reasonably far from a priori and detected obstacles) and reliable trajectory (free from local traps). This avoids the coordination problem between the local collision avoidance controllers and the global planners, when local traps or blocked trajectories exist in the environment. This is due to the refraction index, which causes higher velocities far from obstacles.
- *Completeness.* As the method consists of the propagation of a wave, if there is a path from the initial position to the objective, the method is capable of finding it.

## 2.2 Algorithm Implementation on an Orthogonal Mesh

The Fast Marching Method applies to phenomena that can be described as a wave front propagating normal to itself with a speed function  $F = F(i, j)$ . The main idea is to methodically construct the solution using only upwind values (the so called entropy condition). Let  $T(i, j)$  be the solution surface  $T(i, j)$  at which the curve crosses the point  $(i, j)$ , then it satisfies  $|\nabla T|F = 1$ , the Eikonal equation.

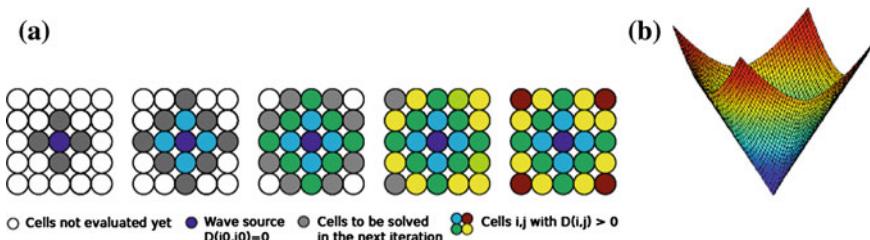
In order to understand how fast marching works, imagine an imprudent visitor that leaves unextinguished fire at some location in a natural reserve. The flame quickly becomes a forest fire, which expands outwards. Fire consumes the reached trees so the fire always propagates forward. We can record the fire front position at different points in time. It appears that the fire traverses the route having the smallest propagation time (and hence, the shortest length if the velocity is constant). In optics and acoustics this fact is known as Fermat's principle or, in a more general form, the least action principle. In plain language, Fermat's principle states that light traveling between two points always chooses the quickest path. Snell's law of refraction follows directly from this principle.

It is necessary to know that the propagation happens from smaller to bigger values of  $T$ . The algorithm classifies the points of the mesh into three sets: black, red and green, because our interface propagates like of a forest fire. Black points are points where the arrival time has been computed and is not going to change in the future. Green points are points that haven't been processed yet, for which the arrival time have not been computed up to now (corresponding with live trees). Red points are those belonging to the propagating wave front, which can be considered as an interface between the black and the green regions of the triangular mesh. In our forest fire example, red points correspond with trees that are currently in flames. Initially, only the source  $x_0$  is marked as black and all points adjacent to it, are marked as red. The remaining points are marked as green. At each iteration, the red with the smallest value of  $T(x)$  is put into the black set. This  $T(x)$  value is calculated using the black points in triangles sharing it. The updated adjacent points are tagged as red. The process continues until all points become black or the goal is reached.

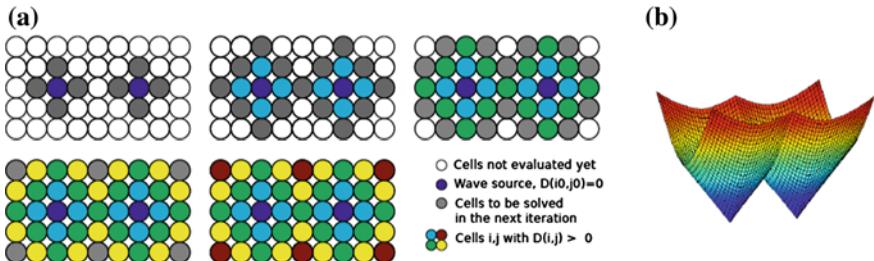
This equation is applied on grid points. Grid points are classified in three different types: alive, trial and far.

- Alive Points (black points) are points where values of  $T$  are known.
- Trial Points (red points) are points around the curve (alive points), where the propagation must be computed. The set of trial points is called narrow band. To compute propagation, points in the narrow band are updated to alive points, while the narrow band advances.
- Far Away Points (green points) are points where the propagation was not computed yet. During the propagation far away points are converted to trial points.

Figure 10a explain this idea: in the first subfigure the black point (alive) represents the initial curve; in 2nd subfigure the value of  $T$  is computed in the neighbourhood of black point; this neighbourhood is converted from far away (green) to trial points (red); in 3rd subfigure the trial point with smallest value of  $T$  is chosen (point A); in 4th subfigure the values of  $T$  are computed in the neighbours of point A, converting them from far away to trial. In fifth subfigure the trial point with smallest value of  $T$  is chosen (for example, "D"); in the last subfigure the neighbours of D are converted from far away to trial. And so on.



**Fig. 10** Scheme of fast marching propagation with an initial point. Different colors (blue to red) represent different arrival times in increasing order. **a** Iteration of FM with one wave in 2D. **b** Time of arrival potential  $D(x)$  (third axis)



**Fig. 11** Scheme of fast marching propagation with two initial points. Different colors (blue to red) represent different arrival times in increasing order. **a** Iteration of FM with two propagating wave (in 2D). **b** Time of arrival potential  $D(x)$  (third axis)

Figure 11 represents the scheme of Fast Marching propagation with two initial points.

### 2.3 Algorithm Implementation on an Triangular Mesh

The numerical basis of the fast marching method and its foremost difference with Dijkstra's algorithm resides in the update procedure. While in Dijkstra's algorithm the path is restricted to the graph edges, and a graph vertex was updated each time from an adjacent vertex, in fast marching, because the path can pass through the triangular faces of the mesh, a vertex has to be updated from a triangle, requiring two supporting vertices. We assume that the update step is applied to a triangle  $(x_1, x_2, x_3)$ , where  $x_1$  is the red point with the smallest arrival time  $T_1 = T(x_1)$ ,  $x_2$  is a point for which some arrival time approximation  $T_2 = T(x_2)$  is available, and  $x_3$  is the red or green point, whose arrival time approximation  $T_3 = T(x_3)$  is that the triangle lies in the plane with  $x_3 = 0$ . In essence, given that the front reaches  $x_1$  at time  $T_1$  and  $x_2$  at time  $T_2$ , the update step has to estimate the time when the front arrives to  $x_3$ , as shown in Fig. 12.

### 2.4 Results of $FM^2$ Method

To illustrate the potential of the proposed method, four working conditions are evaluated (sensor-based operation, map-based operation, combined operations, behaviour in cluttered environment and the computational cost is shown).

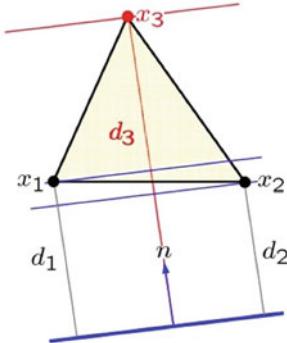
**Triangular Fast Marching update step applied to a triangle  $(x_1, x_2, x_3)$**

- Given  $x_1$  with the smallest arrival time  $d_1 = T(x_1)$  and  $x_2$  with arrival time  $d_2 = T(x_2)$ , it is needed to calculate  $d_3 = T(x_3)$
- The equation of the line is  

$$ax+by-d=0$$
- Substituting the data  

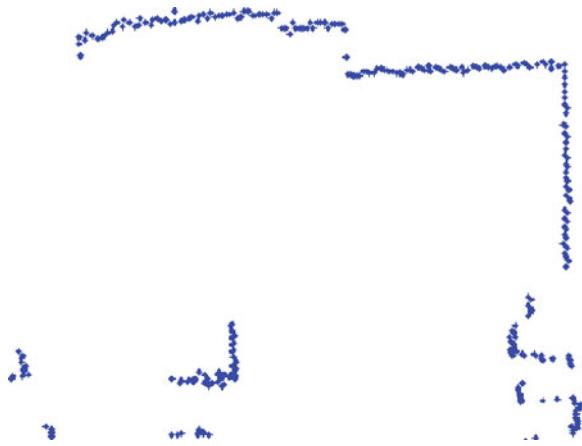
$$\begin{cases} ax_{11} + bx_{12} = d_1 \\ ax_{21} + bx_{22} = d_2 \end{cases}$$
  
and solving the equation, the normal vector is  $n=(b, -a)$
- Finally the time  $d_3=T(x_3)$  is  

$$d_3=ax_{31}+bx_{32}$$



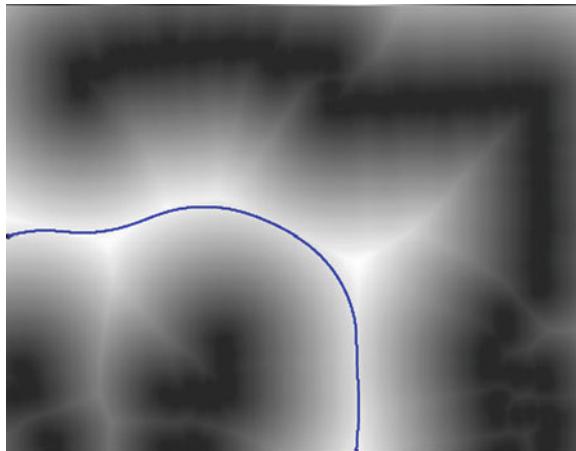
**Fig. 12** Scheme of fast marching update step

**Fig. 13** Laser scan data corresponding to a corner of a corridor of our university



#### 2.4.1 Sensor-Based Planning

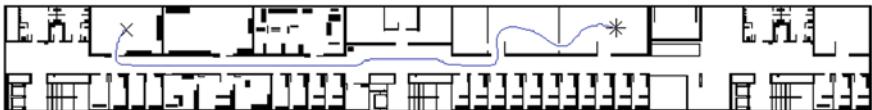
In the first test, the method proposed is applied directly to the data obtained from a laser scan around the robot, where the method obtains a good trade off between trajectory distance, distances to obstacles and smooth overall trajectory as shown in Figs. 13 and 14. These images correspond to a corner of a corridor of our University.



**Fig. 14** Repulsive potential of the scanned data and trajectory obtained with the FM<sup>2</sup> method



**Fig. 15** Slowness potential of FM<sup>2</sup> 1st step for the UC3M Robotics Lab floor



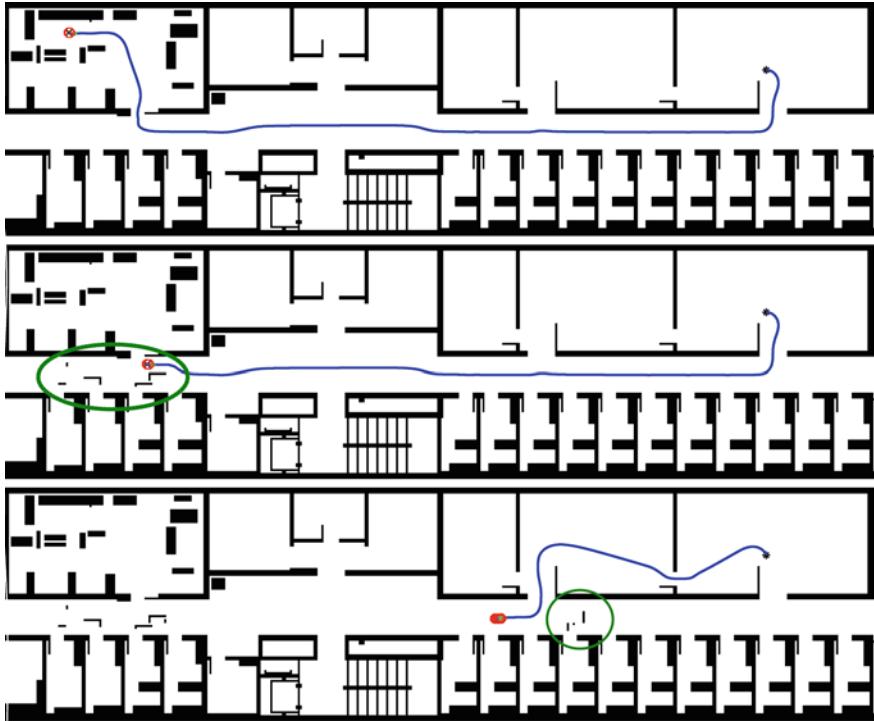
**Fig. 16** Motion trajectory obtained with the FM<sup>2</sup> method for the UC3M Robotics Lab floor

#### 2.4.2 Map-Based Planning

In the second test, in order to show global plan capabilities, the method is applied to the whole plant of the building where the Robotics laboratory is located. The laboratory floor is around 2,000 m<sup>2</sup> (medium size). The results are shown in Figs. 15 and 16.

#### 2.4.3 Combined Planning

The third test shows the combination of the global and local properties of the method. In this case a simple trajectory motion is determined from an initial position to the goal position. During the motion, the robot observes the environment with its laser scan, places it on the map and plans a new trajectory. Local observations (obstacles located in the middle of the corridor) result in slightly modified trajectories to avoid



**Fig. 17** Dynamical evolution of the path when the robot reads information about the new obstacles (marked with *green ellipsoids*) absent in the previous map and the robot cannot pass through the corridor

the obstacles detected (Fig. 17). In the last image in Fig. 17 the detected obstacles blocked the corridor and the sensor based global planner finds a completely different trajectory. It is worth noting that in this case, the fact that the global planning capability takes action, allows automatic replanning of the trajectory. This replanning is not possible with some other methods due to the separation of the two planners.

This technique shows the advantage of a method which is not only local, but also global, that combines sensor based local planning capabilities with global planning capabilities to react quickly to the obstacles while maintaining reliability in the planned trajectory. The method always finds the solution, if one exists.

### 3 Application of the FM<sup>2</sup> to Car-Like Robots

An important kind of robots are the nonholonomic robots, that can't move freely in any desired direction, but they have to accomplish a set of constraints. A typical case are the car-like robots.

In this section, we are going to describe how to apply the FM<sup>2</sup> method to car-like robots. An interesting feature that has not been sufficiently highlighted in the previous sections is that by using the gradient over the second potential, it is possible to calculate a vector field whose field lines are the paths that go from each point to the target, away from obstacles and walls.

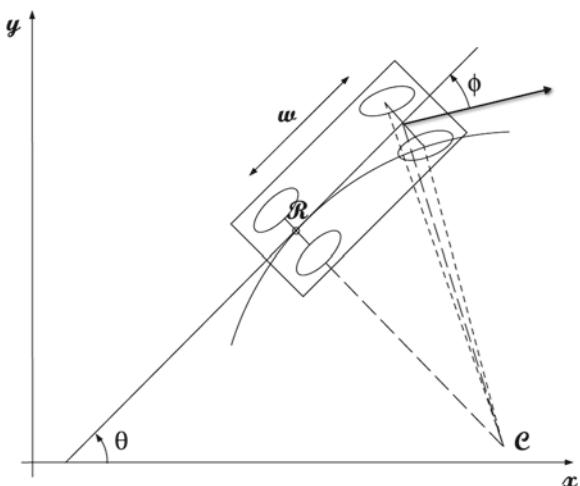
In order to apply the proposed method, it is considered a 3D C-Space of the environment, with the two dimensions of the robot's position and the vehicle's orientation as the third dimension. Computing a trajectory along the C-Space built taking into account the vehicle's dimensions, it is possible to guarantee the absence of collisions. This means we operate over the configuration space instead of the bi-dimensional environment map (see Fig. 19, in which the third dimension is the orientation of the robot, with 21 possible values. These orientations are repeated above and under the principal interval in order to permit manoeuvres). The C-space has been built iteratively placing the vehicle in every position and with every possible angle. This is a slow task, but it can be done offline and once per map.

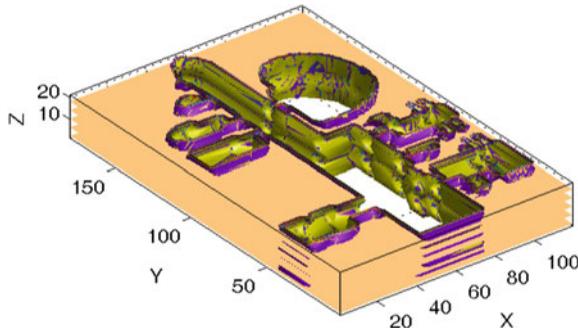
After that, the slowness potential (distance transform) is calculated using the Fast Marching method for this resultant space. The wave is propagated from the walls of the previously calculated C-space.

Based on this slowness map, the Fast Marching Method creates the second potential  $T(\mathbf{x})$  that represents arrival time of the wavefront, and in this way the method gives the arrival time as the fourth axis. The origin of the wave is the goal point, which continues propagating until it reaches the current position of the robot.

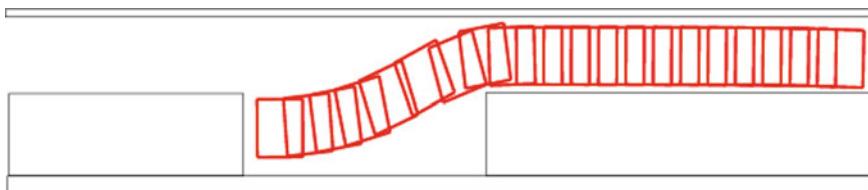
Using this Funnel shaped second Potential the associated  $OXY$  vector field is calculated. This vector field has as field lines the different line paths from the different points of the C-space and all of them finish in the goal point. This lines, also, go away from the obstacles. This vector field is going to be used to move the car-like robot (Fig. 18).

**Fig. 18** A car-like robot





**Fig. 19** Three dimensional C-space of the car-like robot, where the third dimension is the orientation. These orientations are repeated above and under the principal interval in order to permit manoeuvres



**Fig. 20** Parking maneuver using FM<sup>2</sup>-NH

Car-like robots have a limited steering angle causing them to move along paths of bounded curvature. This can be expressed as a constraint on the curvature radius. This constraint can be directly included in the algorithm using the vector field, in form of limits during the path calculation. Figure 20 shows the result to apply the algorithm to a parking manoeuvre.

Finally, starting from the initial position and orientation, the path is constructed step by step, according to the following order:

- The front wheels are aligned with the vector field in the midpoint of the front axis.
- The perpendicular lines to the front and rear wheels are considered and their intersection is taken as center of the step movement.
- With the previously calculated center, the vehicle is moved a circumference arc of length proportional to the vector modulus correspondent to that point.

The previous process is repeated from the new point until the destination point is reached. The final point and orientation is always reached because the funnel potential end at this point and orientation.

Consider the car-like robot shown in Fig. 18. In this figure ( $x, y$ ) is the position of the center of the rear axis,  $\theta$  is the car orientation respect the  $OX$  axis. It is necessary to take into account the non-holonomic constraint

$$\dot{y} \cos \theta - \dot{x} \sin \theta = 0$$

and the car-like movement can be modelled, assuming the distance between the front and rear axes as 1, as

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} v \cos \phi \cos \theta \\ v \cos \phi \sin \theta \\ v \sin \phi \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} v_1 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} v_2 \quad (5)$$

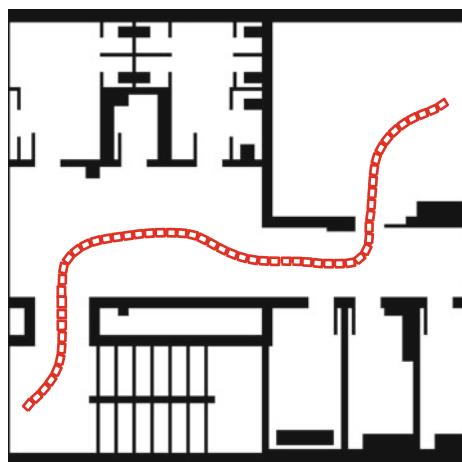
where  $\phi$  is the front wheels orientation,  $v$  is the car velocity and  $v_1, v_2$  are the two control inputs: acceleration of the car and angular velocity of the front wheels.

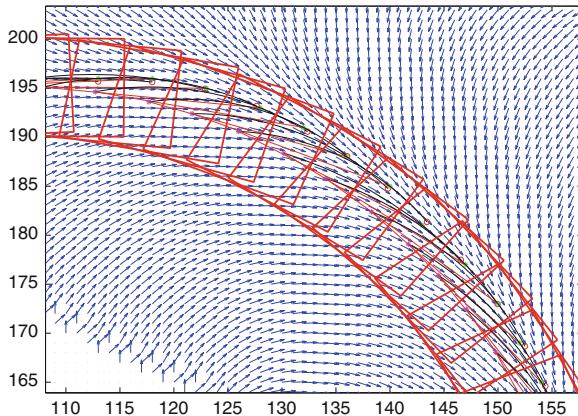
An interesting remark is that in this equation everything is done by the vector field except the control inputs  $v_1, v_2$ : the acceleration of the car and the angular velocity of the front wheels. These control inputs can be deduced and in this way the method not only give the trajectory but also the control inputs to follow that trajectory.

The result of an example of the nonholonomic version of the FM<sup>2</sup> method can be observed in Fig. 21, where a corridor of the university is shown. The corresponding C-space is represented in Fig. 19. The top and the bottom are connected because the angle wraps around  $2\pi$ . The trajectory obtained is smooth and safe.

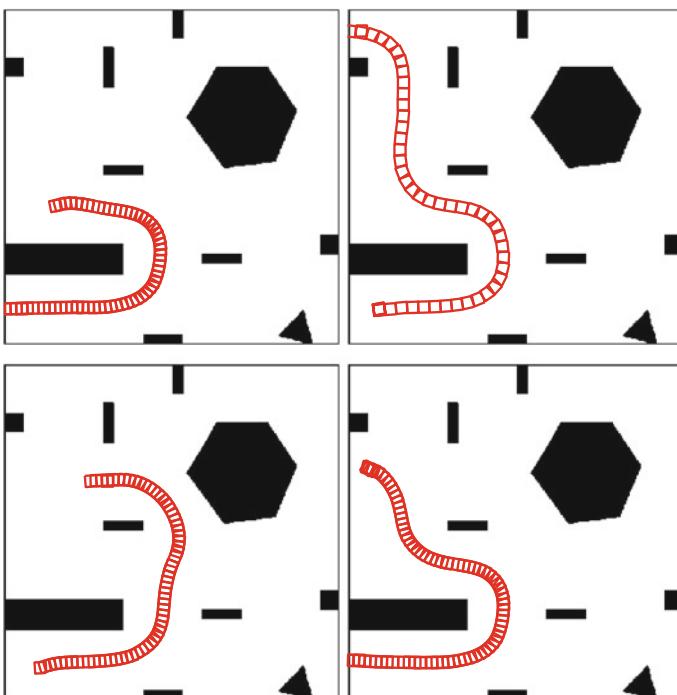
To illustrate the capability of the proposed method, different situations are shown (see Figs. 23 and 24). In the case of the Fig. 23, a simple trajectory is determined from an initial position and orientation to the goal position and orientation. Local observations (obstacles located in the scene) originate slightly modified trajectories to avoid the detected obstacles. We can conclude that the four situations have good trajectories (safe and smooth) between the initial and the final point. In the enlarged image we can see the velocity field (see the Fig. 22) calculated for the movement of the car-like robot, where the vectors have been normalised for a better visualisation. This technique shows the advantage of a method which is not only local, but also

**Fig. 21** FM<sup>2</sup>-NH applied to the car-like robot in the university corridor





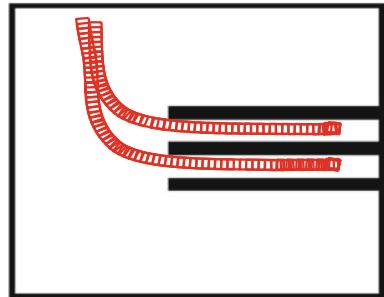
**Fig. 22** Movement of the vehicle on the vector field



**Fig. 23** Different motion trajectories obtained with the proposed method (non holonomic)

global, which combines sensor based local planning capabilities with global planning capabilities to react to the obstacles very quickly while maintaining reliability in the planned trajectory. The proposed method is highly efficient from a computational point of view because the Fast Marching can be implemented with complexity  $O(n)$ , where  $n$  is the number of cells in the environment map.

**Fig. 24** Example trajectory obtained with FM<sup>2</sup>-NH



### 3.1 Comparison with Existing Methods

The common limitation of all the reactive navigation methods is that they cannot guarantee global convergence to the goal location because they use only a fraction of the information available (the local sensory information). Some researchers have worked on introducing global information into the reactive collision avoidance methods to avoid local trap situations. This approach has been adopted by Ulrich and Borenstein [13] which uses a look-ahead verification to analyse the consequences of a given motion a few steps in advance to avoid trap situations. Other authors exploit the information about global environment connectivity to avoid trap situations (Minguez and Montano [14]). Those solutions still maintain the classical two level approach, and require additional complexity at obstacle avoidance level to improve the reliability at this level.

The proposed method is consistent at local and global scale because it guarantees a motion path (if it exists), and does not require global replanning supervision to restart a planning when a local trap is detected or a path is blocked. Furthermore, the path calculated has good safety and smoothness characteristics.

Most of the other methods give paths that are not smooth, even though they only provide a few loose points linked by segments of straight lines. The only methods that give comparable results are based on harmonic functions (the solutions of the equation of Laplace) but they have the problem of slowness.

## 4 How to Deal with Difficulty and Uncertainty in an Outdoor Environment to Plan Trajectories Using the Fast Marching Method. Algorithm Implementation on a Triangular Mesh

This section applies the FM<sup>2</sup> to the problem of finding trajectories for an outdoor robot. The objective is to apply Fast Marching to a 3D triangular mesh that represents the surface terrain to find a trajectory between two points. The proposed method uses a triangular mesh because this kind of grid adapts better to 3D surfaces. The advantages

of this approach are that, in the first step of the method, the algorithm calculates a weight matrix  $W$  that can represents difficulty, refraction index (inverse of speed) or uncertainty based on the information extracted from the 3D surface characteristics and the sensor data of the robot. In the experiments carried out in this work these features are the spherical variance, the gradient of the surface, the height, and also the incertitude in the map because some portions of the map can't be measured directly by the robot. This difficulty matrix is used to define the speed of propagation of the Fast Marching wave in order to find the best path depending on the task requirements, e.g., the trajectory with the fastest path, the least energy consumption, the most plain terrain, the safest path or the known terrain. The method also gives the robot's maximum admissible speed in each point. This depends on difficulty matrix. The results presented in this chapter show that it is possible to model the path characteristics as desired, by varying this difficulty matrix  $W$ .

## 4.1 Matrix $W$ : The Difficulty Map

The proposed method is based on the FM method, changing the speed of propagation of the wave using a potential generated from the 3D environment characteristics and the robot limitations. This way, the method changes the time when the front reaches each point and when the generated trajectory is calculated. This trajectory is not going to be the simple geodesic, but it is going to be modified according to the robot and task needs. To be able to modify this speed, the proposed method creates a weight matrix  $W$ , which is currently built based on the main characteristics of the 3D surface: the *spherical variance*, the *saturated gradient*, the *height* and the uncertainty. Some other characteristics can be added to the method and it will build a different potential surface.

### 4.1.1 Spherical Variance

The spherical variance is a measure the roughness of a surface. It can determine if a zone is crossable or not. In [15], it is presented a method to calculate the roughness degree. This method is based on the normal vector dispersion in each point of the surface:

- In a uniform terrain (low roughness), the normal vectors in a surface will be approximately parallel and, for this reason, they will present a low dispersion.
- On the other hand, in an uneven terrain (high roughness) the normal vectors will present great dispersion due great to changes in their orientation.

The method to calculate the spherical variance is:

1. Given a set of  $n$  normal vectors to a surface, defined by their three components  $\{(x_i, y_i, z_i)\}$ , the module of the sum vector  $\mathbf{R}$  is calculated by:

$$R = \sqrt{\left(\sum_{i=0}^n x_i\right)^2 + \left(\sum_{i=0}^n y_i\right)^2 + \left(\sum_{i=0}^n z_i\right)^2} \quad (6)$$

2. Next, the mean value is normalised by dividing the module  $R$  between the number of data  $n$ , so the value of the result is within  $[0, 1]$ . In this way, we have

$$\frac{R}{n} \in [0, 1] \quad (7)$$

3. Finally, the spherical variance  $S_v$  is defined as the complementary of the previous result.

$$S_v = 1 - \frac{R}{n} \quad (8)$$

when  $S_v = 1$ , there exists a maximum dispersion that can be considered as the maximum roughness degree, and when  $S_v = 0$ , a full alignment exists and the terrain will be completely flat.

#### 4.1.2 Saturated Gradient

The gradient of a surface is a vectorial field. In each point, the gradient point in the direction of the greatest rate of increase of the scalar field in that point, and whose module is the greatest rate of change in that point.

The gradient of  $f(x, y)$  is defined to be the vector field whose components are the partial derivatives of  $f$ . That is:

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right) \quad (9)$$

In order avoid having path slopes greater than the robot can perform the gradient is saturated with that limit. That means that, if the gradient value exceeds that limit, the point will not be included in the list of accessible points determined by the robot limitations.

#### 4.1.3 Construction of Matrix $W$

By using this matrix  $W$  the algorithm modifies the path that the robot is going to follow across the 3D surface. The way the matrix modifies the path is by giving a viscosity value for each point on the surface. It means that the propagation speed of the front end of the FM wave is modified. Hence, the time when the wave reaches each point will depend on that difficulty. It is possible to add as many characteristics as we need to get different paths. These characteristics will modify the viscosity at each point.

The saturated gradient, the spherical variance, and the height are three matrices  $G$ ,  $Sv$ , and  $H$  with the same size as the vertex matrix (the 3D mesh). The value of each vertex of the 3D grid will be determined by the calculated gradient, spherical variance, and the height of each point.

The matrix  $W$  is a weighted average of each surface characteristic we are interested in, and in each case it gives more importance to the more important factors depending on the task requirements.

The values of the component matrices vary from 0 to 1, so the values of matrix  $W$  are also within this range. The components of matrix  $W$  with a value of 0 (less difficult) will be points in the *vertex* matrix with maximum speed. Hence, these are points which the robot can cross without any problem and at its maximum speed. The elements of  $W$  with a value of 1 will be points with a minimum speed, and in that case, the robot will not be able to pass across them.

$$W = a_1 \cdot G + a_2 \cdot Sv + a_3 \cdot H \quad (10)$$

where:

$$\sum_i a_i = 1 \quad (11)$$

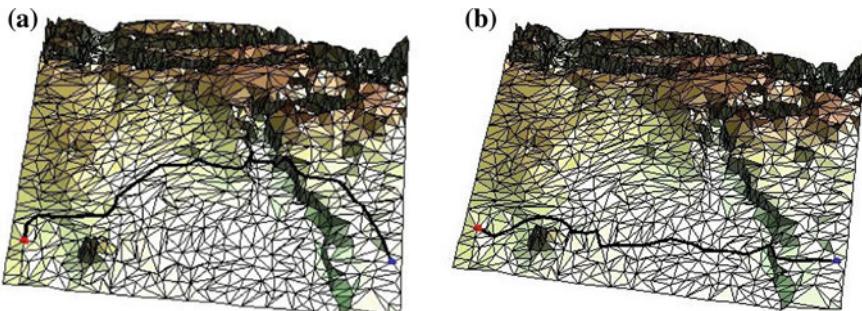
After the difficulty matrix  $W$  is generated, the method runs the FM algorithm over the modified mesh (3D mesh + matrix  $W$ ) to calculate the best trajectory. With the FM method the path found will be the less time path in the  $W$  metrics. If  $W$  is constant, this path will be the shortest because all the points in the surface will have the same ‘speed’ for the front propagation, i.e. the path is the geodesic. With a non constant matrix  $W$ , the proposed method changes that ‘speed’, since this matrix gives information about the difficulty to pass through each point of the surface. The trajectory will be modified depending on the surface conditions and characteristics and according to the robot limitations. Since the method modifies the ‘speed’ of the Fast Marching wave, and in each point  $W$  gives the difficulty that can be interpreted as maximum speed, it gives not only the best trajectory, but also the speed to control the robot.

#### 4.1.4 Test on Data Taken in Advance

As previously stated, in the proposed method we need terrain data that can be an elevation map, global or local laser data or a mixture of all. In relation to the outdoor environment reconstruction, a triangle-based 3D surface is chosen.

The method works in 3D, in order to create a triangular mesh, the algorithm reads the data from the bitmap file to create the three matrices  $X$ ,  $Y$ , and  $Z$  and then, it builds a 3D mesh based on  $X$ ,  $Y$  and  $Z$  coordinates. The first step of the algorithm is to generate a Delaunay triangulation in 3D.

After the mesh is created, the algorithm extracts the vertices and the faces of the triangles. Using these values, the algorithm is able to model the 3D triangular surface.



**Fig. 25** Path calculated when **a**  $W = A$  and when **b**  $W = G$  in a mars map

Several paths over the surface already presented will be obtained between the same initial and final points. Those paths are obtained by varying the values of the weight factors  $a_i$  of matrix  $W$ .

In the case that  $W = A$ , this implies that the difficulty of the path will be determined by the height of every point of the mesh. In Fig. 25a, the path obtained when the height is penalised, without considering the roughness of the surface or its inclination, is presented. As can be observed, the calculated path will try to reach the final point passing through the deepest part of the map.

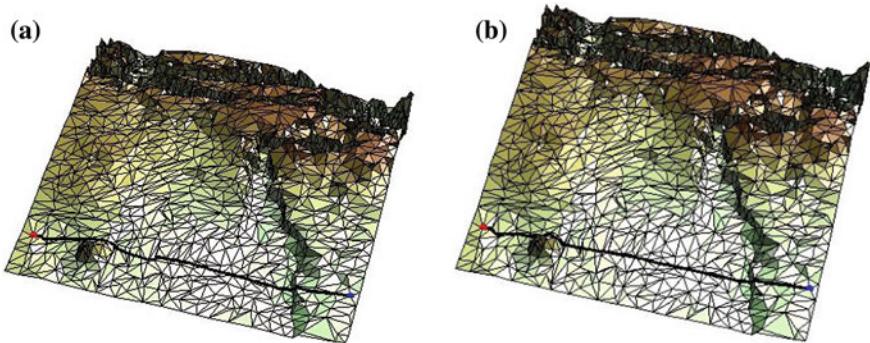
On the other hand, if we decide to calculate the path penalising just the inclination of the surface, then the difficulty matrix is defined as  $W = G$ . In this case, as shown in Fig. 25b, the path will follow the parts with smallest slope.

The general idea proposed in this section is the possibility of combining the different matrices in order to obtain a path that considers the height  $A$ , the roughness  $Sv$ , and the inclination  $G$  of the surface, among others. In the previous figures, it can be observed that, for the selected initial and final points, the height matrix favour that the path goes all the way trying to avoid the highest parts of it. On the other hand, the gradient matrix  $G$  favours the path with smallest slope. Therefore, we can select the values of each weight factor  $a_i$  in order to consider the limitations or features of the robot used.

The final step is to propagate the wave using as refraction index the difficulty matrix  $W$  from the goal point until it gets the present position of the robot and in this funnel shaped potential, the trajectory is calculated by using the gradient method.

Figure 26a shows a view of the path obtained when  $W = Sv$ . As can be observed the result is an intermediate path. Figure 26b shows a view of the path obtained when  $W = 0.20 * A + 0.40 * Sv + 0.40 * G$ .

Moreover, the values of the weight factors  $a_i$  can be changed if the robot to be used is different or modified. It is also important to note that the trajectories calculated are a tentative path for the robot. The path can be modified online by modelling the environment with the robot sensors and recalculating the trajectory in a local area.



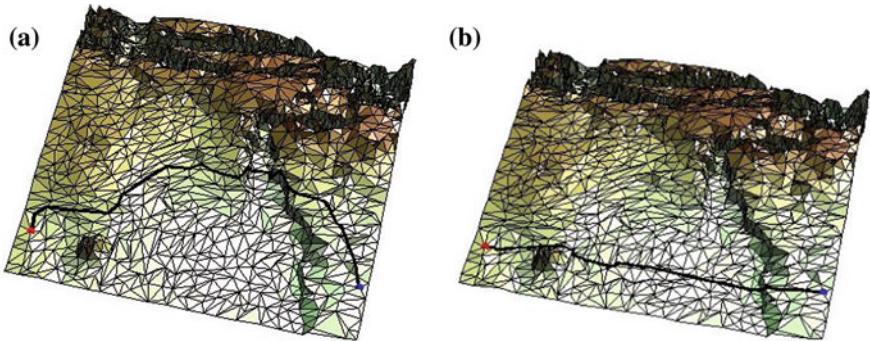
**Fig. 26** Path calculated using **a**  $W = Sv$  and when **b**  $W = 0.20 * A + 0.40 * Sv + 0.40 * G$  in a mars map

#### 4.1.5 Introduction of the Uncertainty in the Slowness Matrix $W$

When there is a certain uncertainty the robot has to modify the trajectory or the velocity. For example, in the case of robot in Mars, if the robot doesn't have enough information of part of the trajectory, because it hasn't visual data of that part, could be better to change the trajectory to zones the robot can visualise.

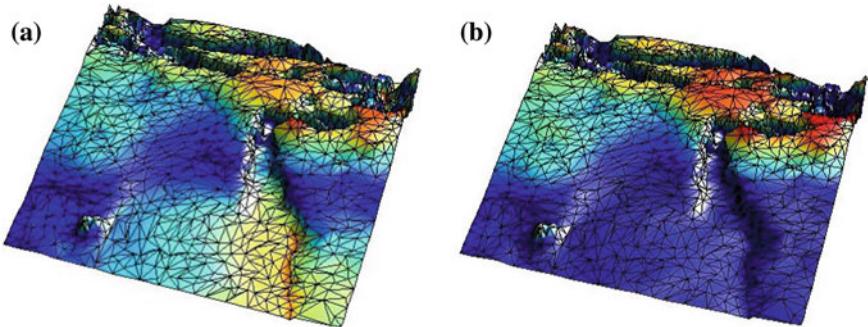
How can we introduce that uncertainty in the map in order to change the trajectory? Fortunately, the viscosity matrix  $W$  can also be understood as an uncertainty matrix, the grey degree can be understood as a measurement of the uncertainty.

For example, suppose that the robot has no data of the points lower than its altitude, in that case the shadow points are represented in the matrix  $W$  with values next to zero (velocity of the media). In Fig. 27 is shown the difference between the robot trajectories without and with uncertain data of the points lower than the robot's altitude. As can be seen in the figure on the right, the trajectory is modified to not to

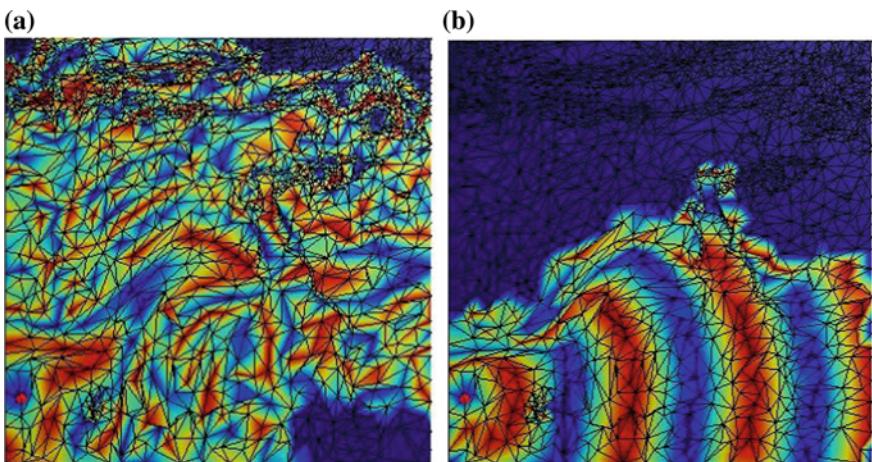


**Fig. 27** Difference between the robot trajectories **a** without and **b** with uncertain data of the points lower than its altitude

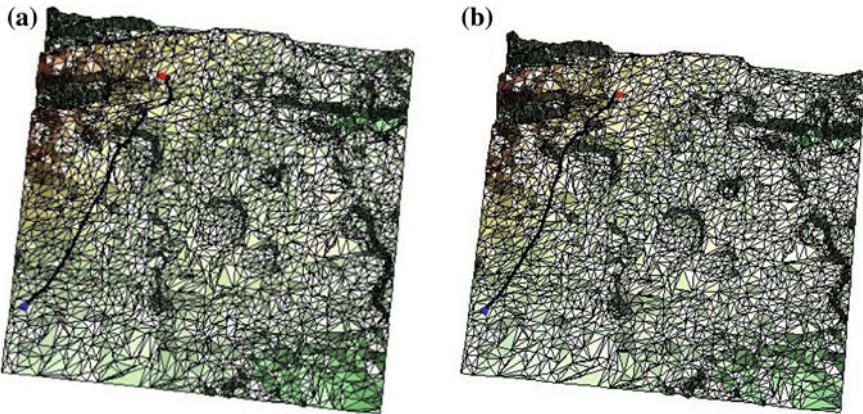
go through the lower areas. In Fig. 28 is shown the difference between the difficulty-uncertainty  $W$  matrices when the robot has and hasn't data of the points lower than its altitude. As can be seen in the right figure the lower parts have a bluish colour due to a bigger uncertainty and lower values in the  $W$  matrix that correspond to lower media velocity. In Fig. 29 is shown the difference between the wave expansion  $D$  matrices when the robot has and hasn't data of the points lower than its altitude. As can be seen, in the figure on the right, the expansion of the wave is more directed to the zone with less incertitude.



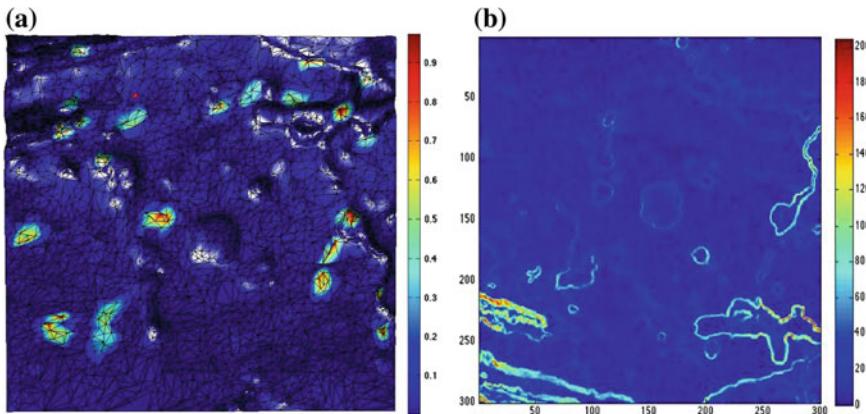
**Fig. 28** Difference between the difficulty-uncertainty  $W$  matrices when the robot **a** has and **b** hasn't data of the points lower than its altitude



**Fig. 29** Difference between the wave expansion  $D$  matrices when the robot has and hasn't data of the points lower than its altitude



**Fig. 30** Difference between the difference in the paths when the gradient **a** is not saturated and **b** when it is



**Fig. 31** **a** Spheric variance and **b** the saturated gradient corresponding to the previous figures

In Fig. 30 is shown the difference in the paths when the gradient is not saturated and when it is.

In Fig. 31 are shown the saturated gradient and the spheric variance corresponding to the previous figures.

## 5 Conclusions and Future Work

As shown along the chapter, the FM and  $\text{FM}^2$  methods are very powerful when applied to robot motion planning. Many different problems can be faced with the same underlying method in addition to minor modifications.

When applied to 2D or 3D environments, the FM<sup>2</sup> method is able to provide efficient solutions in a very short period of time, reaching even real-time applications. However, as it is based on grid maps, it suffers from the curse of dimensionality. The number of cells in an environment representation increases polynomially with the dimensions.

Therefore, future work focuses on applying different heuristics, and include previous experience in the planner in order to boost the plannification process.

## References

1. Sethian JA (1996) A fast marching level set method for monotonically advancing fronts. *Proc Natl Acad Sci* 93:1591–1595
2. Jbabdi S, Bellec P, Toro R, Daunizeau J, Plgrini-Issac M, Benali H (2008) Accurate anisotropic fast marching for diffusion-based geodesic tractography. *Int J Biomed Imaging* 2008:12
3. Li H, Xue Z, Cui K, Wong STC (2011) Diffusion tensor-based fast marching for modeling human brain connectivity network. *Comput Med Imag Graph* 35(3):167–178
4. Yang K, Li M, Liu Y, Jiang C (2010) Multi-points fast marching: a novel method for road extraction. In: Proceedings of the 18th international conference geoinformatics: GIScience in change, geoinformatics, June 2010, pp 1–5
5. Sethian JA (1996) Level set methods. Cambridge University Press, Cambridge
6. Garrido S, Moreno L, Abderrahim M, Blanco D (2009) FM2: a real-time sensor-based feedback controller for mobile robots. *Int J Robot Autom* 24(1):3169–3192
7. Yatziv L, Bartesaghi A, Sapiro G (2005) A fast O(n) implementation of the fast marching algorithm. *J Comput Phys* 212:393–399
8. Adalsteinsson D, Sethian JA (1995) A fast level set method for propagating interfaces. *J Comput Phys* 118(2):269–277
9. Garrido S, Moreno L, Blanco D (2007) Sensor-based global planning for mobile robot navigation. *Robotica* 25:189–199
10. Garrido S, Moreno L, Blanco D (2008) Exploration of 2D and 3D environments using Voronoi transform and fast marching method. *J Intell Robot Syst* 55(1):55–80
11. Valero-Gomez A, Gomez J, Garrido S, Moreno L (2013) The path to efficiency: fast marching method for safer, more efficient mobile robot trajectories. *Robot Autom Mag, IEEE* 20(4):111–120
12. Gomez JV, Vale A, Valente F, Ferreira J, Garrido S, Moreno L (2013) Fast marching in motion planning for Rhombic like vehicles operating in ITER. In: IEEE international conference on robotics and automation, pp 5533–5538
13. Ulrich I, Borenstein J (2000) Vfh\*: local obstacle avoidance with lookahead verification. In: Proceedings of the IEEE international conference on robotics and automation, pp 2505–2511
14. Minguez J, Montano L (2001) Global nearness diagram navigation. In: Proceedings of the IEEE international conference on robotics and automation, Seoul, Korea, pp 33–39
15. Castejon C, Boada B, Blanco D, Moreno L (2005) Traversable region modeling for outdoor navigation. *J Intell Robot Syst* 43(2–4):175–216
16. Alton KR, Mitchel IM (2008) Fast marching methods for stationary Hamilton-Jacob equations with axis-aligned anisotropy. *SIAM J Numer Anal* 47(1):363–385
17. Petres C, Pailhas Y, Evans J, Petillot Y, Lane D (2005) Underwater path planning using fast marching algorithms. *IEEE Oceans 2005 Eur Conf* 2:814–819

# Motion Planning of Large Scale Vehicles for Remote Material Transportation

Alberto Vale and Isabel Ribeiro

**Abstract** The International Thermonuclear Experimental Reactor (ITER) project is a worldwide research experiment that aims to explore nuclear fusion as a viable source of energy for the coming years. Mobile robotics plays an important role in the remote handling systems that perform the maintenance operations in ITER. The Cask and Plug Remote Handling System (CPRHS) is one of the remote handling systems that transports heavy and highly activated in-vessel components between the Tokamak Building and the Hot Cell Building, the two main buildings of the ITER facility. The CPRHS has dimensions similar to an autobus, maximum weight of 100 tons, kinematics of a rhombic like vehicle (two drivable and steerable wheels) and has to move in cluttered environments. The main challenges described in this chapter are the definition of motion planning strategies that cope with the building maps and the cluttered environments. The algorithms were developed and implemented in a standalone application that receives CAD models of the buildings and returns the best trajectories, including reports of the most risky points of collision, and the swept volume of the vehicle along the missions. More than 700 trajectories were computed for different CPRHS types applied in the models of the real scenarios, crucial to proceed with the construction of the Tokamak Building.

**Keywords** Line guidance · Free roaming · Remote handling · ITER and Nuclear fusion facilities

---

A. Vale (✉)

Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, Universidade de Lisboa,  
Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal  
e-mail: alberto.vale@tecnico.ulisboa.pt

I. Ribeiro

Laboratório de Robótica e Sistemas em Engenharia e Ciência, Instituto Superior Técnico,  
Universidade de Lisboa, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal  
e-mail: isabel.ribeiro@tecnico.ulisboa.pt

## 1 Introduction

There is a practical need for developing and exploring nuclear fusion as a source of energy for the humankind benefit. The shortage predictions on fossil fuels, especially with the inevitable oil extraction decline, requires an urgent development and exploration of new sources of energy.

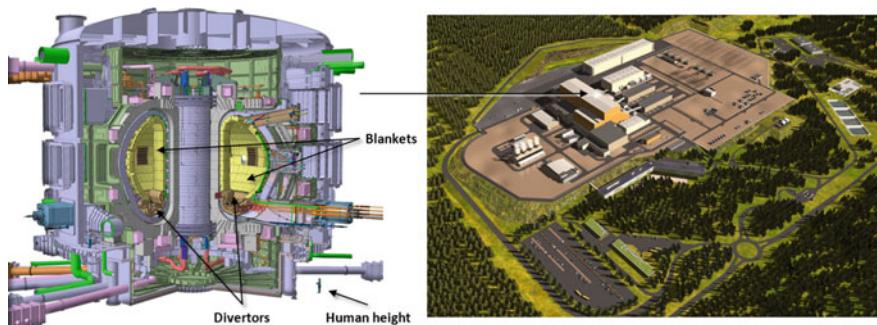
The current energy supply policy is mostly based on fossil fuels (oil, coal and natural gas) representing almost 80 % of the total energy consumption [1]. To worsen this scenario, the world population is expected to grow from 6 to 9 billion people until 2050 [2], resulting on an expressive raise of energy demand.

According to [3], no single technology is likely to provide all of the world's future energy needs and replace the actual oil-based energy infrastructure. It is necessary to achieve a more sustainable mix of fossil fuels and, more importantly, to develop an energy consumption-frame based on new technologies and alternative energies such as solar, geothermal and nuclear, fission and fusion power.

The International Thermonuclear Experimental Reactor (ITER) project is a worldwide research experiment that aims to explore nuclear fusion as a viable source of energy for the coming years. The project is funded by seven member entities: the European Union (EU), India, Japan, China, Russia, South Korea and the United States. The largest experimental tokamak nuclear fusion reactor will be located at the Cadarache facility, in the south of France, as depicted in Fig. 1.

Besides the major scientific objective of exploring the nuclear fusion as a source of energy, the fusion power plants must be safely and effectively maintained through Remote Handling (RH) techniques, due to restrictions on human being in activated areas.

During ITER lifetime, the internal components of the vacuum vessel of the reactor, such as the blanket and divertor modules, will become activated due to exposure to highly energetic neutrons released during the fusion reaction. Additionally, these in-vessel materials might get contaminated with small amounts of radioactive dust.



**Fig. 1** The ITER Tokamak (*left image*) and the scientific buildings and facilities that will house the ITER experiments (*right image*) in Cadarache, south of France

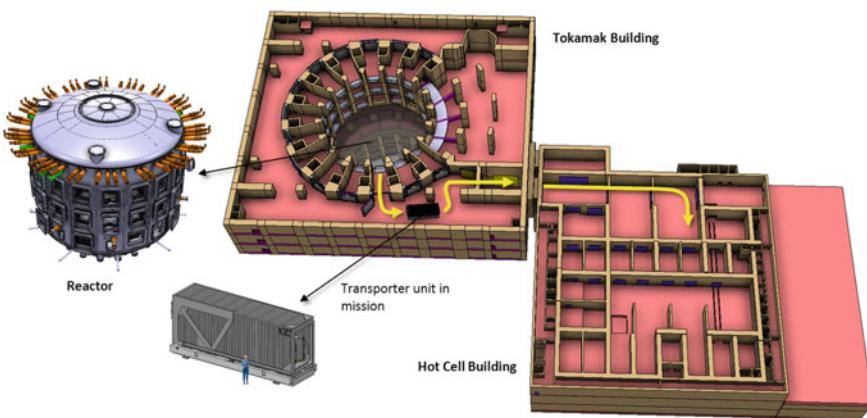
Hence, the components that provide the base functions for the ITER machinery will need to be periodically inspected and upgraded. To manage such operations and provided that human presence will be not authorized in activated areas, the ITER maintenance system will mostly rely on RH devices [4].

## 1.1 The Scenario

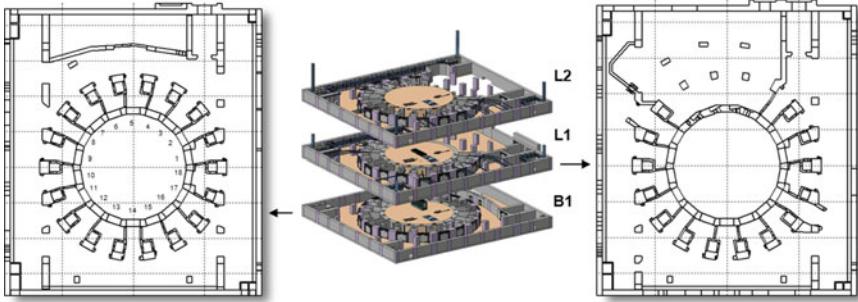
Among the various RH systems that are expected to operate in ITER, this work focus on a large and complex transporter unit that was chosen for the transfer of heavy and contaminated loads between the two main buildings of ITER, the Tokamak Building (TB), lodging the tokamak reactor and with access by vacuum vessel port cells (from this point forward simply identified as “ports”) and the Hot Cell Building (HCB), that will work mainly as a support area. A lift establishes the only interface between the different levels of TB and between the TB and the HCB. Figure 2 represents the interface between the TB and the HCB. It also depicts a mission between one port in one level of TB and a refurbishment docking place in HCB.

The foreseen RH equipment will have a large impact on the design and assembly of the remaining ITER components, namely on building structural aspects and interfaces. Therefore, motion planning studies for the Cask and Plug Remote Handling System (CPRHS) in all of its missions are required for the sake of the feasibility of the ITER buildings design and for the space reservation for the RH missions, carried out by the CPRHS, as described in [5].

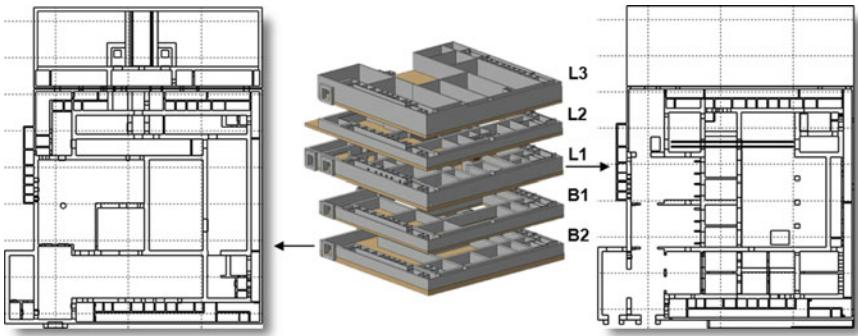
In ITER, the environment in all levels of TB and HCB is mostly composed by static and well structured scenarios, as displayed in Figs. 3 and 4 and each level of the buildings can be modeled using a 2D map representation that will be used for motion



**Fig. 2** A 3D snapshot of the CAD models from the two main buildings in ITER, emphasizing the reactor, the vehicle and a particular mission between the buildings



**Fig. 3** The three levels of Tokamak Building in a split view (*second image*), in particular the 2D maps of the level B1 (*first image*) and the level L1 (*third image*)



**Fig. 4** The five levels of Hot Cell Building in a split view (*second image*), in particular the 2D maps of the level B2 (*first image*) and the level L1 (*third image*)

planning evaluation. The adopted representation for a map is a set of 2D points in the global Cartesian referential defined in the ITER buildings design and a set of line segments. Each line segment connects two different points and it is assumed that there is no crossing between lines. In case of intersection, a 2D point resulted from the intersection is created and each crossed line segment is split in two new line segments, one starting and the other ending in the splitting point, respectively.

In TB, the vehicles can operate in three levels (from bottom to up): B1, L1 and L2, as illustrated in Fig. 3. In HCB, the vehicles may operate in five levels (from bottom to up): B2, B1, L1, L2 and L3, as in Fig. 4.

The entire work developed in this project is applied to the scenario of ITER buildings. However, the same research and development in terms of mobile robots navigation can be applicable to any other type of scenario as warehouses or office type environments.

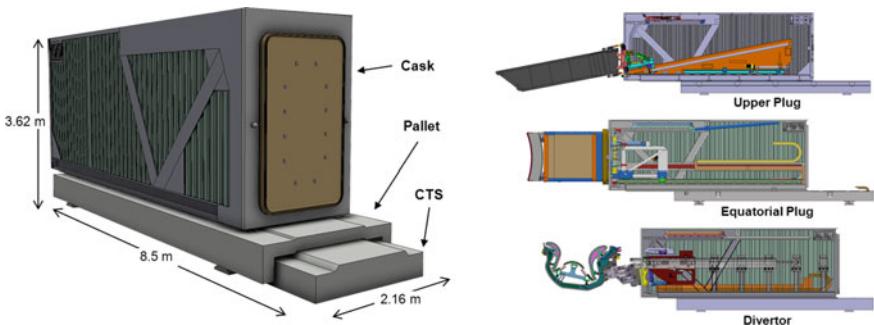
## 1.2 The Vehicle

The CPRHS is a critical element of the ITER remote maintenance system devoted to transportation of components between the TB and HCB. Due to the necessary confinement of contaminated components the CPRHS is defined as Safety Importance Class 1 (SIC-1) plus the mobile nature of the CPRHS brings with it a significant number of complex interfaces with other ITER sub-systems [6]. The geometry of the CPRHS and its payload vary according to the components to be transported and hence, different CPRHS typologies will operate in ITER. As a reference, the largest CPRHS dimensions are  $8.5 \text{ m} \times 2.62 \text{ m} \times 3.62 \text{ m}$  (length  $\times$  width  $\times$  height), as depicted in Fig. 5, and the total weight with the maximum load can reach up to 100 tons.

A CPRHS is composed by three sub-systems: the cask envelope (container that encloses the in-vessel components and the RH tools to be transported), the Cask Transfer System (CTS), which acts as a mobile robot and the pallet (interface between the cask and the CTS equipped with an handling platform to support the cask load and to help on docking procedures). When underneath the pallet, the CTS transports the entire CPRHS, but it can also move independently of the pallet and cask. The CTS has a rhombic like configuration provided by two drivable and steerable wheels, identified as “F”ront and “R”ear wheels, as illustrated in Fig. 6. Given this configuration, the CTS has a higher maneuverability in confined spaces than the traditional cars with Ackerman or tricycle configurations [7].

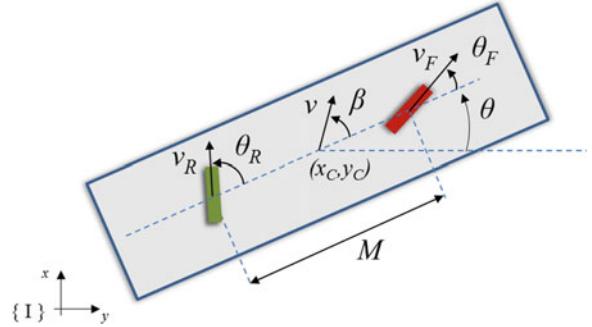
The CTS when operating individually or the CPRHS when carried out by the CTS are, hence, rhombic like vehicles. For simplicity and from this point forward the CPRHS and the CTS when moving alone are identified as the “vehicle”.

As illustrated in Fig. 6, consider the state vector  $q = [x_c \ y_c \ \theta]$  as a representation of the vehicle pose in the frame  $\{\mathbf{I}\}$ , with  $(x_c, y_c)$  the coordinates of the center of the vehicle and  $\theta$  the orientation of the vehicle. Consider  $v$  as the longitudinal speed and  $\beta$  the controllable sideslip angle of the vehicle, both defined in  $\{\mathbf{I}\}$ . The angles and the velocities of the front and rear wheels represented Fig. 6, i.e.,  $\theta_F, \theta_R, v_F$  and



**Fig. 5** The CAD model of the vehicle (*left image*) and three different loads during a disassembling operation (*right image*)

**Fig. 6** The rhombic like configuration



$v_R$ , are the inputs to control the motion of the vehicle, which is not addressed in this work. A kinematic model for a rhombic like vehicle in  $\{I\}$ , that allows the simulation of the vehicle motion directly through the desired longitudinal speed  $v$ , instead of imposing an individual linear speed for each wheel, was introduced in [8] as:

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta + \beta) \\ \sin(\theta + \beta) \\ \frac{\cos \beta \cdot [\tan \theta_F - \tan \theta_R]}{M} \end{bmatrix} \cdot v, \quad (1)$$

where

$$\beta = \arctan \left( \frac{v_F \cdot \sin \theta_F + v_R \cdot \sin \theta_R}{2 \cdot v_R \cdot \cos \theta_R} \right) \quad (2)$$

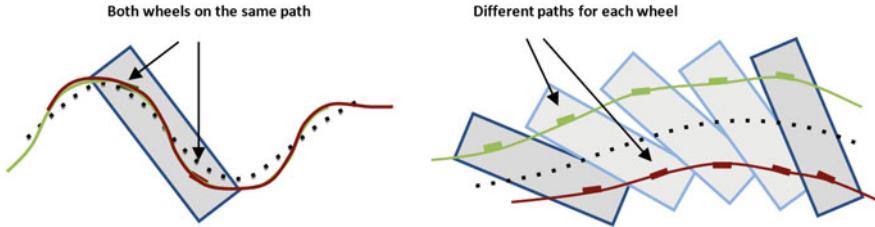
and

$$v = \frac{v_F \cdot \cos \theta_F + v_R \cdot \cos \theta_R}{2 \cdot \cos \beta}. \quad (3)$$

This model entails that the wheels of the vehicle roll without slipping, a constraint inherent to the nonholonomy of rhombic like vehicles, and it also considers a rigid body constraint, common to this type of vehicles, as follows:

$$v_F \cos \theta_F = v_R \cos \theta_R. \quad (4)$$

The drivable and steerable wheels of a rhombic like vehicle are able to follow the same path, in a methodology identified in this work as the line guidance approach, or to follow different paths, herein identified as the free roaming approach, as illustrated in Fig. 7. These two approaches will be addressed later in Sect. 2.

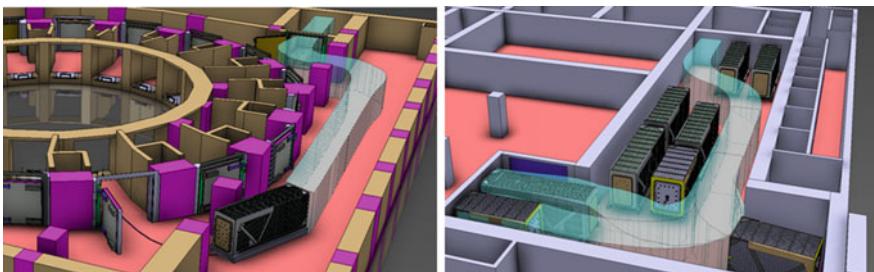


**Fig. 7** The drivable and steerable wheels of a rhombic like vehicle following the same path, feature identified as line guidance (left image) and following different paths, identified as free roaming (right image). The dotted lines represent the path described by the center of the vehicle

### 1.3 Problem Statement

The maintenance operations of transportation in ITER require the vehicle motion throughout the cluttered environments of the TB and the HCB. There are 46 ports in the three levels of TB and one lift that interfaces the different levels of TB and the different levels of HCB. In addition, there are several docking ports and parking places inside the different levels of HCB. The maintenance operations require a mesh of paths between the target points inside the buildings. For instance, the transportation mission of a load for refurbishment requires a path between a port and the lift in TB and then between the lift and a docking port in HCB. During a mission the vehicle sweeps a given volume when follow its path. This volume is important given the scarce free space available in the scenario, or with other vehicles in operation or parked, as illustrated in Fig. 8. The speed along the path is also relevant not only for the mission execution time, but in particular because the dynamics of the vehicle, since it can reach up to 100 tons. From all these reasons, each mission requires an optimized trajectory.

The trajectory optimization problem stated for the vehicle consists on evaluating a trajectory, i.e., a geometric path combined with a speed profile, which guarantees that the vehicle, departing from an initial configuration  $q_S$ , achieves the specified goal,  $q_F$ ,



**Fig. 8** Two missions of remote handling in ITER: from the lift to port 16 in level B1 of TB (left image) and the sequences for moving a parked CPRHS in level B2 of HCB (right image)

without colliding with obstacles and ensuring a safety margin with its surroundings. Specific optimization criteria such as smoothness, path length and obstacle clearance are also considered during the planning phase as well as the vehicle characteristics (dimensions, kinematic and dynamic constraints) and surrounding scenario.

To solve the trajectory optimization problem for each mission, specific information is required, which defines the inputs to the developed algorithms:

1. Environment model: the model of the scenario where the vehicles have to move that constitute relevant information for the definition of a collision free optimal planned solution. From the original CAD models in 3D, it is only used their 2D projection at floor level, as described in Sect. 1.1.
2. Vehicle model: the planning solutions depend directly on the vehicle configuration (geometric, kinematic and dynamic), as described in Sect. 1.2.
3. Initial and goal conditions: the initial and final poses of the vehicle defines the first and the last points of the path, i.e.,  $P_i$  and  $P_f$ .
4. Global trajectory(ies): most of the trajectories in TB share a large common path around the Tokamak, which is identified as a “ring” in each level. First, the path of the ring is evaluated and then, each path for a different port is maximized with the path of the ring. The maximization of different paths is addressed later in Sect. 2.1.5.

Together, these informations define a motion query for the specified mission in the ITER scenarios and are fed as inputs into a trajectory planner. This planner generates a path to be carried out by the vehicles, i.e., a set of Cartesian coordinates (for specific vehicle reference points) and corresponding orientations that geometrically describe the vehicle motion. In addition to the geometric feasibility of the solution, which shall guarantee that the vehicle reaches the goal configuration without colliding with obstacles, it is desirable that the planned solution follows specific criteria requirements:

- Path length: whenever possible, find the shortest possible path, so as to minimize the energy consumption of the on-board batteries.
- Path clearance: increase the minimum distance of the vehicle to the surrounding obstacles of the scenario.
- Path smoothness: the planned solution shall be smooth, minimizing steering maneuvers and jerky motions.
- Maximization of common paths: the vehicle journeys may share common paths through the buildings, wherever possible.

The planner outputs a trajectory to guide the vehicle from an initial configuration,  $q_S$ , to a final configuration,  $q_F$ , using the 2D map representation,  $M$ , and the vehicle model. The geometric solution (a path) is combined with a speed profile, which defines how to move the vehicle along the path at various speeds while satisfying the kinematic and dynamic constraints (maximum/minimum velocities and accelerations).

## 2 Motion Planning

Two main approaches are presented for solving the trajectory optimization problem of rhombic like vehicles: the line guidance and the free roaming (see Fig. 7). The line guidance approach, where both drivable and steerable wheels follow the same path, is used in most of the trajectories. This approach, detailed in Sect. 2.1, outputs an optimized trajectory without maneuvers of alternatively moving forward and backward. In some situations, the resulted trajectory using line guidance is not feasible in the entrances to some ports or to the lift, but may become feasible if including maneuvers. Therefore, an improvement of the line guidance approach using maneuvers is described in Sect. 2.1.4. When the trajectory is not feasible even if using maneuvers, the free roaming approach, detailed in Sect. 2.2, is the last solution adopted. The Sect. 2.3 describes the speed evaluator considered for both approaches, which converts the optimized paths into optimized trajectories. In addition, and in particular for the ITER scenario, specially in the TB, most of the trajectories share a long common part from the single lift in TB to the vicinity of each port of the Tokamak. Therefore, an additional feature was developed for maximizing the common part of different trajectories, which is detailed in Sect. 2.1.5.

The two main approaches, the line guidance and the free roaming, share three stages:

1. Geometric path evaluation: given the environment model and the initial and goal objectives, an initial geometric path is found. At this point the aim is to find a path connecting the initial and goal objectives that can act as an initial condition for the next path optimization stage.
2. Path optimization: this module receives the preceding geometric solution as input and returns an optimized path. The optimization process first applies a spline interpolation to satisfy weaker differential constraints such as smoothness requirements. Afterwards, a clearance based optimization is carried out to guarantee a collision free path that meets the safety requirements. In this study, a minimum safety distance between the vehicle and the obstacles must be guaranteed.
3. Trajectory evaluation: in this final module (described in Sect. 2.3), a velocity function is defined along the optimized path transforming it into a trajectory, which is the output of the proposed planning approach.

The two first stages correspond to the path planning and optimization, while the third stage consists on the trajectory evaluation, including a velocity profile to the path. The two first stages are detailed in the sequel for the line guidance and for the free roaming.

### 2.1 Line Guidance

From previous work of RH in ITER [9], and for safety purposes, buried wired systems were proposed to implement the paths resulted from line guidance. The vehicles

would follow the path, with both wheels following the same path. Given this ITER project requirement, the proposed planning methodology returns directly the path to be followed by the center of the wheels and not the one corresponding to the center of the vehicle. A nominal operation of the vehicle for a specified environment determines a motion between two configurations (2D points with specific orientations). The first step of this planning methodology is to find an initial geometric path, i.e., a set of 2D points, connecting the initial and final configurations.

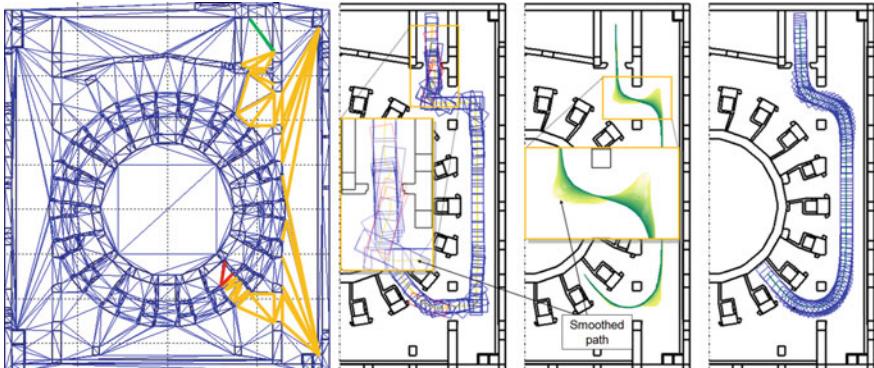
The environment in all levels of TB is composed by static and well structured scenarios. Therefore, the environments can be modeled using a planar map representation, ensuring good geometric properties, like low dimensionality. This encouraged the use of a combinatorial planning approach, instead of other approaches that require more computational effort, as [10, 11]. To handle this first planning objective, the use of a cell decomposition approach is considered, but other combinatorial approaches [12, 13], could also be used. From the cell decomposition approaches, a triangle cell arrangement was adopted, using the Constrained Delaunay Triangulation (CDT) [14], as illustrated in Fig. 9—left.

The adopted representation for a map,  $E$ , is a set of 2D points,  $\mathbf{p}_i$  in a global Cartesian referential of ITER and a set of line segments,  $l_{jk}$ , where each line segment connects two different points,  $\mathbf{p}_k$  and  $\mathbf{p}_l$ , i.e.,

$$E = \{\mathbf{p}_i, l_{jk} | i, j, k = 1, \dots, E_p\} \quad (5)$$

where  $E_p$  is the number of points,  $\mathbf{p}_i = (x, y)$  and  $l_{jk} = \{\mathbf{p}_j + t \cdot \mathbf{p}_k | t \in [0, 1]\}$ .

The CDT, detailed in Sect. 2.1.1, provides the support to generate an initial path, which has a sharp layout and possibly is not feasible. Then, an optimization algorithm



**Fig. 9** Example of the line guidance approach applied in the trajectory evaluation to port 16 in level B1 of TB: the initial map, the Constrained Delaunay Triangulation and the best sequence of triangles painted in a different color (*first image*), the poses of the vehicle along the geometrical path extracted from the sequence of triangles (*second image*), the evaluation of the optimization procedure with the smoothing of the path (*third image*) and the poses of the vehicle along the optimized path (*last image*)

based on elastic bands concept is used to smooth the initial path and to turn it feasible, as detailed in Sect. 2.1.3. In most of the situations, the CDT requires more computational power than the elastic bands, i.e., the initial part of the algorithm is very demanding when compared to the optimization part. Hence, the alternative initialization approach based on Fast Marching Method (FMM), faster than the CDT, is proposed in Sect. 2.1.2. The Sects. 2.1.4 and 2.1.5 details the improvement of line guidance approach with maneuvers and the maximization of common parts of different paths.

### 2.1.1 Constrained Delaunay Triangulation for Initialization

The overall procedure to determine an initial geometric path can using CDT is described:

1. For a specified scenario, the CDT is applied to the corresponding 2D map,  $E$ , yielding a triangle cell decomposition. Let  $C$  denote the set of  $N$  triangle cells so obtained,  $C = \{C_n | n = 1, \dots, N\}$ . Since each triangle has three edges, only some of them may correspond to a wall of the scenario and, therefore, identified as a real edge  $l_{jk}$ . The other edges, possible all of them, are identified as virtual edges since they exist only for computational purposes;
2. To handle a specific motion query, i.e., to connect an initial vehicle configuration,  $q_S$ , to a final configuration,  $q_F$ , the next algorithm's step determines which cells, herein denoted by  $C_I$  and  $C_F$ , contain these two configurations;
3. Using the cell adjacency property, all the possible triangle sequences connecting  $C_I$  to  $C_F$  and composed by consecutive cells that do not share a real edge are evaluated. The desired cells in the sequences are connected by virtual edges, since feasible solutions cannot cross walls represented by real edges. Let  $S = \{S_i | i = 1, \dots, K\}$  be the set of all cell sequences so obtained. If no one of such sequence exists, the algorithm states that there is no solution for the proposed query;
4. The final step converts each cell sequence  $S_i$  into an ordered sequence of points connected by line segments that can be interpreted as a graph. First,  $q_S$  is connected to the middle edge point of the two first cell in each sequence. Then, the middle edge points of two consecutive cells of the sequence are taken as path sample points and are linked by a straight line. Finally, the middle edge point of the last cell in each sequence is connected to  $q_F$ .

The geometric path evaluation module just presented outputs all the possible geometric paths connecting  $q_S$  to  $q_F$  and composed by a set of line segments. To determine the best solution, the shortest path is chosen. To increase the efficiency of the CDT algorithm, the  $A^*$  algorithm [15], is used in alternative to an exhaustive search. Simulations experiments shown how this algorithm can dramatically fasten the search for the shortest triangle cell sequence in complex scenarios such as the TB, that are composed by numerous triangle cells.

Figure 9—first image illustrates the level B1 of TB, the respective CDT of the map and the best sequence of triangles painted in a different color. Figure 9—second

image presents the poses of the vehicle along the geometrical path extracted from the sequence of triangles. This geometrical path is the initial path with a sharp layout and not feasible (clashes with some pillars). The iterative process of the path optimization starting on the initial path is presented in the Fig. 9—third image. A part of the path is zoomed for a better understanding about the modifications along the iterations. The last image in the Fig. 9 presents the poses of the vehicle along the final smooth and feasible path.

### 2.1.2 Fast Marching for Initialization

The initialization path obtained with CDT methodology presents limitations in terms of path smoothness. In complex scenarios, the geometric representation results in a huge number of triangles with rough initial paths still far from the optimal one, as shown in Fig. 9, yielding a large computational effort to optimize the path. These limitations can be overcome by using the Fast Marching Square ( $\text{FM}^2$ ) method [16], providing a better initialization. The  $\text{FM}^2$  is an alternative approach for the initialization, in terms of computational improvement. At the end of the entire process, the optimal final paths are very similar when using the CDT or  $\text{FM}^2$  initializations, as shown later in this chapter.

The FMM is a computational algorithm to solve the arrival time of expanding waves in every point of the space. Conceptually, it can be considered as a continuous version of the Dijkstra's algorithm [17]. It is based on the assumption that the information only flows outwards from the seeding area (wave source). The FMM was proposed by Sethian [18] to approximate the solution of the Eikonal equation, a non-linear partial differential equation encountered in problems of wave propagation [18].

Let assume a 2D map, where  $\mathbf{x} = (x, y)$  is a point on the map with coordinates defined in a Cartesian referential, the front wave arrival time function for every point of the map,  $T(\mathbf{x})$ , and the velocity of the wave propagation  $F(\mathbf{x})$  in each point  $\mathbf{x}$ . Let also assume that a wave starts propagating at  $\mathbf{x}_0 = (x_0, y_0)$  at time  $T(\mathbf{x}_0) = 0$  with velocity  $F(\mathbf{x}) \geq 0$ . The Eikonal equation (6) defines the time of arrival of the propagating frontwave  $T(\mathbf{x})$  at each point  $\mathbf{x}$  of the map, in which the propagation speed depends on  $F(\mathbf{x})$ , according to:

$$|\nabla T(\mathbf{x})|F(\mathbf{x}) = 1 \quad (6)$$

With the discretization of the gradient  $\nabla T(\mathbf{x})$  as in [19], it is possible to solve the Eikonal equation at each point  $\mathbf{x}$ . Using the notation in (7), Eq. (6) can be rewritten as (8):

$$\begin{aligned} T_1 &= \min(T(i - 1, j), T(i + 1, j)) \\ T_2 &= \min(T(i, j - 1), T(i, j + 1)) \end{aligned} \quad (7)$$

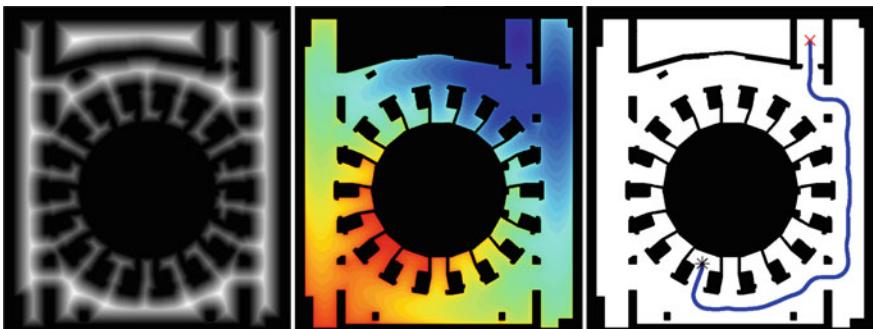
$$\left( \frac{T(i, j) - T_1}{\Delta x} \right)^2 + \left( \frac{T(i, j) - T_2}{\Delta y} \right)^2 = \frac{1}{F(i, j)^2} \quad (8)$$

The FMM consists on solving the Eq.(8) in which all the parameters are known, except  $T(i, j)$ . The process is iterative, starting at the source point of the wave (or waves) where  $T(i_0, j_0) = 0$ . Each iteration obtains the value  $T(i, j)$  for the neighbors of the points evaluated in the previous iteration. Using as an input a binary grid map, in which the velocity  $F(i, j) = 0$  (black) means obstacle and  $F(i, j) = 1$  (white) means free space, the output of the algorithm is a map of distances. These distances are concretely the time of arrival of the expanding wave at every point of the map.

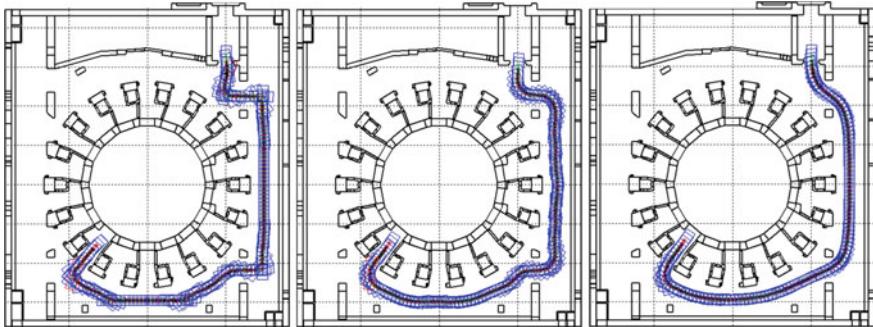
FMM can be directly used as a path planner algorithm. By applying gradient descent from any point of the distance map, a path will be obtained with the source of the wave as a goal point. The paths provided by the FMM are optimal in terms of length, but they do not accomplish the smoothness and safety constraints that most of robotic applications require, because they run too close to obstacles and walls and have sharp curves. However, the FM<sup>2</sup> method includes smoothness and solves partially the safety constraints.

The FM<sup>2</sup> method arises from the application of the FMM twice over the same map. The first time is used to create a map of velocities for the environment, as illustrated in the first image of Fig. 10 and the second time it computes the time of arrival of the wave for every point when the wave is moving at the velocity computed in the previous step. The FM<sup>2</sup> method is based on automatically creating maps of velocities,  $F(\mathbf{x})$ , depending on the environment map in which the velocity of the expanding wave varies depending on the distance to the closest obstacle. The FMM is applied in a first stage to obtain these maps of velocities. In this case, all the obstacles and walls are labeled as wave sources that expand with constant speed. The result is a map of distances in which those cells in the grid that are farther from the obstacles have a higher value, as depicted in Fig. 10—center.

Once this map of distances is computed, it is normalized in order to have values between 0 and 1 and interpreted as relative wave expansion speeds (meaning full stopped or full speed of the wave expansion, respectively). The FMM is applied



**Fig. 10** Map of velocities obtained using all the *black* points of the grid map as a wave source in the FMM (*first image*), map of distances obtained after applying FMM to the map of velocities (*second image*) and the resulted path obtained by applying gradient descent over the map of distances (*third image*)



**Fig. 11** Initial path computed by the CDT (*first image*), initial path computed by the FM<sup>2</sup> (*second image*) and the resulted optimized path (*third image*), which is the same if using any of the previous initializations, but achieved with less iterations if using the second initialization

once again with the goal point as the wave source. During the expansion, the wave will propagate with the velocities indicated in the map previously generated. The propagation ends once the initial point of the path is reached. The resulting map of distances is similar to the one obtained with the standard FMM, but slight differences which make the paths smooth when gradient descent is applied, as shown in Fig. 11. In summary, FM<sup>2</sup> applies FMM twice without any mathematical modification: the first step creates the map of velocities,  $F(\mathbf{x})$ , and the second step computes the time of arrival function,  $T(\mathbf{x})$ , in which gradient descent is applied to find the path. In FMM the map of velocities  $F(\mathbf{x})$  is directly the input binary map.

Similar to the CDT, the FM<sup>2</sup> does not include the rhombic kinematic constraints and, hence, the resulted initial path could be non feasible. Therefore, the optimization process of elastic bands described in Sect. 2.1.3 is required after using any of the initializations methods, CDT or FM<sup>2</sup>. However, the path obtained with FM<sup>2</sup> is smoother and closer to the final optimal solution when compared to the initial geometric paths obtained with CDT, as illustrated in Fig. 11 (a mission from the lift to the port 12, extracted from the several experiments presented in [20]).

The FM<sup>2</sup> avoids local minimum, completeness (finds a path if it exists and notifies in case of no feasible path) and requires less computation power (the map of velocities is independent of the robot shape and pose). In addition, the FM<sup>2</sup> performs a better initialization when compared to the initialization obtained with CDT, as explained in the Sect. 2.1.3. However, FM<sup>2</sup> requires a discretization of the environment and the map of distances has to be computed to each mission, since the final poses are different. In both approaches, CDT or FM<sup>2</sup>, the result is only an initialization path, which is long, with a sharp layout and probably non feasible.

### 2.1.3 Elastic Bands for Smoothing

The initial path retrieved by the CDT or by the FM<sup>2</sup> is not optimized in terms of length, smoothness and most of the times is also non feasible, i.e., results in clashes

with some element in the scenario. An optimization is necessary to smooth the path and to turn it feasible starting from the output of the CDT or the FM<sup>2</sup> initializations. An optimization methodology was implemented, based on the elastic bands method [21]. The original concept associated with this approach appeared in the computer vision field, with the presentation of the so called “snakes” algorithm [22]. A snake is a deformable curve guided by artificial forces that pull it towards image features such as lines and edges. The solution herein proposed with the elastic bands methodology is similar to the snakes approach. Instead of retracting a curve to image features, in the path planning problem, it repels the path out from obstacles. Following this approach, the path is modeled as an elastic band which can be compared to a series of connected springs subjected to two types of forces:

- Internal forces: the internal contraction force simulates the Hooke’s elasticity concept [23, 24], i.e., the magnitude force is proportional to the amplitude of displacement. This modeling approach allows the simulation of the behavior of a stretched band. This is the reason why the paths become retracted and shorter. From this point on, the term “elastic force”,  $F_e$ , is adopted to refer to this force component;
- External forces: the obstacle clearance is achieved using repulsive forces, to keep the path, and consequently the vehicle, away from obstacles.

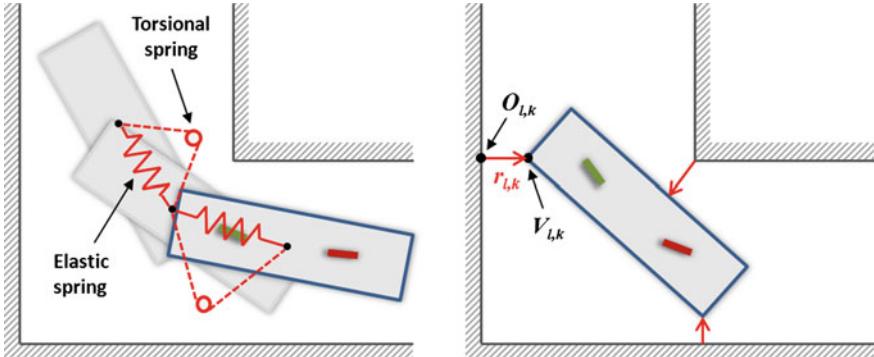
When submitted to these artificial forces, the elastic band is deformed over time becoming a shorter and smoother path, increasing clearance from obstacles. Hooke’s law evaluates the elastic force  $F_e$  applied to path point  $P_i$  as

$$F_e(P_i) = k_e \cdot [(P_{i-1} - P_i) - (P_i - P_{i+1})] \quad (9)$$

where  $k_e$  is the elastic gain and  $P_{i-1}$  and  $P_{i+1}$  are the path points adjacent to  $P_i$ . The elastic band behavior can be controlled through  $k_e$ . The band stretches with high values of  $k_e$  while low values increase the band flexibility.

Using a collision detector algorithm, the nearest obstacle point (OP) to each vehicle pose might be considered. The use of a single OP as a reference to determine the repulsive forces may not be satisfactory to maintain clearance from obstacles due to vehicle dimensions, and therefore, a larger set of obstacle points, such as the k-nearest (k-OPs), must be considered, as illustrated in Fig. 12. This will lead to a more balanced repulsive contribution ensuring effectiveness on most situations. Henceforth, on this formulation, it is considered the set of references formed by the nearest OP to each of the four vehicle’s faces.

In the collision detector algorithm four situations may occur with the nearest points between the vehicle and the scenario: between a corner of the vehicle and a corner of the scenario, between a corner of the vehicle and a wall, between a corner of the scenario and a face of the vehicle and between a face of the vehicle and a wall (the vehicle is in parallel to a wall). In the three first situations, the result is always a point of the vehicle and point in the scenario, defining, as described later, in a distance and a direction for the repulsive force. In the last situation, when the vehicle is in parallel to a wall, an infinite number of points would be expected. However,



**Fig. 12** Elastic band concept: elastic forces to smooth the path (*left image*) and repulsive forces generated by the closest obstacles (*right image*)

any wall or any face of vehicle are line segments. Therefore, when this situation is verified, it is considered the closest points in the boundaries of the line segments.

The overall procedure to evaluate the repulsive force for each path point  $P_i$  is the following:

1. The initial poses (position and orientation) are determined based on the constraint of both wheels placed over the path. These poses were computed considering a forward direction and a backward direction. In the forward direction the rear wheel is fixed on each  $P_i$ . Then, the path point  $P_j$  closest to the front wheel position along the path is determined, such that  $\|P_i - P_j\| = M$ , where  $M$  is the distance between front and rear wheels. The points  $P_i$  and  $P_j$  define the CPRHS/CTS pose. In the backward direction, the same procedure is repeated, but fixing the front wheel for each  $P_i$ , as if the vehicle was executing the path moving backwards. Let  $k = \{1, \dots, K\}$  denote the index of the  $k_{th}$  OP considered on each  $P_i$  related pose and  $l = \{F, B\}$  referring to the forward or backward direction. The  $t_{l,k}$  is the vector defined by the  $k$  obstacle point ( $O_{l,k}$ ) and each wheel point ( $W_F$  = rear wheel and  $W_B$  = front wheel),

$$t_{l,k} = W_l - O_{l,k} \quad (10)$$

This vector defines the repulsive force direction taking into account the position of the wheels. To maintain clearance from obstacles, the force magnitude must vary inversely with the distance of the poses to the obstacles. To carry out this geometric consideration let  $u_{l,k}$  denote the vector taken from the  $O_{l,k}$  to the vehicle nearest point  $V_{l,k}$ ,

$$u_{l,k} = V_{l,k} - O_{l,k} \quad (11)$$

2. Each pair of points ( $O_{l,k}, V_{l,k}$ ) determines a repulsive contribution defined on  $P_i$  given by,

$$r_{l,k}(P_i) = \begin{cases} \frac{t_{l,k}}{\|t_{l,k}\|} f(\|u_{l,k}\|) & \text{if } \|u_{l,k}\| > d_{th} \\ \frac{t_{l,k}}{\|t_{l,k}\|} f_{th} & \text{if } 0 \leq \|u_{l,k}\| \leq d_{th} \end{cases} \quad (12)$$

with  $f$  denoting a monotonically decreasing function, with a maximum reference value,  $f_{th}$ , to avoid outsized magnitude values when a threshold distance,  $d_{th}$ , is exceeded.

3. The repulsive force for each  $P_i$  is determined as a combination of different repulsive contributions, given by (12),

$$F_r(P_i) = k_r \cdot \sum_{l=\{F,B\}} \sum_{k=1}^K r_{l,k}(P_i) \quad (13)$$

with  $k_r$  denoting the repulsive gain.

Once the elastic (9) and the repulsive (13) forces are computed, an update equation procedure that defines the path evolution along each iteration is applied as

$$P_{i,new} = P_{i,old} + k \cdot F_{total}(P_{i,old}) \quad (14)$$

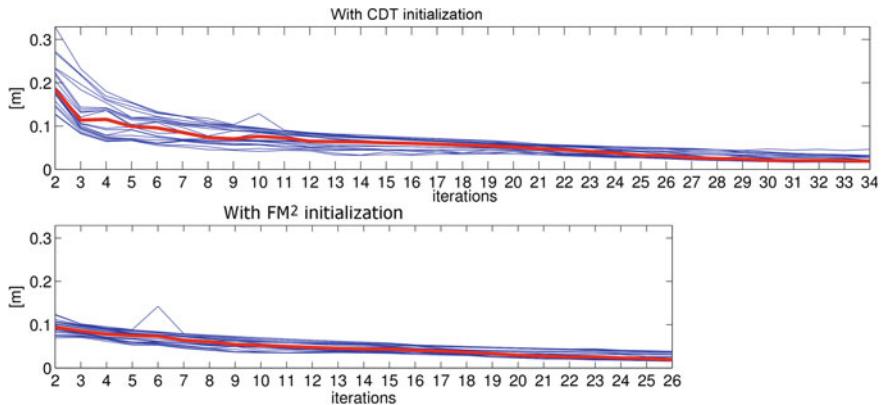
where  $k$  is a normalization factor adding up the total force contribution applied to all points  $P_{i,old}$  and the total force contribution is given by

$$F_{total}(P_{i,old}) = F_e(P_{i,old}) + F_r(P_{i,old}) \quad (15)$$

The stopping criteria is defined by detecting that the magnitude changes on  $F_{total}$  are smaller than a given threshold and by setting a maximum number of iterations.

The path optimization is thus carried out by a path deformation approach where the computed paths are treated as flexible and deformable bands. Elastic interactions smooth the path by removing any existing slack, whereas repulsive forces improve clearance from obstacles. The algorithm is fully described in [25] where the results of line guidance applied in all levels of TB are presented. In case of not enough space, a feasible path may not be possible and the algorithm returns, with warning, a path with clashes or with the minimum distance below a safety margin in certain places.

The computational efforts to obtain the initial path using CDT or FM<sup>2</sup> are similar. However, since the path resulted from FM<sup>2</sup> is closer to the final solution, it is expected that the computational time required by the elastic bands to obtain the final solution is less when using the initial path resulted from FM<sup>2</sup>. The number of iterations to obtain the final path to the port 12 is illustrated in Fig. 13. In this figure, each line represents the variation of each point of the path, i.e., how much the point “moves” in the Cartesian referential when the elastic band is applied. In the first iterations, the points have larger variation, since there are more differences between the elastic forces and repulsive forces. In the last iterations, there is more equilibrium and

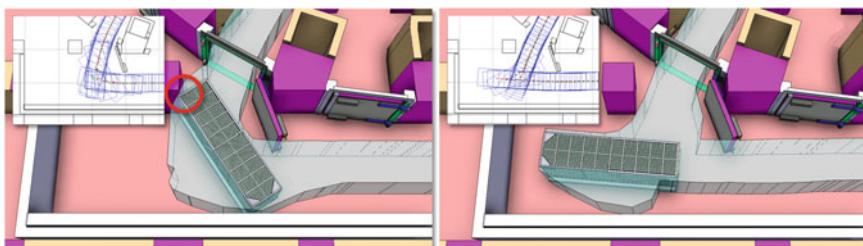


**Fig. 13** Convergence of the elastic bands algorithm when using the CDT initialization (*top image*) and when using the FM<sup>2</sup> initialization (*bottom image*), which requires less iterations

the variation are minimum corresponding to small oscillations. The algorithm stops when the variation is below a threshold value. The elastic bands algorithm takes less iterations to converge using the FM<sup>2</sup> initialization, and thus, less computational time as expected.

#### 2.1.4 Integration of Maneuvers

Due to the confined environment, there are particular situations where the described methodology of line guidance fails to generate feasible solutions, i.e., paths without collisions and above a safety margin. The integration of maneuvers can greatly improve the path planning, by providing a feasible solution where none could be found with the previously methodology and with the advantage of improving the distance to obstacles, as represented in Fig. 14.



**Fig. 14** The line guidance algorithm was not able to generate a path to port 17 in level B1 of TB without clashes (*left image*), but the problem becomes solved with a maneuver (*right image*)

A maneuver exists when the vehicle stops and changes its motion direction (from/to forward to/from backwards). In this paper, a maneuver splits the path in two sub-paths with the constraint that the final pose of the first sub-path is the initial pose of the next sub-path. In both sub-paths the line guidance methodology is considered.

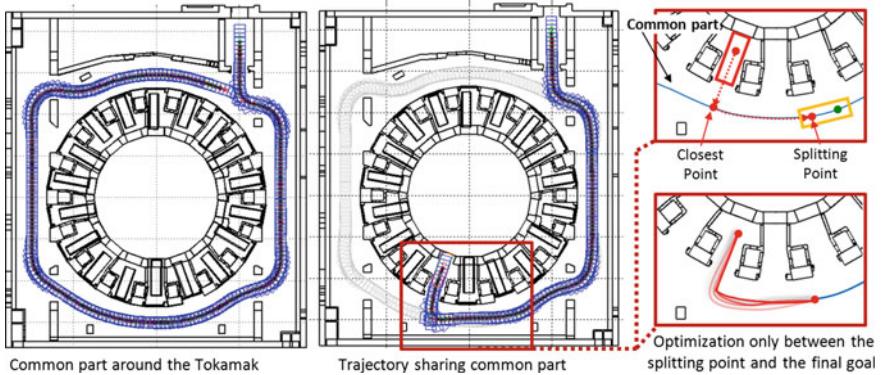
By taking advantage of the vehicle kinematic configuration, the line guidance algorithm was improved to incorporate one or multiple maneuvers. In case  $n$  maneuvers are required, the path is divided in  $n + 1$  sub-paths and the path optimization is applied to each sub-path. An additional constraint has to be taken into account when considering maneuvers. The path should be common to both wheels. However, when following a path, both wheels cannot follow the entire path. For instance, when following in forward direction, it is expected that the front wheel reaches the end of the path before the rear wheel. Similarly, in backwards direction, the rear wheel reaches the beginning of the path before the front wheel. This gap corresponds to the wheel-base,  $M$ . Consequently, this constraint should also be taken into consideration when evaluating maneuvers. Between two consecutive sub-paths of a maneuver, there is a coincident segment of both sub-paths with a length greater or equal to the distance between the wheels.

The decision of including maneuvers is taken when a path without maneuver is not feasible, as illustrated in Fig. 14, where there is a collision, or does not fulfill the minimum safety distance to obstacles. The point(s) of maneuver are introduced manually. It is possible to choose if they are fixed or flexible. In this later case the algorithm can adjust its position during the optimization to obtain the final trajectory. The integration of maneuvers is currently available in [26].

### 2.1.5 Maximization of Common Parts of Different Paths

The geometry of the scenarios in TB and HCB is such that paths for different missions of the CPRHS can, in certain situations, share common parts. In particular, this is noticeable in the galleries around the tokamak where all CPRHS have to travel from/to the lift to/from each of the port cells. The maximization of common parts in different paths minimizes the overall volume required for CPRHS operation, this being a key issue in ITER design and safety. To achieve this goal the line guidance approach described in Sect. 2.1 was improved with a feature to fulfill this objective.

Around the galleries in TB an optimal path to be followed using line guidance navigation is firstly generated, with the initial pose inside the lift and the final pose in the corridor, near the lift, as illustrated in Fig. 15—left. This path is to be used, as much as possible, in all missions from/to the lift to/from each port that are accessible from the gallery. For each of these paths, two issues arise at this stage considering a mission from the lift to a port: (i) where to deviate from this common part of the path to reach a particular port, and (ii) which is the optimal path from this deviation point, to the final goal in the port cell. The last path, i.e., from the deviation point to the final goal, can be optimized using line guidance or free roaming, since the entrance to the port cell is usually critical, given the risk of clashes with the pillars.



**Fig. 15** The common trajectory around the tokamak (*left image*) and the maximization process with the trajectory to port 14 in equatorial level of TB (*right images*)

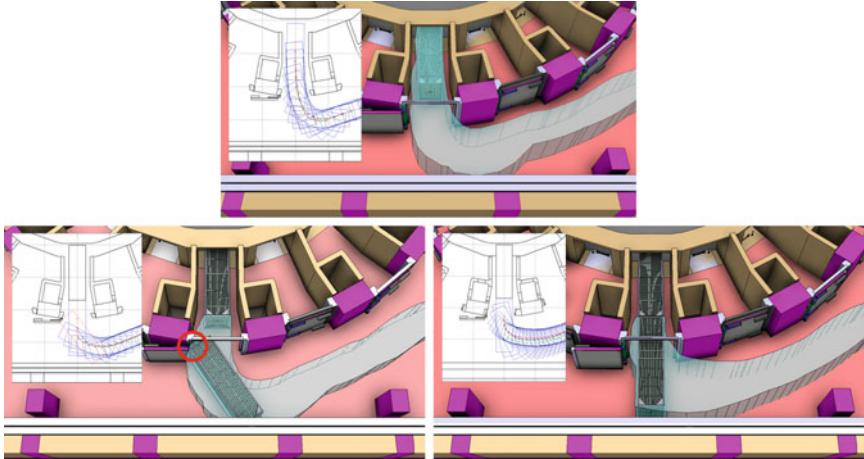
The overall procedure to evaluate an optimized path that considers the maximization of common paths is the following:

1. Assume as input the path starting at the lift and describing a ring around the galleries. This path can be evaluated using the algorithm described in Sect. 2.1.
2. Obtain a second optimized path from the lift to a specific port, using the algorithm of Sect. 2.1.
3. Starting from the end point of the second path (obtained in point 2), the most closest point between the two paths is searched and defined as the Closest Point (CP), as illustrated in Fig. 15. From the CP and crawling backwards the length of the vehicle along the first path a Splitting Point (SP) is defined. The common path is defined between the initial point of both paths and the SP.
4. The path starting in the SP, where the pose of vehicle is frozen, and finishing in the target goal is optimized following the same procedure of line guidance described in Sect. 2.1 or, if it is not possible, using the free roaming described in Sect. 2.2. This means that the path in point 2 is disregarded at this stage.

The resulted path is finally inputted to the speed evaluator, leading to an optimized trajectory.

## 2.2 Free Roaming

The line guidance methodology entails that both vehicle wheels should follow the same physical path and therefore the inherent rhombic flexibility is only partially explored. Figure 16 illustrates part of the scenario in TB of ITER where a CPRHS, acting as a rescue vehicle, has to dock in a Vacuum Vessel Port Cell (VVPC) where another CPRHS is already parked. There is a possible trajectory for the first cask using line guidance, as illustrated in Fig. 16 (top). However, if considering the same



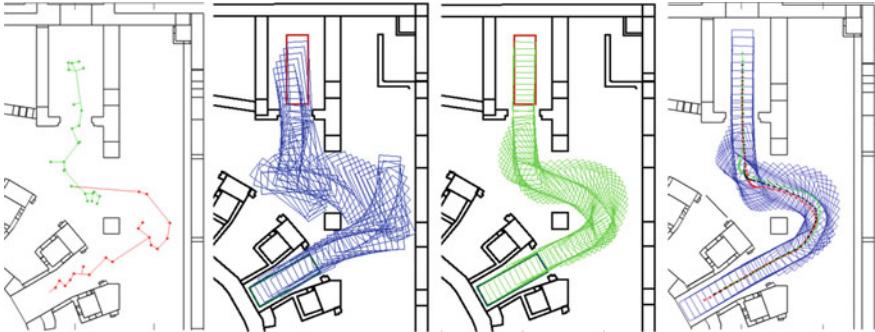
**Fig. 16** Trajectories for port 14 in any level of TB: (top image) the nominal operation is possible with line guidance, (bottom left image) in rescue operation, where a CPRHS is already docked, resulting in collision using the line guidance and (bottom right image) using free roaming results in a feasible trajectory

constraint, the rescue vehicle clashes with the wall, as illustrated in Fig. 16 (bottom left). In several situations as the previous one, no possible trajectory is found for the second cask using the line guidance approach. In case of using independent references for the wheels, i.e., free roaming navigation methodology, a possible path is found, as illustrated in Fig. 16 (bottom right). This solution requires the use of dedicated motion planning techniques, in particular, an efficient path optimization method capable of exploring the high maneuvering ability of the rhombic vehicle.

The proposed free roaming methodology is based on rough paths provided by global planners like the Rapidly-Exploring Random Tree (RRT) [27, 28], or the Probabilistic Roadmap Method (PRM) [10]. The resulted path is optimized using the elastic bands method, but this time, it is redefined, formulating paths as particle systems [21]. Inspired on the rigid body dynamics, consecutive poses along the rough path previously referred are treated as rigid bodies that are repulsion from obstacles through external forces, improving path clearance. Figure 17 illustrates the free roaming process. This formulation allow to explicitly consider the vehicle geometry during the optimization and fully profit from the high maneuverability of rhombic vehicles.

In this approach, the use of rigid body dynamics is restricted to the case of general plane motion, i.e., the particles composing the rigid body move in parallel planes and their motion is neither characterized by pure rotational nor pure translational movements. Therefore, angular variables, such as moments and angular velocities, are scalar quantities.

Consider that a rigid body with a Center of Mass (CoM) denoted by  $C$  is acted by  $N$  external forces,  $F_n$ , with  $\{n = 1, \dots, N\}$ . Following the *Newton's Second Law* and taking the rigid body as a system of particles, the dynamics of  $C$ , with respect



**Fig. 17** The initial path returned by the Rapidly exploring random tree (*first image*), the poses of the vehicle along the path (*second image*), the evaluation of the free roaming (*third image*) and the resulted path (*last image*)

to the inertial frame  $OXY$ , is given by

$$\mathbf{F}_{total} = \sum_{n=1}^N \mathbf{F}_n = m \cdot \mathbf{a}, \quad (16)$$

where  $m$  is the mass of the vehicle (up to 100 tons) and  $\mathbf{a}$  is the linear acceleration of  $C$ . The dynamics of the rigid body motion relative to its body frame,  $CX'Y'$ , is given by

$$\tau_{total} = \sum_{n=1}^N \mathbf{F}_n \times \mathbf{e}_n = I_C \boldsymbol{\alpha}, \quad (17)$$

which entails that the resultant torque about  $C$ ,  $\tau_{total}$ , is a vector with the direction of the angular acceleration,  $\boldsymbol{\alpha}$ , and magnitude  $I_C \boldsymbol{\alpha}$ . In (17),  $I_C$  is the moment of inertia around the perpendicular axis passing through  $C$ , whereas  $\mathbf{e}_n$  corresponds to the position vector of  $\mathbf{F}_n$  relative to the reference frame  $CX'Y'$ . For the case of uniformly accelerated motion, which will be adopted in this formulation,  $\mathbf{a}$  and  $\boldsymbol{\alpha}$  assume constant values over time. From the kinetics viewpoint, the general plane motion of the rigid body can be decomposed as the combination of a translation with linear acceleration,  $\mathbf{a}$ , and a rotation about  $C$  with angular acceleration,  $\boldsymbol{\alpha}$ , given by (16) and (17), respectively.

The path optimization, based on a deformation process, refines and improves the quality of a rough solution path provided by a planner. This rough path, which defines the input for the optimization process, is considered to be a set of collision-free motions connecting the queried initial pose,  $q_S$ , and the final pose,  $q_F$ . From this time forward the rough path will be referred to as query path, as described in Sect. 2.1.3.

In the path optimization process, each of the consecutive vehicle poses that form the query path is treated as a rigid body that is connected with its adjacent poses like a convoy through internal interactions and subjected to external-repulsive forces produced by obstacles in its vicinity. Hence, the path optimization becomes

a path deformation problem, which relies on the principles of rigid body dynamics to iteratively simulate the evolution of each pose on the optimization process. In particular, it is proposed to subject each vehicle pose in the query path to two types of efforts:

- Internal efforts: consecutive poses are kept connected through virtual elastic and torsional springs, which simulate the Hooke's elasticity concept and originate elastic forces and torsional torques. These efforts guarantee smoothness on deformation and help shorten the path, and
- External efforts: repulsive forces repel the rigid poses from obstacles, acting as a collision avoidance feature. Moreover, force eccentricity originates repulsive rotating torques, which re-adapt poses orientation maximizing clearance over the obstacles.

Loosely following the *elastic bands* concept proposed by Quinlan and Khatib in [21], this method, by considering each vehicle pose as a rigid body, enables the path deformation to explicitly consider the vehicle geometry and exploits the rhombic vehicle nature, issues considered here as unattended on similar studies.

The implemented path optimization process based on elastic bands concept is described as follows. Let  $j = \{1, \dots, J\}$  be the index of the consecutive vehicle poses composing the query path, each defined by a configuration vector

$$q_j = \begin{bmatrix} s_j \\ \theta_j \end{bmatrix}, \quad (18)$$

where  $s_j$  and  $\theta_j$  denote the position and the orientation of the pose  $q_j$  relative to a fixed reference frame, respectively. It is stated that  $q_1 = q_S$  and  $q_J = q_F$ .

The elastic force,  $F_E$ , and the torsional torque,  $\tau_T$ , evaluated for the vehicle pose at  $q_j$  are:

$$F_E(q_j) = K_E \cdot [(s_{j+1} - s_j) + (s_{j-1} - s_j)] \quad (19)$$

$$\tau_T(q_j) = K_T \cdot [(\theta_{j+1} - \theta_j) + (\theta_{j-1} - \theta_j)], \quad (20)$$

where  $K_E$ , the elasticity gain and,  $K_T$ , the torsional gain, control the elastic and the torsional avoidance behavior on the path deformation, respectively.

The evaluation of the external efforts due to obstacle proximity relies on a heuristic-based collision detector module, which is capable of determining the set of i-nearest Obstacle Point (OP) to each sampled pose  $q_j$ . The overall procedure to handle the evaluation of the repulsive forces and torques are described as follows:

1. Let  $i = \{1, \dots, I\}$  denote the index of the  $i$ th OP relative to a specific pose  $q_j$ . Let  $u_{j,i}$  be the vector

$$u_{j,i} = V_{j,i} - O_{j,i}, \quad (21)$$

taken from each OP,  $O_{j,i}$ , and the corresponding vehicle nearest point,  $V_{j,i}$ .

2. To improve clearance during path deformation, distance-dependent repulsive forces are defined, where each pair of points  $(O_{j,i}, V_{j,i})$  determines a repulsive

contribution. For a specific vehicle pose,  $q_j$ , the repulsive contributions are defined as

$$r_{j,i} = \frac{u_{j,i}}{\|u_{j,i}\|} \cdot f(\|u_{j,i}\|) \quad (22)$$

where,

$$f(\|u_{j,i}\|) = \max \left( 0, F_{max} - \frac{F_{max}}{d_{max}} \cdot \|u_{j,i}\| \right). \quad (23)$$

In (23), a maximum allowable magnitude,  $F_{max}$ , is assigned to avoid outsized values in the close vicinity of the obstacles, and  $d_{max}$  denotes the distance up to which the repulsive force is applied.

3. For each pose  $q_j$ , the total repulsive force is defined as

$$F_R(q_j) = \sum_{i=1}^I r_{j,i}. \quad (24)$$

Using (17), the net repulsive torque around the  $j$ th pose CoM is defined as

$$\tau_R(q_j) = \sum_{i=1}^I r_{j,i} \times e_{j,i}. \quad (25)$$

The repulsive and elastic forces are combined on a total force contribution as,

$$F_{total}(q_j) = F_R(q_j) + F_E(q_j). \quad (26)$$

Similar approach is valid for the torsional and repulsive torques acting on each pose  $q_j$ . This leads to the definition of a net torque expressed as,

$$\tau_{total}(q_j) = \tau_T(q_j) + \tau_R(q_j). \quad (27)$$

Once determined the efforts acting on each pose, the ensued motion is evaluated through the principles of rigid body dynamics. Equations (16) and (17), are rewritten as

$$a_j = \frac{F_{total}(q_j)}{m} - K_D \cdot v_j \quad (28)$$

$$\alpha_j = \frac{\tau_{total}(q_j)}{I_C} - K_D \omega_j, \quad (29)$$

which provide the linear and angular accelerations for a specific pose  $q_j$ . The last term in the right hand side of (28) and (29) represent damping effects introduced to reduce the oscillatory motion during path deformation. They are controlled through  $K_D$ , herein set equally for both the translational and the rotational motion components.

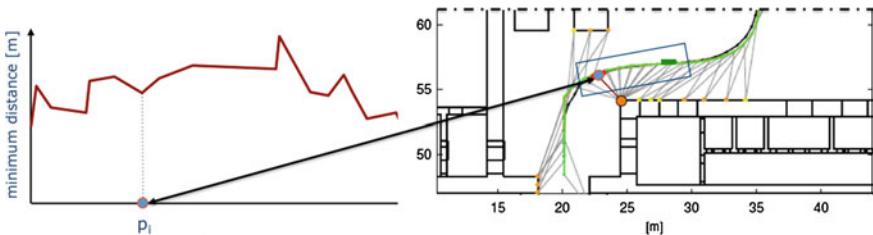
Notice that both  $m$  and  $I_C$  in (28) and (29), do not refer to real vehicle parameters but rather to simple scalars determining the resistance of each pose to change its configuration. The used values in the parameters are presented in Table 1 of Sect. 3.1.

From a starting configuration in the query path,  $q_j$ , this pose is updated iteratively assuming the resulted acceleration and torque, where the referred initial conditions are the previous iterated pose in this process. The stopping criteria is defined by setting a maximum number of iterations. The numerical solution is detailed in [29] and presented with results in all levels of TB and HCB in [30].

### 2.3 From Path to Trajectory

The output of the path optimization module of the planning methodology, using line guidance or free roaming, is a collision free path suitable for execution. To achieve a realistic plan, it is necessary to determine how the vehicle should move along the path satisfying dynamic constraints, i.e., the optimized paths should be parameterized in terms of velocities, converting the paths into trajectories. The definition of the vehicle's velocity along the path assumes a particular importance in this study. The designed trajectories must guarantee that the vehicle performs its motion in the shortest time period satisfying energy minimization. On the other hand, safety requirements are mandatory and the risk of collision shall be reduced. Given the cluttered environment where the CPRHS/CTS moves, an initial approach might define the vehicle speed profile as a function of the distance to the obstacles. The velocity assumes low values when the vehicle is closer to obstacles. Otherwise, the velocity could be higher, under safety levels. To generate this initial speed profile, the minimum distance of each path point in the optimized path to the closest obstacles is identified, as shown in Fig. 18—left. Each value of minimum distance refers to a specific path point,  $P_i$ , and is measured considering the positioning of the vehicle when in this point (front or rear wheel), as if following the optimized path (see Fig. 18—right).

From the evolution of the minimum distance to obstacles, also referred as clearance profile, it is possible to determine an initialization for the vehicle's speed, which



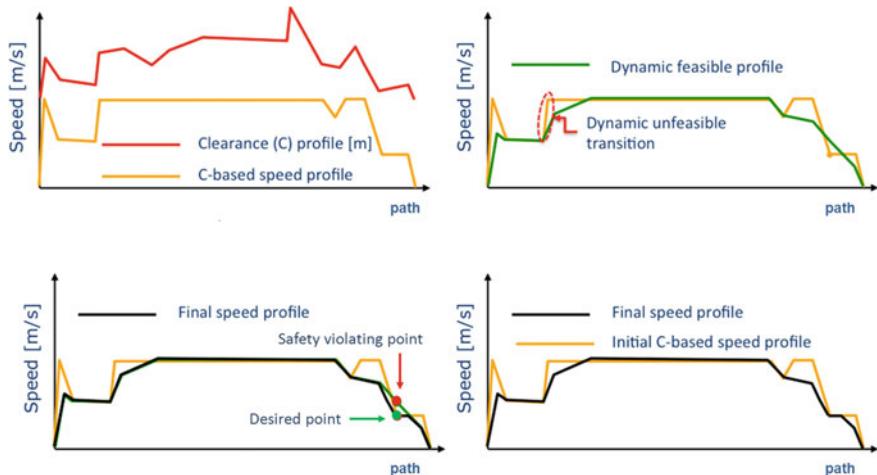
**Fig. 18** Evolution of the minimum distances to obstacles along an optimized path (*left image*) and the respective path in the scenario, part of level B2 of HCB (*right image*)

determines the speed at each reference point  $P_i$ , denoted as  $s_i$ , defined as a linear function of the minimum distance,  $d_i$ , included in the clearance profile as follows.

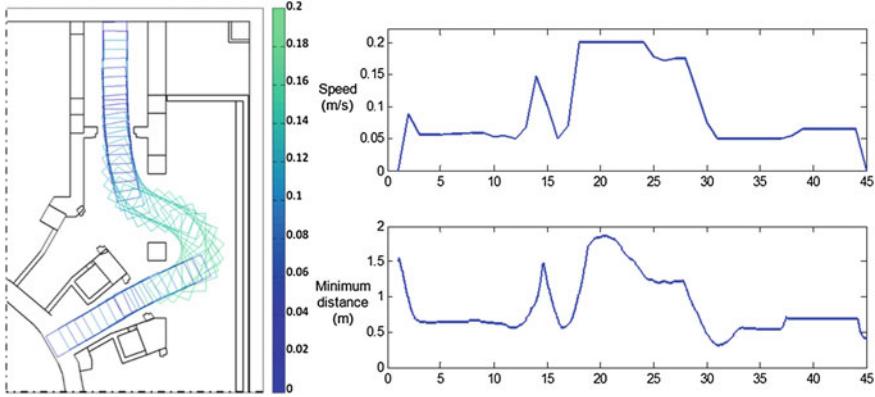
$$s_i = \begin{cases} s_{min} & \text{if } d_i < d_{safe} \\ \alpha(d_i) & \text{if } d_{safe} \leq d_i < d_{th} \\ s_{max} & \text{if } d_i \geq d_{th} \end{cases} \quad (30)$$

A maximum and minimum allowable speed,  $s_{max}$  and  $s_{min}$ , are set to this profile, in order to integrate kinematic constraints. The safety margin is denoted as  $d_{safe}$  and  $d_{th}$  identifies the threshold distance above which  $s_{max}$  is considered. The speed profile thus obtained is saturated when  $d_i$  is above  $d_{th}$  or below  $d_{safe}$  and is referred as C-based speed profile [25], as illustrated in Fig. 19—top left. However, the C-based speed profile is unable to handle vehicle dynamics constraints, meaning that the constraints on the admissible accelerations of the vehicle are ignored. To sidestep this issue, a specific routine was developed, which tests each C-based speed profile transitions, checking whether the accelerations are feasible or not. Whenever a dynamic unfeasible transition is found (e.g., the calculated acceleration is higher than the admissible maximum value,  $a_{max}$ , or lower than the admissible minimum value,  $a_{min}$ ), the routine corrects the speed accordingly. In Fig. 19—top right, the difference between the C-based profile (yellow) and the re-evaluated speed profile based on vehicle dynamics (green) is depicted.

The transition from a clearance based profile to a dynamic feasible profile may lead to a violation of the initial safety restrictions as it is pointed out in Fig. 19—bottom. This means that, in certain situations, the evaluated speed profile may result in values of speed higher than the desired given the obstacle proximity and the maximum level of acceleration/deceleration. When these transgressions are detected, the final



**Fig. 19** Flow diagram for the evaluation of a speed profile that is both compliant with safety (from the proximity with obstacles) and dynamic (from the vehicle) constraints



**Fig. 20** CPRHS speed map for a journey to port cell 2 in level L1 of TB. Each pose is painted in a different color corresponding to a speed value according to the gradient, where the values are in meters per second

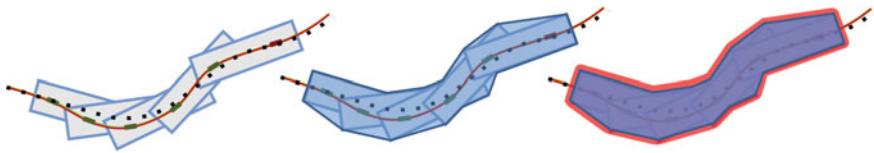
profile is generated by considering both safety requirements and vehicle dynamics constraints as illustrated in the bottom right of Fig. 19. The obtained speed profile can be significantly different from the initial C-based speed profile and when combined with the optimized path, the final optimized trajectory is obtained.

A typical optimized trajectory obtained is shown in Fig. 20 along with the speed variation along the path ( $a_{max} = 0.01 \text{ m/s}^2$ ,  $d_{safe} = 0.3 \text{ m}$ ,  $d_{th} = 1 \text{ m}$ ).

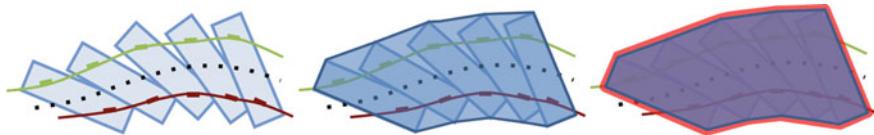
## 2.4 Feasibility Analysis

A vehicle spans an occupancy area when it moves along a path evaluated either using the line guidance or the free roaming methodologies, with or without maneuvers, with or without maximization of common parts among different paths. The occupancy area is evaluated after the path optimization process as follows. The poses are distributed along the path. Then, a polygon is build to fit all the poses. That polygon results from an iterative sum of polygons of each pose, using the polygon clip algorithm described in [31]. The resulted polygon of a path computed by the line guidance is illustrated in the second image of Fig. 21, and computed by the free roaming is illustrated in Fig. 22.

The boundary of the occupancy area means that at least one corner of the vehicle can pass over there. To enlarge the occupancy area with a safety margin, according to the requirements of ITER, a new polygon is computed, containing the initial polygon. The minimum distance between the boundary of the new polygon and the boundary of the initial polygon is always constant and defined by a value identified as the safety margin of 30 cm. The resulted polygons with a safety margin are illustrated in the third images of Figs. 21 and 22.

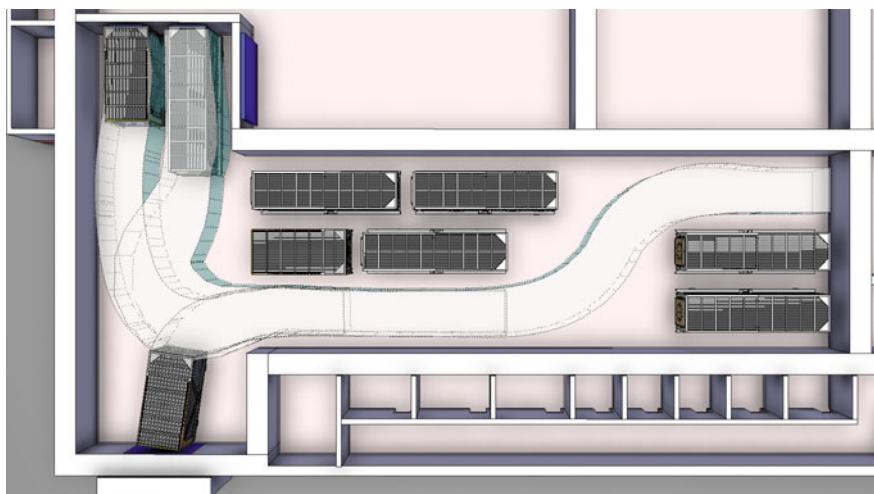


**Fig. 21** A generic path computed by the line guidance algorithm (*first image*), the corresponding occupied area (*second image*) with a safety margin (*third image*)

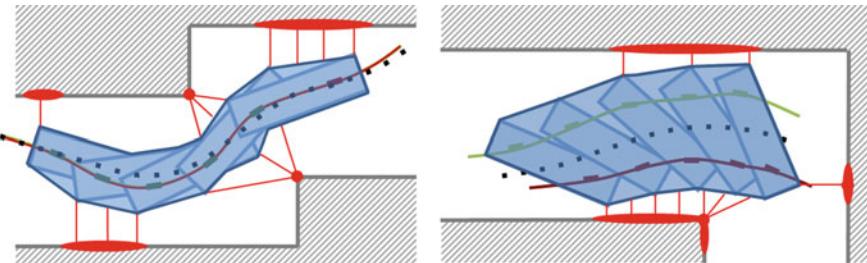


**Fig. 22** A generic path computed by the free roaming algorithm (*first image*), the corresponding occupied area (*second image*) with a safety margin (*third image*)

In terms of 3D, a polygon with the occupancy area is extruded to a swept volume according to the height of the vehicle. This swept volume in the real world results on an additional constrain to the scenario. No other physical elements can be placed in this volume, which must be a free area, as illustrated in Fig. 23. Besides the trajectory optimization, the swept volume is another important output required by the ITER Organization for the buildings design. By integrating the swept volume of a specific mission with new building CAD models it is possible to, immediately, identify if clashes exist and hence, if that mission is feasible.



**Fig. 23** Examples of the swept volume of the casks in level B2 of HCB, where no other elements can be placed



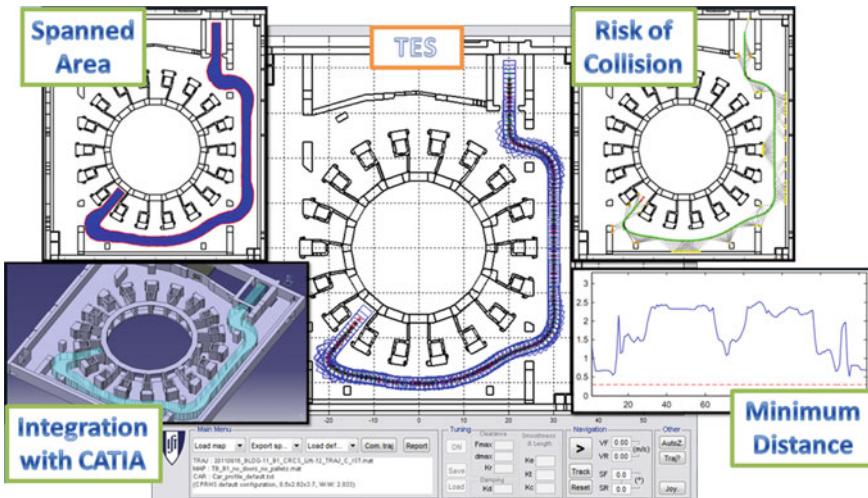
**Fig. 24** The most closest points between the scenarios and the paths evaluated by the line guidance (*left image*) and by the free roaming (*right image*)

In addition, combining the occupancy area with the building model, it is possible to identify the closest points between the scenario and the vehicle when it follows a path. The feasibility analysis becomes a geometrical problem to find the closest points between the polygon of the occupancy area and the segment lines of the scenario. As a result, the most critical points of the scenario, i.e., the points with the highest risk of collision are identified, as illustrated in Fig. 24. If there is an intersection with the polygon, a clash exists.

### 3 Results

The algorithms were implemented in the specially designed software tool Trajectory Evaluator and Simulator (TES), developed under the grants F4E-GRT-016 and F4E-GRT-276-01 of the European Joint Undertaking for ITER and the Development of Fusion Energy. The TES was developed to generate trajectories using line guidance and free roaming approaches, for the evaluation of common parts of different trajectories and for the evaluation of the 3D volume swept by the vehicle in CATIA V5R19 format. The TES receives the models of the buildings and of different vehicles typologies and exports the optimized trajectories and the corresponding 3D swept volume directly to CATIA V5R19.<sup>1</sup> The TES provides also a GUI to preview the trajectory optimization, to manipulate the scenarios (for instance to test modifications in the port doors aperture configuration if necessary), to easily choose the vehicle typology to be used in the simulation and to generate results. Snapshots of TES are presented in Fig. 25, illustrating some features, including the exportation to CATIA V5R19. The output of TES is a set of optimized trajectories, which, under the scope of the previously mentioned grants, were validated by an independent software developed by ASTRIUM SAS [32]. Additionally, TES provides trajectory information

<sup>1</sup>CATIA V5R19 is the required Software tool by the Fusion For Energy (the ITER European Domestic Agency) to work with ITER CAD models.



**Fig. 25** Snapshots of the Trajectory Evaluator and Simulator (TES) software application

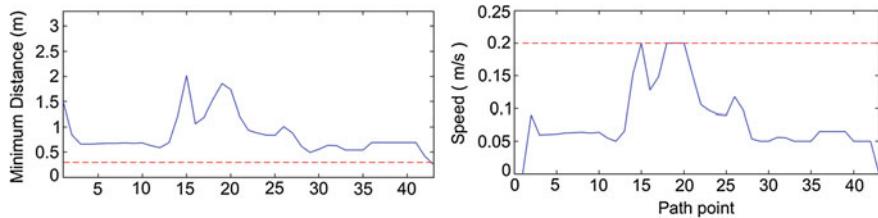
that supports the comparison of the different trajectory scenarios, i.e., both wheels following the same path or following different paths, the ability to evaluate the risk of a clash and the time duration of the journey of a mission.

### 3.1 Trajectories Simulated Results

The results were generated using the maps of the three levels of TB and the four levels of HCB for trajectory optimization (including trajectories for nominal operation of transportation, rescue and parking in HCB). Most of the trajectories are between the lift and a port in TB, and a docking port or a parking place in the HCB. Wherever possible, the line guidance approach is selected to find an optimized trajectory. Figure 26 illustrates an optimized trajectory using line guidance between the lift and the port 2 of level B1 in TB. Figure 26—left illustrates the common path of both wheels, the paths described by the wheels and by the center of the vehicle and a sample of poses along the path. Figure 26—center represents the most critical points of the scenario associated to the path, i.e., the closest obstacles to the vehicle when it crosses each point of the path. The risk level is coded by a gradient color, where red corresponds to the most risky situations. It is noticed that the critical points are most of the times the environment's corners, in particular the entrances to the lift and to the ports. The third image on the right represents the area swept by the vehicle when following the optimized path, i.e., the reserved area where no other elements are allowed aiming at avoiding collisions. Around the swept area it is also presented an extrusion of 30 cm that represents a safety margin; if an element is placed there, no collision is verified,



**Fig. 26** Trajectory to port 2 in level B1 of TB using line guidance (from *left* to *right*): path and sampling poses, the most closest obstacles and the area spanned by the vehicle along the path, with a safety margin

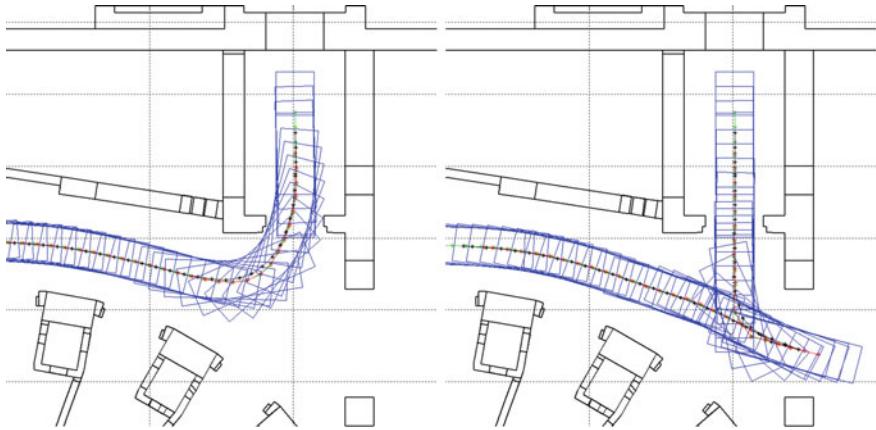


**Fig. 27** The minimum distance to the closest obstacles (*left image*) and the speed profile (*right image*) along the trajectory to port 2 in level B1 of TB using line guidance

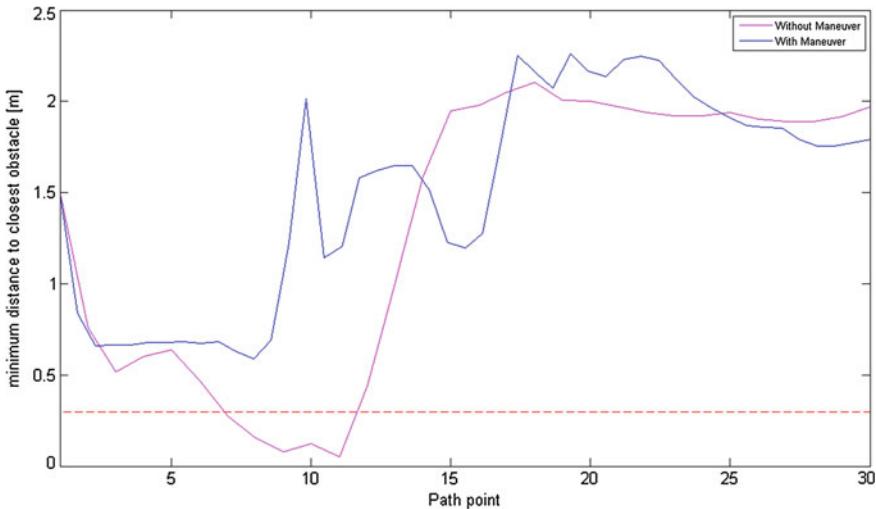
but the risk of clash is high. In Fig. 27 it is quantified the minimum distances to obstacles along the path and the speed profile for the trajectory presented in Fig. 26.

In some situations, the free space constrains the usage of line guidance as illustrated in Fig. 28—left: the optimized path may not result in collision but the safety margin in the entrance to the lift is not fulfilled. Therefore, a maneuver is adopted, as illustrated in the right side of Fig. 28. The minimum distances to obstacles are larger in a trajectory with a maneuver, as illustrated in Fig. 29, mainly between the points 6 and 12 of the path. Each path is a set of points and the horizontal axis in Fig. 29 represents the indexes of those points.

Figure 30 illustrates the example of port 14, where line guidance can not provide a feasible trajectory, even if including maneuvers. The solution is to adopt the free roaming approach. Given the requirement of maximizing the common parts of different trajectories, the free roaming is only adopted in the vicinity of the entrance to the port, while the other part of the trajectory is accomplished using line guidance.



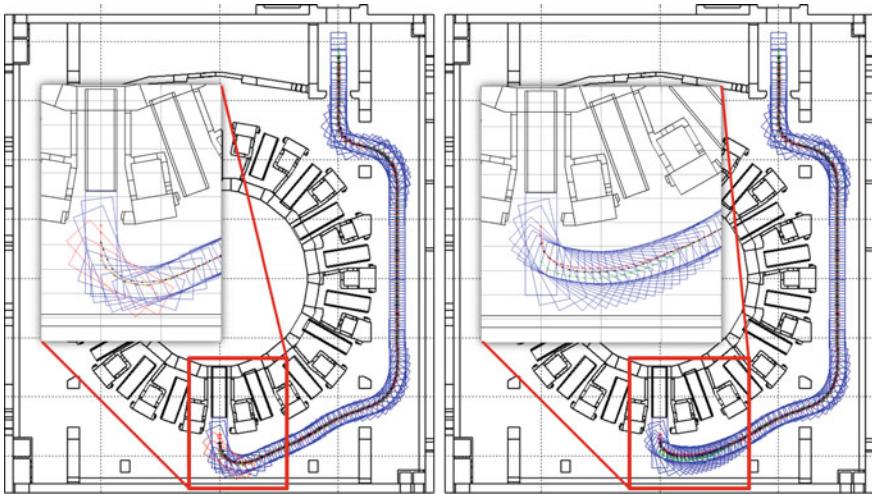
**Fig. 28** Optimized trajectory in the vicinity of the lift using line guidance without maneuver (*left image*) and with a maneuver (*right image*)



**Fig. 29** The minimum distance to the closest obstacles with and without the maneuver in the vicinity of the entrance to the lift

The line guidance is always adopted as the first choice for the trajectory optimization. However, all the feasible trajectories evaluated using the line guidance approach can also be evaluated or improved using the free roaming approach. The line guidance approach can be seen as a particular case of the free roaming, i.e., when the paths of both wheels are coincident free roaming becomes the solution.

In terms of ITER requirements, if a trajectory evaluated using line guidance is feasible for a particular mission, the free roaming is not studied. Even though and for the purpose of this paper. Figure 31 shows the comparison between the two

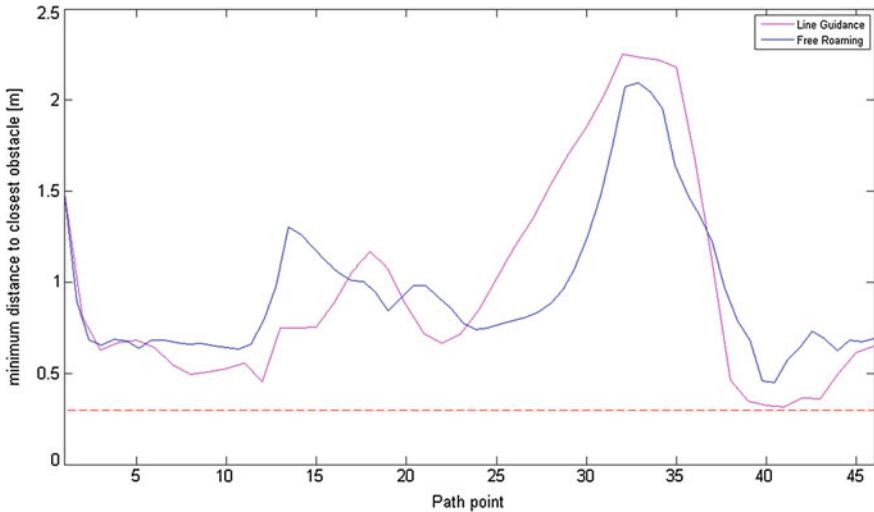


**Fig. 30** Comparison of non feasible trajectory using line guidance (*left image*) and the feasible trajectory using free roaming (*right image*) to port 14 in level B1 of TB



**Fig. 31** Trajectory to port 1 in level B1 of TB (from *left* to *right*): line guidance and the corresponding spanned area, free roaming and the corresponding spanned area

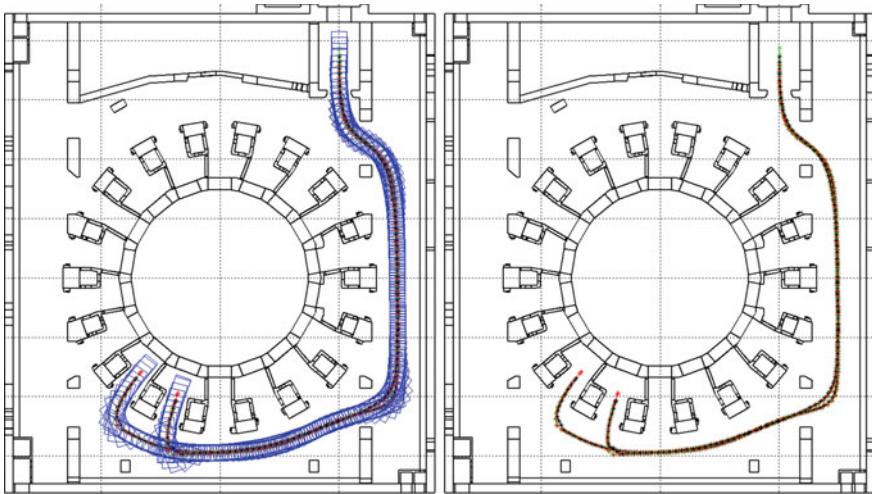
approaches for port 1 in level B1 of TB, where both approaches are feasible. The trajectories are similar, but in the vicinity of the entrances to the lift and to the port, the differences between the paths of each wheel are more relevant. In addition, the space swept by the cask when moving along a path evaluated by the free roaming approach is larger when compared with the line guidance approach. In the example illustrated in Fig. 31, the total swept area using line guidance is  $173.3 \text{ m}^2$ , while using the free roaming is  $182.2 \text{ m}^2$ , i.e., 5 % less. However, the major difference in both approaches is the distances to the closest obstacles when the vehicle is moving along the paths. As illustrated in Fig. 32, the free roaming provides a better safety distance when the



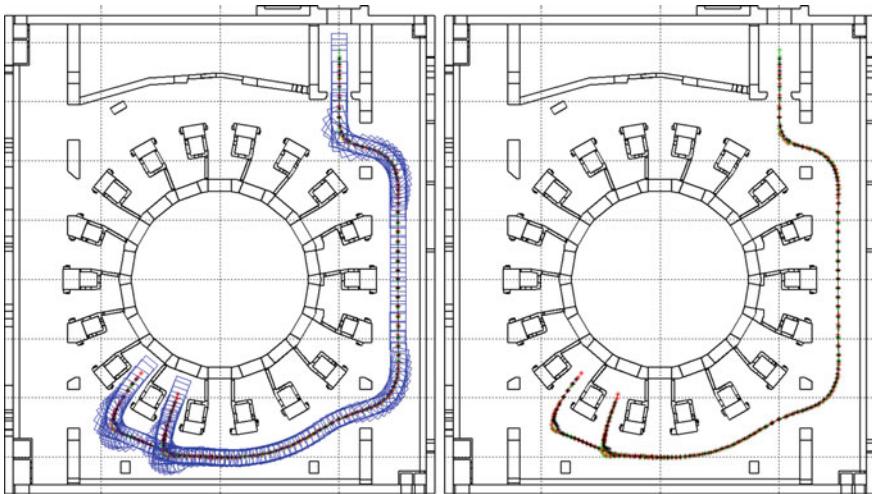
**Fig. 32** Comparison of minimum distance to the closest obstacles between line guidance and free roaming trajectories to port 1 in level B1 of TB

vehicle is closer to the obstacles. In particular, in the entrance to the port (around the trajectory point 40), the minimum distance to the closest obstacle along the path evaluated by the line guidance approach is closer to the safety margin, while the free roaming approach results in a trajectory with additional 15 cm of safety margin. It is a substantial value when the minimum safety margins allowed in ITER are between 10 and 30 cm.

The maximization of common parts from different trajectories is more noticeable for long trajectories, as in the examples illustrated in Figs. 33 and 34: trajectories for ports 12 and 13 in level B1 of TB. Figure 33 illustrates the trajectories evaluated separately. The trajectories are quite similar, with the exception in the vicinity of the ports or closer to the pillars. Figure 34 illustrates both trajectories to the same ports, but maximized with the common path around the tokamak. The main difference is visible outside the lift, since the common path around the tokamak took into consideration the other trajectories that are counter clockwise and requires maneuvers (see the example of Fig. 14) and also the trajectory for port 2 (see Fig. 26). The resulted trajectory of maximization the common part seems to have less waypoints when compared with the trajectories that were evaluated individually. The number of points are equal, with the difference that the points are exactly the same for both trajectories maximized with the common part until the splitting point of the first trajectory (to the port 13). The minimum distances along the trajectories to port 12 and 13 obtained with and without the maximization of the common part are plotted



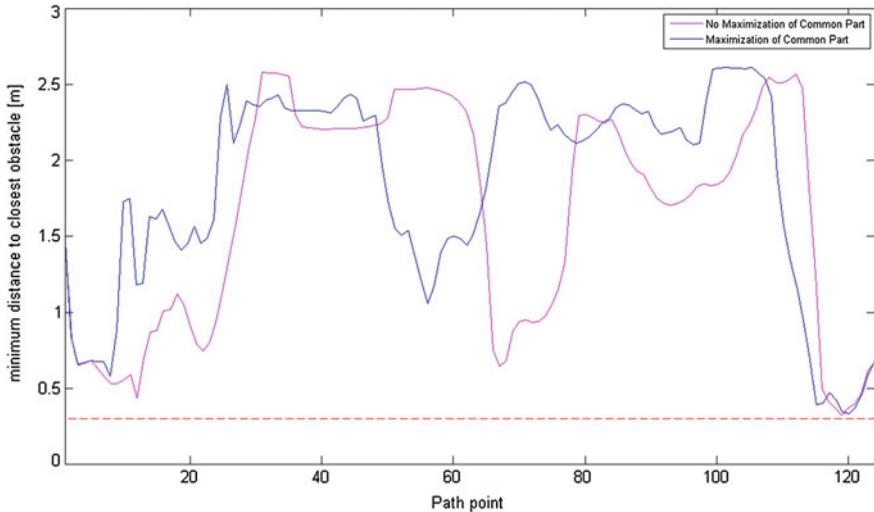
**Fig. 33** Trajectories to ports 12 and 13 in level B1 of TB using line guidance without maximization of common parts: sampling poses (*left image*) and the waypoints (*right image*)



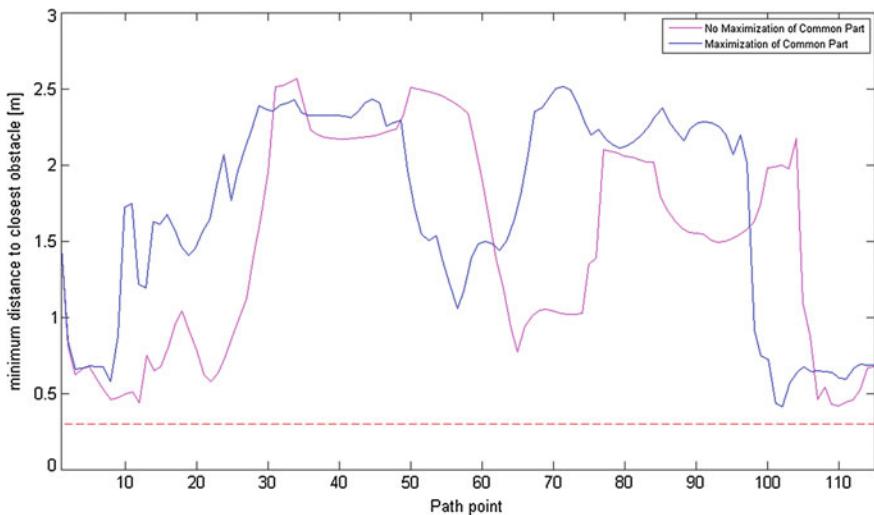
**Fig. 34** Trajectories to ports 12 and 13 in level B1 of TB using line guidance with maximization of common parts: sampling poses (*left image*) and the waypoints (*right image*)

in Figs. 35 and 36. The trajectories resulted from the maximization of common parts provide better results since they decrease the effect of the smoothness and increase the clearance. However, the minimum values, which corresponds to the entrance to the ports, remain approximately the same.

A total of 536 trajectories were optimized: 304 for the 4 levels in HCB and 233 for the 3 levels in TB, for different cask typologies and using the values presented



**Fig. 35** Comparison of the minimum distances to the closest obstacles using line guidance trajectories to port 12, with and without maximization of common part

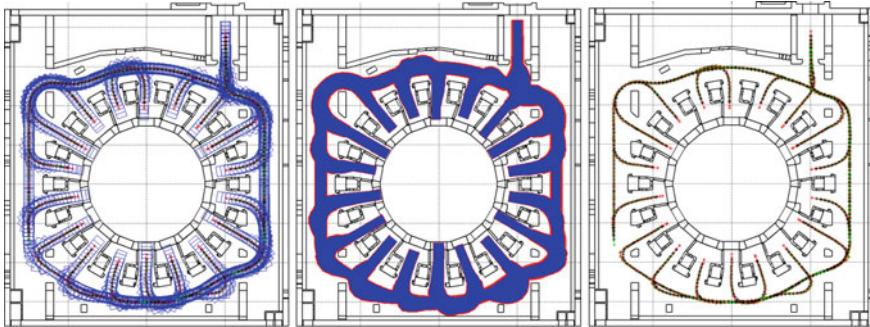


**Fig. 36** Comparison of the minimum distances to the closest obstacles using line guidance trajectories to port 13, with and without maximization of common part

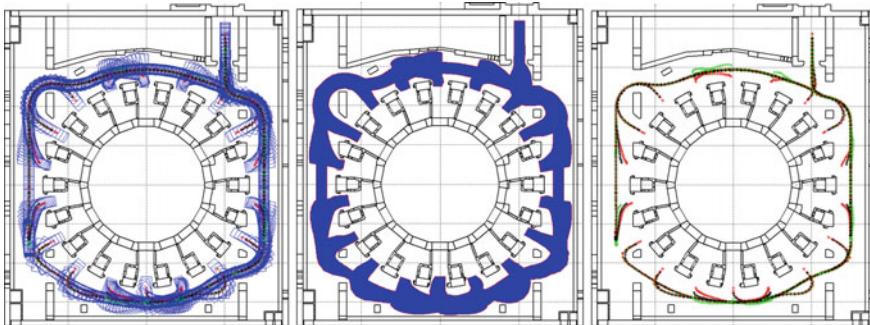
in Table 1. The 3D CAD models of the occupied volumes were generated for all trajectories. Figure 37 illustrates all the trajectories for the nominal operations and the respective occupied volume in level B1 of TB. The common path is similar to a ring around the tokamak. Figure 38 illustrates all the trajectories for the rescue operations

**Table 1** Values used in the experimental results, where  $m$  and  $I_c$  were assumed as 0.5 and 3.29, respectively

Casks	Line guidance				Free roaming					
	$K_e$	$K_r$	$F_{max}$	$d_{max}$	$K_c$	$K_e$	$K_d$	$K_t$	$F_{max}$	$d_{max}$
CPRHS	0.3–0.4	0.05–0.1	1	1	0	1	2	300	1	1
CTS	0.3–0.5	0.05–0.1	1–1.5	0.5–2	0	1	2	300	1	1
Rescue	0.3–0.5	0.05–0.1	1–1.5	0.5–2	2	5	1	500	1	1
Parking	0.3–0.5	0.05–0.1	1–1.5	0.5–2	0	1	2	300	1	1

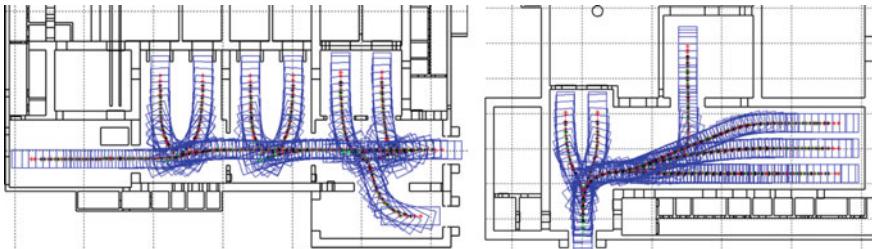


**Fig. 37** The optimized trajectories for all ports in level B1 of TB (*first image*), the respective spanned area (*second image*) and the paths described by the wheels (*third image*)



**Fig. 38** The optimized trajectories for rescue mission for all ports in level B1 of TB (*first image*), the respective spanned area (*second image*) and the paths described by the wheels (*third image*)

and the respective occupied volumes in level B1 of TB (in rescue operations it is assumed a docked cask inside the port). Figure 39 illustrates all the trajectories for the nominal operations in levels B2 and L1 of HCB.

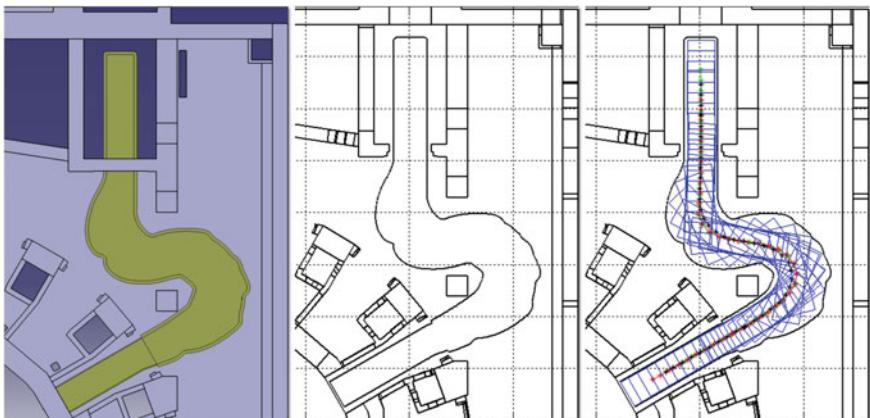


**Fig. 39** The optimized trajectories for all ports in level L1 (*left image*) and in level B2 (*right image*) of HCB

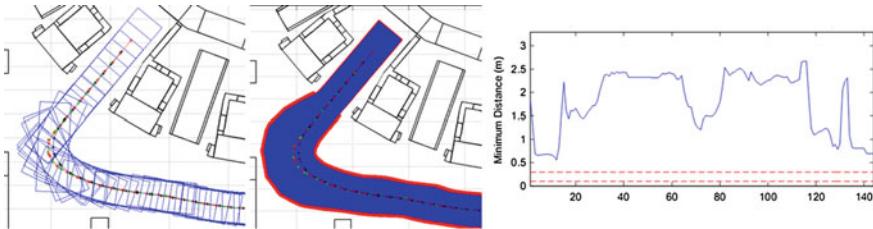
### 3.2 Feasibility Analysis Results

The main goal of the entire work is the motion planning inside the ITER buildings. However, throughout the project, a new challenge was proposed. Despite of walls, pillars and doors, the model of the scenario includes a proposed volume where the vehicle is allowed to move. Therefore, the new challenge is to study the feasibility of the proposed volume, i.e., if it is possible to evaluate a path inside the proposed volume.

For the sake of the work related with the analysis of volume feasibility, a proposed volume is considered feasible if there is at least one feasible trajectory that fits the volume. It is applicable to line guidance or free roaming. To find a feasible path, it is considered a virtual map that results from the 2D projection of the proposed volume, as illustrated in Fig. 40 for the particular case of the mission from lift to the port cell 2 in level B1 of TB. The first image is a snapshot of the 2D view of the proposed



**Fig. 40** The CAD model with the allowed space for the cask navigation (*first image*), the respective geometric representation (*second image*) and the optimized path in the allowed space (*third image*) to the port cell 3 in level B1 of TB

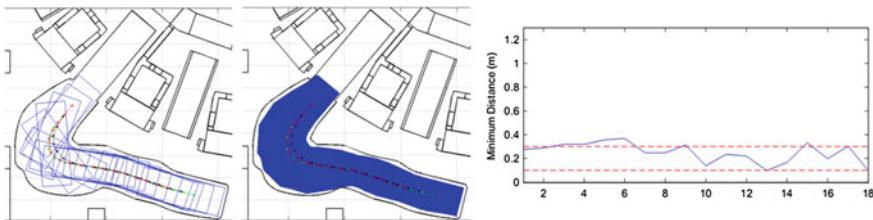


**Fig. 41** The line guidance optimized path to port 12 of level B1 in TB (*first image*), the respective swept area (*second image*) and the distance to the closest obstacles along the path (*third image*)

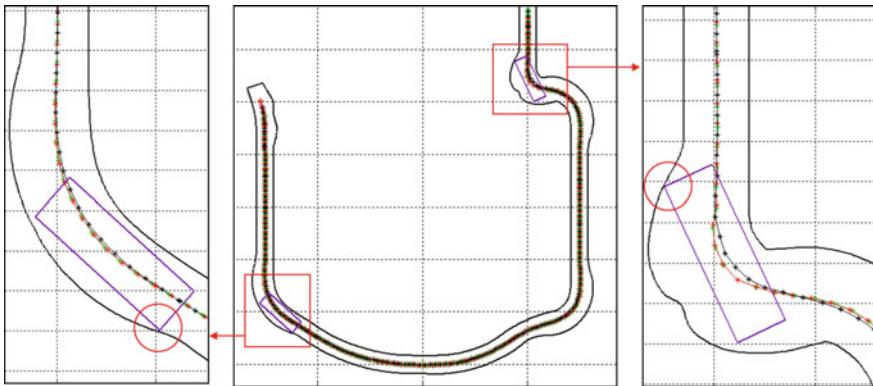
volume in CATIA software, the second image the 2D projection plotted in TES, the last image is an optimized path inside the proposed volume and hence feasible.

When no volume is proposed, there are more free space available in the scenario to find a path, as depicted in Fig. 41. In that case, the path is optimized in terms of smoothness and clearance. The distances along the paths are evaluated against the closest obstacles of the scenario and defines the safety of the mission following that path. When a volume is initially proposed, the space is highly constrained and the goal is only to find a feasible path, as illustrated in Fig. 42. In this case, the distances along the path are evaluated to the proposed volume, leading to lower values.

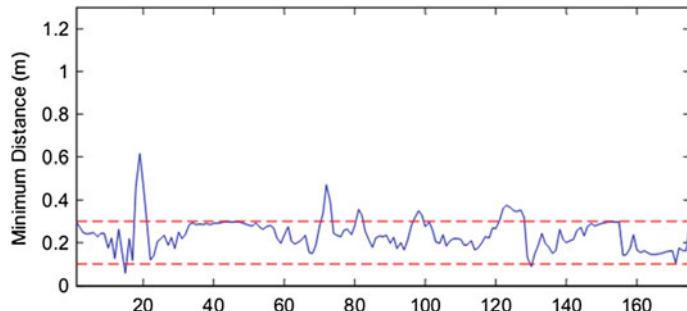
Another important feasibility analysis is the entire trajectory around the Tokamak, called the “ring”, which corresponds to the common part of all trajectories. As example, the Fig. 43 illustrates the proposed volume for level B1 of TB, where a possible path totally inside the proposed volume was identified. The path is represented by the lines described by each wheel and by the center of the vehicle. Figure 44 presents the minimum distances between the vehicle along the path used for the feasibility analysis and the proposed volume that corresponds to the ring. The ring is feasible, since a path was found. However, there are two critical poses where the minimum distance is below 100 mm, which only occurs in the vicinity of the entrance to the lift and slightly above the 100 mm closer to the port 11.



**Fig. 42** The feasibility analysis to port 12 of level B1 in TB (*first image*), the comparison between the proposed area and the swept area (*second image*) and the distance between them (*third image*)



**Fig. 43** The best path inside the allowed space for the *ring* in level L1 of TB and the two critical situations where the minimum distance between cask and the boundary is below a threshold value (*left and right images*)



**Fig. 44** The minimum distances between the cask poses and the boundaries of the allowed volume along the path for the ring in level L1 of TB

Different volumes were proposed for each port in all levels of TB, given the different typologies of the cask expected for operation. As result, a total of 233 trajectories were calculated in TB (3 trajectories for the rings, 46 trajectories for the feasibility analysis of volumes associated to the cask).

## 4 Open Issues and Future Work

This paper presented the trajectory optimization strategies developed for Remote Handling systems that will operate in ITER, for transporting heavy and highly activated in-vessel components between the Tokamak Building and the Hot Cell Building. Two main approaches were developed for trajectory optimization, providing smooth paths that maximize the clearance to obstacles, taking into account the

features of rhombic like vehicles: line guidance (both wheels following the same path) and free roaming (different paths for each wheel). Whenever possible, the line guidance approach with or without maneuvers is adopted as the final solution. If not possible, the free roaming is selected. The trajectories were evaluated maximizing the common paths in the same level of each building. The maximized path is line guidance, since both wheels follows the same path. The branches to each port could also be line guidance or free roaming.

The two main approaches were implemented in a standalone application that receives the 3D CAD models of each level of the buildings, converts them into 2D models and, using the specifications of the missions and the models of the vehicles, returns the best trajectories, including a report of the most risky points of collision and the swept volume of the vehicle along the missions to CATIA V5R19 software. More than 500 hundred trajectories were evaluated, most of them using line guidance, some of them with 1 and few with 2 maneuvers. The most critical points are in the vicinity of the pillars and in the entrance to the lift of ports, where sometimes the free roaming is the only feasible solution. The main conclusions of trajectory optimization were provided to the ITER Organization by the Fusion For Energy and they were crucial to proceed with the construction of the Tokamak Building. For instance the doors aperture profiles (angle and orientation) were adjusted to reduce the risk of clashes.

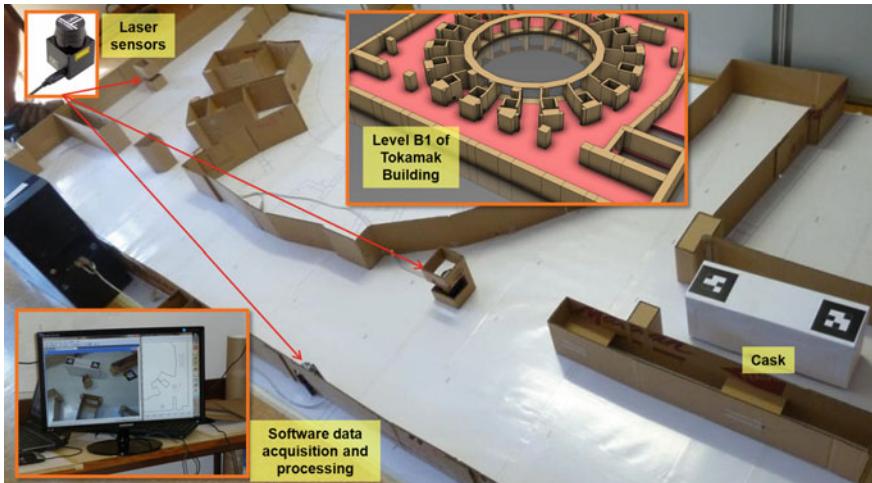
The required trajectories were computed offline. However, the actual implementation can be reverted into a version to run in real time. The decision of selecting line guidance and free roaming is manually, as similar to the decision to include a maneuver and the respective point of maneuver. In the future, the decision of selecting line guidance or free roaming can be done based on the minimum distances to the obstacles, as the decision of including or not maneuvers. The decision of where to put a point of maneuver remains as an open issue. The line guidance approach is a deterministic process, while free roaming may converge to similar solutions, since the initialization is random given the Rapidly-Exploring Random Tree component.

The velocity profile in the trajectory will be improved taking into account approaches with dynamics and the risk of huge vehicle in cluttered environments, which is the next instance of development and improvement of TES.

The next challenge of RH in ITER scenarios after trajectory optimization is the path following. The first simulated results have already achieved in the path following using modified versions of cutting edge approaches, such as Alonzo Kelly controller, Stanley controller, arc path following controller and also a non linear based controller embodied in the Lyapunov function, all resumed in [33]. The first work done only considers the kinematics model, while the inclusion of dynamics is on progress.

The localization of the vehicle, i.e., its position and orientation are also variables to estimate. Given the rad-hard condition in ITER scenarios, different approaches of implementation have been studied. In particular, a set of laser range finders strategically spread over the ITER rooms with Extended Kalman Filtering estimator, as detailed in [34].

The studies of path following and localization has been addressed on simulation environment and also in a experimental setup with a 1:25 scaled mock-up of half of a floor of the TB, as illustrated in Fig. 45. It includes four laser range finder sensors,



**Fig. 45** Experimental setup with a 1:25 scaled mock-up of half of a floor of the TB with four laser range finder sensors, one video camera and a rhombic like vehicle remotely controlled

one video camera and a rhombic like vehicle remotely controlled by a computer. The first results were already achieved and published.

An emerging issue in ITER scenarios is the logistics given the number of casks and the reduced space available to park and move them. The first studies to optimize the layout and the sequence of operations to move each cask, including optimized trajectories, have already been done and reported in [35].

The TES has been upgraded with new features to endorse the new challenges, namely the path following, the localization, logistics and also multiple vehicles operating simultaneously.

**Acknowledgments** The work was supported by the grants F4E-2008-GRT-016 (MS-RH) and F4E-GRT-276-01 (MS-RH) funded by the European Joint Undertaking for ITER and the Development of Fusion for Energy (F4E) and by FCT in the frame of the Contract of Associate Laboratories of Instituto de Plasmas e Fusão Nuclear/IST (PEst-OE/SADG/LA0010/2011) and Laboratório de Robótica e Sistemas em Engenharia e Ciências/IST (PEst-OE/EEI/LA0009/2011). The views expressed in this publication are the sole responsibility of the authors. F4E is not liable for the use which might be made of the information in this publication.

## References

1. Schiffe H-W (2008) World energy congress—energy policy scenarios to 2050, 36(7), pp 2464–2470
2. United Nations Population Fund—State of World Population (2011) People and possibilities in a world of 7 Billion. ISBN 9780897149907:2011

3. Schultz KR (2006) Why fusion? A discussion of energy alternatives. *IEEE Control Syst Mag* 26(2):32–34
4. Ribeiro I, Damiani C, Tesini A, Kakudate S, Siuko M, Neri C (2011) The remote handling systems for ITER. *Fusion Eng Des* 86:471–477
5. Gutiérrez C, Damiani C, Irving M, Friconneau JP, Tesini A, Ribeiro MI, Vale A (2009) ITER transfer cask system: status of design, issues and future developments. In: Proceedings of the 9th international symposium on fusion nuclear energy
6. Locke D, Gutiérrez CG, Damiani C, Gracia V, Friconneau JP, Martins J-P, Blight J (2011) Transfer cask system design activities: status and plan. *Fusion Eng Des* 86:2101–2104
7. Dudek G, Jenkin M (2010) Computational principles of mobile robotics, 2nd edn. Cambridge University Press, New York
8. Wang D (2001) Trajectory planning for a four-wheel-steering vehicle. *Electronic engineering*, pp 3320–3325
9. Ribeiro MI, Lima P, Aparício R, Ferreira R (1997) Conceptual study on flexible guidance and navigation for ITER remote handling transport casks. In: Proceedings of the 17th IEEE/NPSS symposium on fusion engineering, San Diego, pp 969–972
10. Kavraki LE, Svestka P, Latombe J-C, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans Robotics Autom* 12:566–580
11. LaValle SM, Rapidly-exploring random trees: a new tool for path planning, Computer Science Department, Iowa State University, TR: 98-11
12. Lozano-Pérez T, Wesley MA (1979) An algorithm for planning collision-free paths among polyhedral obstacles. In: Communications of the ACM, vol 22, New York, USA, pp 560–570
13. Canny JF (1985) A Voronoi method for the piano-movers problem. In: Proceedings of IEEE international conference on robotics and automation, pp 530–535
14. Chew LP (1987) Constrained delaunay triangulations. In: Proceedings of the third annual symposium on computational geometry, Waterloo, Ontario, Canada, pp 215–222
15. Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans Syst Sci Cybern* 4:100–107
16. Garrido S, Moreno L, Abderrahim M, Blanco D (2009) FM2: a real-time sensor-based feedback controller for mobile robots. *Int J Robot Autom* 24(1):3169–3192
17. Dijkstra EW (1959) A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1):269–271
18. Sethian JA (1996) A fast marching level set method for monotonically advancing fronts. *Proc Nat Acad Sci* 93(4):1591–1595
19. Osher S, Sethian JA (1988) Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J Comput Phys* 79(1):12–49
20. Gómez JV, Vale A, Valente F, Ferreira J, Garrido S, Moreno L (2013) Fast marching in motion planning for Rhombic like vehicles operating in ITER. In: Proceedings of the 2013 IEEE international conference on robotics and automation, Karlsruhe, Germany, pp 5513–5518
21. Quinlan S, Khatib O (1993) Elastic bands: connecting path planning and control. In: Proceedings IEEE conference robotics and automation, vol 2, Atlanta, pp 802–807
22. Kass M, Witkin A, Terzopoulos D (1988) Snakes: active contour models. *Int J Comput Vis* 1(4):321–331
23. Kang F, Zhong-Ci S (1995) Mathematical theory of elastic structures. Science Press, Beijing
24. Beer FP, Johnston ER, DeWolf JT (2002) Mechanics of materials. McGraw Hill, New York
25. Valente F, Vale A, Fonte D, Ribeiro I (2011) Optimized trajectories of the transfer cask system in ITER. *Fusion Eng Des* 86:1967–1970
26. Fonte D, Valente F, Vale A, Ribeiro I (2010) A motion planning methodology for Rhombic-like vehicles for Iter remote handling operations. In: Proceedings of the 7th IFAC symposium on intelligent autonomous vehicles, Lecce, Italy, pp 106–111
27. Kuffner JJ Jr, LaValle SM (2000) RRT-connect: an efficient approach to single-query path planning. In: IEEE international conference on robotics and automation
28. LaValle SM, Kuffner JJ (2001) Rapidly-exploring random trees: progress and prospects. In: Donald BR, Lynch KM, Rus D (eds) Algorithmic and computational robotics: new directions, A K Peters, Wellesley, pp 293–308

29. Fonte D, Valente F, Vale A, Ribeiro I (2011) Path optimization of Rhombic-like vehicles: an approach based on rigid body dynamic. In: Proceedings of the 15th IEEE international conference on advanced robotics, Tallinn, Estonia, pp 106–111
30. Vale A, Fonte D, Valente F, Ferreira J, Ribeiro I, González C (2013) Flexible path optimization for the cask and plug remote handling system in ITER. *Fusion Eng Des* 88:1900–1903
31. Vatti BR (1992) A generic solution to polygon clipping. *Commun ACM* 35(7):56–63
32. Rubianys P, Reig C, Gazeau E, Marmie J, Etchegoin N (2010) Definition, development and operation of a comprehensive virtual model of the ITER buildings, ATS and TCS. In: 26th symposium on fusion technology, Porto
33. Silva N, Baglivo L, Vale A, Cecco M (2013) Four new path following controllers for Rhombic like vehicles. In: Proceedings of the 2013 IEEE international conference on robotics and automation, Karlsruhe, Germany, pp 3189–3196
34. Ferreira J, Vale A, Ventura R (2013) Vehicle localization system using off-board range sensor network. In: Proceedings of the 8th IFAC intelligent autonomous vehicles symposium, vol 8, part 1, Gold Coast, Australia, pp 102–107
35. Ventura R, Ferreira J, Filip I, Vale A (2013) Logistics management for storing multiple cask plug and remote handling systems in ITER. *Fusion Eng Des* 88:2062–2066

# Car-Like Robot Manoeuvre Generation

F. Gomez-Bravo

**Abstract** This chapter describes a methodology for planning feasible car-like robot manoeuvres. The kinematics constraints and the model of the vehicle are taken into account, so that a canonical set of manoeuvres are designed. From these basic trajectories, three strategies of motion are defined in order to combine them and achieve different and flexible ways of motion. The basic of the manoeuvre design is addressed in detail by considering mathematical elements of differential geometrics. The problem of collision-free motion planning is solved by means of an algorithm that is capable of turning a collision-free route into a feasible sequence of collision-free manoeuvres.

**Keywords** Nonholonomic constraints · Manoeuvre generation · Car-like robots

## 1 Introduction

Motion planning of wheeled vehicles has attracted the interest of the scientific community along the last two decades [6, 7, 14, 18, 22, 37, 40, 41, 45]. Wheeled locomotion imposes nonintegrable kinematics constraints on the vehicle motion, known as nonholonomic. As a consequent, the generalized velocity vector of the system has to verify a set of non-singular equations along any admissible path [24, 36, 39, 44].

There are different approaches to solve motion planning problems of wheeled robots [4, 19, 26]. Due to the nonholonomic nature of these systems, traditional path planning techniques have been adapted in order to obtain paths accomplishing kinematics constraints [23, 38, 45]. These techniques provide derivable and continuous curvature trajectories based on different searching techniques. However, when cluttered scenarios are involved, complex trajectories may be needed and traditional methods could not provide suitable solutions. Vehicle motion in constrained

---

F. Gomez-Bravo (✉)

Department of Electronic Engineering, Computer System and Automation,  
University of Huelva, Huelva, Spain  
e-mail: fernando.gomez@iesia.uhu.es

scenarios (narrow corridors, very close and enveloping obstacles, etc.) imposes severe restrictions on the generation of admissible paths. In these cases, it is difficult to obtain a solution accomplishing continuity in the sign of the speed, i.e., the vehicle has to stop and change forward to backward motion or vice versa, and the path has to present cusps. In order to manage these situations, obtaining solutions based on the concept of manoeuvre can be an effective option.

The systems most frequently using this concept of manoeuvre are wheeled vehicles. This chapter focuses on the description of a methodology for wheeled robot manoeuvre generation, and pay special attention to those vehicles with Ackerman configuration, i.e. car-like robots [14, 18, 41]. The application of this type of vehicles for autonomous navigation has increased enormously in recent years [29, 31, 40, 45, 46].

Planning admissible manoeuvres for car-like vehicles represents a theoretical challenge due to the constraint in the curvature imposed by the Ackerman steering system. One of the first significant contributions in this area was conducted by Dubins (1957) [8]. This approach consider the generation of curves of minimal length with constrained curvature when the vehicle can not reverse. Melzak (1961) [33] studied the motion on a plane of a mobile object with curvature limitations. Later, Reeds and Shepp (1990) extended Dubins results by solving the problem when the vehicle can shift into reverse and the path is allowed to have cusps [43]. In [19], Latombe (1991) proved the full controllability of a car-like robot by using a set of canonical manoeuvres. Latombe stated that a feasible path between any two configurations, can be obtained by a suitable combination of these manoeuvres. In [39] Murray and Sastry (1993) presented a motion and manoeuvre planning approach based on nonlinear control theory. Further works have been developed from these early results [9, 10, 22, 38, 41, 45, 47]. The use of different spline curves has also been proposed in order to perform parking manoeuvres [13, 27].

However, solutions based on nonlinear control theory, Reeds and Shepp curves or Latombe manoeuvres, sometimes differ from manoeuvres performed by humans drivers. They consider only optimal length solutions, that may not perfectly fit to the robot's environment in accordance with other criteria (distance to obstacles, reverse distance, smoothness in the steering angle changes and others).

The manoeuvre generation methodology proposed in this chapter represents an alternative to Latombe or Reeds and Shepp approaches. It is based also on a set of canonical manoeuvres, which differs from that presented by Latombe in the way the manoeuvres are defined (heuristic considerations), and in appearance, that more closely resembles the manoeuvres performed by human drivers. From this approach not only solutions similar to Reeds and Shepp trajectories can be obtained but also a wide family of different solutions can be computed. These solutions could optimize certain criteria better than the shorter trajectories provided by traditional methods [32].

This chapter is devoted to explain in detail the basis of the manoeuvres design, presenting the fundamentals of some differential geometry concepts and its application to a car-like vehicle. It also illustrates a new version of an algorithm that applies this motion strategy for collision-free motion planning [14, 15, 30]

The approach presented in this work is based on two basic elements: restricted manoeuvres and connecting paths. From this two concepts, a solution containing admissible trajectories (complex or simple, depending on what is needed) can be provided. These two elements can be incorporated for planning collision-free manoeuvres into different traditional planning methods: Road Map Method, Potential Field Methods, Randomize Algorithm (see Chaps. 1, 2, 8 and 9 of this book).

A restricted manoeuvre is a path or a sequence of paths for changing only one of the vehicle state variables [12, 14]. Thus, restricted manoeuvres represent a set of canonical paths.

Connecting paths are trajectories defined by a sequence of restricted manoeuvres, so that different types of paths can be build: from the simplest solutions (such as straight lines) to complex manoeuvres including multiple cusps.

The restricted manoeuvres proposed in this chapter are referred to a local coordinate frame, the one attached to the vehicle in the initial configuration. For the case of a vehicle moving in a plane, three types of restricted manoeuvres are defined:  $\Phi^x$  that provide a change in the x coordinate;  $\Phi^\theta$  that provides a change in the orientation of the vehicle, leaving it in the same Cartesian configuration;  $\Phi^y$  that provides a change in the y coordinate. Due to the way in which these manoeuvres are defined, the trajectory can be determined, by means of simple algebraic equations, as a function of the initial and the goal configuration, without integrating the vehicle kinematics model.

The general planning algorithm presented in this chapter solves the path planning problem in two steps. Firstly a solution is obtained from the application of any of the traditional planning methodologies without considering the kinematics constraint. Secondly a simplifying process is applied so that the route provided by this method is turned into a feasible path by means of the connecting paths.

The proposed approach presents several advantages. First of all, it provides, in a short period of time, with feasible solutions in complex environments in which difficult manoeuvres have to be planned. Secondly, it can be applied to different type of searching algorithm. Finally the solutions provided by this method are very similar to those a human driver would perform in real scenarios.

This chapter is organized as follow. The theoretical framework applied for motion planning purposes is introduced in Sect. 2. Also in this section, the kinematics characteristics of car-like vehicles are introduced. Section 3 presents a methodology for generating car-like vehicle manoeuvres, based on the concepts of restricted manoeuvre and connecting path. Section 4 is devoted to illustrate a procedure for incorporate this methodology into traditional planning algorithm. Finally, in the Sect. 5, a set of experiments involving car-like vehicles manoeuvres generation in different scenarios are presented. The chapter closes with the conclusions and two Appendix where some theoretical concept are detailed.

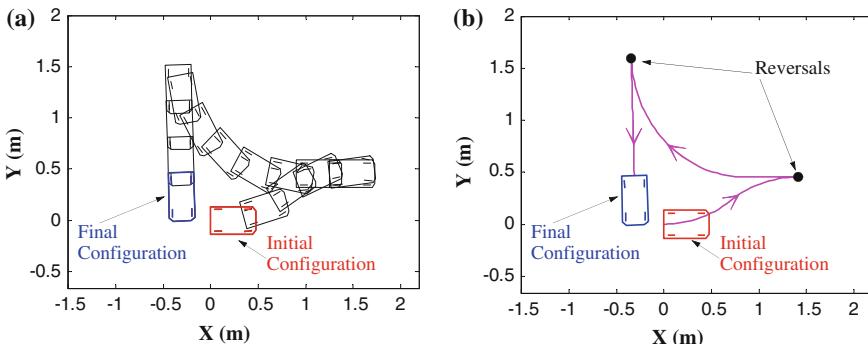
## 2 Motion Planning in System with Kinematics Constraints

Motion Planning of wheeled vehicles requires especial attention. Particularly, navigating in cluttered scenarios requires the capability of generating trajectories that present discontinuities in the sign of the velocity. In situations where the geometry of the environment prevents navigating without stopping the vehicle, it is possible to obtain suitable solutions based on the concept of manoeuvre.

Generally speaking, a manoeuvre is an operation on a vehicle for changing its course. However, the concept of manoeuvre requires a further definition in order to be applied as a planning tool. In [19], the concept of manoeuvre is defined for a car-like robot. To this end, the idea of reversal is introduced. In a trajectory, a reversal is a point where the linear speed of the robot changes the sign. Thus, a manoeuvre is defined as a concatenation of paths separated by reversals, see Fig. 1.

Nevertheless, not all vehicles can manoeuvre in the same way. Special attention has to be paid to those with non integrable kinematics constraints, known as non-holonomic systems. Wheeled vehicles fall into this category. These vehicles can only perform admissible trajectories by accomplishing the nonholonomic constraints. As consequence, planning trajectories for nonholonomic systems has to pay special attention on generating admissible motions. There are different techniques to tackle this challenge [21].

On the one hand, some authors have proposed the application of nonlinear control strategies for motion planning purposes. An admissible control law is applied to the system's mathematical model in order to drive it to a desired configuration. Then, by running a simulation, a feasible trajectory is obtained. In this sense, Murray and Sastry (1993) developed in [39] a strategy for steering wheeled vehicles by using sinusoids as control inputs. Tilbury (1995), in [47, 48], extended this approach to drive wheeled vehicles with trailers. In [41] Paromtchik et al. (1998) proposed a discontinuous control law that allows to perform parallel-parking manoeuvres. More recent applications of this control strategies can be found in Qu et al. (2006) [42], and Michalek and Kozowski (2010) [34].



**Fig. 1** Manoeuvre: **a** a car-like vehicle manoeuvring; **b** trajectories separated by reversals

The design of admissible control strategies for nonholonomic systems is deeply determined by the Brockett's necessary condition for asymptotic stability [2]. Moreover, obtaining the trajectory requires performing the numeric integration of the vehicle's kinematics model. Then no previous knowledge about the shape of the resulting path or the control signal evolution exists. Thus complex or unusual solutions are sometimes achieved.

On the other hand, traditional geometrics approaches, like the one presented in Dubins's (1957), [8], or in Reeds and Shepp (1990), [43], are based on building a piecewise trajectory by using a systematic combination of basics admissible paths: circular trajectories,  $C$ , and straight segments,  $S$  ( $C^-$  denotes a clockwise/anticlockwise circular segment followed in reverse,  $C^+$  indicates a clockwise/anticlockwise circular segment followed in forward direction, and  $S^{-/+}$  denotes a straight segment followed in reverse or in forward direction). Not all possible combinations are optimal solutions. These works establish the way of achieving the shorter path a car can travel between two configurations. In [22], Laumont et al. (1994) described a general path planning algorithm inspired in Reeds and Shepp approach. More recent contributions based in these methodologies can be found in Fraichrd et al. (2004), [9], Giordano et al. (2009), [10], and Chitsaz et al. (2009) [3].

These strategies aim to obtain the shortest path or a time optimal trajectory for a wheeled robot with unitary speed and constrained curvature. However, there is a huge amount of trajectories that are not time optimal solution and could optimize others different criteria. In fact, the manoeuvres presented in this chapter are out of the group considered as optimal in [43], because they are built by considering the sequences:  $C^+C^+S^-$ ;  $C^-C^-S^+$ ;  $S^+C^-S^+$  and  $S^-C^+S^-$ . However, as will be shown before, by means of a combinational procedure and a simplification process, from this methodology shortest trajectories as well as other solutions, different from manoeuvres presented in [43], can be achieved.

The procedure and formalism applied for designing these manoeuvres differs from the technique presented in [43]. However, the basis is similar to the one presented in [19] and the formalism is inspired in the approach presented in [22]. In this chapter, admissible paths are generated by a sequence of vector field belonging to the tangent space of the system. A set of vectors field is selected, so that any configuration is reachable. Later, a set of canonical manoeuvres are defined by using this vectors.

The next subsection introduces the basis for manoeuvre generation in vehicles with nonholonomic constraints. Further, these concepts will be applied to vehicles with Ackermann steering configuration, i.e. car-like vehicles, so that a technique for planning admissible manoeuvres for this type of systems can be developed [12, 14, 17].

## 2.1 Motion Generation in Nonholonomic Systems

Given a system, with a configuration space  $\Omega$  of dimension  $n$ , and  $m$  nonholonomic constraints, it is possible to find in the tangent space a distribution  $\Delta$ , with dimension

$(n - m)$ , spanned by a set of  $m$  linearly independent vectors field  $\mathbf{V}_i$ , such that the vector speed of the system accomplishes:  $\mathbf{V}_{\text{sys}} \in \Delta$ , [5, 19, 22]. That is:

$$\exists \Delta = \text{span}\{\mathbf{V}_1, \dots, \mathbf{V}_m\}/\mathbf{V}_{\text{sys}} = \sum_i \alpha_i \mathbf{V}_i \in \Delta \quad \alpha_i \in R \quad (1)$$

Let be  $L(\tau)$  a parametrized trajectory (expressed as a functions of the parameter  $\tau \in R$ ) representing the evolution of the system in the configuration space. It is said that  $L(\tau)$  follows vector  $\mathbf{V}_i$  from  $a$  to  $b$  if:

$$\frac{d}{d\tau} L(\tau)|_{\tau \in [a,b]} = \mathbf{V}_i|_{L(\tau)} \quad (2)$$

That is, the system follows  $\mathbf{V}_i$  if the tangent vector along the trajectory  $L(\tau)$  (when  $\tau \in [a, b]$ ), is equal to  $\mathbf{V}_i$ . For a more intuitive understanding the parameter  $\tau$  could be interpreted as time. However, as will be shown below,  $\tau$  could also be associated with other meanings such as the path length.

Controllability of nonholonomic systems has been detailed by Latombe in [19]. This property is related with the system capability of following the flow of certain class of vector field (see Appendix 1 for details). Latombe demonstrated that a controllable system with a  $n$  dimensional configuration space  $\Omega$  and  $m$  nonholonomic equality constraints, can move between any two configurations of an open connected subset of  $\Omega$  by following the flow of a finite sequence of vectors of a distribution  $\Delta$ . This distribution has to accomplish a necessary and sufficient condition: the dimension of  $CLA(\Delta)$  must to be equal to  $n$  (being  $CLA(\Delta)$  the Control Lie Algebra associated with  $\Delta$ , that is, the distribution generated by the set of vectors that spans  $\Delta$  and all their Lie brackets operations recursively computed, see Appendix 1 and [5, 19] for details).

The methodology presented in this chapter is inspired in this idea. Thus, if a set of vectors field are defined and the distribution spanned by them accomplishes the controllability condition, feasible trajectories can be generated by following the flow of these vectors. In order to formalize this procedure, the following concepts are introduced.

Let be  $p_0 \in \Omega$ ,  ${}^{p_0}L$  is defined as the set of parametrized trajectories  $L_j(\tau)$  accomplishing:

$$L_j(0) = p_0 \quad (3)$$

If  $\mathbf{V}_j$  is a vector field defined around  $p_0$ , there exists only one parametric trajectory  $L_j(\tau)$  starting at  $p_0$  and following  $\mathbf{V}_j$ , that is:

$$L_j(\tau) \in {}^{p_0}L \quad \text{and} \quad \frac{d}{d\tau} L_j(\tau) = \mathbf{V}_j|_{L_j(\tau)} \quad (4)$$

The relation between  $L_j(\tau)$  and  $\mathbf{V}_j$  can be expressed by using an exponential formalism (see [22] for details):

$$p(\tau) = L_j(\tau) = p_0 \circ e^{\tau V_j} \quad (5)$$

This expression defines an operation over  $\Omega$  with the following meaning: “ $p(\tau)$  is reached if starting from  $p_0$  the vector  $V_i$  is followed along a change on the path parameter equal to  $\tau$ ”. This meaning is equally valid whether  $p(\tau)$  is considered a temporary variable, or it is interpreted as the path length.

Thus, if a trajectory starts at  $p_0$ , and follows the vector  $V_1$  during a value  $\tau = t_1$ , the point  $p_1$  will be reached. If, from  $p_1$ , the system follows the vector  $V_2$  during a value  $\tau = t_2$ , a new point  $p_2$  will be reached. This operation can be expressed as:

$$p_2(t_1, t_2, p_0) = p_1 \circ e^{t_2 V_2} = p_0 \circ e^{t_1 V_1} \circ e^{t_2 V_2} \quad (6)$$

This formalism has been previously applied for car-like robot manoeuvre generation in [12, 22] as will be introduced in the next section.

It is remarkable (see [22] for details) that in general if  $V_i \neq V_j$ :

$$p_0 \circ e^{t_1 V_i} \circ e^{t_2 V_j} \neq p_0 \circ e^{t_1 V_i + t_2 V_j} \quad (7)$$

However, it is straightforward to show that:

$$p_0 \circ e^{t_1 V_j} \circ e^{t_2 V_j} = p_0 \circ e^{(t_1 + t_2)V_j} \quad (8)$$

**Definition 1** A Generic Path is a motion generated from an arbitrary initial configuration following a sequence of vectors field.

Thus, for instance, let  $\Phi$  be a Generic Path, defined by any starting configuration  $p \in \Omega$  and following the sequence  $[V_1, \dots, V_m]$ . Then  $\Phi$  can be defined by the expression:

$$\Phi(\tau_1, \dots, \tau_m, p) \equiv p \circ e^{\tau_1 V_1} \circ \dots \circ e^{\tau_m V_m} \quad (9)$$

Given a Generic Path defined by a sequence of  $m$  vectors  $V_i$ , a particular path can be obtained by specifying an initial configuration  $p_0$  and a  $m$ -tuple  $[t_1, \dots, t_m]$ . This path can be generated by chaining the  $n$  trajectories defined in the following way:

$$\begin{aligned} L_1(\tau_1, p_0) &\equiv p_0 \circ e^{\tau_1 V_1} \quad \text{with } \tau_1 \in [0, t_1] \quad \text{and} \quad p_1 = p_0 \circ e^{\tau_1 V_1}; \\ L_2(\tau_2, p_1) &\equiv p_1 \circ e^{\tau_2 V_2} = (p_0 \circ e^{\tau_1 V_1}) \circ e^{\tau_2 V_2} \quad \text{with } \tau_2 \in [0, t_2] \quad \text{and} \quad p_2 = p_0 \circ e^{\tau_1 V_1} \circ e^{\tau_2 V_2}; \\ &\vdots \\ L_m(\tau_m, p_{m-1}) &\equiv p_{m-1} \circ e^{\tau_m V_m} = p_0 \circ e^{\tau_1 V_1} \circ \dots \circ e^{\tau_m V_m} \quad \text{with } \tau_m \in [0, t_m] \end{aligned} \quad (10)$$

**Definition 2** Let  $\Phi_1$  and  $\Phi_2$  be two Generic Path:

$$\Phi_1(\tau_1, p) \equiv p \circ e^{\tau_1 V_1} \quad \text{and} \quad \Phi_2(\tau_2, p) \equiv p \circ e^{\tau_2 V_2} \quad (11)$$

the composition of this two Generic Paths is a new Generic Path defined as:

$$\Upsilon(\tau_1, \tau_2, p) = \Phi_1 \otimes \Phi_2 \equiv p \circ e^{\tau_1 V_1} \circ e^{\tau_2 V_2} \quad (12)$$

It is easy to extend this definition to the case of two Generic Paths that follow a  $m$  and a  $k$  sequence of vectors respectively.

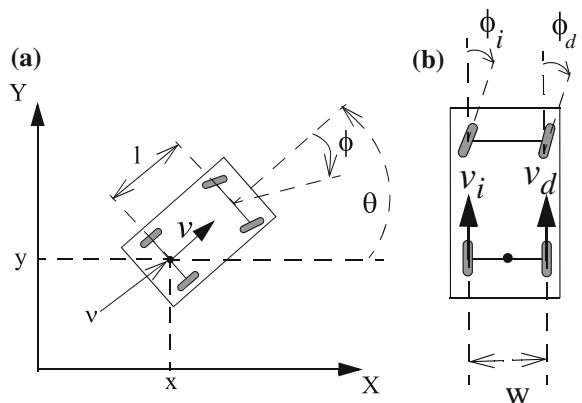
**Definition 3** Let  $\Phi_1$  be a Generic Path, following the vector sequence  $[V_1, \dots, V_m]$ , and  $\Phi_2$  be a Generic Path, following the vector sequence  $[V'_1, \dots, V'_k]$ . The Generic Path defined as  $\Upsilon(\tau_1, \dots, \tau_{m+k}, p) = \Phi_1 \otimes \Phi_2$  is the one following the sequence of vectors  $[V_1, \dots, V_m, V'_1, \dots, V'_k]$ .

Generally speaking, this formalism can be applied for motion planning purpose to different types of nonholonomic systems. In particular, in this chapter, it will be applied to a car-like vehicle in order to develop a methodology for planning manoeuvres. To this end, a set of vectors, accomplishing the controllability condition, will be defined, and a set of feasible manoeuvres will be generated from them. Manoeuvres presented below are based on the principles described above, especially in the expressions (9) and (10), and Definitions (2) and (3).

## 2.2 Car-Like Vehicle Kinematics

Nowadays there are numerous applications of conventional vehicles for autonomous navigation. In this sense, car-like vehicles are the most popular human transportation system, and real applications of car-like robots have increased along the last decade [1, 29, 31, 40, 41, 46]. Figure 2 shows an approximate representation of this type of vehicle. Traditionally, the motion of this system is determined by the steering angle of the forward wheels and the velocity of the rear wheels, see Fig. 2b.

**Fig. 2** Car-like vehicle model: **a** rear reference point and central steering angle; **b** forward steering wheels and rear velocities



Note that, the car-like model presented in Chap. 8, corresponds to a front wheel drive vehicle, that is, the action of the velocity control is applied to the front wheels rather than to the rear wheels. The results presented in the present chapter can be extended to this type of vehicles by considering geometrics relations between the rear and front velocities.

Due to the nonholonomic constraints the vehicle has only two degree of freedom, and the steering angles and rear velocities are usually expressed as a function of a virtual central steering angle  $\phi$  and the velocity of the central rear point  $v$  (see Fig. 2a), in the following way:

$$\phi_i = \frac{l \cdot \tan \phi}{l - \frac{w}{2} \cdot \tan \phi}; \quad \phi_d = \frac{l \cdot \tan \phi}{l + \frac{w}{2} \cdot \tan \phi} \quad (13)$$

and

$$v_i = \left( \frac{l - \frac{w}{2} \cdot \tan \phi}{l} \right) \cdot v; \quad v_d = \left( \frac{l + \frac{w}{2} \cdot \tan \phi}{l} \right) \cdot v \quad (14)$$

where  $l$  is the distance between the front and the rear wheels, and  $w$  is the distance between to wheels on the same axis.

Traditionally, an Ackerman mechanism has been applied in order to implement the steering system [35]. This mechanism bounds the steering angle and, as a consequence, it also bounds the curvature of any admissible path.

Although there are some complex mathematical models that further describe the dynamic behaviour, the methodology addressed in this chapter will consider the kinematic model in order to develop a technique for manoeuvres planning.

The vehicle is geometrically modelled as a rectangle that moves in  $W = R^2$ . The configuration space is  $R^2 * S^1$ . Then, each configuration is represented with a tuple  $(x, y, \theta)$ , where  $(x, y)$  are the coordinates of the midpoint between the rear wheels, and  $\theta$  is the angle between the longitudinal axis of the rectangle and the axis  $X$  of the global reference frame [14, 19, 22]. Kinematics of this vehicle can be modelled with de following state equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v(t) \\ v(t) \cdot \rho(t) \end{bmatrix} \quad (15)$$

where  $v(t)$  and  $\rho(t)$  are the control parameters. As was mentioned before,  $v(t)$  is the lineal velocity of the rear reference point, and  $\rho(t)$  is the curvature of the path described by the same point. The relation between the curvature and the radius of curvature  $r$  is:

$$r(t) = \frac{1}{\rho(t)} \quad (16)$$

Both magnitudes can be used for defining the control of the vehicle, and are determined by the steering angle  $\phi$  according to the expression:

$$r(t) = \frac{1}{\rho(t)} = \frac{l}{\tan [\phi(t)]} \quad (17)$$

Under the assumption that the wheels do not slip, the following nonholonomic constraints can be written [19]:

$$\dot{x} \cdot \sin \theta - \dot{y} \cdot \cos \theta = 0 \quad (18)$$

An additional constraint has to be taken into account. Due to the Ackerman mechanism, the values of the steering angle are usually bounded i.e.,  $\phi \in [-\phi_{max}, \phi_{max}]$  with  $\phi_{max} > 0$ . This relation can be also expressed as:

$$|\phi| \leq \phi_{max} \quad (19)$$

This feature deeply constraints the possible values of the state velocity and also the values of the control parameter  $\rho(t)$ . According to Eqs. (17) and (19),  $|\rho(t)|$  is upper bounded:

$$|\rho(t)| \leq \rho_{max} = \frac{\tan \phi_{max}}{l} \quad (20)$$

Then, from (15) and (20) the following kinematics constraint is established:

$$|\dot{\theta}| \leq |v(t)| \cdot \rho_{max} \quad (21)$$

Given that

$$|v(t)| = |\dot{x} \cdot \cos \theta + \dot{y} \sin \theta| \quad (22)$$

the inequality constraint (21) can be written as follow:

$$|\dot{x} \cdot \cos \theta + \dot{y} \sin \theta| - \frac{|\dot{\theta}|}{\rho_{max}} \geq 0 \quad (23)$$

Thus, a vector defining a Generic Path must accomplish both constraints (18) and (23). As it is illustrated in [22] the row vectors of the matrix of the model (15) accomplish (18) and are linearly independent. Therefore, they span the distribution associated to this constraint. These vectors are:

$$\Psi_1 = [\cos \theta, \sin \theta, 0]^T; \quad \Psi_2 = [0, 0, 1]^T \quad (24)$$

It can be shown that vector  $\Psi_1$  represents a situation in which the vehicle move straight ahead with constant heading  $\theta$  and constant linear velocity equal to 1. Vector  $\Psi_2$  represents a situation in which the vehicle rotates around the rear reference point. Obviously, this last vector does not accomplish (23). As a consequence, this vector can not be used for defining a feasible generic path. However, it is possible to define two vectors, from a linear combination of  $\Psi_1$  and  $\Psi_2$  that accomplish both (18), and (23):

$$\begin{aligned}\Psi_3 &= \Psi_1 + c_1 \cdot \Psi_2 = [\cos \theta, \sin \theta, c_1]^T \\ \Psi_4 &= \Psi_1 + c_2 \cdot \Psi_2 = [\cos \theta, \sin \theta, c_2]^T\end{aligned}\tag{25}$$

where  $c_1 \geq |\rho_{max}|$  and  $c_2 \leq -|\rho_{max}|$ . In this chapter the values  $c_1 = |\rho_{max}|$  and  $c_2 = -|\rho_{max}|$  have been chosen.

These vectors are associated to the following situations.  $\Psi_3$  represents the vehicle moving along a circular arc with positive constant steering angle and linear velocity equal to 1. Likewise,  $\Psi_4$  represents a similar situation but with negative constant steering angle. Due to this interpretation on the meaning of the vectors, in trajectories defined by expressions such as:

$$p(\tau) = p_0 \circ e^{\tau \Psi_i} \quad \text{with} \quad \Psi_i \in \{\Psi_1, \Psi_3, \Psi_4\} \tag{26}$$

the parameter  $\tau$  can be interpreted as the length of the path described by the rear point of the vehicle, and also:

$$[dx, dy, d\theta]^T = \Psi_i \cdot d\tau \quad \text{with} \quad \Psi_i \in \{\Psi_1, \Psi_3, \Psi_4\} \tag{27}$$

Moreover, it can be demonstrated that the vectors set  $\Psi_c = \{\Psi_1, \Psi_3, \Psi_4\}$  accomplishes the controllability condition (see Appendix 1 for details). Therefore, any configuration can be reached using a generic path following the flow of these vectors. This property will be used in the following section in order to define a methodology for manoeuvres generation.

### 3 Car-Like Vehicle Manoeuvres Generation

In [19, 20] a set of two basic manoeuvres for a car-like vehicle are shown. On the one hand, the first manoeuvre allows the robot to perform sidewise motion, on the other hand, the second type, allows the robot to rotate around the rear reference point without violating the kinematics constraint. These motions are characterized by the use of straight segments and arcs of circumference, and were developed to demonstrate that any two configurations can be connected by a finite sequence of manoeuvres. In Fact, in [19] it is formally demonstrated, based on the controllability property of car-like systems, that if there exist a collision-free path between any two configurations then there also exists a feasible collision-free path connecting both configurations. Even more, it is demonstrated that this path can be built from a finite sequence of this manoeuvres set.

The approach presented in this section is inspired in these ideas but improve in different aspects the results presented in the aforementioned works. Manoeuvres introduced below, and the procedure to combine them, represent a more flexible way of motion, providing a wide collection of solutions that avoid repetitive patterns of motion and best suit the environment's features. Thus, more efficient trajectories are achieved, in more resemblance to the manoeuvres performed by human drivers.

The main characteristic of the basic manoeuvres presented below consists on the capability of changing the value of one configuration variable, maintaining, at the end of the manoeuvre, unchanged the value of the others. This characteristic is common to other planning techniques used in non-holonomic systems (as is the case of free-floating manipulators [16, 36]. This feature will be exploited to define a generic type of manoeuvre (restricted manoeuvre) and a more general concept, very powerful for manoeuvre generation: connecting paths.

### 3.1 Restricted Manoeuvres

The following definition can be introduced in order to establish the concept of restricted manoeuvre.

**Definition 4** A restricted manoeuvre is a generic path that generates a change in the value of one of the state variables, maintaining unchanged the value of the others.

Thus, it can be defined a collection of restricted manoeuvres representing a set of canonical paths, that is, any two points of the configuration space can be connected through a suitable combination of manoeuvres belonging to this set, by considering the change of each configuration variable individually.

From a formal point of view, this definition avoids the necessity of including reversals in a generic path in order to be considered as a manoeuvre. As will be shown before, this idea allows obtaining a large diversity of solution and avoids motion pattern repetition.

Practical examples of restricted manoeuvres for car-like vehicles have been previously reported and applied in [6, 11, 12, 14, 29, 32]. All of them represent generic paths accomplishing Definition 4 in a reference frame  $\Sigma_L$  where the initial vehicle's heading is equal to zero and the origin is located at the manoeuvre starting point. Three basic groups of manoeuvres have been defined in [6, 11, 12]:

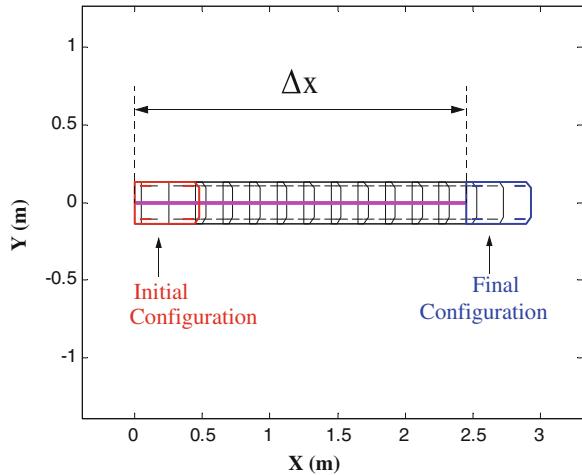
- $\Phi^x$ , allows changing the value of the variable  $x$ .
- $\Phi^y$ , allows changing the value of the variable  $y$ .
- $\Phi^\theta$ , allows changing the value of the variable  $\theta$ .

These manoeuvres have been developed by using the set of vectors defined in Sect. (2.2):  $\Psi_c = \{\Psi_1, \Psi_3, \Psi_4\}$ . As the motions are built by following the flow of these vectors, the parameters defining the trajectory will represent the lengths of the paths described by the rear point of the vehicle. Therefore, the total length of the trajectory will be the sum of the value of all the parameters.

#### 3.1.1 Manoeuvre $\Phi^x$

This is the simplest example of a restricted manoeuvre. It is a straight displacement, that generates only an increment in  $x$ , referred to the reference frame  $\Sigma_L$ , see Fig. 3.

**Fig. 3** Restricted manoeuvre  $\Phi^x$



This manoeuvre is defined by the expression.

$$\Phi^x(s_1) \equiv p_0 \circ e^{s_1 \Psi_1} \quad (28)$$

The value of  $s_1$  is related with  $\Delta x$  in the following way:

$$\Delta x = s_1 \quad (29)$$

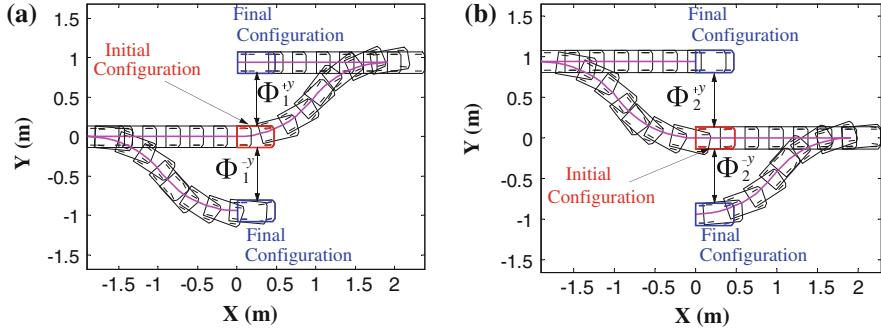
A positive value of  $s_1$  will provide a forward motion and a positive value of  $\Delta x$ . In contrast, a negative value of  $s_1$  will produce a backward movement and a negative value of  $\Delta x$ . Note that the definition of  $p_0$  is omitted, due to the fact that these restricted manoeuvres are always defined in the origin of the reference frame  $\Sigma_L$ .

### 3.1.2 Manoeuvre $\Phi^y$

In this case the initial and final value of the variables  $\theta$  and  $x$  are the same, while the value of the variable  $y$  is changed. Therefore, in the reference frame  $\Sigma_L$ , this motion allows moving the vehicle from the configuration  $(0, 0, 0)$  to  $(0, \Delta y, 0)$ .

Depending on the sign of  $\Delta y$  two types of motion can be established. Thus, if  $\Delta y > 0$  a manoeuvre is defined in the following way:

$$\Phi^{+y} \equiv p_0 \circ e^{s_3 \Psi_3} \circ e^{s_2 \Psi_4} \circ e^{s_1 \Psi_1} \quad (30)$$



**Fig. 4** Restricted manoeuvres: **a**  $\Phi_1^y$  (+y and  $-y$ ); **b**  $\Phi_2^y$  (+y and  $-y$ )

As is demonstrated in [12], the relation between  $\Delta y$  and  $s_1, s_2, s_3$  are:

$$\begin{aligned} \theta_\phi &= \arccos\left(1 - \frac{|\Delta y|}{r_1 - r_2}\right) \\ s_3 &= r_1 \cdot \theta_\phi \\ s_2 &= \frac{-s_3 \cdot r_2}{r_1} \\ s_1 &= (r_2 - r_1) \sin \theta_\phi \end{aligned} \tag{31}$$

where:

$$r_1 = \frac{1}{c_1} \text{ and } r_2 = \frac{1}{c_2}$$

being  $c_1$  and  $c_2$  the values associated to the vectors  $\Psi_3$  and  $\Psi_4$ , defined in (25). Note that  $\Delta y$  has to accomplish:  $\Delta y \leq 2 \cdot (r_1 - r_2)$ . If largest increments on  $y$  are needed it will be required to perform various manoeuvres.

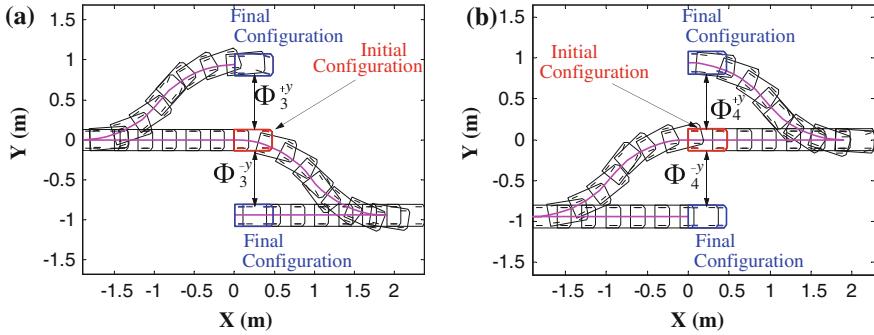
Note also that, from Eq. (31), two possible values of  $\theta_\phi$  can be obtained. Thus depending on this selection, two different manoeuvres can be generated:  $\Phi_1^{+y}$  ( $\theta_\phi > 0$ ), see Fig. 4a, and  $\Phi_2^{+y}$  ( $\theta_\phi < 0$ ), see Fig. 4b.

In a similar way, two new manoeuvres can be defined for a negative increment on  $y$  by using the same vector fields but in an inverse sequence, that is,

$$\Phi^{-y} \equiv p_0 \circ e^{s_1 \Psi_1} \circ e^{s_2 \Psi_4} \circ e^{s_3 \Psi_3} \tag{32}$$

where  $s_1, s_2$  and  $s_3$  are defined by (31). If  $\theta_\phi > 0$ ,  $\Phi_1^{-y}$  is generated see Fig. 4a. Likewise, the election of  $\theta_\phi < 0$  provides  $\Phi_2^{-y}$ , see Fig. 4b.

Finally, using a different sequence of vector, four additional manoeuvres can be defined in order to produce an increment on  $y$  (see Fig. 5):



**Fig. 5** Restricted manoeuvres: **a**  $\Phi_3^y$  ( $+y$  and  $-y$ ); **b**  $\Phi_4^y$  ( $+y$  and  $-y$ )

$$\begin{aligned}
 \Phi_3^{+y} &\equiv p_0 \circ e^{s_1 \Psi_1} \circ e^{s_3 \Psi_3} \circ e^{s_2 \Psi_4} & \theta_\phi > 0 \\
 \Phi_3^{-y} &\equiv p_0 \circ e^{s_2 \Psi_4} \circ e^{s_3 \Psi_3} \circ e^{s_1 \Psi_1} & \theta_\phi > 0 \\
 \Phi_4^{+y} &\equiv p_0 \circ e^{s_1 \Psi_1} \circ e^{s_3 \Psi_3} \circ e^{s_2 \Psi_4} & \theta_\phi < 0 \\
 \Phi_4^{-y} &\equiv p_0 \circ e^{s_2 \Psi_4} \circ e^{s_3 \Psi_3} \circ e^{s_1 \Psi_1} & \theta_\phi < 0
 \end{aligned} \tag{33}$$

where  $s_1, s_2, s_3$  and  $\theta_\phi$  are again defined by (31).

### 3.1.3 Manoeuvre $\Phi^\theta$

The design of this manoeuvre involves changing the vehicle's orientation while the Cartesian position of the reference point remains unchanged at the end of the manoeuvre. Thus, in the reference frame  $\Sigma_L$ , the vehicle will move from configuration  $(0, 0, 0)$  to  $(0, 0, \Delta\theta)$ , see Fig. 6.

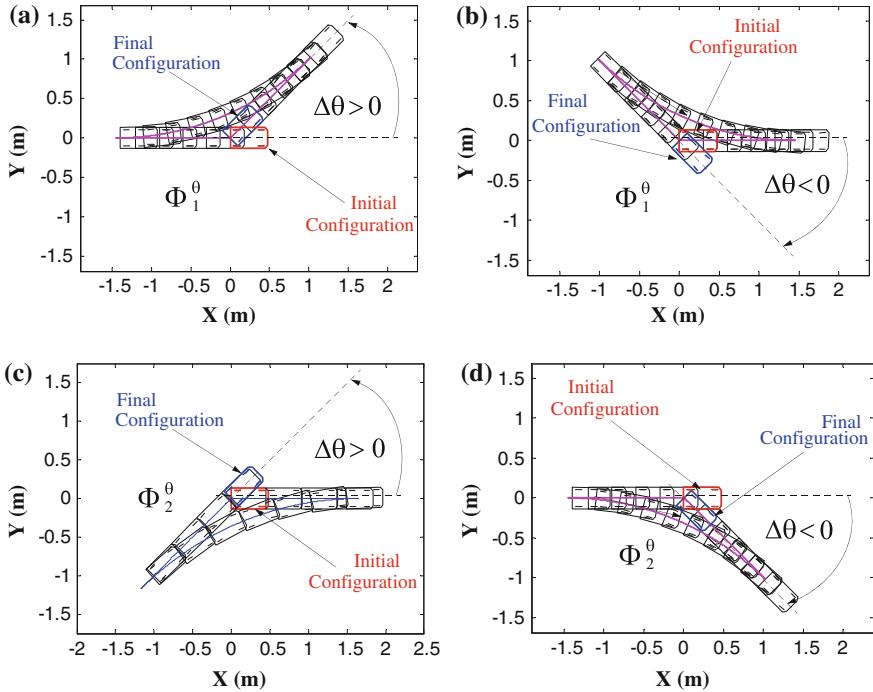
Following the same ideas presented in [12] it can be shown that a change on the variable  $\theta$  can be obtained from a restricted manoeuvre defined in the form (see Fig. 6 a and b):

$$\Phi_1^\theta \equiv p_0 \circ e^{s_1 \Psi_1} \circ e^{s_2 \Psi_3} \circ e^{s_3 \Psi_1} \tag{34}$$

The relation between  $\Delta\theta$  and  $s_1, s_2, s_3$  are

$$\begin{aligned}
 s_1 &= \frac{r_1(\cos \Delta\theta - 1)}{\sin \Delta\theta} \\
 s_2 &= r_1 \cdot \Delta\theta \\
 s_3 &= s_1
 \end{aligned} \tag{35}$$

where:  $r_1 = \frac{1}{c_1}$ , being  $c_1$  the value associated to  $\Psi_3$  in (25) and  $\Delta\theta \in ]-\pi, 0[ \cup ]0, \pi[$ .



**Fig. 6** **a**  $\Phi_1^0$  ( $\Delta\theta > 0$ ); **b**  $\Phi_1^0$  ( $\Delta\theta < 0$ ); **c**  $\Phi_2^0$  ( $\Delta\theta > 0$ ); **d**  $\Phi_2^0$  ( $\Delta\theta < 0$ )

Additionally, a new manoeuvre can be defined by using vector  $\Psi_4$  instead of vector  $\Psi_3$ , see Fig. 6c–d:

$$\Phi_2^0 \equiv p_0 \circ e^{s_1 \Psi_1} \circ e^{s_2 \Psi_4} \circ e^{s_3 \Psi_1} \quad (36)$$

where  $s_1$ ,  $s_2$  and  $s_3$  are determined by the same expressions presented in (35), changing  $r_1$  and  $c_1$  by  $r_2$  and  $c_2$ .

A relevant feature of this method for manoeuvres design consists on the definition of the manoeuvre parameter: they are obtained from explicit algebraic functions of the increments of the dependent variables. Therefore, given a desired change in a variable, the correspondent manoeuvre can be determined by applying the equations presented before, without using any numerical integration method or any other computational methodology with convergence problems.

### 3.2 Manoeuvres Generation: Connecting Paths

From an arbitrary distribution of obstacles in the work space, the collision free space  $\Omega_{free}$  is defines as the subset of  $\Omega$  in which no collision exists. In [19] (pp. 428–429),

it is demonstrated that given any two configurations located in a connected subset of  $\Omega_{free}$ , there exists a feasible collision-free path between them. Moreover, it is also shown that this path can be generated by combining a sequence of canonical manoeuvres. Thus, in order to take advantage of this property in a flexible way, the concept of connecting path has been introduced [14].

**Definition 5** A Connecting path is a sequence of restricted manoeuvres that allows the vehicle to move between two arbitrary configurations.

Thus, based on this concept, there are many possibilities for obtaining feasible paths. In this chapter, connecting paths have been built using the operation introduced in (2), that defines the generic paths composition. Using this formalism three groups of connecting have been considered:

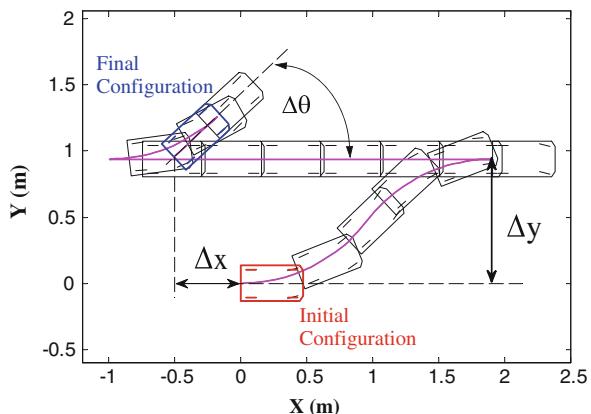
- Connecting path  $\Gamma$ : defined as:  $\Gamma \equiv \Phi^y \otimes \Phi^x \otimes \Phi^\theta$ .
- Connecting path  $\Lambda$ : defined as:  $\Lambda \equiv \Phi^\theta \otimes \Phi^x \otimes \Phi^\theta$ .
- Connecting path  $\Xi$ : defined as:  $\Xi \equiv \Phi^x \otimes \Phi^\theta \otimes \Phi^x$ .

### 3.2.1 Connecting Path $\Gamma$

Let  $(x_0, y_0, \theta_0)$  and  $(x_f, y_f, \theta_f)$  be two arbitrary configuration expressed in the global reference frame. Then, in the reference frame  $\Sigma_L$ , a connecting path will move the vehicle from  $(0,0,0)$  to  $(\Delta x, \Delta y, \Delta\theta)$  with:  $\Delta x = x_f - x_0$ ;  $\Delta y = y_f - y_0$ ,  $\Delta\theta = \theta_f - \theta_0$ . Connecting path  $\Gamma$  is defined as the combination of three restricted manoeuvres (see Fig. 7):

- Manoeuvre  $\Phi^y$  for achieving the increment  $\Delta y$ .
- Manoeuvre  $\Phi^x$  for achieving the increment  $\Delta x$ .
- Manoeuvre  $\Phi^\theta$  for achieving the increment  $\Delta\theta$ .

**Fig. 7** Connecting Path  $\Gamma$  definition



In Sect. 3.1 different manoeuvres  $\Phi^y$  and  $\Phi^\theta$  have been presented. Then, connecting path  $\Gamma$  represents a group of generic paths that can be defined by considering all possible combination of these restricted manoeuvres. For this purpose, the following convention is introduced. Manoeuvre  $\Phi_1^y$  will represent:  $\Phi_1^{+y}$  if a positive increment on  $y$  is needed, or  $\Phi_1^{-y}$  if a negative increment is considered. The same convention is applied for the definition of  $\Phi_2^y$ ,  $\Phi_3^y$ ,  $\Phi_4^y$ ,  $\Phi_1^\theta$  and  $\Phi_2^\theta$  arranging in six groups all the manoeuvres presented in Sects. 3.1.2 and 3.1.3. Therefore, by combining these manoeuvres, eight different connecting path  $\Gamma$  can be defined:

$$\begin{aligned} \Gamma_1 &\equiv \Phi_1^y \otimes \Phi^x \otimes \Phi_1^\theta & \Gamma_2 &\equiv \Phi_1^y \otimes \Phi^x \otimes \Phi_2^\theta \\ \Gamma_3 &\equiv \Phi_2^y \otimes \Phi^x \otimes \Phi_1^\theta & \Gamma_4 &\equiv \Phi_2^y \otimes \Phi^x \otimes \Phi_2^\theta \\ \Gamma_5 &\equiv \Phi_3^y \otimes \Phi^x \otimes \Phi_1^\theta & \Gamma_6 &\equiv \Phi_3^y \otimes \Phi^x \otimes \Phi_2^\theta \\ \Gamma_7 &\equiv \Phi_4^y \otimes \Phi^x \otimes \Phi_1^\theta & \Gamma_8 &\equiv \Phi_4^y \otimes \Phi^x \otimes \Phi_2^\theta \end{aligned} \quad (37)$$

As a consequence of these definitions, each connecting path is a generic path defined by five parameters. As a example, let consider the case of  $\Gamma_1$ . By taking into account (28), (30), (34) and (37) it can be written:

$$\begin{aligned} \Gamma_1 &\equiv \Phi_1^y(s_1, s_2, s_3) \otimes \Phi^x(s_m) \otimes \Phi_1^\theta(s_1', s_2', s_3') \\ &= p_0 \circ e^{s_3\Psi_3} \circ e^{s_2\Psi_4} \circ e^{s_1\Psi_1} \circ e^{s_m\Psi_1} \circ e^{s_1'\Psi_1} \circ e^{s_2'\Psi_3} \circ e^{s_3'\Psi_1} \end{aligned} \quad (38)$$

This expression can be simplified as:

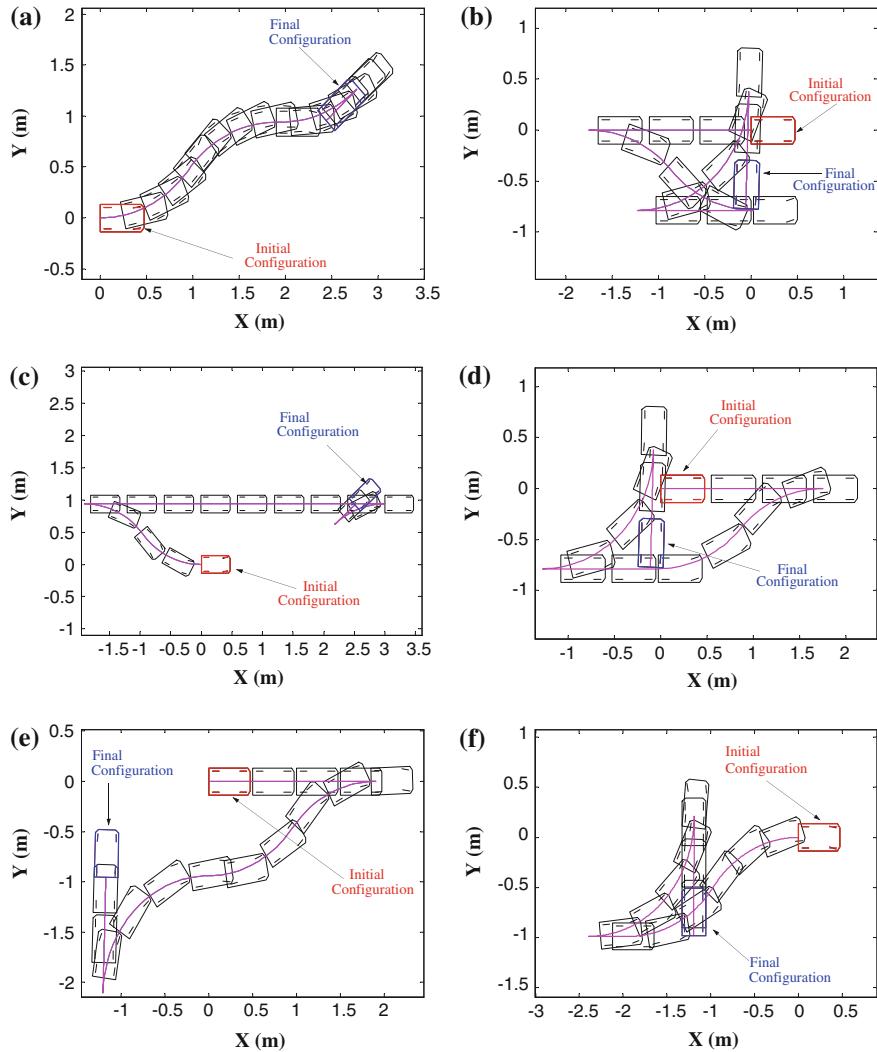
$$\Gamma_1 \equiv p_0 \circ e^{s_3\Psi_3} \circ e^{s_2\Psi_4} \circ e^{s_d\Psi_1} \circ e^{s_2'\Psi_3} \circ e^{s_3'\Psi_1} \quad (39)$$

where  $s_2, s_3, s_d = s_1 + s_m + s_1'$ ,  $s_2'$  and  $s_3'$  are determined by  $\Delta x$ ,  $\Delta y$  and  $\Delta\theta$ , considering the expressions (29), (31) and (35). In Appendix 2, the calculus of these values is detailed. The total length of the connecting path is

$$S_t = s_2 + s_3 + s_d + s_2' + s_3' \quad (40)$$

By using this methodology, any two configurations will define a family of connecting paths ( $\Gamma$ ) that represent a group of feasible trajectories linking them. Based on the simplification applied in Eq. (39), this methodology presents an outstanding property: the capability of including reversals only when necessary, adapting the shape of the trajectory to the specific values of the initial and final configuration. Figure 8 shown several examples of connecting paths  $\Gamma$  defined by various configurations. Note that trajectories look different, as the relative position of the configuration changes.

Similar to de definition of  $\Gamma$  there could be defined five more possible connecting paths just by permuting the sequence in the increments of the state variables. The procedure for obtaining the value of the parameters is similar to the one detailed in



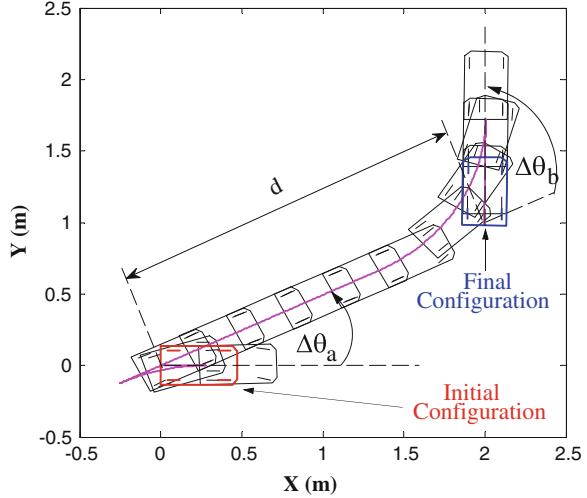
**Fig. 8** a–f Different connecting paths  $\Gamma$

this section. However, according to the author experience, resulting manoeuvres of these combinations are very similar to de set of solutions achieved with  $\Gamma$  and the connecting paths defined in subsequent sections.

### 3.2.2 Connecting Path $\Lambda$

The design of  $\Lambda$  is inspired in the following idea: the vehicle must initially change orientation a value equal to  $\Delta\theta_a$  in order to go along a straight line until it arrives

**Fig. 9** Connecting path  $\Lambda$  definition



to  $(x_f, y_f)$ , Fig. 9. The length of this straight motion,  $d$ , is equal to the Euclidean distance between the initial and the final configurations. Once located there, it must execute a reorientation manoeuvre for changing the vehicle's orientation a value equal to  $\Delta\theta_b$ , which makes the vehicle reaches the final correct heading, see Fig. 9. Then,  $\Lambda$  is composed of three basic manoeuvres:

- Manoeuvre  $\Phi^\theta$  for achieving the increment  $\Delta\theta_a$ .
- Manoeuvre  $\Phi^x$  for achieving the displacement  $d$ .
- Manoeuvre  $\Phi^\theta$  for achieving the increment  $\Delta\theta_b$ .

Similarly as was done for  $\Gamma$  definition, and considering two possible reorientation manoeuvres  $\Phi_1^\theta$  and  $\Phi_2^\theta$ , a group of four connecting paths can be established:

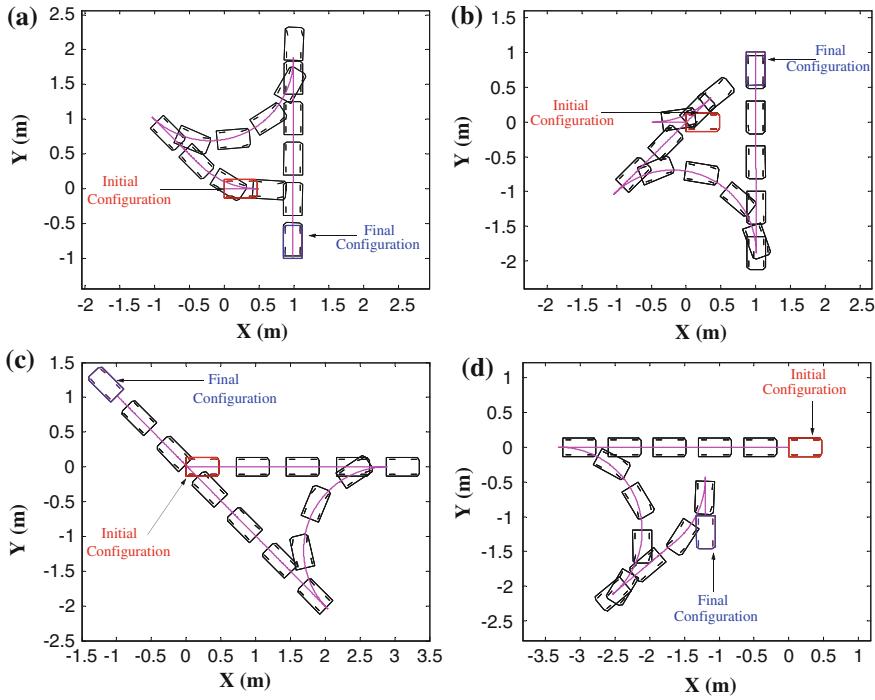
$$\begin{aligned} \Lambda_1 &\equiv \Phi_1^\theta \otimes \Phi^x \otimes \Phi_1^\theta & \Lambda_2 &\equiv \Phi_1^\theta \otimes \Phi^x \otimes \Phi_2^\theta \\ \Lambda_3 &\equiv \Phi_2^\theta \otimes \Phi^x \otimes \Phi_1^\theta & \Lambda_4 &\equiv \Phi_2^\theta \otimes \Phi^x \otimes \Phi_2^\theta \end{aligned} \quad (41)$$

Again, each connecting path is defined by five parameters, determined by  $\Delta x$ ,  $\Delta y$  and  $\Delta\theta$ , which can be computed from (29) and (35). As an example let consider the definition of  $\Lambda_1$ :

$$\begin{aligned} \Lambda_1 &\equiv \Phi_1^\theta(s_1, s_2, s_3) \otimes \Phi^x(s_m) \otimes \Phi_1^\theta(s_1', s_2', s_3') \\ &= p_0 \circ e^{s_1 \Psi_1} \circ e^{s_2 \Psi_3} \circ e^{s_3 \Psi_1} \circ e^{s_m \Psi_1} \circ e^{s_1' \Psi_1} \circ e^{s_2' \Psi_3} \circ e^{s_3' \Psi_1} \end{aligned} \quad (42)$$

This expression can be simplified as:

$$\Lambda_1 \equiv p_0 \circ e^{s_1 \Psi_1} \circ e^{s_2 \Psi_3} \circ e^{s_d \Psi_1} \circ e^{s_2' \Psi_3} \circ e^{s_3' \Psi_1} \quad (43)$$



**Fig. 10** a–d Different connecting paths  $\Lambda$

The calculus of  $s_1, s_2, s_d = s_3 + s_m + s_1', s_2'$  and  $s_3'$  is detailed in Appendix 2. The total length of the connecting path is

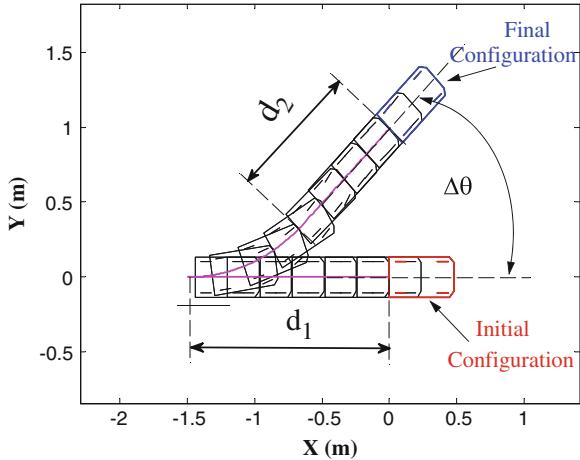
$$S_t = s_1 + s_2 + s_d + s_2' + s_3' \quad (44)$$

Figure 10 illustrates several examples of connecting paths  $\Lambda$  defined by various configurations. Again, the paths presents reversal when necessary and the shape of the trajectory depends on the relative position of the initial and goal configuration.

### 3.2.3 Connecting Path $\Xi$

This connecting path is designed to allow the vehicle to arrive to the target configuration following a straight line, see Fig. 11. Therefore,  $\Xi$  makes the vehicle move along a straight line a distance equal to  $d_1$ , there the vehicle performs a reorientation manoeuvre for achieving an increment  $\Delta\theta$ , and finally it moves along a straight line a distance equal to  $d_2$ . Thus, the connecting path is composed of three basic manoeuvres:

**Fig. 11** Connecting path  $\mathcal{E}$  definition



- Manoeuvre  $\Phi^x$  for achieving the displacement  $d_1$ .
- Manoeuvre  $\Phi^\theta$  for achieving the increment  $\Delta\theta$ .
- Manoeuvre  $\Phi^x$  for achieving the displacement  $d_2$ .

In accordance with the conventions previously introduced in Sect. 3.2.1, this connecting path can be designed in two different ways: by using manoeuvre  $\Phi_1^\theta$  or by considering manoeuvre  $\Phi_2^\theta$ . Consequently, a set of two connecting paths can be defined:

$$\mathcal{E}_1 \equiv \Phi^x \otimes \Phi_1^\theta \otimes \Phi^x \quad \mathcal{E}_2 \equiv \Phi^x \otimes \Phi_2^\theta \otimes \Phi^x \quad (45)$$

Differently from  $\Gamma$  and  $\Lambda$ , each of the connecting paths  $\mathcal{E}$  is defined by fewer parameters (three), determined by  $\Delta x$ ,  $\Delta y$  and  $\Delta\theta$ , which can be computed from (29) and (35). As a example let consider the definition of  $\mathcal{E}_1$ :

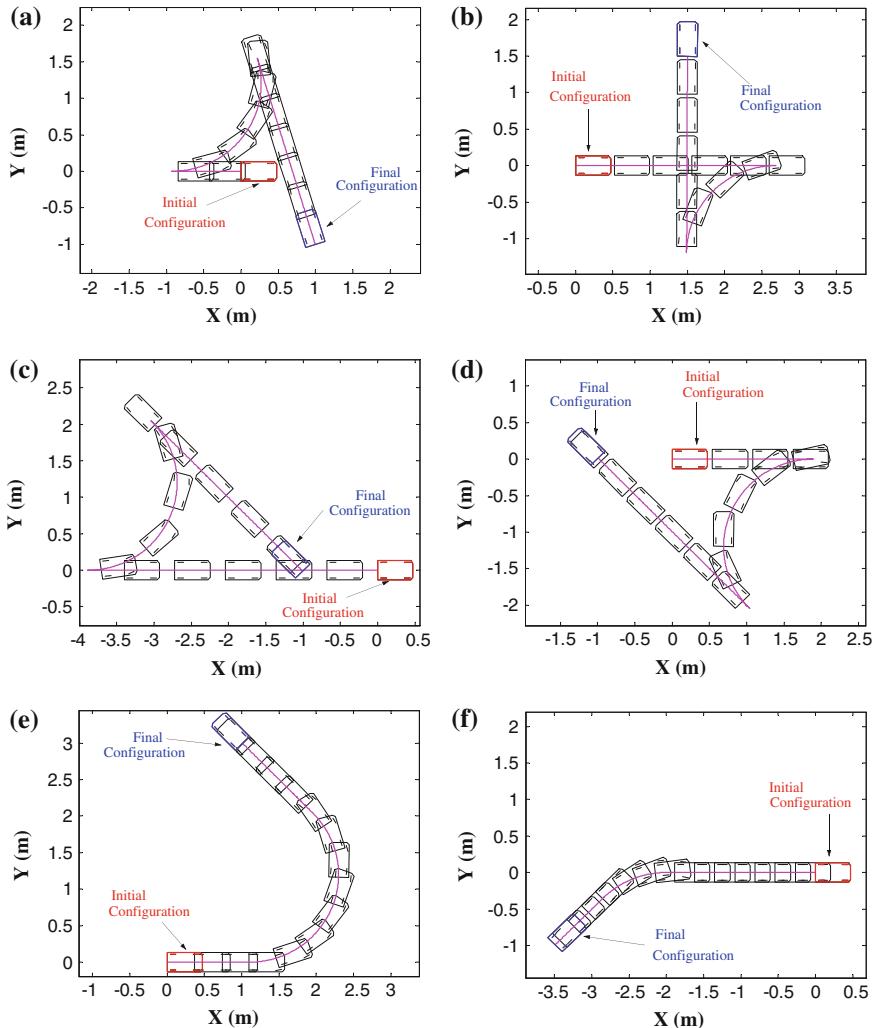
$$\begin{aligned} \mathcal{E}_1 &\equiv \Phi^x(s_1) \otimes \Phi_1^\theta(s_2, s_3, s_4) \otimes \Phi^x(s_5) \\ &= p_0 \circ e^{s_1 \Psi_1} \circ e^{s_2 \Psi_1} \circ e^{s_3 \Psi_3} \circ e^{s_4 \Psi_1} \circ e^{s_5 \Psi_1} \end{aligned} \quad (46)$$

This expression can be simplified as:

$$\mathcal{E}_1 \equiv p_0 \circ e^{s_a \Psi_1} \circ e^{s_b \Psi_3} \circ e^{s_c \Psi_1} \quad (47)$$

The calculus of  $s_a = s_1 + s_2$ ,  $s_b = s_3$  and  $s_c = s_4 + s_5$  is detailed in Appendix 2. The total length of the connecting path is

$$S_t = s_a + s_b + s_c \quad (48)$$



**Fig. 12** a–f Different connecting paths  $\mathcal{E}$

Figure 12 illustrates several examples of connecting paths  $\mathcal{E}$ . It shows that, depending on the relative position of the configurations to be linked, this connecting path can provide a large variety of paths with very different shapes: from paths including two inverter (Fig. 12a–d) to very simple trajectories without cusps (Fig. 12e–f).

As has been shown, all the parameters defining a connecting path depend only on the initial and the final configuration. Thus, given two configurations, each of the motion strategies presented before will provide a feasible solution with an associate cost  $S_t$ .

## 4 Planning Collision Free Manoeuvres from a Collision Free Route

Given an scenario, with a distribution of obstacles, and given an initial and a final configuration, it is highly unlikely to obtain a collision-free path by using a simple connecting path. Thus, different procedures can be designed in order to solve a motion planning problem by applying the methodology presented before.

The approach proposed in this chapter consists on applying, in a first step, a search algorithm in order to obtain a collision-free route, i.e., a set of intermediate collision-free configurations (known as nodes) that yield a connection between the initial and the goal configuration. In the second step, a sequence of collision-free connecting paths can be generated, so that an effective obstacle avoidance is obtained.

To this end, at the first stage different searching method can be applied: visibility graph, potential field, Voronoi diagrams, randomly-based algorithm, or any other of the algorithms presented in Chaps. 1, 2, 8 or 9 of this book. In the second step, the collision-free route is turned into a collision-free sequence of feasible manoeuvres by means of a filtering algorithm.

An example of the combination of this methodology with the algorithm Rapidly exploring Random Tree (RRT), [23–25], can be found in [14, 30]; in [32] it is combined with a manoeuvre selection process based on a multicriteria decision-making technique; an example of the combination of this methodology with a Voronoy diagram Algorithm can be found in [29]; combination of this methodology with a visibility graph can be found in [28].

The solution addressed in this chapter is a different version of the proposed in these previous works, since the optimization procedure is implemented within the filtering algorithm rather than over the set of solutions provided by the algorithm.

This section is devoted to present the filtering algorithm. Then, it is supposed that a Map representing the environment is available, and a route connecting the initial and the goal configuration has been established. The route will define a sequence of very close nodes, assuming that direct motion between them could not satisfy the kinematics constraints.

According to the method presented in Sect. 3.2 it is possible to establish an admissible connection between two arbitrary configurations of the route by using any of the proposed connecting paths. The parameters defining the connecting path will be obtained from the configurations associated to both nodes. Moreover, from this definition it is easy to check if, along the path, the vehicle intersects any obstacle. Thus, given two nodes of the route, if a collision-free manoeuvre allows the vehicle motion between both configurations, all intermediate nodes could be replaced by this new path.

The proposed methodology is based on the algorithm Path Generation, see Algorithm 1, that gradually replaces sections of the route by collision-free connecting paths. By applying this method, the original searching algorithm, that provides the collision free route, will maintain its computational natural speed, being applied without considering the nonholonomic constraints of the vehicle. Moreover, the filtering

algorithm is the responsible of transforming the route in a sequence of feasible paths accomplishing the kinematics constraints.

Thus, this technique is based on checking the possible connections between pair of nodes by using the connecting paths defined in Sect. 3.2. In this way, the original route is segmented in a sequence of motions accomplishing the kinematics constraints and the non-collision condition.

For this purpose, the Algorithm 1 starts by testing the connection between the initial and the final configuration. Each of the connecting paths is tested. If a set of feasible solutions exist, the algorithm selects the one with lower trajectory cost  $L$  and the solution is provided. Nevertheless, if none of the connecting paths is a feasible solution, the algorithm will check the connection between the initial configuration and other node of the route. The process is performed until a collision-free connection is achieved between two nodes. Then, the section of the route defined by both nodes will be replaced by the selected collision-free manoeuvre. At this stage the algorithm starts checking new connections between nodes in order to transform the remaining section of the route into a sequence of connecting paths. Various version of this algorithm have been presented in [14, 28–30] where different criteria for the route segmentation are applied; namely by changing the definition of the nodes that determines the section to check. The algorithm presented below, is a variation of these original versions. On the one hand the nodes of the route are tested one by one starting from the goal configuration; on the other hand, the selection of the connecting path is implemented by taking into account the manoeuvre cost  $L$ .

---

### Algorithm 1 Path Generation

---

**Require:** *OriginalRoute*

**Ensure:** *FinalSolution*.

```

1: final = 0
2: j = 1
3: while (Not final) do
4:   collision = 1
5:   i = NodesNumber
6:   while (collision) do
7:     if (FeasibleConection(j, i, OriginalRoute)) then
8:       FeasibleSection  $\leftarrow$  FeasibleConection(j, i, OriginalRoute)
9:       AddSection(FinalSolution, FeasibleSection)
10:      j = i
11:      collision = 0
12:    else
13:      i = i-1
14:    end if
15:    if (i-j=1) then
16:      FeasibleSection  $\leftarrow$  BasicManoeuvreComposition(j, i, OriginalRoute)
17:      AddSection(FinalSolution, FeasibleSection)
18:      collision = 0
19:    end if
20:  end while
21: end while

```

---

The function `FeasibleConection()` (see Algorithm 2) is the responsible for defining the path and checking the vehicle collisions. It provides, if it exists, the collision-free connecting path with the lower cost  $L$  between configurations  $j$  and  $i$ .

Within this algorithm, the function `ConectingPathGeneration` implements the trajectory generation, linking configurations  $j$  and  $i$ . The index  $m$  is used for addressing the type of connecting path, being  $m = 1$  associated with  $\mathcal{E}_1$ ,  $m = 2$  with  $\mathcal{E}_2$ , ... until  $m = 13$  is associated with  $\Gamma_8$ .

The collision test is performed in the function `TestColision`. It can be implemented following different methods. On the one hand, the test can be performed by checking directly the intersection of the obstacles with the vehicle's contour along the path. On the other hand, it can be implemented by using the configuration space. An occupancy grid map can be previously defined over the configuration space, so that each configuration is associated to a grid that contains the information about the collision state. Then, a connecting path can be tested by sampling the trajectory in the configuration space and checking the grids associated with the samples. Other alternatives have been explored in the literature, thus, it is possible to obtain a template of the space where the vehicle moves along the manoeuvre [14, 28, 30]. By this way, the existence of collisions can be tested without computing the trajectory, just using this template. In any case, collision detection can be easily done, whether using geometric algorithm or by applying occupancy grid maps.

The calculus of the trajectory cost  $L$  can be implemented according to different criteria: length of the trajectory ( $S_t$ ), distance to obstacles, reverse distance, smoothness in the steering angle changes, see [32] for details. The function `FeasibleConection` applied the value of  $L$  for selecting the connecting path that minimizes the trajectory cost.

Within the Path Generation algorithm, The function `AddSection()` links the collision-free feasible connections so that, at the end, `FinalSolution` will contain a sequence of feasible paths connecting the initial and the goal configuration.

If a collision-free motion between  $j$  and  $i$  is not possible, index  $i$  is decreased and the connection between  $j$  and  $i - 1$  is tested. Oddly, all the connections fail, including the connection between two consecutive nodes. In this case, two possibilities can be considered. The first consists on discarding the search for a solution. However, if the algorithm applied for the route generation assures the existence of non constrained collision-free motions between consecutive nodes, a strategy similar to the one presented in [19] can be applied, connecting both nodes by a repetitive sequence of sidewise and reorientation manoeuvres. These two alternatives can be implemented in the function `BasicManoeuvreComposition()`.

As an example of the efficiency of this methodology, a feasible path achieved by this procedure is shown in Fig. 13. Due to the curvature constraint this scenario prevents the vehicle to move between the initial and final configuration without stopping and changing the sign of the velocity. This figure represents, with continuous line, the path described by the vehicle's rear point and some intermediate configurations drawn in dashed line. This solution has been obtained by applying the filtering algorithm to a route generated by a RRT algorithm.

**Algorithm 2** Feasible Connection

---

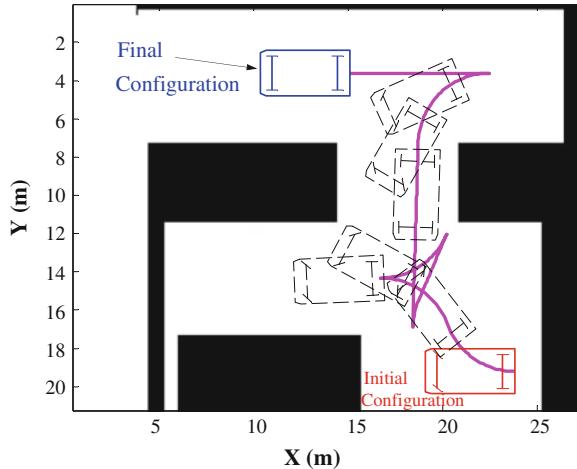
**Require:**  $j, i, \text{OriginalRoute}$

**Ensure:**  $\text{Trajectory}$  if a feasible connection exists,  $\text{Null}$  otherwise.

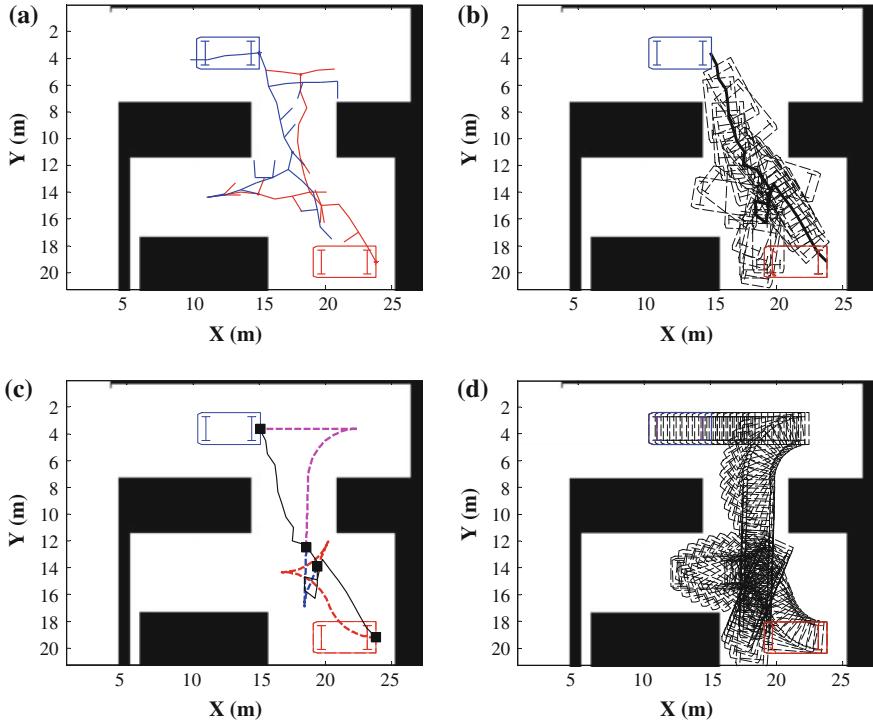
- 1:  $L_{min} = 0$
- 2: **for**  $m=1$  to  $13$  **do**
- 3:    $collision = 1$
- 4:    $(Path, T_t) \leftarrow \text{ConectingPathGeneration}(m, \text{OriginalRoute}[j], \text{OriginalRoute}[i])$
- 5:    $collision = \text{TestColision}(Path)$
- 6:    $L \leftarrow (T_t, \text{other criteria...})$
- 7:   **if** ( $Collision=0$ ) and ( $L_{min} = 0$ ) **then**
- 8:      $L_{min} = L$     // this is the first feasible connection
- 9:   **end if**
- 10:   **if** ( $Collision=0$ ) and ( $L \leq L_{min}$ ) **then**
- 11:      $Trajectory \leftarrow Path$
- 12:      $L_{min} = L$
- 13:   **end if**
- 14: **end for**
- 15: **if**  $collision = 1$  **then**
- 16:   **return**  $\text{Null}$
- 17: **else**
- 18:   **return**  $Trajectory$
- 19: **end if**

---

**Fig. 13** Manoeuvering in a constrained scenario



In Fig. 14 four different moments of this planning methodology are presented. In Fig. 14a the trees provided by RRT are shown. In Fig. 14b the route derived from the trees is drawn in continuous line, also the associated configurations are represented using dashed lines. The algorithm builds the solution of Fig. 13 by chaining three connecting paths:  $\Gamma_1$ ,  $\Xi_2$  and  $\Xi_2$ . Figure 14c illustrates the way in which the route is sectioned in order to be turned into a feasible sequence of manoeuvres. It presents, in dashed line, the final path and in continuous line, the original route. Both lines have been split by placing square markers at the limit of each sections, showing the



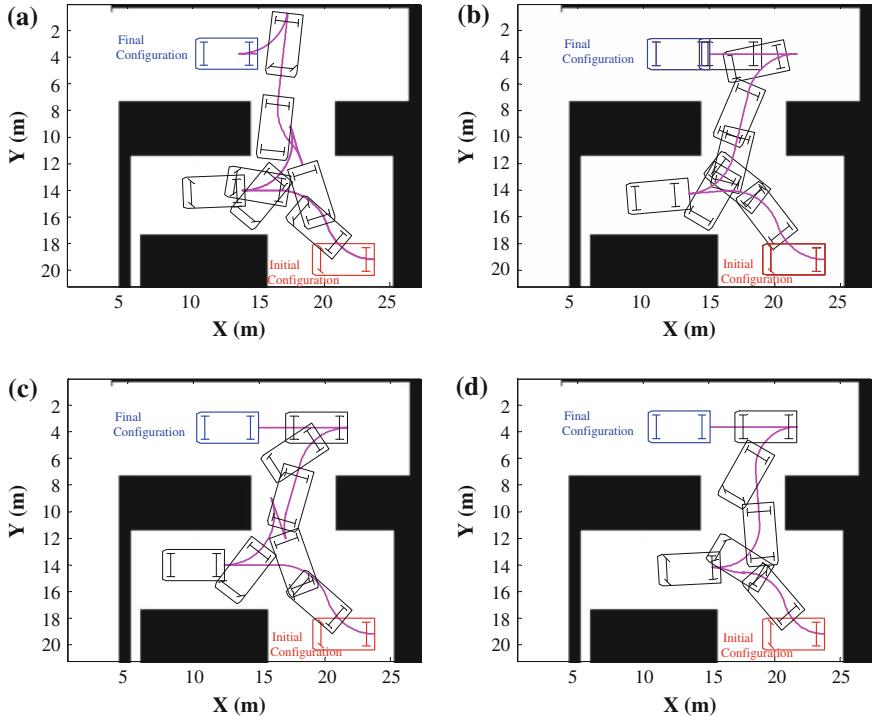
**Fig. 14** Planning process sequence: **a** RRT's trees; **b** the route derived from the trees; **c** sections of the route and sequence of feasible manoeuvres; **d** shadow of the car along the manoeuvre

piece of path associated to each connecting path. Finally, in Fig. 14d the shadow of the vehicle along the trajectory is shown, in order to demonstrate that the generated trajectory is a collision-free solution.

Regarding the smoothness in the curvature changes, it represents a question beyond the scope of this chapter. However, it is worth mentioning several contributions that address solutions to continuous curvature trajectories generation, based on path defined from circular and straight segments: [9, 13, 38]. They can be combined with the approach presented in this chapter.

## 5 Experimental Results

This section is devoted to present some results achieved with the proposed approach. Different scenarios have been considered in order to illustrate the applicability and efficiency of this methodology.



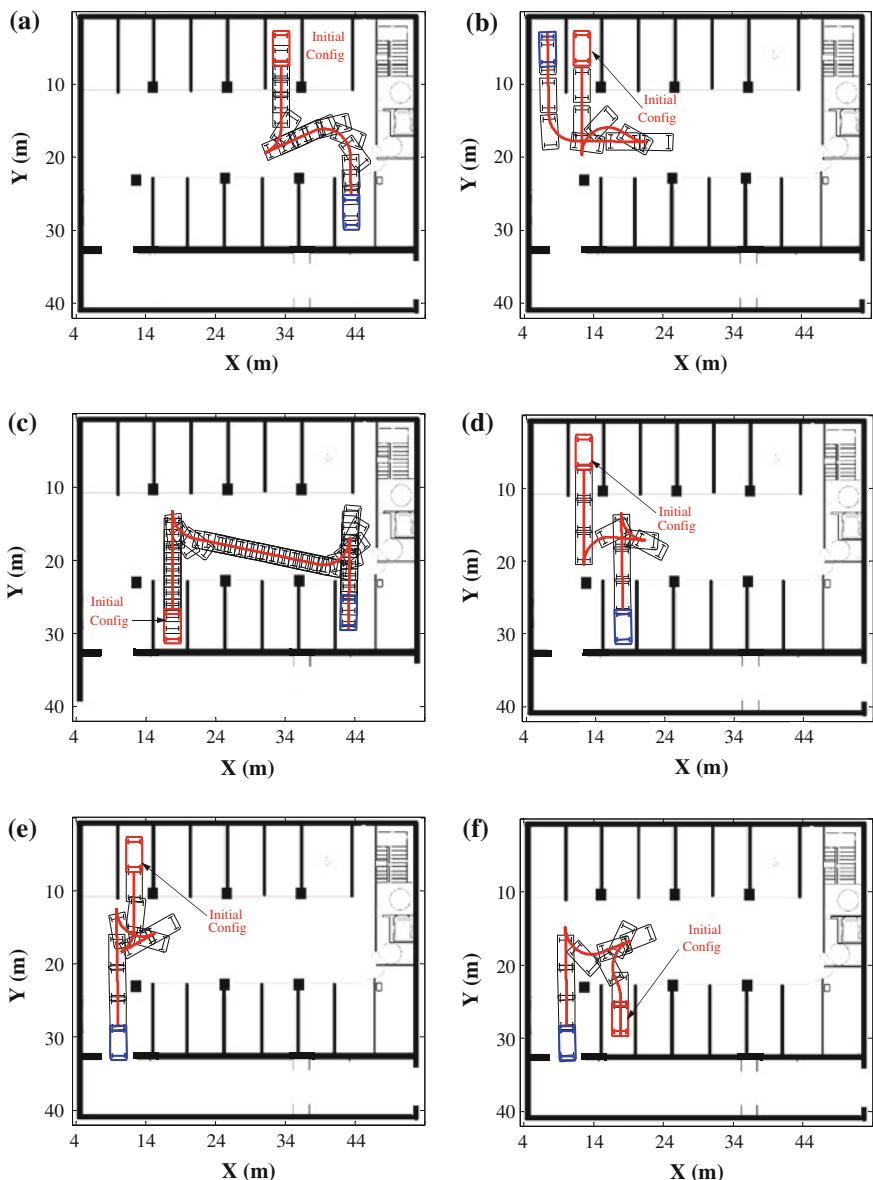
**Fig. 15 a–d** A set of solutions to the planning problem introduced in Sect. 4

As was mentioned before, the algorithm presented in the previous section can be applied to routes generated by several search algorithm. Namely, the results presented below have been obtained by using RRT. A study with comparisons of the combination of this filtering algorithm whit other searching techniques is shown in [28].

Regarding the dimension of the vehicle, the length is 4.72 m, the vehicle's width is 2.36 m,  $l$  is equal to 4.32 m and the minimum radius of curvature is 4.32 m, thus the maximum steering angle is  $\frac{\pi}{2}$  rad.

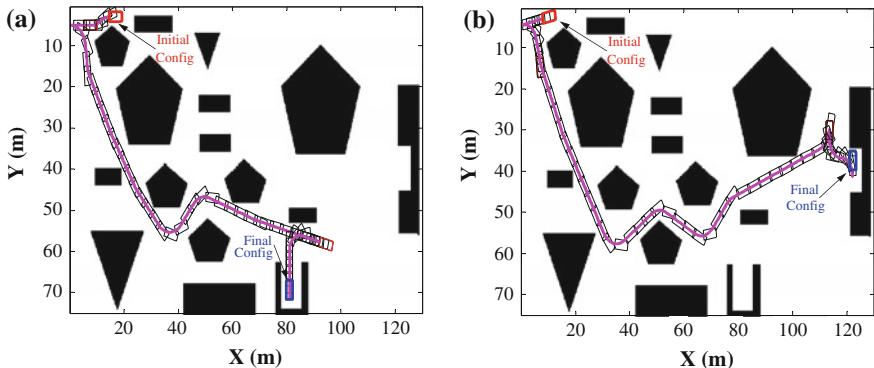
Due to the random nature of RRT, different experiments in the same scenario, and with the same initial and final conditions, provide different routes. Therefore, successive applications of RRT and the filtering algorithm also generates different final solutions. In figure Fig. 15 four different trajectories, solving a situation similar to the one presented in Fig. 13, are shown. This solution set approximately illustrates the admissible way of motions of the vehicle in this scenario. The combination of RRT's with the filtering algorithm, is characterized by this feature: it provides a wide collection of different solutions for a particular situation. This property has been exploited in [32], where decision making techniques were applied in order to select the manoeuvre that best fits a specific scenario.

This approach work efficiently when daily life circumstances are involved. As an example, in Fig. 16 a set of six trajectories are shown. Each one represents a motion



**Fig. 16** Different experiments for car-like vehicle manoeuvring: **a–f** manoeuvring for changing the parking place; **e–f** garage exit manoeuvres

of the vehicle in a garage where it has to move from one to another parking place,



**Fig. 17** a–b Two experiments in a large scenario

see Fig. 16a–d. Moreover, manoeuvres to leave the garage have been also obtained. Particularly, two exit paths that begin at two different parking places are represented in Fig. 16e–f. In these experiments, initial and final configuration have been selected so that simple motions between them are not feasible. Thus, the proposed methodology proves that suitable and feasible solutions can be obtained from the concepts of restricted manoeuvre and connecting path when, constrained scenarios are involved.

Finally, there have been also performed experiments in large scale scenarios, where short manoeuvres has to be combined with large trajectories without cusps. In Fig. 17 two experiments are presented, in a scenario 80 m. wide and 130 m. long. Each experiment has been defined so that initial and final configuration requires short motion among closed obstacles. However, motion between them can be obtained by a continuous trajectories without stopping the vehicle for changing the velocity sign. It is remarkable that the algorithm provides trajectories with and without cusp, incorporating manoeuvres only when direct motion is not possible.

Computing time depends drastically on two main aspects: the methodology for manoeuvre evaluation (defined in function `FeasibleConection()`) and the scenario complexity.

In the experiments presented in this section, collision test has been implemented by checking a grid map of the Cartesian space with a discrete representation of the vehicle's contour. The cost of a manoeuvre is the sum of the path length plus factor that weight the proximity to the obstacles along the path. This factor increases in value every time a configuration lays too close to an obstacle.

The average time of the experiments in Fig. 16 is 1.5 s. with a dispersion of 0.3 s. Experiments in Fig. 17 took an average of 1.8 s. All of them were programmed using Matlab V-7.11 on a PC with an Intel Core i3 1.4 GHz processor.

## 6 Conclusions

This chapter is devoted to present an approach for car-like robot manoeuvre generation. The proposed technique is described in a context of differential geometry concepts related with the vehicles kinematics constraints. The procedure is based on the definition of a set of basic manoeuvres (restricted manoeuvres) that allow changing the value of only one of the configuration variables. Thus, these manoeuvres represent a set of canonical paths, that is, any two point of the configuration space can be connected by a suitable combination of them. In order to achieve a suitable manoeuvre combination a heuristic procedure has been established by introducing the concept of connecting paths. Three basic connecting paths have been defined:  $\Gamma$ ,  $\Lambda$  and  $\Xi$ . These connecting paths represent a flexible way of motion, providing efficient and feasible trajectories with different patterns of movement in resemblance to the manoeuvres performed by human drivers. The final step of the approach consist on applying a filter algorithm to a collision-free route (provided by one of the traditional searching algorithm) so that the original route is transformed into a sequence of feasible collision-free connecting paths, implementing an effective obstacle avoidance. The efficiency of the approach is illustrated by a set of experimental results involving daily live scenarios.

## Appendix 1: Controllability Condition

This Appendix introduces some concepts and mathematical formulation that allow to understand the car-like vehicles controllability. Further explanation regarding the application of differential geometry can be found in [4, 19, 26]

**Definition 6** Let  $\Psi_i(q)$  and  $\Psi_j(q)$  be two vector field, the Lie bracket  $[\Psi_i(q), \Psi_j(q)]$  is a third vector field defined by:

$$[\Psi_i(q), \Psi_j(q)] = \frac{\partial \Psi_j(q)}{\partial q} \Psi_i(q) - \frac{\partial \Psi_i(q)}{\partial q} \Psi_j(q) \quad (49)$$

**Theorem 1** Let be a system, whit a configuration space  $\Omega$  of dimension  $n$ , with  $m$  nonholonomic constraints, and a distribution  $\Delta$ , with dimension  $(n - m)$ , spanned by a set of vectors field  $\Psi_i(q)$  accomplishing the  $m$  constraints. This system is fully controllable if the dimension of the distribution generated by the set of vectors and the recursive computation of all their Lie brackets, usually known as  $CLA(\Delta)$ , is equal to  $n$ .

This theorem is a summary of all fundamental theorems about the controllability of nonholonomic systems presented and demonstrated in [19, 22].

Let consider de vectors  $\Psi_1$ ,  $\Psi_3$  and  $\Psi_4$ , defined in (24) and (25). It is straightforward to demonstrate that the dimension of  $\Delta_c = \text{span}\{\Psi_1, \Psi_3, \Psi_4\}$  is 2, i.e.  $\text{rank} [\Psi_1, \Psi_3, \Psi_4] = 2$ .

The recursive Lie brackets of this vector set can be computed as:

$$\begin{aligned}\Psi_5 &= [\Psi_1, \Psi_3] = [c_1 \sin \theta, -c_1 \cos \theta, 0] \\ \Psi_6 &= [\Psi_1, \Psi_4] = [c_2 \sin \theta, -c_2 \cos \theta, 0] \\ \Psi_7 &= [\Psi_3, \Psi_4] = [(c_2 - c_1) \sin \theta, (c_1 - c_2) \cos \theta, 0]\end{aligned}\tag{50}$$

Thus, it is easy to show that:

$$\text{rank} [\Psi_1, \Psi_3, \Psi_4, \Psi_5, \Psi_6, \Psi_7] = 3\tag{51}$$

Therefore, the dimension of  $\text{CLA}(\Delta_c) = 3$ , and the system is fully controllable, that is, any configuration can be reached by using a finite sequence of paths that follow vectors  $\Psi_1, \Psi_3$  and  $\Psi_4$ .

## Appendix 2: Parameters Calculation

### $\Gamma_1$ Parameters Calculation

According to the Sect.(3.2.1) the expression defining this connecting path is:

$$\begin{aligned}\Gamma_1 &\equiv \Phi_1^y \otimes \Phi^x \otimes \Phi_1^\theta \\ &= p_0 \circ e^{s_3 \Psi_3} \circ e^{s_2 \Psi_4} \circ e^{s_d \Psi_1} \circ e^{s_2' \Psi_3} \circ e^{s_3' \Psi_1}\end{aligned}\tag{52}$$

The objective is to determine the values of the parameters as a function of the initial and the final configurations. Let  $(x_0, y_0, \theta_0)$  and  $(x_f, y_f, \theta_f)$  be respectively the initial and final configuration expressed in the global frame  $\Sigma_G$ , and let  $p_0^G = [x_0, y_0, \theta_0, 1]^T$  and  $p_f^G = [x_f, y_f, \theta_f, 1]^T$  be the extended vector containing these configurations.

First of all, both configurations has to be expressed in the frame  $\Sigma_L$  in order to obtain the desired increment of the state variables in this frame.

Given that the origin of  $\Sigma_L$  is located at the manoeuvre starting point, i.e. in  $[x_0, y_0]$ , where the initial vehicle's heading is equal to zero, it is easy to demonstrate that the relation between  $p^G$ , the extended vector of a configuration in  $\Sigma_G$ , and  $p^L$ , the extended vector of this configuration in  $\Sigma_L$ , is determined by the following expressions:

$$p^G = T_L^G \cdot p^L \quad \text{with} \quad T_L^G = \begin{pmatrix} \cos \theta_0 & -\sin \theta_0 & 0 & x_0 \\ \sin \theta_0 & \cos \theta_0 & 0 & y_0 \\ 0 & 0 & 1 & \theta_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\tag{53}$$

Thus, initial and final vectors can be written in  $\Sigma_L$  in accordance with:

$$p_0^L = T_G^L \cdot p_0^G \quad p_f^L = T_G^L \cdot p_f^G \quad \text{with} \quad T_G^L = (T_L^G)^{-1} \quad (54)$$

Obviously,  $p_0^L = [0, 0, 0, 1]^T$  and the desired increments in  $\Sigma_L$  are:

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta \theta \\ 1 \end{pmatrix} = p_f^L = T_G^L \cdot p_f^G \quad (55)$$

Then,  $s_1, s_2$  and  $s_3$ , can be calculated from  $\Delta y$  by applying Eq. (31);  $s_m = \Delta x$ ; and  $s_1', s_2'$  and  $s_3'$  can be obtained from  $\Delta \theta$  by applying Eq. (35). Finally and following the reasoning about the simplification process expressed in Sect. (3.2.1)  $s_d = s_1 + s_m + s_1'$ . That way, the five parameters  $s_2, s_3, s_d, s_2'$  and  $s_3'$  of Eq. (52) have been determined as a function of two arbitrary configurations to be connected. The calculus of the rest of the  $\Gamma$  connecting paths is similar to the one presented in this section.

## A1 Parameters Calculation

According to the Sect. (3.2.2) the expression defining this connecting path is:

$$\begin{aligned} \Lambda_1 &\equiv \Phi_1^\theta \otimes \Phi^x \otimes \Phi_1^\theta \\ &= p_0 \circ e^{s_1 \Psi_1} \circ e^{s_2 \Psi_3} \circ e^{s_d \Psi_1} \circ e^{s_2' \Psi_3} \circ e^{s_3' \Psi_1} \end{aligned} \quad (56)$$

The procedure for determining the parameters is based on the calculus of the vector  $[\Delta x, \Delta y, \Delta \theta, 1]^T$  from the initial and the final configurations, as was done in the previous section. From this vector the following values are calculated:

$$d = \sqrt{(\Delta x^2 + \Delta y^2)}; \quad \Delta \theta_a = \arctan\left(\frac{\Delta y}{\Delta x}\right); \quad \Delta \theta_b = \Delta \theta - \Delta \theta_a \quad (57)$$

Then,  $s_1, s_2$  and  $s_3$ , can be calculated from  $\Delta \theta_a$  by applying Eq. (35);  $s_m = d$ ; and  $s_1', s_2'$  and  $s_3'$  can be obtained from  $\Delta \theta_b$  by applying again Eq. (35).

Finally and following the reasoning about the simplification process  $s_d = s_3 + s_m + s_1'$ . Therefore, the five parameters  $s_1, s_2, s_d, s_2'$  and  $s_3'$  of Eq. (56) have been determined as a function of two arbitrary configurations to be connected. The calculus of the rest of the  $\Lambda$  connecting paths is similar to the one presented in this section.

## **$\Xi_1$ Parameters Calculation**

According to the Sect. (3.2.3) the expression defining this connecting path is:

$$\begin{aligned}\Xi_1 &\equiv \Phi^x \otimes \Phi_1^\theta \otimes \Phi^x \\ &= p_0 \circ e^{s_a \Psi_1} \circ e^{s_b \Psi_3} \circ e^{s_c \Psi_1}\end{aligned}\quad (58)$$

The procedure for determining the parameters is based on the calculus of the vector  $[\Delta x, \Delta y, \Delta\theta, 1]^T$  from the initial and the final configurations, as was done in the previous section. From this vector the following values are calculated:

$$b = \Delta y - \Delta x \cdot \tan \Delta\theta; \quad d_1 = \frac{-b}{\tan \Delta\theta}; \quad \theta_m = \text{atan2}(\Delta y, \Delta x - d_1) \quad (59)$$

$$s = \begin{cases} 1 & \text{if } \Delta\theta = \theta_m; \\ -1 & \text{otherwise} \end{cases}; \quad d_2 = s \cdot \sqrt{\Delta y^2 + (\Delta x - d_1)^2}; \quad (60)$$

Then,  $s_1 = d_1$ ;  $s_2, s_3$  and  $s_4$ , can be calculated from  $\Delta\theta$  by applying Eq. (35); and  $s_5 = d_2$ . Again, following the reasoning about the simplification process  $s_a = s_1 + s_2$ ,  $s_b = s_3$ , and  $s_c = s_4 + s_5$ . Thus, the parameters of Eq. (58) have been determined as a function of two arbitrary configurations to be connected. The calculus of  $\Xi_2$  is similar to the one presented in this section.

## References

1. Brandao AS, Sasaki AS, Castelano C, Cruz RR, Carelli R (2012) Autonomous navigation with obstacle avoidance for a car-like robot. In: Robotics symposium and Latin American robotics symposium (SBR-LARS). IEEE, Brazilian, pp 156–161
2. Brockett RW (1983) Asymptotic stability and feedback stabilization. In: Brockett RW, Millman RS, Sussmann HJ (eds) Differential geometric control theory, pp 181–191
3. Chitsaz H, LaValle SM, Balkcom DJ, Mason MT (2009) Minimum wheel-rotation paths for differential-drive mobile robots. Int J Robot Res 28(1):66–80
4. Choset H, Lynch KM, Hutchinson S, Kantor GA, Burgard W, Kavraki LE, Thrun S (2005) Principles of robot motion: theory, algorithms, and implementations. MIT Press, Cambridge
5. Coelho P, Nunes U (2003) Lie algebra application to mobile robot control: a tutorial. Robotica 21(5):483–493
6. Cuesta F, Gomez-Bravo F, Ollero A (2004) Parking manoeuvres of industrial-like electrical vehicles with and without trailer. IEEE Trans Ind Electr 51(2):257–269
7. Demirli K, Khoshnejad M (2009) Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy sensor-based controller. Fuzzy Sets Syst 160(19):2876–2891
8. Dubins LE (1957) On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. Am J Math 79:497–516
9. Fraichard T, Scheuer A (2004) From reeds and Shepp's to continuous-curvature paths. IEEE Trans Robot 20(6):1025–1035
10. Giordano PR, Vendittelli M (2009) Shortest paths to obstacles for a polygonal Dubins car. IEEE Trans Robot 25(5):1184–1191

11. Gomez-Bravo F (2001) Planificación de maniobras en sistemas robóticos no holónomos. aplicaciones en robots móviles. Ph.D. Thesis, University of Seville
12. Gomez-Bravo F, Cuesta F, Ollero A (2001) Parallel and diagonal parking in nonholonomic autonomous vehicles. *Eng Appl Artif Intell* 14(1):419–434
13. Gomez-Bravo F, Cuesta F, Ollero A, Viguria A (2008) Continuous curvature path generation based on  $\beta$ -spline curves for parking manoeuvres. *Robot Auton Syst* 56(4):360–372
14. Gomez-Bravo F, Ollero A, Cuesta F, Lopez DA (2008) A new approach for car-like robots manoeuvring based on RRT. *Robotica, Instrumentaao e control* 15(71):10–14
15. Gomez-Bravo F, Lopez DA, Cuesta F, Ollero A (2007) RRT-D: a motion planning approach for autonomous vehicles based on wireless sensor network information. In: Proceedings of the 6th IFAC symposium on intelligent autonomous vehicles
16. Gomez-Bravo F, Ollero A (1995) Dynamic path planning of free floating manipulator. In: Proceedings of the 1995 ICAR conference, vol 1. pp 447–457
17. Gomez-Bravo F, Ollero A, Cuesta F, López DA (2007) A new approach for car-like robots manoeuvring based on RRT. In: Proceedings of the 7th conference on mobile robots and competitions
18. Lamiriaux F, Lammond JP (2001) Smooth motion planning for car-like vehicles. *IEEE Trans Robot Autom* 17(4):498–501
19. Latombe JC (1991) Robot motion planning. Kluwer Academic Publisher
20. Laumond JP (1986) Feasible trajectories for mobile robots with kinematic and environment constraints. In: Intelligent autonomous systems, An international conference. North-Holland Publishing Co, pp 346–354
21. Laumond JP, Sekhavat S, Lamiriaux F (1998) Guidelines in nonholonomics motion planning for mobile robots. In: Robot motion planning and control, pp 1–53
22. Laumont J, Jacobs P, Taix M, Murray M (1994) A motion planner for nonholonomic mobile robots. *IEEE Trans Robot Autom* 10(5):577–593
23. LaValle SM, Kuffner JJ (1999) Randomized kinodynamic planning. In: Proceedings IEEE international conference on robotics and automation, pp 473–479
24. LaValle SM, Kuffner JJ (2001) Rapidly-exploring random trees: progress and prospects. In: Algorithmic and computational robotics. New Directions
25. LaValle SM (1998) Rapidly-exploring random trees: a new tool for path planning. Computer Science Department, Iowa State University (TR 98-11)
26. LaValle SM (2006) Planning algorithms. Cambridge University Press
27. Lini G, Piazz A, Consolini L (2011) Multi-optimization of  $\eta$  3-splines for autonomous parking. In: 50th IEEE conference on decision and control and European control conference (CDC-ECC). IEEE, pp 6367–6372
28. Lopez Garcia DA (2011) Nuevas aportaciones en algoritmos de planificación para la ejecución de maniobras en robots autónomos no holónomos. Ph.D. Thesis, University of Huelva
29. López García DA, Gomez-Bravo F (2012) VODEC: a fast Voronoi algorithm for car-like robot path planning in dynamic scenarios. *Robotica* 30(07):1189–1201
30. Lopez D, Gomez-Bravo F, Cuesta F, Ollero A (2006) Planificación de trayectorias con el algoritmo RRT. aplicación a robots no holónomos. *RIAII* 3(3):56–67
31. Markoff J (2010) Google cars drive themselves, in traffic. *The New York Times* 10:A1
32. Martín Ramos J, López García D, Gómez-Bravo F, Blanco Morón A (2010) Application of multicriteria decision-making techniques to manoeuvre planning in nonholonomic robots. *Expert Syst Appl* 37(5):3962–3976
33. Melzak Z (1961) Plane motion with curvature limitations. *J Soc Ind Appl Math* 9(3):422–432
34. Michalek M, Kozowski K (2010) Vector-field-orientation feedback control method for a differentially driven vehicle. *IEEE Trans Control Syst Tech* 18(1):45–65
35. Miller G, Reed R, Wheeler F (1991) Optimum Ackerman for improved steering axle tire wear on trucks. Technical Report 912693, SAE Technical Paper
36. Mukherjee R, Anderson A (1994) A surface integral approach to the motion planning of nonholonomic systems. *J Dyn Syst, Meas, Control* 116(3):315–325

37. Munoz VF, Ollero A (1995) Smooth trajectory planning method for mobile robots. In: Proceedings of the congress on computational engineering in system applications. Lille, Francia, pp 700–705
38. Munoz VF, Cerezo AG, Cruz A (1999) A mobile robots trajectory planning approach under motion restrictions. *J Integr Comput-Aided Eng* 6(4):331–347
39. Murray RM, Sastry S (1993) Nonholonomics motion planning: steering using sinusoids. *IEEE Trans Autom Control* 38(5):700–716
40. Ollero A, Arrue BC, Ferruz J, Heredia G, Cuesta F, Lopez-Pichaco F, Nogales C (1999) Control and perception components for autonomous vehicle guidance. Application to the ROMEO vehicles. *Control Eng Pract* 7(10):1291–1299
41. Paromtchik I, Laugier C, Gusev S, Sekhavat S (1998) Motion control for autonomous car maneuvering. In: Proceedings of the international conference on control, automation, robotics and vision, pp 136–140
42. Qu Z, Wang J, Plaisted CE, Hull RA (2006) Global-stabilizing near-optimal control design for nonholonomic chained systems. *IEEE Trans Autom Control* 51(9):1440–1456
43. Reeds J, Shepp L (1990) Optimal paths for a car that goes both forwards and backwards. *Pac J Math* 145(2):367–393
44. Rumsyantsev VV (2000) Forms of Hamiltons principle for nonholonomic systems. *Facta Univ Ser Mech, Autom Control Robot* 2(19):1035–1048
45. Scheuer A, Fraichard T (1997) Continuous-curvature path planning for car-like vehicles. *Proc IROS* 97:997–1002
46. Thrun S (2006) Winning the DARPA Grand challenge. In: PKDD 2006. LNCS (LNAI), vol 4213. Springer, Heidelberg, pp 4–4
47. Tilbury D, Murray RM, Shankar Sastry S (1995) Trajectory generation for the n-trailer problem using Goursat normal form. *IEEE Trans Autom Control* 40(5):802–819
48. Tilbury D, Laumond JP, Murray R, Sastry S, Walsh G (1992) Steering car-like systems with trailers using sinusoids. In: Proceedings IEEE international conference on robotics and automation. IEEE, pp 1993–1998

# Vehicle Autonomy Using Cooperative Perception for Mobility-on-Demand Systems

**Seong-Woo Kim, Tirthankar Bandyopadhyay, Baoxing Qin, Zhuang Jie Chong, Wei Liu, Xiaotong Shen, Scott Pendleton, James Guo Ming Fu, Marcelo H. Ang Jr., Emilio Frazzoli and Daniela Rus**

**Abstract** A holy grail of research in urban transportation systems is to increase throughput of people while minimizing the requirement of building additional road and rail networks. The promising new paradigm of Mobility-on-Demand (MoD), where shared personal transportation vehicles provide necessary service, is fast becoming a viable and preferred alternative to the traditional framework of having either public fixed route service or privately owned vehicles. This chapter looks at innovative approaches that enable an MoD system to be economical in development, robust in operation, efficient and sustainable in deployment. We develop algorithms to efficiently use the combination of prior information with minimalistic sensing with only a single 2-D LIDAR, utilizing vehicle odometry and prior road information which may not have accurate metric information but is topologically consistent. Additionally, we use the rich capability of cooperative perception, which can far extend perception range without expensive long-range sensors, by exchanging local perception information with other vehicles or infrastructure via wireless communications. The augmented perception capability enables a vehicle to see the oncoming traffic situation ahead even beyond human line-of-sight and field-of-view, which thereby contributes to traffic flow efficiency and safety improvement through long-term perspective driving, e.g., early obstacle detection and avoidance, and early lane changing. This chapter develops these ideas, presents results in demonstration and provides insights of the motion and operation planning for a case study of a fully autonomous vehicle deployment to the general public.

---

S.-W. Kim  
Seoul National University, Seoul, Korea

T. Bandyopadhyay (✉) · J.G.M. Fu  
Singapore MIT Alliance for Research and Technology, Singapore, Singapore  
e-mail: tirtha.bandy@gmail.com

B. Qin · Z.J. Chong · W. Liu · X. Shen · S. Pendleton · M.H. Ang Jr.  
National University of Singapore, Singapore, Singapore

E. Frazzoli · D. Rus  
Massachusetts Institute of Technology, Boston, USA

**Keywords** Capítulo 11.-Mobility-on-demand systems · Cooperative perception · Autonomous vehicles · Driver assistance

## 1 Introduction

A holy grail of research in urban transportation systems is to increase throughput of people while minimizing the requirement of building additional road and rail networks. The promising new paradigm of Mobility-on-Demand (MoD), where shared personal transportation vehicles provide necessary service, is fast becoming a viable and preferred alternative [1] to the traditional framework of having either public fixed route service or privately owned vehicles. Many such MoD systems are already available commercially where the shared transportation modules are either cars [2, 3], bikes [4] or shared vans [3, 5]. These systems aim to reduce ownership of private vehicles and increase effective resource utilization.

Our research group at Singapore-MIT Alliance for Research and Technology looks at technologies that use vehicle automation as a component of MoD systems. In order for the systems to be viable, they have to be economical in development, robust in operation, efficient and sustainable in deployment. To achieve these challenging goals, we develop algorithms to efficiently use the combination of prior information with minimalistic sensing. In accordance with minimalistic sensing, our vehicles are localized reliably in the campus using only one single 2D LIDAR (Light Detection And Ranging), odometry and prior road information.

In the same context, we use the rich capability of cooperative perception, which can far extend perception range without expensive long-range sensors, by exchanging local perception information with other vehicles or infrastructure via wireless communications. The augmented perception capability enables a vehicle to see the oncoming traffic situation ahead even beyond human line-of-sight and field-of-view, which thereby contributes to traffic flow efficiency and safety improvement through long-term perspective driving, e.g., early obstacle detection and avoidance, and early lane changing [6].

As a first step of system development and deployment, we focus on the crowded campus environment of National University of Singapore, where demands on transportation and the mobility challenges due to pedestrians and other vehicles are significant. Figure 1 shows our personal transporters in operation. We have demonstrated this system successfully with accumulated autonomous run mileage of over 100 km in the campus environment.

While there has been significant progress in autonomous driving in terms of technological capability, a major challenge that remains is development of public policy that stems from a positive public perception towards adopting such a system. To get an insight into such questions and the feedback from public, we had various demonstrations to visiting dignitaries and researchers from around the world. We also had a field deployment of the system on public grounds in the Science Center where we gained valuable feedback and insights with the interaction with members of the public, some of which we present in this paper.



**Fig. 1** Our autonomous personal transporters in operation, providing Mobility-on-Demand service in a campus environment of National University of Singapore

The contribution of this chapter can be summarized as the following:

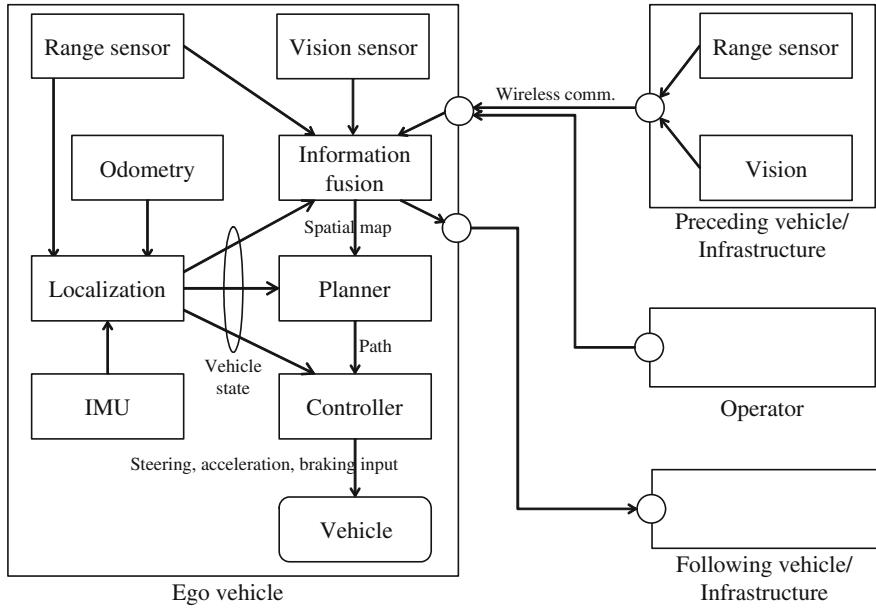
- We introduce our autonomous vehicles for Mobility-on-Demand systems.
- We propose cooperative perception for vehicle automation and planning.
- We present the case for autonomous vehicle deployment to the general public.

This chapter is outlined as follows. Section 2 provides the overview of our systems. Section 3 presents how our autonomous vehicles perceive urban road environment using local range sensors. Section 4 addresses how our systems use remote information from other vehicles or infrastructure via wireless communications in the context of cooperative perception. Section 5 presents driving assistance capability using cooperative perception via MoD systems, which enables a see-through collision warning, overtaking and lane changing assistance. In Sect. 6, we discuss the impact and next steps for autonomous vehicles in MoD systems as highlighted through the experiments and case studies of public demonstrations. We conclude this chapter in Sect. 7.

## 2 System Overview

We have operated a number of autonomous vehicles for MoD service at NUS campus, which are implemented on mass-produced automotive vehicles as shown in Fig. 1. The overall architecture of vehicles is presented in Fig. 2.

The vehicles start to move toward the next destination by self-driving upon the requests of users or operators. For this purpose, an autonomous driving system is necessary, which consists of several subsystems such as a perception, planning and control system. For the perception, the vehicles are equipped with range sensors like 2D LIDARs, a vision camera, and wireless interface IEEE 802.11g, IEEE 802.11n, 3G HSDPA (High-Speed Downlink Packet Access), and 4G LTE (Long Term Evolution).



**Fig. 2** Overall system architecture

The software architecture of this system was established on Robot Operating System (ROS) suite [7] using only open source libraries. Detail specifications are available in [8].

In Fig. 2, an odometry system is used for motion estimation such as moving direction and speed. The range sensors are used for vehicle detection and tracking. The vision sensors are used for identification and classification of pedestrians and vehicles, and provide driver-friendly visual traffic information.

The vehicles exchange their local perception results with other vehicles or infrastructure via wireless communications where the information is delivered in a form of network messages. Since there exists a trade-off between information quantity and communication performance, the message profile, e.g., transmission period and message size, should be carefully chosen according to driver preferences and application requirements. The detailed message profiles and communication performance investigation will be provided in Sect. 4.

All local and remote sensing information is properly fused at the box of information fusion in Fig. 2. Compared to data fusion of on-board sensing information, data fusion of remote sensing information on the road includes a number of practical challenges [9], among which this chapter focuses on map merging problem and sensor multi-modality.

After the information fusion procedure, the fused information can be delivered to the local planner for self-driving, or vehicles following behind or nearby infrastructure for the purpose of cooperative perception.

From the perspective of local planner, the vehicles can provide autonomous collision avoidance and lane changing through our several intelligent algorithms based on local perception and cooperative perception, which include vehicle detection, pedestrian detection, intention aware planning [10], localization [11], on-road recognition [12] and path planning [13] and cooperative perception [6].

From the perspective of cooperative perception, it can be definitely beneficial to both human drivers and autonomous vehicles for safe and comfortable driving, because the fused information includes oncoming traffic situation ahead. In this context, we provide a method to let a driver know the moment when the driver should be careful, such as hidden obstacle detection or sudden braking of preceding vehicles beyond line-of-sight. The notification is performed by visual and sound alarms for a human driver, which enables a driver to focus on driving until any dangerous situation is detected by our system. Moreover, the subsystem notifies a driver when there are any vehicles coming from behind or at blind spots, which can contribute to safe lane changing or overtaking. This notification can trigger path re-planning earlier for safe and comfort autonomous driving including smooth deceleration/acceleration and early lane changing.

From the next section, we first investigate local perception and localization using local range sensors. Then, we move our focus to cooperative perception, cooperative driving assistance, and cooperative autonomous driving.

### **3 Synthetic 2D LIDAR for Vehicle Localization in 3D Urban Environment**

Vehicle localization is a fundamental requirement for autonomous vehicle, which is the problem of determining the pose of a robot relative to a given map of the environment [14]. This section presents our precise localization algorithm for vehicles in 3D urban environment using one 2D LIDAR and odometry information. A novel idea of synthetic 2D LIDAR is proposed to solve the localization problem on a virtual 2D plane. A Monte Carlo Localization scheme is adopted for vehicle position estimation, based on synthetic LIDAR measurements and odometry information.

#### ***3.1 Localization on a Virtual Plane***

Vehicle localization on a planar surface has been studied for decades and many algorithms have been proposed. The 2D scan-matching algorithm may be the most popular choice due to its accuracy and robustness [15]. However, it cannot be directly applied for vehicles moving in the 3D world. Since outdoor road has many up-and-downs, laser points from a planar LIDAR may cast on the road surface, rather than the desired vertical objects, as discussed in [16]. Our previous research in [11] uses

a tilted-down LIDAR to extract road boundary features on urban road, and then uses these features for vehicle localization. Actually, there are many other salient features in urban environment that can benefit localization. What features to extract, how to extract them, and how to feed them into the localization scheme are questions to be further addressed.

3D range data is usually desired to extract features for robot navigating in the 3D world [17]. In this paper, we use a tilted-down LIDAR to generate 3D point cloud of the environment in a push-broom configuration.<sup>1</sup>

Rather than directly apply 3D scan-matching with the raw data [16], we try to extract features from the 3D point cloud, and use the vertical features for localization. The assumption of our method is that urban environment is rich in vertical surfaces, such as curbs, wall of buildings, and even vertical tree trunks.

The vertical world assumption is actually a popular assumption used in many works in the literature. Harrison and Newman in [18] proposed a method to generate high quality 3D laser range data while the robot is moving. By exploiting the assumption of vertical world, useful information (e.g., roll and pitch angles) can be inferred. Kohlbrecher et al. in [19] achieved 2D SLAM and 6-DOF pose estimation using one single 2D LIDAR and an IMU (Inertial Measurement Unit). Although not explicitly explained, the underlying assumption in the work is that the environment contains many vertical surfaces. Weingarten et al. in [20] used this assumption to realize fast structured environment reconstruction. Rezaei et al. proposed a navigation method in a 3D scenario with 2D LIDAR data in [21].

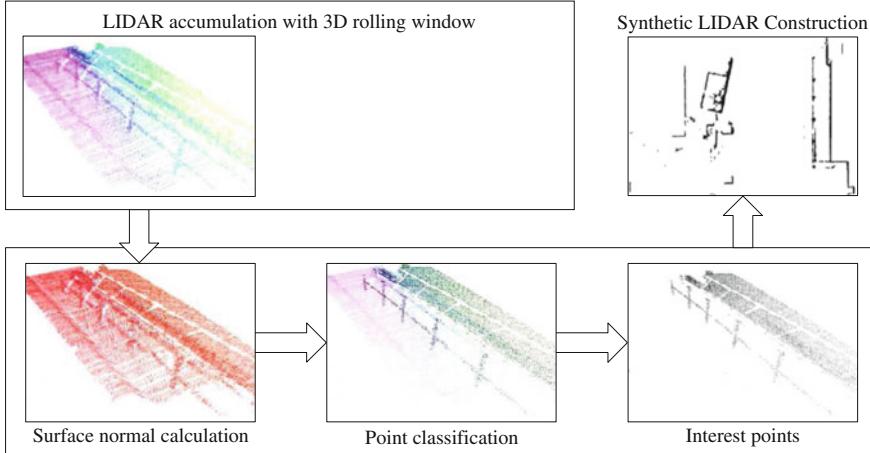
In this work, since outdoor environments may have more arbitrary-shape objects other than structural environments, a classification step has to be taken before using the vertical assumption. In the classification procedure, laser points cast on the vertical surfaces are extracted based on surface normal estimation. When the tilted-down LIDAR sweeps the environment, some vertical surfaces will be swept from bottom to up in consecutive laser scans. If we take a bird's eye view for this scanning process and project the vertical features on to a virtual horizontal plane, it is exactly the same as a robot with a horizontal LIDAR moving on a 2D surface. From a mathematical point of view, the vertical surface constrains how laser points at different height should match with each other. With the above intuition, the idea of synthetic LIDAR is proposed. A synthetic LIDAR is a planar 2D LIDAR on the projected virtual plane, where the end points of its laser beams are the projected points from vertical surface in the 3D environment. Figure 3 shows an example of the synthetic LIDAR.

Before looking at the technical details, let us understand the key idea of the synthetic LIDAR first. In the upper left of Fig. 3, 2D LIDAR readings are accumulated over 3D space. The accumulated 3D points are processed to extract interest points. Finally, the post-processed points are mapped into an occupancy grid map for localization and planning.

The idea of synthetic LIDAR helps to solve the 3D localization problem on a 2D plane. Although a vehicle is moving in the 3D world with 6-DOF, generally

---

<sup>1</sup>The push-broom configuration is exactly “tilted-down LIDAR” configuration. The LIDAR is the broom. As the vehicle moves, the LIDAR works in a push-broom way.



**Fig. 3** Construction of synthetic LIDAR

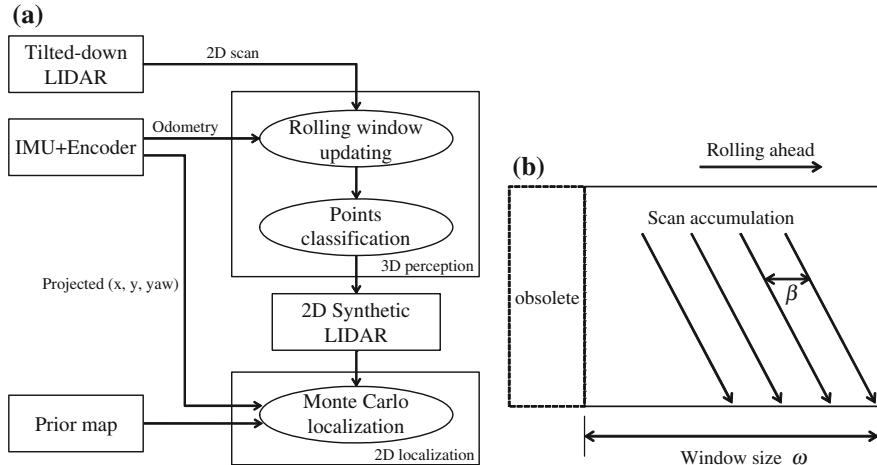
speaking, ground based vehicle is mostly interested in its 2D pose vector ( $x$ ,  $y$ , yaw). By projecting the 3D vertical features onto a virtual plane, 2D occupancy grid map can be used by marking those vertical features. This way, an a-priori map can be obtained using SLAM with the idea of synthetic LIDAR. It should be clarified that our algorithm only applies to an environment with only one vertical traversable level. For cases with more traversable levels, some other 2.5D or full 3D algorithms may be used, for example [22].

The localization system mainly consists of two parts, 3D perception to extract key feature points, and 2D localization to solve the localization on the horizontal plane. The synthetic LIDAR serves as a bridge to connect the 3D world and the 2D virtual plane, as shown in Fig. 4a. In this work, we use the map-aided localization method where the origin of the coordinate is set to the left-bottom corner of the map.

The system uses an IMU and a wheel encoder to provide 6-DOF odometry information, a 2D tilted-down LIDAR to provide laser scans, and an occupancy grid map serving as a prior for localization. A simple dead reckoning is used to obtain the odometry information. Assuming the distance measured by a wheel encoder at  $n$ -th time step is  $r_n$ , and the rotation is given by a pitch  $\theta$  and a yaw  $\Psi$ , the change in position of the vehicle is given by:

$$\begin{pmatrix} \Delta x_n \\ \Delta y_n \\ \Delta z_n \end{pmatrix} = \begin{pmatrix} \cos(\theta_n) \cos(\Psi_n) \\ \cos(\theta_n) \sin(\Psi_n) \\ -\sin(\theta_n) \end{pmatrix} \cdot (r_n - r_{n-1}). \quad (1)$$

The 3D perception assumes that odometry system is accurate enough in a short time period, and accumulates the laser scans for 3D range data. A classification procedure is then applied to extract interest points from the accumulated data. The extracted laser points are then projected onto a virtual horizontal plane (by ignoring their  $z$



**Fig. 4** Localization system overview. **a** Localization flow chart. **b** 3D rolling window

values), and a synthetic 2D LIDAR is constructed. The 2D localization fuses odometry information from odometry and measurements from the synthetic 2D LIDAR in a Monte Carlo Localization scheme. With a prior map of vertical features generated beforehand, localization on the 2D horizontal plane is achieved.

### 3.2 3D Perception

One of the requirements of the synthetic LIDAR is excellent adaptability that fits with different types of environment in urban scenario. We achieve this by properly extracting interest points from a reconstructed environment model, before the synthetic LIDAR is built.

To be able to recognize features that are perpendicular to the ground, an accurate model of the world is necessary. There are numerous ways that allow building an accurate environmental model, which includes nodding LIDAR and Velodyne [23]. As introduced in Sect. 3.1, a fixed, tilted down single planar LIDAR enables the reconstruction of the environment accurately by sweeping across the ground surface. This is an attractive solution since it is low cost and only requires rigid mounting of the sensor. It also allows optimization to be done that allows real-time computation of feature extraction that is unique to this configuration.

### 3.2.1 3D Rolling Window

The reconstruction uses rolling window sampling to maintain high probability of reflecting more recent samples by the ranging sensor. As such, a 3D rolling window is used to accumulate different scans recorded in a short distance. The size of the

window is flexible and the rolling window forms a local map of the 3D environment, i.e., it rolls together with the vehicle, where new incoming scans will be added into the window, and the old samples get discarded.

More specifically, given the window size  $w$ , the points  $p$  in  $n$ -th scan,  $P_n$  is accumulated according to

$$P_n = \bigcup_{k=n-w}^n \{p_k, \dots, p_n\} \quad n > w. \quad (2)$$

As shown in Fig. 4b,  $w$  is used to control the number of accumulated scans such that the size of the window would not grow unbounded. Also, a new scan is only inserted when a sufficient distance,  $\beta$  is achieved. This has two effects, a small  $\beta$  will have denser points but the overall window size will become shorter and vice versa. In our case, it was set to  $\beta = 0.02$  m and  $w = 100$  achieve the right compromise. The rolling window works in the odometry frame of the system, where each scan from a physical LIDAR is projected based on the odometry information derived from IMU and wheel encoder.

### 3.2.2 Point Classification

To extract features that are perpendicular to the ground, surface normal needs to be estimated. While many method exists [24], we used normal estimation proposed by [25]. It is based on first order 3D plane fitting, where the normal of each point in the space is approximated by performing least-square plane fitting to a points local neighborhood  $P^K$  [26]. The plane is represented by a point  $x$ , its normal vector  $\mathbf{n}$  and distance  $d_i$  from a point  $p_i \in P^K$ , where  $d_i$  is defined as

$$d_i = (p_i - x) \cdot \mathbf{n}. \quad (3)$$

By taking  $x = \bar{p} = \frac{1}{k} \sum_{i=1}^k p_i$  as the centroid of  $p_k$ , the values of  $\mathbf{n}$  can be computed in a least-square sense such that  $d_i = 0$ . The solution for  $\mathbf{n}$  is given by computing the eigenvalue and eigenvector of the following covariance matrix  $C \in \mathbb{R}^{3 \times 3}$  of  $P^K$  [27]:

$$C = \frac{1}{k} \sum_{i=1}^k (p_i - \bar{p}) \cdot (p_i - \bar{p})^T, C \cdot \mathbf{v}_j = \lambda_j \cdot \mathbf{v}_j, j \in \{0, 1, 2\}, \quad (4)$$

where  $k$  is the number of points in the local neighborhood,  $\bar{p}$  as the centroid of the neighbors,  $\lambda_j$  is the  $j$ th eigenvalue with  $\mathbf{v}_j$  as the  $j$ th eigenvector.

The principal components of  $P^K$  corresponds to the eigenvectors  $\mathbf{v}_j$ . Hence, the approximation of  $\mathbf{n}$  can be found from the smallest eigenvalue  $\lambda_0$ . Once the normal vector  $\mathbf{n}$  is found, the vertical points can then be obtained by simply taking the threshold of  $\mathbf{n}$  along the  $z$  axis, e.g. 0.5. This can vary depending on how noisy the sensor data is.

To find the local neighborhood points efficiently, KD-tree [28] is built from all the points obtained from the rolling window and perform a fixed radius search at each point. Although the surface normal can be calculated as a whole, performing normal calculation at each point in the rolling window can be very expensive. To further reduce the computation complexity, two successive rolling windows are maintained, where

$$P_{n+1}^\phi = P_n^\phi \cup \Phi(P_{n+1} \setminus P_n), \quad (5)$$

where  $\Phi$  can be any points classification function,  $P^\phi$  consists of the processed points and  $P$  contains the raw points. This way, surface normal calculation is only required for the much smaller rolling window  $P_{n+1} \setminus P_n$ . In other words, this ensures that classification will only perform on the newly accumulated point cloud and the processed points from the previous instance can be reused.

### 3.2.3 Synthetic LIDAR Construction

The result from the classified points consists of a collection of interest points in 3D. For the construction of synthetic LIDAR, the interest points in 3D is projected into virtual horizontal plane ( $z = 0$ ). It can be seen that this synthetic LIDAR has a very special feature: the ability to see-through the obstacles. This is possible due to the way that its laser scans are synthesized. During the 3D data accumulation, the collection of laser points is not performed at a single time stamp at the horizontal plane, but in a short time interval while the tilted-down LIDAR sweep the local environment. The objects at a distance (but within the sensor range) will not be blocked by the objects that is nearby but lower than the mounting height of the tilted LIDAR. Laser points cast on the faraway vertical surfaces will be also kept in the synthesis of the laser scan, contributing to the “see-through” characteristics.

This synthetic LIDAR is comprehensive in the sense that it is as if having multiple short ranged LIDARs arranged at different height at a forward facing configuration, with the height adapted to wherever there exist a vertical surface in the environment. The construction of synthetic LIDAR is completed by placing the virtual sensor at the base of the vehicle and performs transformation of all the interest points from odometry to the vehicle’s base. In many applications where a standard LIDAR is desired (equally spaced angle increment), the synthetic LIDAR can be further reconstructed to fulfill this constraint. This would involve performing ray tracing at each fixed angle increment to obtain minimum range value from the possible end points. The overall 3D perception can be summarized in Fig. 3. The 3D perception is done with the Point Cloud library [29] which provides many of the operations described in this section.

### 3.3 Online Localization

This paper adopts the Monte Carlo Localization (MCL) scheme in [30] to estimate the vehicle pose. MCL is a probabilistic localization method based on Bayes' Theorem and Monte Carlo idea [14]. The core of MCL is a particle filter, where the belief of vehicle position is maintained by a set of particles. MCL mainly consists of three steps, prediction, correction, and resampling. For the motion model which is required for the prediction step, Pseudo-3D odometry motion model from our previous work [11] is used. The choice of measurement model is discussed in the following.

#### 3.3.2 Virtual LIDAR Measurement Model

To incorporate the measurement into localization, a measurement model is needed for the synthetic LIDAR. The likelihood model is adopted for the synthetic LIDAR. Since the end points of virtual beams are the projection of interest points from vertical surfaces, it is possible that different points from different vertical surfaces may have the same angle. In other works, there exist two laser beams with the same angle while having two different range values. For this reason, synthetic 2D LIDAR is a peculiar LIDAR that only detects vertical surfaces, and can also see-through these surfaces. In light of this, the likelihood model which only requires the end points of laser beams is well suited for the synthetic LIDAR.

### 3.4 Experimental Results

In this experiment, SICK LMS-151 LIDAR is mounted in the upper-front and tilted-down for localization. A 4-layer LIDAR, SICK LD-MRS400001 is mounted at the waist level for obstacle detection. Both rear wheels of the golf cart are mounted with encoders that provide an estimate of the distance traveled. An IMU MicroStrain 3DM-GX3-25 is mounted at the center of the real axle to provide orientation information of the vehicle. The localization algorithm is tested in the Engineering Campus of National University of Singapore, where the road is up-and-down and many high buildings exist off the road.

A prior map is first generated with graph SLAM techniques by using the synthetic LIDAR as the input. To perform pose optimization, [31] is used as front end to detect loop closure. Then, the fully optimized pose is recovered using optimization library from [32]. To evaluate the quality of the recovered map built from synthetic LIDAR, the map is projected onto a satellite map, as shown in Fig. 5. The map shows consistency with good correlation with the satellite map, with an area of about 550 m × 487 m. Although there are discrepancies towards the left side of the map due to uniform longitudinal features along the road, the overall topology is maintained. This shows that the map can be used for accurate localization.



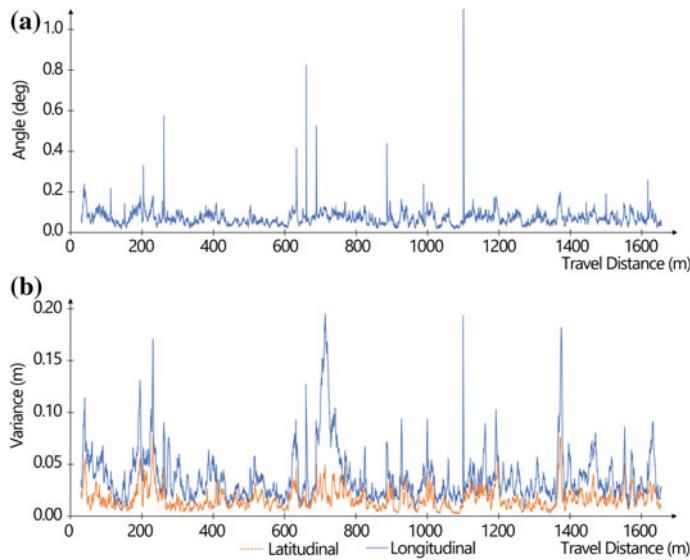
**Fig. 5** Mapping of the NUS engineering area

The synthetic LIDAR using SICK LMS-151 is able to perform at a rate of 50 Hz output on a laptop with Core i7 processor, showing that the synthetic LIDAR can be used to perform a real-time localization. The localization results are shown in Fig. 6. Judging from the prior map, the localization result from our algorithm always aligns with our driving path where a parallel line with the road boundary is clearly shown in the long stretch of road. Since our algorithm does not rely on GPS, our estimation still performs well near areas crowded by tall buildings. Note that in the experiment, a rough initial position is given and hence localization is mostly concerned with pose tracking. However, the system is able to cope with small kidnapping problems, e.g. brief data error from the LIDAR since the odometry system is still able to provide information. Should a large kidnapping occur, e.g. the vehicle was moved in between placed without turning on the localization module, a rough initial position may be provided to speed up the convergence rate.



**Fig. 6** Localization result of AMCL within NUS Engineering Campus

Figure 7 shows “localization variance” versus “driving distance”. The angle estimation variance is generally less than  $1^\circ$ , as shown in Fig. 7a. Figure 7b shows “position estimation variance” versus “driving distance” in longitudinal and lateral direction relative to the vehicle. It is shown that during the whole test, variance in both directions remain small. The worse variance occurs longitudinally, at a value about 0.2 m. This suggests the localization algorithm has high confidence about its pose estimates. At the same time, it is seen that the lateral variance is generally smaller than the longitudinal one. This is in-line with the fact that in an urban road environments, features in the lateral direction are much richer than those from the longitudinal one, as discussed in our previous work [11].



**Fig. 7** Localization variance results in terms of angle and position. **a** Angle variance. **b** Position variance

## 4 Cooperative Perception for Situational Awareness and Vehicle Control

In MoD systems, multiple autonomous vehicles are operating at the same time and equipped with radio devices to handle remote requests from users or operators. In the scenario where multiple vehicles can communicate with other vehicles or infrastructure, perception range can be far extended up to the boundary of connected vehicles by exchanging local perception information. The extended perception range can reach even beyond line-of-sight or field-of-view according to the network connectivity. This augmented perception capability can contribute to safety improvement and traffic flow efficiency [6, 33].

### 4.1 Cooperative Perception

*Cooperative perception* is defined as sharing local sensing information with others via wireless communications. The preceding vehicles highly affect the driving decision of the ego driver. In this sense, a leader is defined as a preceding vehicle (1) connected via cooperative perception and (2) observable by the ego vehicle<sup>2</sup> through its local

---

<sup>2</sup>Ego vehicle is a reference point of sensor fusion and a target of planning and control.



**Fig. 8** Concept of leader string where  $i$  is an ego vehicle.  $i + 1$  and  $i + 2$  are the first and second leader of  $i$ , respectively.  $i - 1$  is a following vehicle behind ego vehicle  $i$

sensors or remote information via cooperative perception. Let  $\mathcal{V}_i = \{i + 1, \dots\}$  be a *leader string* of a vehicle  $i$ , where,  $j, j + 1 \in \mathcal{V}_i$ ,  $j$  and  $j + 1$  are connected via cooperative perception and  $j + 1$  is observable by local sensors of  $j$ . The concept of leader string is depicted in Fig. 8.

In a broad sense, following vehicles of ego vehicle  $i$  can be included in the leader string  $\mathcal{V}_i$ , although a vehicle  $i - 1$  is not a leader literally. The information of a vehicle  $i - 1$  can be beneficial for blind spot detection or lane changing assistance, because the following vehicle  $i - 1$  can watch the ego vehicle in the third-person view from behind the ego vehicle.

In vehicle driving scenarios, sensing information is typically dealt with as a map. Let  $\mathcal{M} = \{\dots, m, \dots\}$  be a map for navigation of vehicles, which consists of the set of points obtained and filtered from sensors, where  $m \in \mathbb{R}^n$ ,  $n = 2$ , or 3. Given a position  $p \in \mathcal{M}$  in a map,  $\mathcal{M}[p] \rightarrow \mathbb{R}$  can be defined in several ways such as the height of obstacle in case of  $p \in \mathbb{R}^2$ , or the belief that the position  $p$  is obstacle-free.  $\mathcal{M}_i$  denotes a map of a vehicle  $i$ .

Now, we can formulate cooperative perception as follows:

$$\mathcal{M}_i = \mathcal{M}_i \bigcup_{j \in \mathcal{V}_i} \mathcal{M}_j, \quad (6)$$

where the operation  $\bigcup$  is called *map merging*. The map merging operation is merely a set union operation, if  $\mathcal{M}_i$  and  $\mathcal{M}_{j \in \mathcal{V}_i}$  are mapped into a global coordinate frame such as GPS coordinates. However, the observation from sensors is typically mapped into a local coordinate frame. Also, there is no guarantee that the initial poses of vehicles are known. In this case, the relative pose between vehicles is necessary to merge different spatial information. The relative pose can be defined as  $q = (\tau, \theta)$ , where  $\tau$  and  $\theta$  correspond to translation and rotation, respectively. We define a transformation operator as  $p \otimes q = R(\theta)p + \tau$ , where  $R(\theta)$  is a rotation matrix. Finally, (6) can be rewritten in a more general form as follows:

$$\mathcal{M}_i = \mathcal{M}_i \bigcup_{j \in \mathcal{V}_i} \mathcal{M}_j \otimes q_{i,j}, \quad (7)$$

where  $q_{i,j}$  is a relative pose between a vehicle  $i$  and  $j$ . One of the key challenges to solve (7) is to obtain an accurate  $q_{i,j}$ .

## 4.2 Map Merging Problem

The primary problem of map merging can be formulated as follows:

$$q_{i,j}^* = \arg \max_{q_{i,j}} S(\mathcal{M}_i, \mathcal{M}_j, q_{i,j}), \quad (8)$$

where the similarity measure  $S$  is defined as

$$\sum_p \mathcal{L}(\mathcal{M}_i[p], (\mathcal{M}_j \otimes q_{i,j})[p]), \quad (9)$$

where  $\mathcal{L}(a_i, a_j)$  is a point-to-point similarity measure, which is positive if  $a_i = a_j$ ; 0, otherwise. Equations (8) and (9) attempt to find the relative pose that maximizes the overlapping area between two maps.

Equations (8) and (9) can be extended to more than two vehicles. Let  $Q_i = \{q_{i,i+1}, \dots, q_{i,i+N}\}$  be the set of relative poses w.r.t the ego vehicle, where  $N$  is the number of leader vehicles. The problem can be rewritten as follows:

$$Q_i^* = \arg \max_{Q_i} S(\mathcal{M}_i, \dots, \mathcal{M}_{i+N}, Q_i), \quad (10)$$

where the similarity measure  $S$  is defined as

$$\sum_p \mathcal{L}(\mathcal{M}_i[p], \dots, (\mathcal{M}_j \otimes q_{i,j})[p], \dots), \quad (11)$$

where  $\mathcal{L}(a_i, \dots, a_{i+N})$  is positive if  $a_i = \dots = a_{i+N}$ ; 0, otherwise. There are various ways to implement the similarity measure. In this work, we use Iterative Closet Point (ICP) [34, 35] and Correlative Scan Matching (CSM) [36] to quantify and implement the measure.

Algorithm 1 is an essential algorithm to solve Eqs. (8)–(11). The first step LeaderVehicleDetection responds for vehicle detection and estimating the leader vehicle's pose using on-board LIDAR and return the initial relative pose  $q_{i,j}^0$ . The detail leader vehicle detection method is described as follows.

---

**Algorithm 1:** On-Road Map Merging Algorithm

---

```

input :  $\mathcal{M}_i, \mathcal{M}_j$ 
output: Merged map
begin
     $q_{i,j}^0 \leftarrow \text{LeaderVehicleDetection}(\mathcal{M}_i)$ 
     $q_{i,j}^* \leftarrow \text{ScanMatching}(\mathcal{M}_i, \mathcal{M}_j, q_{i,j}^0)$ 
    return  $\mathcal{M}_i \cup (\mathcal{M}_j \otimes q_{i,j}^*)$ 
end

```

---

One of the distinct characteristics of scan data on the road is that features are not obvious to match due to bushes, trees, or pedestrians. Nonetheless, scan data of a leader has relatively clear features such as the backside or side of vehicles. The initial pose of a leader can be guessed from the features. More specifically, the proposed system keeps checking whether there are consecutive points lying in the desired path ahead. For this task, let  $\mathcal{D}_i$  be the set of positions that a vehicle  $i$  will/can move to, typically the set of the center positions of the current lane. To find the leader position, the proposed system keeps watching straight or slightly-curved lines in the following point set:

$$\{p | r_{th} > \|p - p_l\|, p_l \in \mathcal{D}_i\}, \quad (12)$$

where  $r_{th}$  is the maximum boundary that a vehicle can go away from the desired path, typically half of the lane width. Equation (12) can significantly reduce the search space and false positives of leader vehicle detection.

If (1) the detected points compose a straight or slightly curved line, and (2) the line is almost orthogonal to the desired path, the center position of the line and the orthogonal angle to the line are used as  $t_0$  and  $\theta_0$ , respectively, which is used as the initial transformation  $q_{i,j}^0 = (t_0, \theta_0)$ .

Based on the estimated relative pose  $q_{i,j}^0$ , ICP and CSM are employed for ScanMatching to match the scans within the overlap area to maximize  $S(M_i, M_j, q_{i,j})$  and refined the relative pose accordingly. Finally, the map is merged according to (7). More detail information is available in [6].

### 4.3 Sensor Multi-Modality

Given the relative poses between different vehicles, it is easy to perform map merging for sensory information such as a laser scanner whose readings are a set of physical quantities recorded with their spatial coordinates. However, for the sensory modality of vision whose reading is an image, map merging is not a straightforward task, because the vision image is the result of perspective projection unlike laser scan data.

To deal with the problem, we use the Inverse Perspective Mapping (IPM) [37] method that can map these projected pixels to their spatial coordinates on the road surface. For a fixed-mount on-board vision camera, its pose in the vehicle coordinate and the pose relative to the road surface are usually known. Together with its intrinsic parameters from a calibration process, the perspective transformation matrix from road surface to vision camera image can be calculated. The inverse operation of this perspective matrix will restore the original road surface from the camera image. Each pixel of the image generates one color point on the ground. By applying IPM operation, camera sensing information is transformed from the projected image pixels into the spatial coordinate, with which map merging can be performed.

In the merged map of ego vehicle, three types of information represented: (1) vehicle poses, including poses of ego vehicle and other vehicles; (2) laser scan points; and (3) color points of road surface from vision. By supporting multiple sensory modalities, the cooperation perception system helps ego vehicle to perceive not only occluded vehicles and obstacles, but also road surface that may be out of its sensing ability.

Figure 10a shows an example of the map merging result fused with information from range sensor and vision sensor on the road.

#### 4.4 Experimental Results

In the experiments, we used four vehicles consisting of one Mitsubishi iMiEV and three Yamaha golf cars, in which one vehicle participated as a moving obstacle. Among four vehicles, three vehicle were equipped with the proposed cooperative perception system. The setup of three vehicles is represented in Fig. 8, where  $i$ ,  $i + 1$  and  $i + 2$  are corresponding to the ego vehicle, the first leader and the second leader, respectively. In Fig. 8, the second leader transmits its sensing information to the first leader via wireless communications, while moving forward. With the support of our system, the first leader merges the remote information with its local sensing information, and then transmits the combined information to the ego vehicle via wireless communications, while moving forward as well. We conducted experiments using the vehicles on a campus road at NUS on sunny or cloudy days between 12 and 5 PM. The rule of the road is to drive on the left and the speed limit is 40 km/h.

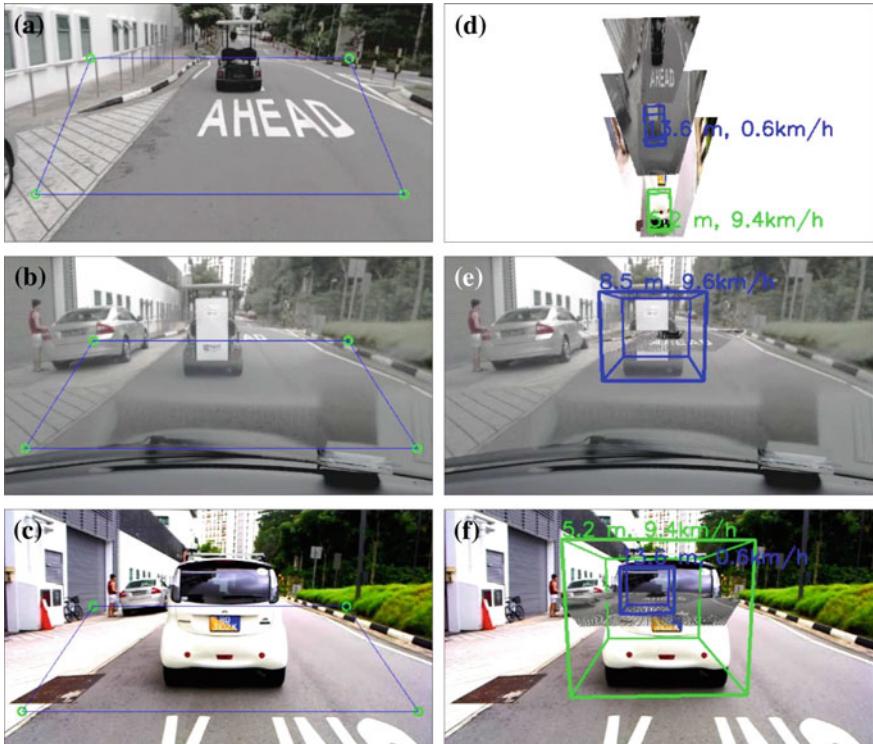
Table 1 shows the performance of map merging algorithm according to the scan matching methods and leader detection method. The LDR is obtained from LIDAR-based LeaderVehicleDetection. It is used as the initial condition for both the ICP and CSM scan matching algorithm. In this work, the ICP algorithm from [38] is used. The comparison is done using a ground truth generated by particle-filter-based localization using LIDAR. In the ICP,  $d_{th} = 2.5$  m is used. For CSM, two levels of 0.5 and 0.1 m search grids are used. It is also set to search within a window of  $10\text{ m} \times 10\text{ m} \times 60^\circ$ , and all computations are done with a desktop computer equipped with Core i7 CPU and a Nvidia GTX770 graphics card. Note that among the obtained

**Table 1** Performance of map merging in terms of translation and rotation error

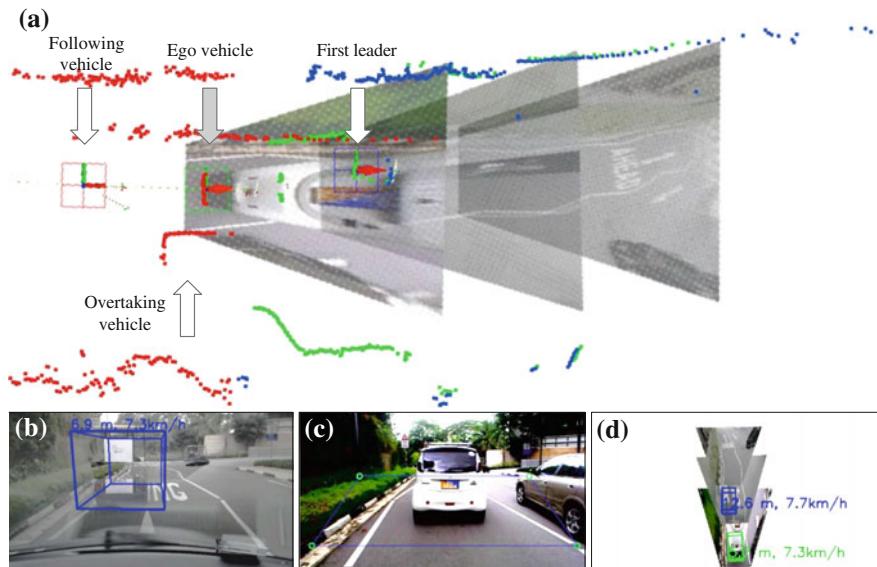
Average		LDR	ICP	CSM
Translation error (m)	First leader	0.73	0.54	0.44
	Second leader	1.81	1.76	1.11
Orientation error (degree)	First leader	14.44	12.14	2.77
	Second leader	17.65	13.15	3.51
Computation time (ms)		6	6	108

$q = (\tau, \theta)$ , the translation  $\tau$  can be measured more accurately by range sensors. This accuracy asymmetry between position and orientation results in lever-arm effects. In Table 1, it was found that there is significant improvement of the merged map from the second leader. While CSM performed slower than ICP, it is able to robustly estimate the relative pose since CSM is robust to initialization error. We conclude that the results using the both scan matching approaches perform sufficient for our requirements.

Figures 9 and 10 show the snapshots of cooperative perception visualization. In Fig. 9a–c show the camera views of the ego vehicle, the first leader and the second leader, respectively. (e) and (f) shows see-through views of the first and second leader, respectively. (d) shows the third person view of the ego vehicle, where driver's seat is lifted at 30 m from the ground. Figure 9 shows the satellite view of the ego vehicle at the same moment of Fig. 9. Note that the ego vehicle cannot see oncoming traffic



**Fig. 9** Map merging results fused with information from range sensor and vision sensor, where the cuboids represent a simplified vehicle model. **a** Second leader's camera view. **b** First leader's camera view. **c** Ego vehicle's camera view. **d** Ego vehicle's lifted seat view (30 m). **e** First leader's see-through view. **f** Ego vehicle's see-through view



**Fig. 10** All-around view supported by cooperative perception. **a** Ego vehicle’s satellite view. **b** Ego vehicle’s see-through view. **c** Following vehicle’s camera view. **d** Ego vehicle’s third person view

situation ahead due to the first leader and the limitation of line-of-sight. However, the ego vehicle equipped with the cooperative perception system can see the second leader and the preceding vehicle of the second leader, and traffic sign on road surface, e.g., “AHEAD”.

Figure 10a–d show how all-around view is supported by a following vehicle connected via cooperative perception. Figure 10a is the satellite view of the ego vehicle, where the grey arrow indicates the ego vehicle. Note that there is another vehicle behind the ego vehicle. The ego vehicle is moving between the first leader and the following. In Fig. 10a, the ego vehicle can detect a vehicle approaching behind, i.e., the overtaking vehicle, because the following vehicle connected via cooperative perception can tell the ego vehicle about what is going on at blind spot of the ego vehicle, which can greatly improve situational awareness.

Figure 10b is a see-through view of the ego vehicle. Figure 10c is a view of the following vehicle  $i - 1$ , which can be seen by the ego vehicle. (d) is the third person view of the ego vehicle, where driver’s seat is lifted at 30 m from the ground. All-around view characteristics of Fig. 10 can be applied to the lane changing and overtaking assistance. In addition, a normal vision camera can sense the adjacent left and right lane, as we can see Fig. 10a and d. A fisheye lens can be considered for multiple lane detection. In this work, we used a processed image and laser scan as a message profile, whose average size is 6.5 K. The average delay is less than 100 ms on IEEE 802.11g, IEE 802.11n, 4G LTE.

## 5 Automated Early Collision Avoidance Using Cooperative Perception

If the system can let a driver know the moment when the driver has to drive carefully or watch the display through proper visual, audio, or tactile feedback, a driver can focus on driving itself without having to keep watching a driving assistance display. The forward collision warning can trigger path-replanning earlier for early hidden obstacle avoidance. In this context, one of the most desired functions is forward collision warning.

### 5.1 See-Through Forward Collision Warning

Many forward collision warning algorithms have been proposed along with risk assessment method, which includes time-based, and distance-based approaches. Firstly, the time-based methods usually use Time-To-Collision (TTC), which can be formulated as

$$TTC_{i,j} = \frac{g_{i,j}}{v_i - v_j}, \quad j \in \mathcal{V}_i, \quad (13)$$

where  $TTC_{i,j}$  is corresponding to TTC,  $g_{i,j}$  is the distance between a vehicle  $i$  and  $j$ , and  $v_i$  is the speed of a vehicle  $i$ . A forward collision warning is activated if  $TTC_{i,j}$  is less than a certain threshold time, typically 2–3 s according to safety requirements [39].

In distance-based methods, a forward collision warning is activated if, given two vehicles  $i$  and  $j$ , the recommended safety gap  $r_{i,j}$  is larger than the distance between two vehicles  $g_{i,j}$ , which can be restated as follows: A forward collision warning is activated if

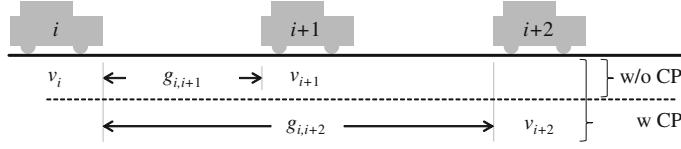
$$g_{i,j} < r_{i,j}, \quad j \in \mathcal{V}_i, \quad (14)$$

where  $r_{i,j}$  is a minimum distance to avoid collision to a preceding vehicle  $j$ , which can be formulated as follows [40]:

$$r_{i,j} = (v_i - v_j)/2\gamma^2 + (v_i - v_j) * T_{rs}, \quad j \in \mathcal{V}_i, \quad (15)$$

where  $\gamma$  is the deceleration of the ego vehicle, e.g.,  $-0.2\text{ g}$  ( $\approx -2\text{ m/s}$ ).  $T_{rs}$  is the response time of a driver, e.g., 0.5–1.5 s.

Note that one of the great advantages of our system is that  $j$  is not limited to only  $i + 1$ . In our system, thanks to the capability of a see-through view,  $j$  can be beyond the first leader according to the connectivity via cooperative perception, as shown in Fig. 11. This see-through characteristic enables the driver to avoid hidden obstacles earlier than without cooperative perception.



**Fig. 11** Comparison of collision warning with and without cooperative perception

## 5.2 All-Around View Using Cooperative Perception

Not surprisingly, cooperative perception enables all-around view without having to install sensors covering all sides of a vehicle. In particular, a following vehicle can see the ego vehicle in a third-person view, including its blind spots. Figure 10a shows one snapshot of the all-around view of the ego vehicle on the road. Furthermore, our system can detect a vehicle approaching toward the ego vehicle using a spatio-temporal moving obstacle detection and tracking method. From the following, we investigate the benefit of all-around view in terms of overtaking assistance and lane changing assistance.

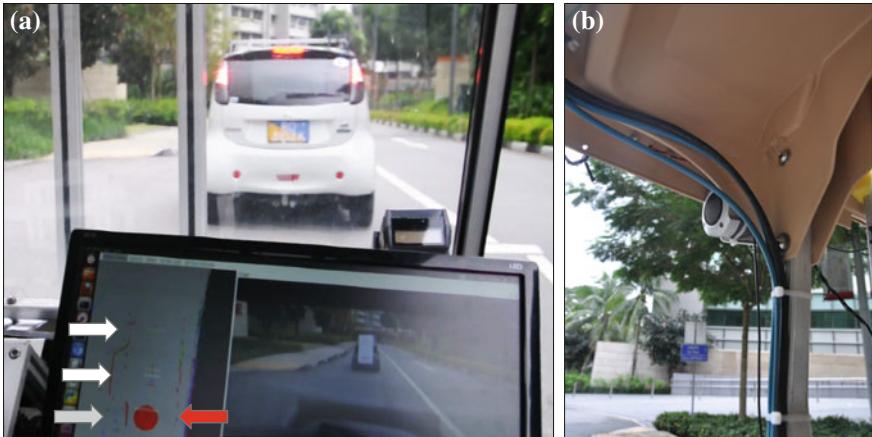
## 5.3 Overtaking and Lane Changing Assistance

When a driver should drive slowly or stop due to a slow-moving truck or obstacle ahead, the human or autonomous driver should decide whether to wait longer in the current lane or change lanes, which is an important issue for overtaking on a single-lane road.

Many overtaking assistance methods have been proposed according to the requirements, sensing capability and sensor configurations. In principle, the overtaking decision should be determined with the consideration of (1) the number of lanes, (2) the speed of a preceding vehicle, (3) cut-in space availability, (4) distance from an oncoming vehicle, and (5) the existence of another overtaking vehicle from behind as in Fig. 10d. Our system can inform (3), (4), (5) that are difficult to be provided without cooperative perception. Equations (14) and (15) can be applied to read-end collision warning at lane changing, which are corresponding to  $TTC_{j,i}$  and  $r_{j,i}$ , respectively, where the vehicle  $j$  is approaching toward the ego vehicle behind in an adjacent lane, and  $j < i$ .

## 5.4 Feedback to a Driver

Once a forward collision is expected, or overtaking is not possible, or lane changing is not possible, it should be notified to a driver through visual, audio, or tactile feedback. In Fig. 12, some examples of visual warnings are illustrated for a forward



**Fig. 12** See through forward collision warning is provided both as a visual warning (seen in **a**) as well as audio warning on speakers (seen in **b**)

collision warning, overtaking possibility, and lane-changing possibility, respectively. In the next subsection, we evaluate how our system can contribute to the see-through forward collision warning, overtaking assistance, and lane changing assistance are, through real experiments on the road using vehicles equipped with our system. Furthermore, we will address in the next experimental results how the feedback can be used for autonomous driver from the perspective of planning and control.

## 5.5 Experimental Results

We performed (1) see-through forward collision warning tests with real human drivers using real vehicles equipped with the driving assistance system, and (2) automated early lane change experiments using our autonomous vehicles.

### 5.5.1 See-Through Forward Collision Warning

The test scenario is as follows. Three vehicles move forward on single-lane road like Fig. 1, where a test driver drives the ego vehicle. Then, the second leader suddenly stops at a certain position. To avoid collision, the first leader and the ego vehicle stop accordingly. Real tests were conducted with three human drivers on the road eight times.

In this experiment, the second leader was moving forward at 2–5 m/s before sudden braking. Accordingly, TTC was set to 15 s. These are somewhat conservative parameters primarily for safety concerns. Note that the brake lights of the second leader were turned off. It makes the first leader difficult to notice whether the second

leader decelerates or not. The warning sound includes two loud beeps that last for 2 s per one warning activation.

We used IEEE 802.11n as a radio interface of vehicle-to-vehicle communications. Laser scan and compressed vision image were used as a message profile. The average communication delay between the first leader and the ego vehicle is 16 ms by measurement. IEEE 802.11n performed well at more than 30 m distance on the road.

Figure 12a shows one camera shots captured at the front passenger seat. The left-bottom screen of Fig. 12a shows the satellite view, where the three left-most arrows indicate the ego vehicle, the first leader, and the second leader as like Fig. 9a. The red circle indicated by vertical red arrow represents a high collision probability, because the vehicle the first leader is suddenly stopped in front of the first leader.

In principle, the test driver cannot see the second leader due to the limitation of line-of-sight. However, the test driver can see the second leader through the cooperative perception system. The right bottom screen of Fig. 12a is the see-through view. The color of the red dot becomes green when no collision is expected. The collision warning sign turned red along with a loud sound alarm using a speaker in Fig. 12b. The first leader also stopped to avoid collision.

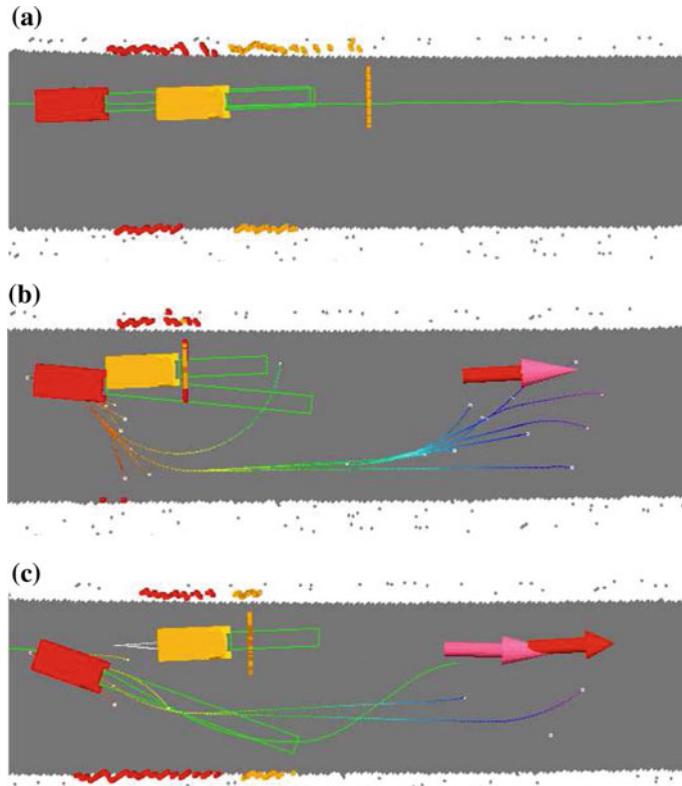
Average forward collision warning activating time is 2.1 s according to the preset TTC and the distance from the second leader. The test drivers pushed the brake pedal averagely 0.82 s later after a forward collision warning sound is activating, which is a time from processing the sound signal via making a decision in a brain to actuating foot muscles. The delay is widely ranged from 0.7 s to 1 s according to the drivers. In this experiment, the test drivers detected the sudden braking of the second leader earlier than the first leader and accordingly stopped earlier as much as 0.33 s. With the support of cooperative perception, the test driver has more time to cope with the traffic situation.

### 5.5.2 Early Automated Lane Changing

The see-through collision warning can be used for triggering path-replanning for collision avoidance. Figure 13 shows simulations of the scenario. These simulations utilized a RRT\* path planner [41].

Once a collision warning is triggered, the motion planner is called to search a feasible trajectory for obstacle avoidance. Thanks to cooperative perception, the planning space can be extended to as far as the neighboring vehicles connected via wireless communications can perceive; this allows motion planning to be conducted in a long-term perspective. To alleviate the time-constraint issue of long-term planning, the anytime RRT\* [42] is used, where a feasible trajectory can be quickly found and the solution path will then be asymptotically optimized.

Additionally, while cooperative perception could provide an enlarged planning space, the inherent sensing uncertainty and transmission delay need to be carefully considered. A cost map based approach is proposed in our previous work in [13] to handle this issue. Specifically, the sensing uncertainty and transmission delay are factored into a cooperative-perception-based cost map, then an anytime RRT\* algorithm is employed to search for an optimal trajectory subject to this cost map.



**Fig. 13** Early lane changing with the support of a see-through collision warning using cooperative perception. The leader is represented by an *orange rectangle*, and the ego vehicle is represented by a *red rectangle*. Both vehicles are traveling with a lane from *left to right*, with the lane blocked by an obstruction shown as a *vertical line*. Feasible vehicle trajectories are shown by *multicolored curved lines*. **a** shows the point of sudden obstacle appearance, **b** shows path plan of ego vehicle without cooperative perception, and **c** shows planning with the see-through forward collision warning

In this manner, the solution trajectory will be located in the regions with lowest collision risk.

In Fig. 13a shows a situation without cooperative perception, during which a sudden obstacle appears in front of the leader; only the leader detects the obstacle whereas the following ego vehicle has a limited line-of-sight range only up to the rear of the leader. In Fig. 13b, the both of the leader and the ego vehicle suddenly stop, then the ego vehicle performs re-path planning to overtake the stopped leader. However, it is hard to find the feasible path to overtake the leader in the case of Fig. 13b, because the ego vehicle and the leader are too close after the sudden stop. In Fig. 13c, early lane changing is triggered with the support of the see-through collision warning. Since sufficient space is occupied to overtake the leader, the path planner can find the overtaking path promptly.

## 6 Results and Impact

Since the end of 2011, our main testing grounds have been within the campus grounds of the National University of Singapore. There are two main testing sites: One on a road involving vehicular and pedestrian movement, and another on a non-road area involving heavy pedestrian movement.

Our work in 2013 mainly focused on the second site involving heavy pedestrian movement. Pedestrian interaction is usually with students who are busy rushing off to their next lectures. Students' movements are tied to their lecture schedules. It is common to see groups of students appear around the same time, rushing off to the bus-stop to catch the approaching bus for their next lecture. This presents a nice testing grounds for our vehicle to not only prove the reliability of the vehicle's implemented dynamic safety zone but as well as to further improve the vehicle's mobility within a densely crowded pedestrian environment.

### 6.1 An Invitation to a Live Showcase

The Singapore Science Centre held its first ever Street Fair from 8–11 November 2013. This was in conjunction with the celebrations of its 35th Anniversary in the ongoing endeavor to promote science and technology to the public. The SMART Autonomous Vehicles Group was invited to exhibit the driverless buggy at this Street Fair. We were also given the opportunity to showcase the vehicle in operation amongst the pedestrian crowd. And the most fun part of this was giving out free rides. Figure. 14, captures one such ride.

This was an exciting milestone to our research work. This was the first time the vehicle was operating outside of university campus. This was a very important validation to our work as public interaction with our vehicle was involved—not just with the vehicle motion but as passengers as well.

While roboticists in the community of autonomous vehicles are more comfortable with how such driverless vehicles work, the same cannot be said with the general public. One of the purposes for this live showcase was to raise public awareness of the maturity of driverless technology and that driverless vehicles can be very safe. This was achieved by allowing the public to freely interact with the vehicle. This provided a good opportunity for everyone to get comfortable with a moving vehicle without any human driver. People would learn from first-hand experience that the vehicle will slow down when someone comes near to the vehicle and that the vehicle would reliably come to a halt when anything comes too near it. Free rides were given as well. This allowed the public to have a first-hand experience in sitting in a self-driving vehicle that would bring them to their destination reliably, safely and in a comfortable manner.



**Fig. 14** Live showcase at the Singapore Science Center Street Fair

## 6.2 Humans Are Unpredictable

Most path planners make a basic assumption, i.e., pedestrian movement will largely remain unchanged if there is no sudden perturbation to the overall state of the environment. This is a very safe assumption as people generally tend to have a certain destination they want to walk to and this destination largely remains unchanged. This predicated behavior, however, was not entirely observed at the Science Centre. People behaved differently with the vehicle upon learning that it was driving all by itself. On the one hand, there were those who were unsure if the vehicle would knock them down as there was no human controller. And on the other hand, there were those who were jumping directly in front of the vehicle, from all possible angles, trying to find a scenario where the autonomous vehicle would fail. Thankfully, the implemented dynamic safety zone in the vehicle worked as intended.

It was an interesting phenomenon that people behaved very differently upon knowing that the vehicle is self-driven. Most path planners would assume that pedestrians would take a more conservative approach in their movement in the presence of a vehicle. However, what we observed was a number of pedestrians acting in a more “aggressive” manner—their motion would abruptly change and intersect with the vehicle’s path. It was also observed that this unpredictability in human behavior usually happened to people who have never seen a driverless vehicle before.

## 6.3 Safety Proven

Allowing the public to interact with the vehicle both as a passenger and as a pedestrian helped to reinforce the reliability of the system. Using laser sensors with a range of

50 m, the vehicle is able to have an early detection of obstacles. In a dynamic safety zone, the speed of the vehicle is monitored in relation to the presence of near-by pedestrians. This allows the vehicle to stop in a gradual manner in the presence of an approaching obstacle. The dynamic safety zone also allows the vehicle to automatically adapt to the speed of the moving crowd, allowing the vehicle to still move as opposed to being completely stationary. The dynamic safety zone also allows passengers in the vehicle to not experience “vehicle jerkiness” in a crowded environment.

## ***6.4 The Next Step***

The main thrust of this research work is to use driverless vehicles for Mobility on Demand. One of the key elements to make this a reality is to bridge the gap of between public acceptance of driverless vehicles and the state-of-the-art enabling technologies for driverless vehicles. The experience at the Singapore Science Centre definitely aided to not only raise public awareness of driverless vehicles, but also public acceptance of it through experiential interaction with the driverless buggy. It is important to continually engage the public with regards to the development of autonomous vehicles. After all, the public will eventually be the end-users of such a system.

Ongoing research work aims at enabling the driverless vehicle with greater mobility within a dense pedestrian environment while maintaining an overall satisfactory user experience.

## **7 Conclusion**

In this chapter, we introduced vehicle autonomy using cooperative perception to enable Mobility-on-Demand system. Our MoD system currently operates in a restricted section of the National University of Singapore campus, where autonomous personal transporters have been autonomously servicing numerous requests from visitors and dignitaries during the course of multiple demonstrations. The rich sensing and actuating capabilities of autonomous vehicles can improve traffic flow and safety of MoD services. We proposed and demonstrated driving assistance system and autonomous driving system using cooperative perception, which include a see-through/lifted-seat/satellite view, a see-through forward collision warning, and overtaking/lane changing assistance. All these functions and systems are in operation with a number of autonomous vehicles in the NUS campus road. We have extended the test road to the outside of the campus.

## References

1. Mitchell WJ, Borroni-Bird CE, Burns LD (2010) Reinventing the automobile: personal urban mobility for the 21st century. MIT Press, Cambridge
2. Car2Go. [Online]. Available: <http://www.car2go.com/>
3. SMOVE. [Online]. Available: <http://smove.sg/>
4. Shaheen SA, Guzman S, Zhang H (2010) Bikesharing in Europe, the Americas, and Asia. *Transp Res Res J Transp Res Board* 2143(1):159–167
5. Bouraoui L, Boussard C, Charlot F, Holguin C, Nashashibi F, Parent M, Resende P (2011) An on-demand personal automated transport system: the citymobil demonstration in La Rochelle. In: IEEE Intelligent vehicles symposium (IV), pp 1086–1091
6. Kim S-W, Qin B, Chong ZJ, Shen X, Liu W, Ang MH Jr, Frazzoli E, Rus D (2014) Multivehicle cooperative driving using cooperative perception: design and experimental validation. In: IEEE Transaction on Intelligent Transportation Systems
7. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) Ros: an open-source robot operating system. In: ICRA workshop on open source software, vol 3. no 2
8. Chong ZJ, Qin B, Bandyopadhyay T, Wongpiromsarn T, Rebsamen B, Dai P, Kim S-W, Ang MH Jr, Hsu D, Rus D, Frazzoli E (2012) Autonomy for mobility on demand. In: IEEE/RSJ international conference Intelligent Robots and systems, Oct 2012
9. Kim S-W, Chong ZJ, Qin B, Shen X, Cheng Z, Liu W, Ang MH Jr (2013) Cooperative perception for autonomous vehicle control on the road: motivation and experimental results. In: IEEE/RSJ international conference intelligent Robots and systems, Nov 2013
10. Bandyopadhyay T, Jie CZ, Hsu D, Ang MH Jr, Rus D, Frazzoli E (2012) Intention-aware pedestrian avoidance. In: International symposium on experimental robotics, June 2012
11. Qin B, Chong Z, Bandyopadhyay T, Ang MH, Frazzoli E, Rus D (2012) Curb-intersection feature based Monte Carlo localization on urban roads. In: Robotics and automation (ICRA), 2012 IEEE international conference on, pp 2640–2646
12. Qin B, Lie W, Shen X, Chong ZJ, Bandyopadhyay T, Ang MH Jr, Frazzoli E, Rus D (2013) A general framework for road marking detection and analysis. In: 16th international IEEE conference on intelligent transportation systems (ITSC)
13. Liu W, Kim S-W, Chong ZJ, Shen X, Ang MH Jr (2013) Motion planning using cooperative perception on urban road. In: IEEE international conferences on Cybernetics and intelligent systems, and robotics, automation and mechatronics
14. Thrun S, Burgard W, Fox D (2005) Probabilistic robotics. MIT Press, Cambridge
15. Thrun S (2000) A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In: Proceedings of the robotics and automation, 2000, ICRA'00, IEEE international conference on, vol 1. pp 321–328
16. Baldwin I, Newman P (2012) Road vehicle localization with 2d push-broom lidar and 3d priors. In: Robotics and automation (ICRA), 2012 IEEE international conference on, pp 2611–2617
17. Nüchter A, Lingemann K, Hertzberg J, Surmann H (2007) 6D SLAM—3D mapping outdoor environments. *J Field Robot* 24(8–9):699–722
18. Harrison A, Newman P (2008) High quality 3d laser ranging under general vehicle motion. In: Robotics and automation, (2008) ICRA 2008, IEEE international conference on, pp 7–12
19. Kohlbrecher S, von Stryk O, Meyer J, Klingauf U (2011) A flexible and scalable slam system with full 3d motion estimation. In: IEEE international symposium of safety, security, and rescue robotics (SSRR), (2011) on, pp 155–160
20. Weingarten J, Gruener G, Siegwart R (2003) A fast and robust 3d feature extraction algorithm for structured environment reconstruction. In: International conference on advanced robotics
21. Rezaei S, Guivant J, Nieto J, Nebot EM (2004) Simultaneous information and global motion analysis (SIGMA) for car-like robots. In: Proceedings of IEEE international conference of the robotics and automation, 2004, ICRA'04. 2004, vol 2. pp 1939–1944
22. Kümmerle R, Triebel R, Pfaff P, Burgard W (2008) Monte Carlo localization in outdoor terrains using multilevel surface maps. *J Field Robot* 25(6–7):346–359

23. Velodyne Lidar. [Online]. Available: <http://velodynelidar.com/>
24. Klasing K, Althoff D, Wollherr D, Buss M (2009) Comparison of surface normal estimation methods for range sensing applications. In: IEEE international conference on Robotics and automation, (2009) ICRA'09, pp 3206–3211
25. Berkmann J, Caelli T (1994) Computation of surface geometry and segmentation using covariance techniques. *Pattern Anal Mach Intell IEEE Trans* 16(11):1114–1116
26. Shakarji CM et al (1998) Least-squares fitting algorithms of the nist algorithm testing system. *J Res-Natl Inst Stand Technol* 103:633–641
27. Rusu RB (2010) Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intell* 24(4):345–348
28. Muja M, Lowe DG (2009) Fast approximate nearest neighbors with automatic algorithm configuration. In: VISAPP (1), pp 331–340
29. Rusu RB, Cousins S (2011) 3d is here: point cloud library (pcl). In: IEEE international conference on Robotics and automation (ICRA), 2011, pp 1–4
30. ROS AMCL. [Online]. Available: <http://www.ros.org/wiki/amcl>
31. Tipaldi GD, Arras KO (2010) Flirt-interest regions for 2d range data. In: IEEE international conference on Robotics and automation (ICRA), 2010, pp 3616–3622
32. Kaess M, Ranganathan A, Dellaert F (2008) Isam: incremental smoothing and mapping. *Robot IEEE Trans* 24(6):1365–1378
33. Kim S-W, Gwon G-P, Choi S-T, Kang S-N, Shin M-O, Yoo I-S, Lee E-D, Seo S-W (2012) Multiple vehicle driving control for traffic flow efficiency. In: IEEE intelligent vehicles symposium, pp 462–468, June 2012
34. Besl PJ, McKay ND (1992) Method for registration of 3-d shapes. In: Robotics-DL tentative, international society for optics and photonics, pp 586–606
35. Yang C, Medioni G (1992) Object modelling by registration of multiple range images. *Image vis comput* 10(3):145–155
36. Olson EB (2009) Real-time correlative scan matching. In: IEEE international conference on Robotics and automation, ICRA'09, pp 4387–4393
37. Mallot HA, Blthoff HH, Little J, Bohrer S (1991) Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biol cybern* 64(3):177–185
38. Claraco JLB (2008) Development of scientific applications with the mobile robot programming toolkit., Machine Perception and Intelligent Robotics Laboratory, University of Málaga. MRPT reference book, Málaga
39. Dagan E, Mano O, Stein GP, Shashua A (2004) Forward collision warning with a single camera. In: Intelligent vehicles symposium, 2004 IEEE, pp 37–42
40. Parasuraman R, Hancock P, Olofinboba O (1997) Alarm effectiveness in driver-centred collision-warning systems. *Ergonomics* 40(3):390–399
41. Karaman S, Walter MR, Perez A, Frazzoli E, Teller S (2011) Anytime motion planning using the RRT\*. In: IEEE international conference on Robotics and automation, May 2011
42. Karaman S (2011) Anytime motion planning using the RRT\*. In: IEEE international conference on Robotics and automation (ICRA), pp 1478–1483

# Motion Planning of a Spherical Mobile Robot

Qiang Zhan

**Abstract** As a new member of mobile robot family, spherical mobile robot (shortly spherical robot) is well-known for its compact structure and agile motion, but its special motion principle and nonholonomic characteristic complicate its motion planning. This chapter first overviews the previous research work of the motion planning of spherical robot, and then introduces the structure and motion principle of spherical robot BHQ-1, and last presents one kinematic motion planning method and one dynamic motion planning method for BHQ-1 respectively. Compared with other motion planning methods of spherical robot, those two methods realize the motion planning of a spherical robot in 3D space and focus more on practical applications.

**Keywords** Spherical mobile robot · Nonholonomic constraints · Kinematic planning · Dynamic planning

## 1 Introduction

Spherical mobile robot (shortly spherical robot) is a new type of mobile robot boomed in recent decades [1–8, 18], which usually has a ball-shaped outer shell to include all its mechanism, control system and batteries inside. Different from those traditional mobile robots, such as wheeled robot, legged robot and tracked robot, spherical robot has no apparent locomotion mechanism and its outer shell works as that. Although many different kinds of spherical robots have been developed, there are mainly two principles to realize its motion: center of gravity displacement and angular momentum conservation. Spherical robot is characterized as compact structure and agile motion, which make it very suitable to be applied in those unmanned environments. For example, like a tumbler a spherical robot can never overturn, even if suffered with

---

Q. Zhan (✉)

Laboratory of Complex Mechanism and Intelligent Control (CMIC), Robotics Institute,  
Beijing University of Aeronautics and Astronautics, No. 37, Xueyuan Road,  
Beijing, Haidian District, China  
e-mail: qzhan@buaa.edu.cn

collision or falling down it can resume stability quickly. The research on spherical robot mainly focuses on the mechanism design and motion control.

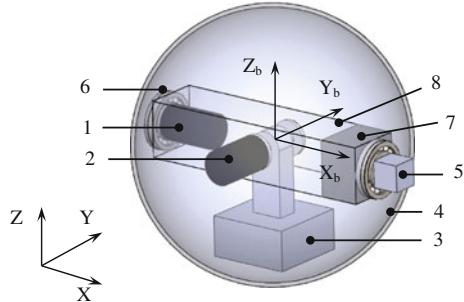
From the control point of view, spherical robot is a nonholonomic system that can control more configuration variables than the number of its degrees of freedom or control inputs but accompanied with much more complexity. Aarne Halme et al. established the kinematic motion model and dynamic motion equation of a spherical robot in one dimensional space by regarding it as a rolling disk, and analyzed its such motion capabilities as uphill motion and overrunning an obstacle as well as some basic motion features [1, 13]. Antonio Bicchi et al. deduced a planar quasi-static kinematic model of a spherical robot by linking a unicycle and a plate-ball system together through some constraints, and planned its motion through solving a set of nonlinear equations [2, 14, 15]. Bhattacharya and Agrawal deduced a first-order mathematical motion model of a spherical robot under the constraints of non-slip and angular momentum conservation, and presented three types of motion planners by considering feasibility, minimum energy and minimum time separately [3]. Mukherjee et al. presented two geometric motion planning strategies to realize the partial and complete reconfiguration of a spherical robot respectively, and the partial reconfiguration strategy uses spherical triangles to bring the sphere to a desired position and a specific orientation and the complete reconfiguration strategy generates a four-steps motion to move the sphere along a trajectory composed of straight lines and curves [4, 16]. Javadi et al. established a dynamic model of a spherical robot with Newton formulation and presented a trajectory planning method by directly calculating the best solution of each step-motor's movement [5]. Cameron et al. discussed the kinematic and dynamic modeling of nonholonomic system and deduced a simplified Boltzmann-Hamel equation for both holonomic and nonholonomic systems [9]. Zhan et al. established a dynamic model of a spherical robot with the simplified Boltzmann-Hamel equation, based on which the motion of a spherical robot is divided into linear motion and circular motion so as to realize complex trajectory planning by dividing it into line segments and curve segments [10]. Chen et al. presented a time and energy optimal trajectory planning method based on quasi-velocity motion model and Hamiltonian function, and discussed the influence of three key factors on the shape and direction of the planned trajectory [11]. Jaimez et al. established the dynamic model of a spherical robot Omnidola with Newton-Euler equations and compared its actual motions with the simulated ones through experiments [18].

In the following of this chapter, a brief introduction of spherical robot BHQ-1 will be given first, and then one motion planning method based on kinematics and one motion planning method based on dynamics will be introduced separately.

## 2 Brief Introduction of Spherical Robot BHQ-1

BHQ-1 is the first kind of spherical robot designed by our lab and the first prototype was implemented in 2001 [8], which is designed for the exploration of unmanned environments. As shown in Fig. 1, BHQ-1 is mainly composed of two motors, one

**Fig. 1** Structure of spherical robot BHQ-1 (1: motor 1, 2: motor 2, 3: mass, 4: shell, 5: camera, 6: bearing, 7: controller & battery, 8: hollow axle)



hollow axle, one mass, one camera, one controller and battery combination, and one ball-shaped shell. In Fig. 1 frame  $\{X_b Y_b Z_b\}$  is a body frame attached to the hollow axle and its origin is coincident with the geometric center of the sphere. The hollow axle connects with the shell through two ball bearings at the two ends and serves as a chassis or frame to install other components, so the outer shell can rotate around the axis of the hollow axle freely and the camera installed on the hollow axle can keep a relatively steady posture no matter BHQ-1 is moving or static. Motor 1 is installed on the hollow axle but its output axle is fixed to the shell, so its rotation can result in the displacement of the mass along  $Y_b$  direction. Motor 2 is also installed on the hollow axle and its output axle is fixed to a link so as to drive the mass along  $X_b$  direction. Installed on the hollow axle the camera is used to take pictures of environments which can be transmitted to a remote control center through a wireless image transmission system. According to the received pictures an operator can not only observe the environment but also control the motion of the spherical robot through a joy stick.

The motion principle of the spherical robot is that the rotations of motor 1 and motor 2 make the mass rotate about axes  $X_b$  and  $Y_b$  respectively and result in the displacement of the center of gravity of the whole system, which produces a displacement moment to counteract the friction moment and makes the robot move. As shown in Fig. 1, when motor 1 rotates and motor 2 keeps still, the mass, the hollow axle, the controller and battery combination, and motor 2 will rotate about the axis of the hollow axle. If the angle displacement  $\theta \geq \theta_0$  ( $\theta_0$  is the angle displacement of the mass to balance the moment caused by static friction), the robot will move forward or backward. Because the moment caused by dynamic friction is less than that caused by static friction, the mass will stay at a position where the angle displacement of the mass is less than  $\theta_0$ . So the system is balanced and the robot can go forward or backward continuously. If motor 1 and motor 2 both rotate the mass will rotate around both axes  $X_b$  and  $Y_b$ , and the compound motion of the mass will produce a displacement gravity moment to cause the robot to turn to the side where the mass stays. For example, if the mass moves to the  $+X_b$  direction the robot will

turn to  $+X_b$  direction and if the mass moves to the  $-Y_b$  direction the robot will turn to the  $-Y_b$  direction. So any required motion of spherical robot BHQ-1 can be easily achieved by the separate control or compound control of two motors.

### 3 Kinematics Based Motion Planning of BHQ-1

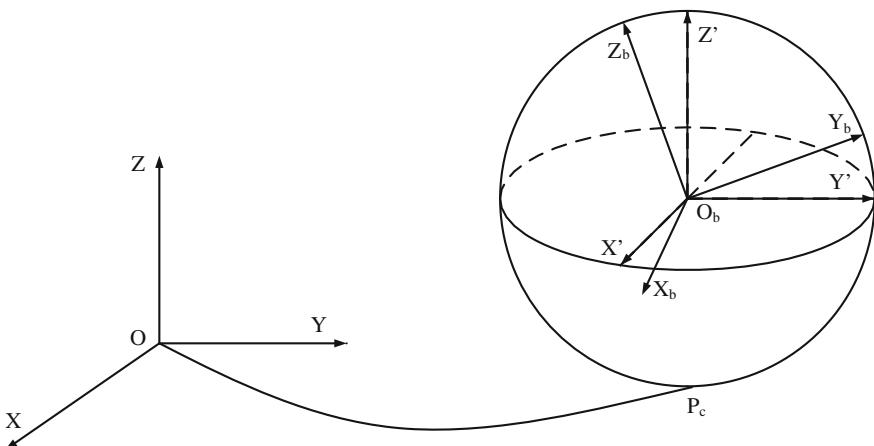
#### 3.1 Nonholonomic Constraint Equations of BHQ-1

In order to describe the configuration of spherical robot BHQ-1, such following frames are established as shown in Fig. 2. Frame  $\{OXYZ\}$  is the reference frame, frame  $\{O_bX'Y'Z'\}$  is a body reference frame with its origin locating at the geometric center of the sphere and its orientation the same as that of reference frame  $\{OXYZ\}$ , frame  $\{O_bX_bY_bZ_b\}$  is the body frame fixed to the hollow axle of BHQ-1 and its origin is the geometric center of the sphere. It's obvious that BHQ-1 cannot move along  $Z$  axis, so it requires five variables to describe its configuration:  $x, y, \psi, \theta, \varphi$ , among which  $x, y$  are the position coordinates of the geometric center of BHQ-1 expressed in the reference frame  $\{OXYZ\}$ ,  $\psi, \theta, \varphi$  are the  $ZXZ$  Euler angle to describe the orientation of BHQ-1.

When spherical robot BHQ-1 moves on the ground, it will come under a velocity constraint due to rolling without slipping: the velocity of the contact point of BHQ-1 and the ground must be the same. Then the following velocity constraint equations can be deduced.

$$\begin{cases} \dot{x} + r(\dot{\psi} \cos \psi \sin \theta - \dot{\theta} \sin \psi) = 0 \\ \dot{y} + r(\dot{\psi} \sin \psi \sin \theta + \dot{\theta} \cos \psi) = 0 \end{cases} \quad (1)$$

where,  $r$  is the radius of BHQ-1.



**Fig. 2** Frames describing the configuration of BHQ-1

For nonholonomic systems quasi-coordinates are widely used due to its several advantages when compared with generalized coordinates. For example, the nonholonomic constraints could be expressed more easily with quasi-coordinates and the projections of kinetic energy can be expressed more simply with quasi-velocities. Normally, the left part of the nonholonomic constraint equations of some systems can be chosen as quasi-velocities and the choice of other quasi-velocities should facilitate the calculation [12].

Here, five quasi-velocities  $\omega_1, \omega_2, \omega_3, \omega_4, \omega_5$  of spherical robot BHQ-1 are chosen as

$$\begin{cases} \omega_1 = \dot{\phi} \sin \psi \sin \theta + \dot{\theta} \cos \psi \\ \omega_2 = -\dot{\phi} \cos \psi \sin \theta + \dot{\theta} \sin \psi \\ \omega_3 = \dot{\psi} + \dot{\phi} \cos \theta \\ \omega_4 = \dot{x} - r\omega_2 \\ \omega_5 = \dot{y} + r\omega_1 \end{cases} \quad (2)$$

where  $\omega_1, \omega_2, \omega_3$  are the projections of the angle velocities of BHQ-1 on the three axes of frame  $\{O_bX'Y'Z'\}$ ,  $\omega_4, \omega_5$  are defined according to the rolling without slipping constraint equations (1). It is easy to get  $\omega_4 = 0, \omega_5 = 0$ .

### 3.2 Optimized Motion Planning Based on Hamiltonian Function

Spherical robot includes all the energy sources inside its shell, so the available energy sources are limited to its size and structure. In order to make a spherical robot move further with the limited energy sources, time and energy based optimized motion planning is greatly preferred.

When spherical robot BHQ-1 moves on the ground, its hollow axle will always keep horizontal except it turns aside. Furthermore, for the ZXZ Euler angles the first two rotations angles  $\psi$  and  $\theta$  cannot result in the rotation of the hollow axle around  $Y_b$  axis, that means only  $\varphi$  can do that, so we can suppose  $\varphi = 0$  in order to simplify the motion planning problem, and then the configuration of BHQ-1 is simplified as  $P = [x, y, \psi, \theta]^T$ . So Eq. (2) can be simplified as

$$\begin{cases} \omega_1 = \dot{\theta} \cos \psi \\ \omega_2 = \dot{\theta} \sin \psi \\ \omega_3 = \dot{\psi} \\ \omega_4 = \dot{x} - r\omega_2 = 0 \\ \omega_5 = \dot{y} + r\omega_1 = 0 \end{cases} \quad (3)$$

From Eq. (3) we can get the kinematics model of BHQ-1 as

$$\begin{cases} \dot{x} = r\omega_2 \\ \dot{y} = -r\omega_1 \\ \dot{\psi} = \omega_3 \\ \dot{\theta} = \omega_1 \sec \psi \end{cases} \quad (4)$$

Rewrite Eq. (4) in the matrix form as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & r & 0 \\ -r & 0 & 0 \\ 0 & 0 & 1 \\ \sec \psi & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = f(P, u), u = [\omega_1 \ \omega_2 \ \omega_3]^T \quad (5)$$

In order to plan an optimized trajectory from the initial configuration  $P_i = [x_i, y_i, \psi_i, \theta_i]^T$  to the goal configuration  $P_g = [x_g, y_g, \psi_g, \theta_g]^T$ , following cost function is introduced.

$$J = \int_0^{t_g} \left[ k + \frac{1}{2}(1-k)(b_1\omega_1^2 + b_2\omega_2^2 + b_3\omega_3^2) \right] dt, \quad (b_1 \geq 0, b_2 \geq 0, b_3 \geq 0, 0 \leq k \leq 1) \quad (6)$$

where,  $k$  describes the tendency of the function to the least time or the least energy, if  $k$  is much smaller the function trends to approach the least energy more, if  $k$  is much bigger the function trends to approach the least time more;  $b_1, b_2, b_3$  describes the weight values of three angle velocities  $\omega_1, \omega_2, \omega_3$ .

A Hamiltonian function is constructed as follows.

$$\begin{aligned} H &= L + \lambda^T f(P, u) \\ &= \left[ k + \frac{1}{2}(1-k)(b_1\omega_1^2 + b_2\omega_2^2 + b_3\omega_3^2) \right] + \lambda_1 r\omega_2 - \lambda_2 r\omega_1 + \lambda_3 \omega_3 + \lambda_4 \omega_1 \sec \psi \end{aligned} \quad (7)$$

where,  $\lambda = [\lambda_1, \lambda_2, \lambda_3, \lambda_4]^T$  is the Lagrange multiplier vector. In order to optimize the trajectory of BHQ-1,  $\dot{\lambda} = -(\frac{\partial H}{\partial P})^T$  must be satisfied, namely

$$\begin{cases} \dot{\lambda}_1 = -\frac{\partial H}{\partial x} = 0 \\ \dot{\lambda}_2 = -\frac{\partial H}{\partial y} = 0 \\ \dot{\lambda}_3 = -\frac{\partial H}{\partial \psi} = -\lambda_4 \omega_1 \sin \psi \sec^2 \psi \\ \dot{\lambda}_4 = -\frac{\partial H}{\partial \theta} = 0 \end{cases} \quad (8)$$

From Eq. (8) we can find that  $\lambda_1, \lambda_2, \lambda_4$  are all constants, but  $\lambda_3$  is a variable on the entire trajectory. In order to optimize the quasi-velocities  $\omega_1, \omega_2, \omega_3$ , the entire trajectory should satisfy  $\frac{\partial H}{\partial u} = 0$ , namely

$$\begin{cases} \frac{\partial H}{\partial \omega_1} = 0 \Rightarrow (1-k)b_1\omega_1 - \lambda_2r + \lambda_4 \sec \psi = 0 \\ \frac{\partial H}{\partial \omega_2} = 0 \Rightarrow (1-k)b_2\omega_2 + \lambda_1r = 0 \\ \frac{\partial H}{\partial \omega_3} = 0 \Rightarrow (1-k)b_3\omega_3 + \lambda_3 = 0 \end{cases} \quad (9)$$

From Eq. (9) the optimized quasi-velocities can be got as following.

$$\begin{cases} \omega_1 = \frac{\lambda_2r - \lambda_4 \sec \psi}{b_1(1-k)} \\ \omega_2 = -\frac{\lambda_1r}{b_2(1-k)} \\ \omega_3 = -\frac{\lambda_3}{b_3(1-k)} \end{cases} \quad (10)$$

Because on the entire optimized trajectory Hamiltonian function must be 0 [9], namely

$$H = L + \lambda^T f(x, u) = k + \frac{1}{2}(1-k)(b_1\omega_1^2 + b_2\omega_2^2 + b_3\omega_3^2) + \lambda_1r\omega_2 - \lambda_2r\omega_1 + \lambda_3\omega_3 + \lambda_4\omega_1 \sec \psi = 0 \quad (11)$$

Substitute the three optimized angle velocities in Eq. (10) for those in Eq. (11) we can get

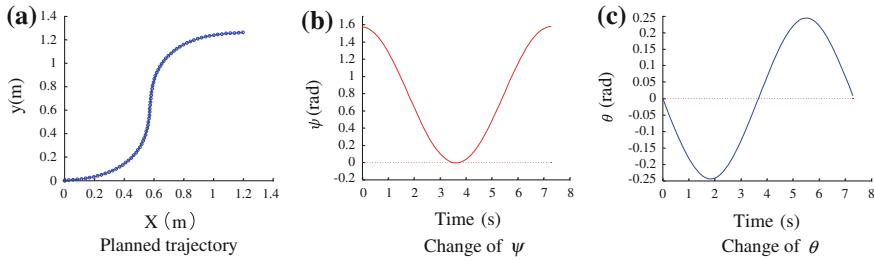
$$\lambda_3 = \pm \sqrt{-b_3 \frac{2b_1b_2(k^2 - k) + r^2(b_1\lambda_1^2 + b_2\lambda_2^2) - 2rb_2\lambda_2\lambda_4 \sec \psi + b_2\lambda_4^2 \sec^2 \psi}{b_1b_2}} \quad (12)$$

From the above equation we can find  $\lambda_3 = \lambda_3(\psi)$ . The symbol of  $\lambda_3$  can be got from experiments, and for spherical robot BHQ-1 we choose it as a negative one according to experience. Then the trajectory equation of spherical robot BHQ-1 can be deduced from Eq. (4) as follows.

$$\begin{cases} \dot{x} = r\omega_2 \\ \dot{\psi} = \omega_3 \\ \dot{y} = -r\omega_1 \\ \dot{\psi} = \omega_3 \end{cases} \Rightarrow \begin{cases} \frac{dx}{d\psi} = \frac{r\omega_2}{\omega_3} = \frac{\lambda_1r^2b_3}{\lambda_3b_2} = h_1(\psi) \\ \frac{dy}{d\psi} = -\frac{r\omega_1}{\omega_3} = \frac{(\lambda_2r - \lambda_4 \sec \psi)r b_3}{\lambda_3b_1} = h_2(\psi) \end{cases} \quad (13)$$

From Eq. (13) we can find that  $\frac{dx}{d\psi}, \frac{dy}{d\psi}$  are all functions of  $\psi$ , which means the calculation of  $x, y$  can be greatly simplified because it can be got by integrating the above equation from initial  $\psi = 0$  to final  $\psi = \psi_g$ , namely

$$\begin{cases} x = \int_0^{\psi_g} h_1(\psi) d\psi \\ y = \int_0^{\psi_g} h_2(\psi) d\psi \end{cases} \quad (14)$$



**Fig. 3** Trajectory planning simulations **a** Planned trajectory. **b** Change of  $\psi$ . **c** Change of  $\theta$

It's clear that Eq. (14) has no variables of time  $t$ , so there is no need to find the final time  $t_g$  when computing  $x$  and  $y$ . However,  $\psi$  cannot be a monotone function, so it should be divided into several segments and the piecewise points are those that make  $\dot{\psi} = 0$ . Thus in each segment  $\psi$  will increase or decrease monotonously. From Eq. (4) we can find that  $\omega_3 = 0$  must be satisfied in order to make  $\dot{\psi} = 0$ , and then from Eqs. (8) and (10) we can get  $\lambda_3 = \lambda_3(\psi) = 0$ , so those piecewise points can be decided according to the equation. So the optimized trajectory of spherical robot BHQ-1 can be deduced.

In real applications a group of suitable or optimized coefficients  $\lambda_1, \lambda_2, \lambda_4$  should be decided first for the given goal position and orientation  $(x_g, y_g, \psi_g, \theta_g)$ , and which are usually got according to experiences.

For spherical robot BHQ-1, suppose the initial configuration is  $P_i = [0, 0, \frac{\pi}{2}, 0]$  and the final configuration is  $p_g = [1.25, 1.25, \frac{\pi}{2}, 0]$ , then the optimized motion can be planned as follows. First, choose a group of coefficients  $\lambda_1, \lambda_2, \lambda_4$ , here according experience we choose  $\lambda_1 = 0.5, \lambda_2 = 0.5, \lambda_4 = 0.3$ . Then we choose  $k = 0.5, b_1 = b_2 = b_3 = 1$ . The optimized trajectory and the changes of two orientation variables of spherical robot BHQ-1 are shown in Fig. 3. From the simulations we can find that the planned trajectory and the curves of two orientations are all smooth.

### 3.3 The Influence of $\lambda_1, \lambda_2, \lambda_4$ on Planned Trajectory

In order to facilitate the choice of coefficients  $\lambda_1, \lambda_2, \lambda_4$ , of which the influence on the trajectory shape and the moving direction of robot BHQ-1 are discussed by a group of simulations. Here, suppose  $\psi_g = \frac{2\pi}{5}$ .

First, let  $\lambda_2$  and  $\lambda_4$  be constants and let  $\lambda_1$  change from  $-1.5$  to  $1.5$ , different planned trajectories are shown in Fig. 4. From the simulation results we can find that the change of  $\lambda_1$  can affect the trajectory shape greatly and the symbol of  $\lambda_1$  can affect the moving direction of BHQ-1 along  $Y$  direction.

Then let  $\lambda_1$  and  $\lambda_4$  be constants and let  $\lambda_2$  change from  $-0.9$  to  $0.9$ , those different planned trajectories are shown in Fig. 5. From the simulation results we can find that

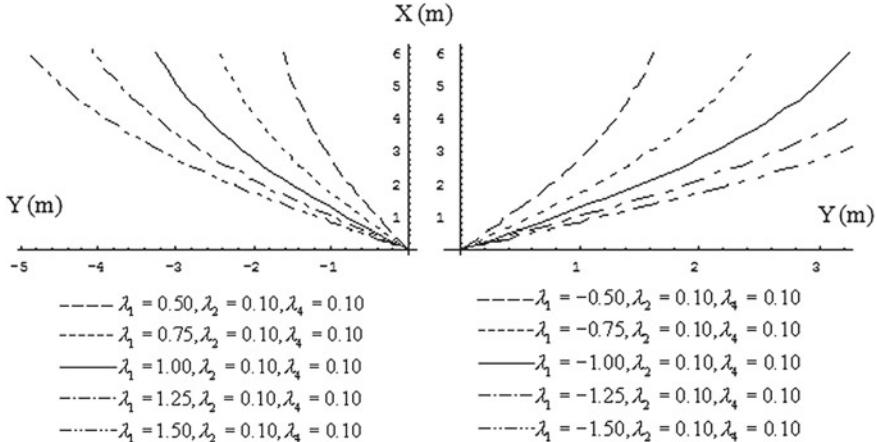


Fig. 4 Trajectory planning results when  $\lambda_1$  changes

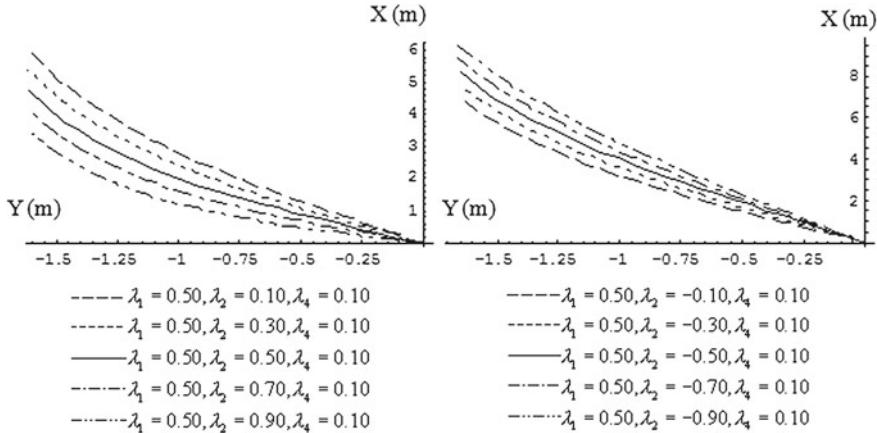
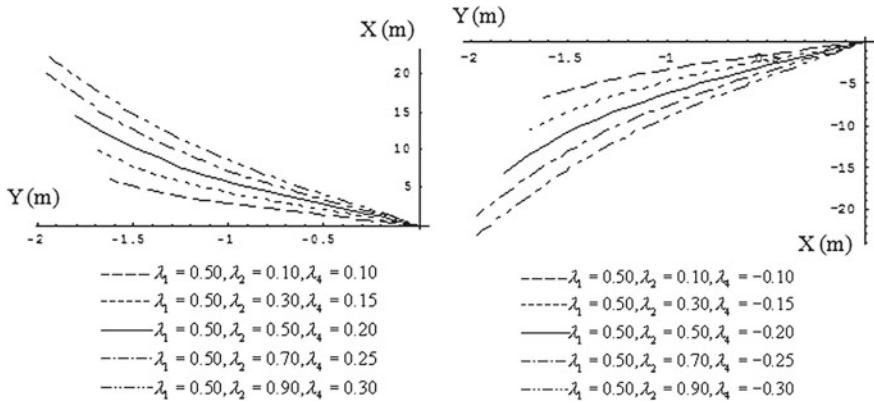


Fig. 5 Trajectory planning results when  $\lambda_2$  changes

the change of  $\lambda_2$  affects the trajectory shape little and the symbol of  $\lambda_2$  cannot change the moving direction of spherical robot BHQ-1.

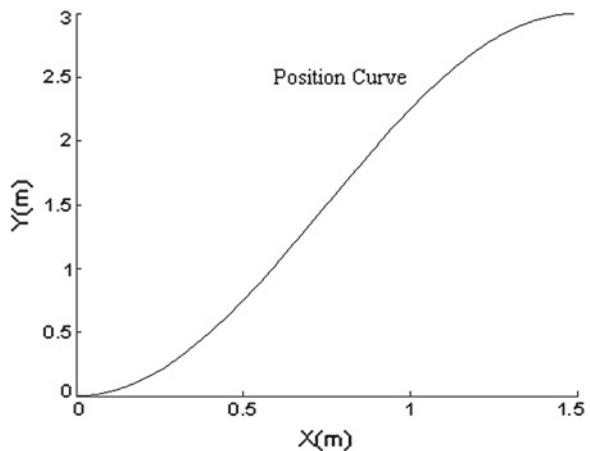
At last, let  $\lambda_1$  and  $\lambda_2$  be constants and let  $\lambda_4$  change from  $-0.3$  to  $0.3$ , those different planned trajectories are shown in Fig. 6. From the simulation results we can find that the change of  $\lambda_4$  can greatly affect the trajectory shape and the final position  $(x, y)$ , and the symbol of  $\lambda_4$  can change the moving direction of BHQ-1 along  $X$  direction.

The above simulations reveal the influence of  $\lambda_1, \lambda_2, \lambda_4$  on the planned trajectory of spherical robot BHQ-1 respectively, which can help to decide a group of suitable  $\lambda_1, \lambda_2, \lambda_4$  for a real application by experience.



**Fig. 6** Trajectory planning results when  $\lambda_4$  changes

**Fig. 7** Optimal trajectory from (0, 0) to (1.5, 3) by shooting method



One way to directly get a group of optimized  $\lambda_1, \lambda_2, \lambda_4$  has been proposed in [17], which is called “shooting” method. Using the “shooting” method, we can plan an optimal trajectory of spherical robot BHQ-1 from start position (0, 0) to final position (1.5, 3), as shown in Fig. 7. Here,  $k = 0.5$ ,  $b_1 = b_2 = b_3 = 1$ ,  $\lambda_1 = 0.08$ ,  $\lambda_2 = 0.78$ ,  $\lambda_4 = 0.56$ . Although the “shooting” method can get the optimized variables, it’s not always effective for some cases.

### 3.4 Motion Planning Experiments

In order to validate the proposed trajectory planning method motion experiments of spherical robot BHQ-1 avoiding an obstacle were done. BHQ-1 is planned to move



**Fig. 8** Motion planning experiments of BHQ-1

straight first, but there is an obstacle in its path, when it detects the obstacle it will avoid it. An infrared sensor was used by BHQ-1 to detect obstacles, and the motion commands were sent to BHQ-1 from a PC through a wireless system. The radius of the experimental spherical robot BHQ-1 is 200 mm, and its total mass is about 2.5 kg.

Figure 8 shows some pictures of one experiment during which spherical robot BHQ-1 avoided an obstacle successfully. To be honest, there were also several cases that BHQ-1 could not avoid the obstacle successfully due to some practical reasons, such as the motion errors, the delay on re-planning.

## 4 Dynamics Based Motion Planning of BHQ-1

### 4.1 Dynamic Model of BHQ-1

Compared with kinematics based motion planning, dynamics based motion planning can achieve more steady motion and better performance when meeting unpredicted external disturbances. But not all the dynamic modeling methods can be used for nonholonomic systems except Gibbs-Appell equation, improved Lagrange equation, Kane equation and Boltzmann-Hamel equation, etc. However, it is always difficult to use those methods to establish a simplified dynamic model of a spherical robot that can be used in real applications due to the complex deduction procedures and time-consuming computations.

From D'Alembert-Lagrange principle:  $\sum_{k=1}^n \left( \frac{d}{dt} \frac{\partial T}{\partial \dot{q}_k} - \frac{\partial T}{\partial q_k} - Q_k \right) \delta q_k = 0$ , Cameron et al. deduced a simplified Boltzmann-Hamel equation that can be applied to both holonomic system and nonholonomic system [9], shown in the following.

$$\frac{d}{dt} \frac{\partial \bar{E}}{\partial \omega_I} + \sum_{j=1}^n \sum_{i=1}^n \eta_{iI} \gamma_{ij} \frac{\partial \bar{E}}{\partial \omega_j} - \sum_{j=1}^n \eta_{jI} \frac{\partial \bar{E}}{\partial q_j} = M_I \quad (15)$$

where,  $\omega$  is the vector of quasi-velocities,  $E$  is the kinetic energy,  $M$  is the generalized driving force,  $\eta$  and  $\gamma$  are coefficients,  $I$  denotes the independent quasi-coordinates,  $n$  is the number of the generalized coordinate  $q_j$ ,  $t$  is the time.

Different from the traditional Boltzmann-Hamel equation, the new one is expressed explicitly in terms of generalized coordinate  $q_j$  and coefficient  $\gamma$  can

be easily calculated. So the dynamic model has a more compact expression and can be used more easily.

A configuration vector  $P = [x, y, \psi, \theta, \varphi]^T$  is used to describe the position and orientation of spherical robot BHQ-1, and the definition of those variables are the same as that in Sect. 3.

Rewrite Eq. (2) in matrix form as

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \\ \omega_5 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cos \psi & \sin \theta \sin \psi \\ 0 & 0 & 0 & \sin \psi & -\sin \theta \cos \psi \\ 0 & 0 & 1 & 0 & \cos \theta \\ 1 & 0 & 0 & -r \sin \psi & r \sin \theta \cos \psi \\ 0 & 1 & 0 & r \cos \psi & r \sin \theta \sin \psi \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \alpha \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} \quad (16)$$

where  $\alpha$  is a  $5 \times 5$  transformation matrix. From Eq. (16) we can deduce

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} 0 & r & 0 & 1 & 0 \\ -r & 0 & 0 & 0 & 1 \\ -\sin \psi \cot \theta & \cos \psi \cot \theta & 1 & 0 & 0 \\ \cos \psi & \sin \psi & 0 & 0 & 0 \\ \sin \psi \csc \theta & -\cos \psi \csc \theta & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \\ \omega_5 \end{bmatrix} = \beta \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \\ \omega_5 \end{bmatrix} \quad (17)$$

where  $\beta$  is also a  $5 \times 5$  transformation matrix.

Coefficient  $\gamma_{ij}$  in Eq. (15) can be calculated by  $\alpha$  and  $\beta$  according to the following equation.

$$\gamma_{ij} = \sum_{k=1}^5 \sum_{s=1}^5 \omega_s \beta_{ks} \left( \frac{\partial \alpha_{ij}}{\partial q_k} - \frac{\partial \alpha_{kj}}{\partial q_i} \right) \quad (18)$$

Kinetic energy  $\bar{E}$  of spherical robot BHQ-1 is

$$\bar{E} = \frac{1}{2} m (\dot{x}^2 + \dot{y}^2) + \frac{1}{2} \times \frac{2}{5} m r^2 \cdot (\dot{\psi}^2 + \dot{\theta}^2 + \dot{\varphi}^2 + 2\dot{\psi}\dot{\varphi} \cos \theta) \quad (19)$$

where  $m$  is the total mass of BHQ-1. Equation (19) can be expressed by quasi-velocities as

$$\bar{E} = \frac{1}{2} m \left[ \frac{5}{7} r^2 (\omega_1^2 + \omega_2^2) + \frac{2}{5} r^2 \omega_3^2 + 2r\omega_2\omega_4 - 2r\omega_1\omega_5 + \omega_4^2 + \omega_5^2 \right] \quad (20)$$

From Eq. (20) we can get

$$\begin{cases} \frac{\partial \bar{E}}{\partial \omega_1} = \frac{7}{5}mr^2\omega_1 - mr\omega_5 \\ \frac{\partial \bar{E}}{\partial \omega_2} = \frac{7}{5}mr^2\omega_2 + mr\omega_4 \\ \frac{\partial \bar{E}}{\partial \omega_3} = \frac{2}{5}mr^2\omega_3 \end{cases} \quad (21)$$

Because  $\omega_4 = \omega_5 = 0$ , we can get the simplified form of Eq. (21) as

$$\begin{cases} \frac{\partial \bar{E}}{\partial \omega_1} = \frac{7}{5}mr^2\omega_1 \\ \frac{\partial \bar{E}}{\partial \omega_2} = \frac{7}{5}mr^2\omega_2 \\ \frac{\partial \bar{E}}{\partial \omega_3} = \frac{2}{5}mr^2\omega_3 \end{cases} \quad (22)$$

From Eq. (19) we can get  $\frac{\partial \bar{E}}{\partial x} = \frac{\partial \bar{E}}{\partial y} = \frac{\partial \bar{E}}{\partial \psi} = \frac{\partial \bar{E}}{\partial \varphi} = 0$ .

Substituting those calculated  $\eta$ ,  $\gamma$  and Eq. (21) for those in Eq. (15) the dynamic model of spherical robot BHQ-1 can be deduced as

$$\begin{cases} \frac{7}{5}mr^2\dot{\omega}_1 = m_1^0 - rf_2^0 \\ \frac{7}{5}mr^2\dot{\omega}_2 = m_2^0 + rf_1^0 \\ \frac{2}{5}mr^2\dot{\omega}_3 = m_3^0 \end{cases} \quad (23)$$

where,  $m_1^0, m_2^0, m_3^0$  are the projections of the principal moment  $m^0$  on the three axes of the body reference frame  $\{O_bX'Y'Z'\}$ ,  $f_1^0, f_2^0$  are the projections of the principal force  $f^0$  on axes  $X', Y'$  of frame  $\{O_bX'Y'Z'\}$ ,  $f^0$  and  $m^0$  are the principal force and principal moment imposed on the geometric center of spherical robot BHQ-1 respectively.

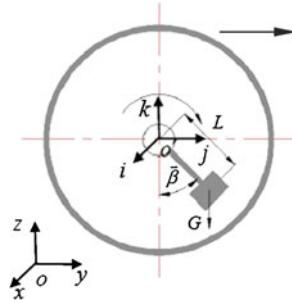
## 4.2 Motion Planning Based on Dynamic Model of BHQ-1

### 4.2.1 Linear Trajectory Planning

In Fig. 9, frame  $\{o'ijk\}$  is located on the geometric center of BHQ-1 and its orientation is the same as that of frame  $\{oxyz\}$ . When spherical robot BHQ-1 moves along a linear trajectory its hollow axle and those installed components will rotate around axis  $i$  to reach a high position supposed as the one shown in Fig. 9.

Because there is no rotation about axes  $j$  and  $k$ ,  $\psi = 0$  and  $\varphi = 0$  are obtained, substituting them for the variables in Eq. (16), we can get those quasi-velocities as

$$\omega_1 = \dot{\theta}, \omega_2 = 0, \omega_3 = 0, \omega_4 = 0, \omega_5 = 0 \quad (24)$$



**Fig. 9** Straight motion

In Fig. 9, the gravity direction is along the  $-k$  direction (downward vertically), so the projection of gravity on plane  $io'j$  is zero, that it to say  $f_1^0 = 0$ ,  $f_2^0 = 0$ , substituting them for the variables in the dynamic model (23), the following simplified dynamic model of BHQ-1 can be got.

$$\begin{cases} m_1^0 = \frac{7}{5}mr^2\ddot{\theta} \\ m_2^0 = 0 \\ m_3^0 = 0 \end{cases} \quad (25)$$

So the gravity moment exists only around axis  $i$  and the mass sways only in the plane  $o'kj$ . From Eq. (25) and

$$m_1^0 = mg\vec{l}_j + \vec{f}r \quad (26)$$

we can get the driving moment of motor 1 is

$$M_1(t) = mg\vec{l}_j + mL^2\ddot{\beta} = -\frac{7}{5}mr\ddot{y}(t) - \vec{f} \cdot r + mL^2\ddot{\beta}(t) \quad (27)$$

where,  $L$  is the distance between the center of the mass and the rotation axis of the hollow axle,  $\vec{l}_j$  is the projection vector of  $L$  on axis  $j$ ,  $\beta$  is the angle that the mass deviates from axis  $k$ ,  $f$  is the friction vector imposed on the spherical robot by the ground,  $g$  is the gravity acceleration. The driving moment of motor 2 is

$$M_2(t) = 0. \quad (28)$$

If spherical robot BHQ-1 moves along a straight trajectory with a constant velocity it is obvious that  $\ddot{y} = 0$ . So we get  $\dot{\omega}_1 = 0$  from Eq. (16),  $\ddot{\theta} = 0$  from Eq. (24) and  $m_1^0 = 0$  from Eq. (25), substituting them for the variables in Eq. (26) we can get

$$\vec{l}_j = -\frac{\vec{f} \cdot \vec{r}}{mg} \quad (29)$$

$$\beta(t) = \arcsin \frac{\vec{l}_j}{L} = \arcsin \left( -\frac{\vec{f} \cdot \vec{r}}{mgL} \right) \quad (30)$$

Thus when spherical robot BHQ-1 moves along a straight trajectory with a constant velocity the driving moments of two motors are

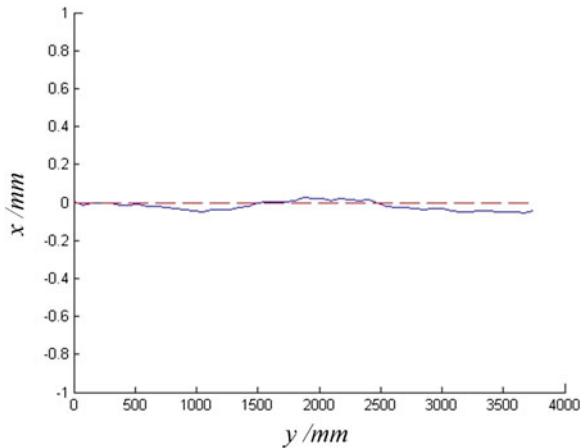
$$\begin{cases} M_1 = mg\vec{l}_j = -\vec{f}r \\ M_2 = 0 \end{cases} \quad (31)$$

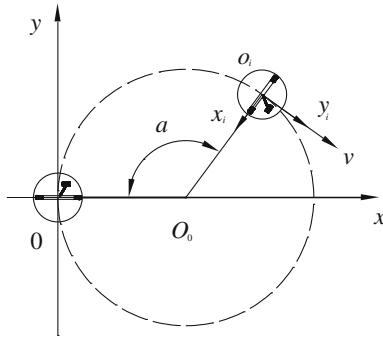
The simulation results of BHQ-1 moving straight are shown in Fig. 10, where the dashed line is the planned trajectory in theory or the target trajectory and the solid line is the planned trajectory by adding 1% noise disturbance. From the simulation we can conclude that BHQ-1 can realize motion along a linear trajectory with the deduced dynamic model.

#### 4.2.2 Circular Trajectory Planning

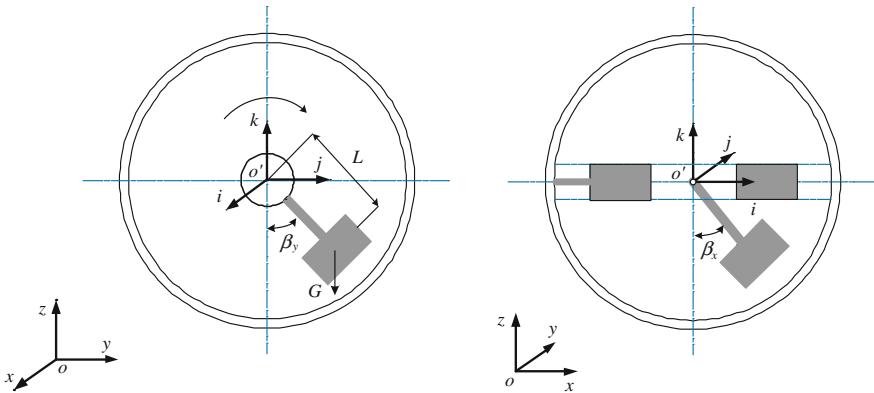
Assume spherical robot BHQ-1 moves along a circular trajectory from the initial configuration to the final configuration, as shown in Figs. 11 and 12. In Fig. 11, a frame  $\{o_i x_i y_i\}$  is established on the geometric center of BHQ-1 with its axis  $x_i$  pointing to the center of the circular trajectory and its axis  $y_i$  pointing to the tangential direction of the circular trajectory.

**Fig. 10** Simulation results of BHQ-1 moving straight





**Fig. 11** Theoretic trajectory of circular motion of BHQ-1



**Fig. 12** Two views of the mass during circular motion

With similar derivation to that of linear trajectory planning, we can deduce  $f_1^0 = 0$ ,  $f_2^0 = 0$ ,  $m_3^0$ . Let  $p = \frac{7mr^2}{5}$  and substitute the above variables for those Eq.(23), we can get

$$\begin{cases} m_1^0 = p\dot{\omega}_1 \\ m_2^0 = p\dot{\omega}_2 \\ \dot{\omega}_3 = 0 \end{cases} \quad (32)$$

Because  $m_1^0$ ,  $m_2^0$  are projections of the principal moment on axes  $i$  and  $j$ , it's easy to get

$$\begin{cases} m_1^0(t) = mg \vec{l}_y + \vec{j}_y \cdot r \\ m_2^0(t) = mg \vec{l}_x + \vec{f}_x \cdot r \end{cases} \quad (33)$$

where,  $\vec{l}_x$ ,  $\vec{l}_y$  are the projections of length  $L$  on axes  $x$  and  $y$  respectively,  $\vec{f}_x$ ,  $\vec{f}_y$  are the projections of the friction force  $f$  on axes  $x$  and  $y$  respectively.

From Eqs. (16), (32) and (33), we can get

$$\begin{cases} \vec{l}_x = \frac{1}{mg} \left( \frac{p}{r} \ddot{x} - \vec{f}_x \cdot r \right) \\ \vec{l}_y = \frac{1}{mg} \left( -\frac{p}{r} \ddot{y} - \vec{f}_x \cdot r \right) \end{cases} \quad (34)$$

Through coordinates transformation we can get

$$\begin{bmatrix} l_{xi} \\ l_{yi} \\ 0 \end{bmatrix} = R(z, -\alpha) \cdot \begin{bmatrix} l_x \\ l_y \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(-\alpha) & -\sin(-\alpha) & 0 \\ \sin(-\alpha) & \cos(-\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} l_x \\ l_y \\ 0 \end{bmatrix} \quad (35)$$

where,  $\alpha$  is the angle that spherical robot BHQ-1 has moved along the circular trajectory from origin  $o$  (as shown in Fig. 11),  $l_x, l_y$  are the norms of  $\vec{l}_x, \vec{l}_y$  respectively,  $\vec{l}_{xi}, \vec{l}_{yi}$  are the projections of length  $L$  on axes  $x_i$  and  $y_i$  respectively (shown in Fig. 11),  $l_{xi}, l_{yi}$  are the norms of  $\vec{l}_{xi}, \vec{l}_{yi}$  respectively.

From Eqs. (34) and (35) we can obtain

$$\begin{cases} l_{xi} = \frac{p}{mgr} (\ddot{x} \cos \alpha - \ddot{y} \sin \alpha) \\ l_{yi} = -\frac{p}{mgr} (\ddot{x} \sin \alpha + \ddot{y} \cos \alpha) + \frac{f \cdot r}{mg} \end{cases} \quad (36)$$

Because  $\alpha = \omega t$ , the driving moment of motor 1 can be got as

$$M_1 = mg \vec{l}_j + m \ddot{\beta}_y L^2 = -\frac{p}{r} (\ddot{x} \sin \alpha + \ddot{y} \cos \alpha) + m \ddot{\beta}_y L^2 \quad (37)$$

where,  $\vec{l}_j$  is the projection vector of  $L$  on axis  $j$ ,  $\beta_y$  is the angle that the mass deviates from axis  $k$  measured on plane  $o'kj$  (as shown in Fig. 12).

The driving moment of motor 2 is

$$M_2 = mg \vec{l}_i + m \ddot{\beta}_x L^2 = \frac{p}{r} (\ddot{x} \cos \alpha - \ddot{y} \sin \alpha) + m \ddot{\beta}_x L^2 \quad (38)$$

where,  $\vec{l}_i$  is the projection vector of  $L$  on axis  $i$ ,  $\beta_x$  is the angle that the mass deviates from axis  $k$  measured on planes  $o'ki$  (as shown in Fig. 12).

If spherical robot BHQ-1 moves along a circular trajectory with a fixed velocity its acceleration  $A = \frac{v^2}{R}$  should point to the center of the circular trajectory and the tangential velocity of circular trajectory  $v = \omega R$  should be a constant. Where,  $R$  is the radius of the circular trajectory, is the angle velocity of BHQ-1. The projections

of the acceleration A on axes x and y are

$$\begin{cases} A_x = -A \cos(\pi - \alpha) = -A \cos(\pi - \omega t) \\ A_y = -A \sin(\pi - \alpha) = -A \sin(\pi - \omega t) \end{cases} \quad (39)$$

From,  $\begin{cases} \ddot{x} = A_x \\ \ddot{y} = A_y \end{cases}$ , Eqs.(16), (32), (39) we can get

$$\begin{cases} m_1^0(t) = \frac{pA}{r} \sin(\omega t) \\ m_2^0(t) = \frac{pA}{r} \cos(\omega t) \end{cases} \quad (40)$$

From Eqs.(2), (17), (36) and (40) and the values of p and A, we can get

$$\begin{cases} l_{xi} = \frac{7rv^2}{5gR} \\ l_{yi} = \frac{f \cdot r}{mg} \end{cases} \quad (41)$$

From Eq.(41) we can get

$$\begin{cases} \beta_{x1} = \arcsin\left(\frac{\vec{l}_i}{L}\right) = \arcsin\left(\frac{7rv^2}{5gRL}\right) \\ \beta_{y1} = \arcsin\left(\frac{\vec{l}_j}{L}\right) = \arcsin\left(\frac{fr}{mgL}\right) \end{cases} \quad (42)$$

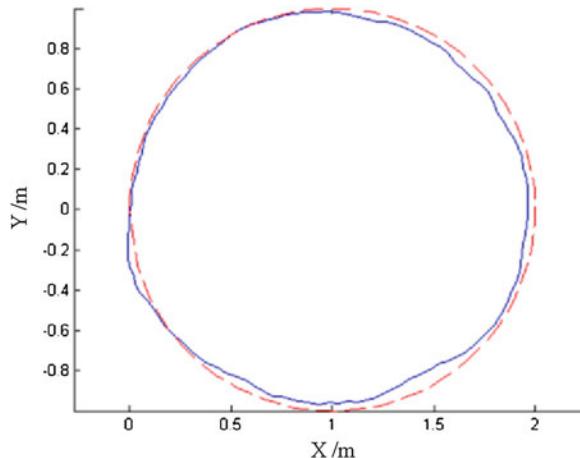
If spherical robot BHQ-1 moves along a circular trajectory with a fixed velocity the driving moments of two motors are

$$\begin{cases} M_1 = mg \cdot l_j = fr \\ M_2 = mg \cdot l_i = \frac{7mr v^2}{5R} \end{cases} \quad (43)$$

A simulation result of spherical robot BHQ-1 moving along a circular trajectory is shown in Fig. 13, where the dashed circle is the planned trajectory in theory or the target trajectory and the solid circle is the planned trajectory by adding 1% noise disturbance. From the simulation we can conclude that BHQ-1 can realize circular motion with the deduced dynamic model.

#### 4.2.3 Complex Trajectory Planning

Theoretically any complex trajectory can be approximately divided into line segments and curve segments, so the deduced linear trajectory motion planning model and circular trajectory motion planning model can also be used to plan the motion of complex trajectories. In order to validate that, a motion planning simulation of spherical robot BHQ-1 moving along a complex trajectory was presented in Fig. 14.

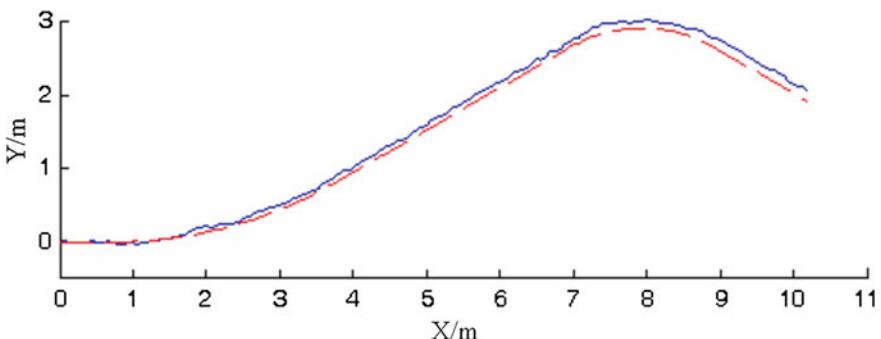


**Fig. 13** Simulation of circular motion of BHQ-1

The given trajectory is composed of three straight line segments and two curves, which are described by functions as

$$y = \begin{cases} 0 & 0 \leq x < 1 \\ 6 - \sqrt{36 - (x - 1)^2} & 1 \leq x < 4 \\ \frac{\sqrt{3}}{3}x + 6 - \frac{13}{3}\sqrt{3} & 4 \leq x < 7 \\ 6 - 3\sqrt{3} - \sqrt{4 - (x - 8)^2} & 7 \leq x < 9 \\ -\frac{\sqrt{3}}{3}x + 6 + \sqrt{3} & 9 \leq x \leq 10 \end{cases} \quad (44)$$

In Fig. 14, the given trajectory is shown in dashed line and the planned trajectory is shown in solid line. 1 % disturbance noise was introduced to the dynamic trajectory planning method in order to test the robustness of the method. From the simulation we



**Fig. 14** Motion simulation of a complex trajectory planning of BHQ-1

can find that the dynamic motion planning method can plan an accurate trajectory for spherical robot BHQ-1, and the small error comes from the noise added intentionally. If there is no disturbance noise the planned trajectory will superpose the target one.

## 5 Conclusion

Spherical mobile robot has compact structure and flexible motion, but because of its special nonholonomic characteristic, traditional motion planning methods proposed for wheeled mobile robots cannot be applied to it. This chapter introduces two motion planning methods for spherical robot BHQ-1, one is a kinematics based motion planning method and another is a dynamics based motion planning method. The kinematic motion planning method uses Hamiltonian function to realize the time and energy based optimal motion planning, and the characteristics of three coefficients  $\lambda_1, \lambda_2, \lambda_4$  are revealed through simulations. The dynamic motion planning method uses a simplified Boltzmann-Hamel equation to get the dynamic model of spherical robot BHQ-1, and the moments of two motors to realize the linear motion and circular motion are deduced respectively. Simulations and experiments are provided in order to validate those motion planning methods. It should be noted that although the two methods are proposed for spherical robot BHQ-1, which can be also used by other spherical robots with similar structure or similar motion principle.

Thanks very much for the help and research work of my students JIA Chuan, LIU Zengbo, CHI Xing and SHANG Zhimeng.

## References

1. Halme A, Schonberg T, Wang Y (1996) Motion control of a spherical mobile robot. In: 4th IEEE international workshop on advanced motion control AMC'96. Mie University, Japan, pp 100–106
2. Bicchi A, Balluchi A, Prattichizzo D, Gorelli A (1997) Introducing the “SPHERICAL”: an experimental testbed for research and teaching in nonholonomy. In: Proceedings of the 1997 IEEE international conference on robotics and automation, Albuquerque, New Mexico, April 1997, pp 2620–2625
3. Bhattacharya S, Agrawal SK (2000) Design, experiments and motion planning of a spherical rolling robot. In: Proceedings of the 2000 IEEE international conference on robotics and automation, San Francisco, CA, pp 1207–1212
4. Mukherjee R, Minor MA, Pukrushpan JT (1999) Simple motion planning strategies for spherobot: a spherical mobile robot. In: Proceedings of IEEE international conference on decision and control, Phoenix, Arizona, pp 2132–2137
5. Amir Homayoun Javadi A, Mojabi P (2002) Introducing august: a novel strategy for an omnidirectional spherical rolling robot. In: Proceedings of IEEE international conference on robotics and automation, pp 3527–3533
6. Michaud F, Caron S (2002) Roball: the rolling robot. Auton Robots 12(2):211–222
7. Zhan Q (2001) Kinematic structure of a new type of moving mechanism of lunar vehicles. In: Proceedings of the second symposium on moon exploring technology, Beijing, pp 328–330

8. Qiang Z, Yao C, Caixia Y (2011) Design, analysis and experiments of an omnidirectional spherical robot. In: 2011 IEEE international conference on robotics and automation, Shanghai, 9–13 May, pp 4921–4926
9. Cameron M, Book Wayne J (1997) Modeling mechanisms with nonholonomic joints using the Botzmann-Hamel equations. *Int J Robot Res* 16(1):47–59
10. Zhan Q, Zhou T, Chen M, Cai S (2006) Dynamic trajectory planning of a spherical mobile robot. In: IEEE conference on robotics, automation and mechatronics (RAM), pp 714–719
11. Ming C, Qiang Z, Zengbo L, Yao C (2008) Optimized trajectory planning based on Hamiltonian function of a spherical robot. *High Technol Lett* 14(31):71–75
12. Fengxiang Mei (1985) Basal dynamics for nonholonomic system. Beijing Industrial College Press, Beijing
13. Halme A, Suomela J, Schonberg T, Wang YA (1996) Spherical mobile micro-robot for scientific applications. In: Proceedings of ASTRA 96, ESTEC, Noordwijk, The Netherlands, November 1996
14. Camicia C, Conticell F, Bicchi A (2000) Nonholonomic kinematics and dynamics of the spherical. In: Proceedings of the 2000 IEEE/RSJ international conference on intelligent robots and systems, pp 806–810
15. Bicchi A, Marigo AA (2002) local-local planning algorithm for rolling objects. In: Proceedings of the 2002 IEEE international conference on robotics and automation, May 2002, pp 1759–1764
16. Mukherjee R, Das T (2002) Feedback stabilization of a spherical mobile robot. In: Proceedings of IEEE international conference on intelligent robots and systems, vol. 3. pp 2154–2162
17. Press William H, Flannery Brian P, Teukolsky Saul A, Vetterling William T (1988) Numerical recipes in C: the art of scientific computing. Cambridge University Press, Cambridge
18. Jaimez M, Castillo JJ, García F, Cabrera JA (2012) Designing and modelling of Omnidola, a spherical mobile robot. *Mech Based Des Struct Mach: Int J* 40(4):383–399

**Part IV**

**Motion Planning for Legged Robots**

# A Minimum Jerk-Impedance Controller for Planning Stable and Safe Walking Patterns of Biped Robots

Amira Aloulou and Olfa Boubaker

**Abstract** Minimum Jerk based control is part of optimal control laws. Its main contribution resides in the generation of smooth trajectories allowing the avoidance of sudden and abrupt motion. This chapter proposes the elaboration of appropriate control laws, with controller parameters computed offline, able to produce stable smooth and safe walking cycles for bipedal robots evolving in the three dimensional space. To alternate footsteps, Minimum Jerk and Impedance control principles are used to switch successively between single support, impact and double support phases. A new methodology of Minimum Jerk control is proposed to produce human like trajectories. Its originality mostly relies on the generation of Cartesian three-dimensional reference trajectories that do combine benefits of trigonometric and polynomial functions. When considering the impact and double support phases, an appropriate impedance control law is proposed to ensure the robot stability and safe balance during the contact with the ground. Simulation results performed on a 15 link/26 degrees of freedom Humanoid robot with a weight of 70 kg and a height of 1.73 m walking at a velocity of  $0.6 \text{ m s}^{-1}$ , show that the dynamics of the robot during the swing phase are very attractive since smooth trajectories without dynamic vibrations are observed and a stable and safe elastic contact takes place while achieving the constrained phases even in presence of sensory noise and uncertainties on the environment stiffness.

**Keywords** Biped robots · Minimum Jerk control · Impedance control · Safe walking patterns

---

A. Aloulou · O. Boubaker (✉)  
National Institute of Applied Sciences and Technology, INSAT,  
Centre Urbain Nord, BP 676, 1080 Tunis, Tunisia  
e-mail: olfa.boubaker@insat.rnu.tn

## 1 Introduction

Despite the numerous technology advances in the field of Humanoid Robotics, mobility and ability to move like a human being remains an essential required feature. In that context, gait pattern generation seems to be the key problem to research dedicated to walking robots [1]. Actually, there are two major walking patterns to be found in bipedal robotics: static walking and dynamic walking [2, 3] and several stability criteria may be used depending on the walking pattern selected. The recourse to a given stability criterion ensures the bipedal robot's balance at every moment in order to avoid falls and collapses. Besides, whether considering the static or dynamic walking pattern, each gait pattern involves the same various phases that are the single support phase, the impacts with the ground and the double support phase. A satisfying control strategy for a gait pattern has to provide good dynamic performances in these different modes in terms of similarity with human gait by guaranteeing at the same time stability, smoothness and safety.

Thus, in this work, to control bipedal robots during the single support stage, we have chosen to focus on Minimum Jerk based control strategy. The main benefit of such approach resides in the generation of smooth trajectories in order to avoid any abrupt motion and consequently limit the robot vibrations [4]. The concept of Jerk Minimization has been developed by Hogan, the pioneer in using the Minimum Jerk principle for robotic systems to reproduce realistic human arm movements [5, 6] and antagonistic muscles [7]. Minimum Jerk Theory comes from the finding that the degree of smoothing of a curve can be quantified by a function counting the number of shocks performed [8]. Hogan named this function Jerk and associated it mathematically to the third time derivative of a given trajectory. Among researchers having recourse to the Minimum Jerk criterion, there are divided opinions between those using trigonometric curves and others using polynomial curves to describe the robot trajectory. Actually, very few research papers consider trigonometric functions to describe the Jerk function [9–11]. They note that all joints involved in the movement are less oscillatory. It seems that most works dealing with the Minimum Jerk criterion are based on polynomial trajectories. As proved by Amirabdollahian et al. [12], the use of polynomial trajectories has some advantages. The control of the movement is easily achievable since the first and second derivatives of the polynomial are known. Also, some studies [13] show that Minimum Jerk control laws based on polynomials of high degrees are more effective because the dynamics of the robot are smoother and the trajectory references are easily followed by the actuators involved. Finally, for applications using real-time control, the trajectories can be corrected or adjusted at any time by simply redefining the polynomials describing the trajectory or by the superposition of a new path to the previous one [14]. Another issue that has also divided opinions among researchers is the space on which reference trajectories must be planned: the Cartesian or the joint space. Actually, Kyriakopoulos and Saridis in [8] are the first to raise the issue of choice of planned trajectories in the Cartesian space or in the joint one. They show that if the minimization problem and its solutions are formulated in the joint space, only physical limitations of the joints actuators will

be included in the constraints statement. However, in a realistic environment, obstacles exist and are causing changes in the trajectory direction. Therefore, generating reference trajectories based on the Minimum Jerk criterion may be done whether in the joint or Cartesian space. The space's choice should only be determined according to the constraints and the shape of the desired trajectory. Finally, it can be noted that very few works using Minimum Jerk criterion were devoted to optimize humanoid motion through gait pattern generation [15–17].

On the other side, to control bipedal robots during the IP and DSP, we have chosen in this work to focus on impedance control strategy. Such control approach was originally proposed by Hogan in [18]. Its goal is to establish a dynamic relation between the end-effector position and the contact force [19]. There are two methodologies stemming from the impedance control law: the classical impedance [6] and dynamic impedance based control laws [20]. The first approach does not take into consideration the dynamic of the robotic system and includes the active stiffness control. In opposition, the dynamic impedance control is based on two assumptions: the consideration of the constraint dynamic model when an external force is applied and the environment characterization by three parameters: inertia, damping and active stiffness. Thus, the dynamic impedance control law represents an efficient control law to overcome difficulties raised in the impact phase. Moreover, it guarantees a stable and safe elastic contact with the ground as it takes into consideration environmental parameters related to the nature of the ground and the contact type. As a result, many research works have recourse to this control law to generate bipedal robots walking gaits as [20–24].

To produce stable, smooth and safe walking cycles for bipedal robots evolving in the 3D space, this chapter proposes the elaboration of appropriate control strategies to be implemented to biped robots. Indeed, during the swing phase, a Minimum Jerk based control law is produced in order to generate a semi-ellipsoidal trajectory for the swing foot while an appropriate impedance control law is proposed to ensure the robot stability and safe balance at foot landings on the ground. Minimum Jerk control is inspired by the human brain cognitive such that trajectories are planned in the Cartesian space system whereas controllers are expressed in the joint space. For the IP and DSP, the impedance control law is designed such that it ensures stable and safe impacts with the ground. Also, it allows the environment characterization through inertia, damping and active stiffness parameters inspired from the recent work [25] where sufficient conditions of stability and discussions about safety during the impacts are given.

This chapter is then organized as follows: in the next section, the novel approach of Jerk optimal control and the impedance control laws are designed. The stability conditions for the two control approaches are rigorously given. Section 3 is dedicated to the application of the proposed approach to a bipedal robot prototype. Indeed, the anthropomorphic model of the Humanoid robot is presented and kinematic and dynamic models of the lower body are developed. Simulations performed on the Humanoid robot do validate both designed control laws and show the generation of a satisfactory walking gait pattern even in presence of measurement noise and uncertainties on the environment stiffness.

## 2 Stable and Safe Gait Pattern

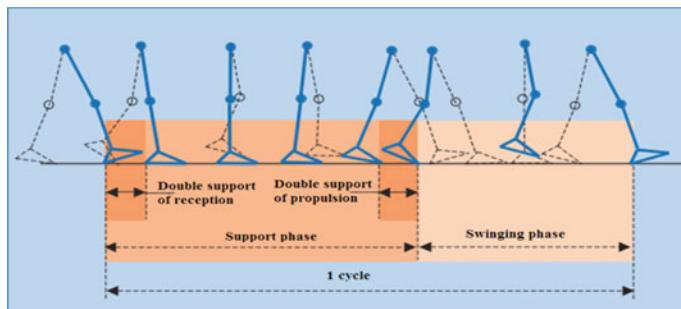
### 2.1 The Gait Cycle

A walking gait cycle involves various phases such as single support phase, impacts with the ground and the double support phase [26]. According to Kajita and his colleagues [2, 3], whether considering human or artificial gait, the walking function is based on an alternate displacement of the two legs with a support point permanently in contact with the ground. The leg that is totally in contact with the ground is called the supporting leg whereas the leg starting the foot step is the swinging leg also called free leg. For each lower limb, a human walking cycle is always composed of two phases (see Fig. 1) [27]:

- A support phase where the foot remains in contact with the ground. This stage starts at the first foot/ground contact and ends when the foot toe is completely off the ground. This phase represents 60 % of the whole walking cycle.
- A swinging phase where the foot is free, without any ground contact. This stage starts at the completion of the supporting phase and it ends when the second foot begins its own swinging phase. This phase generally corresponds to the 40 % left of the whole walking cycle.

For a usual walking gait, the lower limb playing the role of supporting leg ensures the three main functions of support, damping and propulsion while the swinging leg is being moved from rear to front [28]. Two distinct phases of double support are often considered:

- The double support of reception that takes place at the initial foot/ground contact of the previously free leg and is proceeding with the whole weight transfer of the other limb to the current leg.



**Fig. 1** Phases of a walking cycle

- The double support of propulsion occurs at the level of the previously supporting leg at the instant the foot is landing off the ground. This phase also corresponds to the weight transfer from this limb to the other one that becomes the new supporting leg.

Artificial walking aims at reproducing all phases composing a natural walking gait. However, walking bipedal robots are not able to provide a sustained rhythm of walk. Thus, a supplementary phase has to be added and considered. Indeed, a typical walking cycle includes three main stages [26, 29–31]: The single support phase (SSP), the impact phase (IP) and the double support phase (DSP).

The SSP occurs when one limb is pivoted to the ground while the other is swinging from the rear to the front. At the beginning of this stage, the heel of the forward foot is lifted with the toe used as a pivot. When a sufficient rotational motion is done, the foot is to be completely off the ground and swings in the air. The free dynamic model corresponding to the SSP is described by:

$$M(\theta)\ddot{\theta} + H(\theta, \dot{\theta}) + G(\theta) = D.U \quad (1)$$

where  $\theta, \dot{\theta}, \ddot{\theta} \in R^n$  are the joint position vector, the joint velocity vector and the joint acceleration vector of the bipedal robot, respectively.  $M(\theta) \in R^{n \times n}$  is the inertia matrix,  $H(\theta, \dot{\theta}) \in R^n$  is the vector of the Coriolis and centripetal forces  $G(\theta) \in R^n$  and is the gravity vector. The matrix  $D \in R^{n \times n}$  is a nonsingular input map matrix whereas  $U \in R^n$  is the control input vector.

The IP occurs when the toe of the forward foot starts touching the ground. The impact between the toe of the swing leg and the ground takes place during an infinitesimal length of time [32]. The DSP occurs when both limbs remain in contact with the ground. This phase begins with the heel of the forward foot touching the ground. Then the foot rotational motion continues until the entire sole of the foot becomes in contact with the ground. This stage finally ends with the toe of the rear foot taking off the ground. The length of this phase depends on the walking cycle's rhythm. The constrained dynamic model during the IP and DSP of the bipedal robot is generally described by [33]:

$$M(\theta)\ddot{\theta} + H(\theta, \dot{\theta}) + G(\theta) = DU + \frac{\partial c(\theta)^T}{\partial \theta} F \quad (2)$$

where  $C(\theta) \in R^3$  is the contact point and  $F \in R^3$  is the contact force with the ground. During the gait cycle, the supporting foot does not change its position and orientation, and the whole part of its sole is in contact with the ground. As soon as the third phase of the swing foot ends, the foot of the supporting leg goes into its own first stage of the swing motion.

## 2.2 Minimum Jerk Based Control for the Swing Phase

### 2.2.1 Minimum Jerk Control: Theoretical Foundations

According to [5], the Jerk is defined as the third time derivative of a given trajectory  $\beta(t)$  such that:

$$\ddot{\beta}(t) = \frac{d^3\beta(t)}{dt^3} \quad (3)$$

The main benefit of Minimum Jerk based control resides in the generation of smooth trajectories in order to avoid any abrupt motion and consequently limit the robot vibrations [4]. To find among all possible trajectories, the one that allows the achievement of the smoothest motion, a Jerk cost must be assigned. Thus, for a trajectory  $\beta(t)$  describing a particular path starting at and ending at  $t_f$ , the Minimum Jerk cost criterion is generally defined by [34]:

$$C_{Jerk} = \min \frac{1}{2} \int_{t=t_0}^{t_f} \ddot{\beta}(t)^2 dt \quad (4)$$

Even if not essential, some additional terms could be included in the criterion function to minimize a weighted sum of multiple criteria. In [35], for example, the objective function to be minimized is the integral of a weighted sum of squared jerk and the execution time. Among all possible solutions, the following fifth order polynomial trajectories seems to be the most recommended one in the literature [36]:

$$\beta(t) = at^5 + bt^4 + ct^3 + dt^2 + et + f \quad (5)$$

where  $a, b, c, d, e, f$  are constants to be determined for each trajectory  $\beta(t)$ . The corresponding velocity and acceleration functions are easily deduced as:

$$\dot{\beta}(t) = 5at^4 + 4bt^3 + 3ct^2 + 2dt + e \quad (6)$$

$$\ddot{\beta}(t) = 20at^3 + 12bt^2 + 6ct + 2d \quad (7)$$

To compute the parameters  $a, b, c, d, e$  and  $f$  two main methods are used in the literature: The Point-to-point method and the Via-point method. The Point-to-point method requires the expression of the function to be minimized and the values of positions, velocities and accelerations of only two points corresponding to the initial and final time of the movement. The control algorithm corresponding to the Point-to-point method only needs to run once. For each a trajectory  $\beta(t)$  the following relation is used to compute the parameters  $a, b, c, d, e$  and  $f$  [37]:

$$\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = \begin{pmatrix} t_0^5 & t_0^4 & t_0^3 & t_0^2 & t_0 & 1 \\ t_f^5 & t_f^4 & t_f^3 & t_f^2 & t_f & 1 \\ 5t_0^4 & 4t_0^3 & 3t_0^2 & 2t_0 & 1 & 0 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 & 2t_f & 1 & 0 \\ 20t_0^3 & 12t_0^2 & 6t_0 & 2 & 0 & 0 \\ 20t_f^3 & 12t_f^2 & 6t_f & 2 & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \beta_0 \\ \dot{\beta}_f \\ \ddot{\beta}_0 \\ \dot{\beta}_f \\ \ddot{\beta}_0 \\ \ddot{\beta}_f \end{pmatrix} \quad (8)$$

where:

$\beta_0$ ,  $\dot{\beta}_f$  and  $\ddot{\beta}_0$  are respectively the position, velocity and acceleration of the trajectory  $\beta(t)$  at  $t_0$ .

$\beta_f$ ,  $\dot{\beta}_f$  and  $\ddot{\beta}_f$  are respectively the position, velocity and acceleration of the trajectory  $\beta(t)$  at  $t_f$ .

For the Via-point method, such approach is recommended when obstacles occur in the operating space where the robotic system evolves. In such cases, not only the initial and final positions must be specified but also a number of desired intermediate positions characterized by the time at which these positions must be reached. Therefore, the number of intermediate points determines the accuracy of the reference trajectory. This method implies that the algorithm is executed several times. If only one desired intermediate position is specified, the parameters  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$  and  $f$  are computed as follows [38]:

$$\begin{pmatrix} 5t_0^4 & 4t_0^3 & 3t_0^2 & 2t_0 & 1 & 0 \\ 5t_d^4 & 4t_d^3 & 3t_d^2 & 2t_d & 1 & 0 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 & 2t_f & 1 & 0 \\ 20t_0^3 & 12t_0^2 & 6t_0 & 2 & 0 & 0 \\ 20t_d^3 & 12t_d^2 & 6t_d & 2 & 0 & 0 \\ 20t_f^3 & 12t_f^2 & 6t_f & 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_d \\ \beta_f \\ \dot{\beta}_0 \\ \dot{\beta}_d \\ \dot{\beta}_f \end{pmatrix} \quad (9)$$

where:

- $t_d$  is an intermediate instant satisfying  $t_0 < t_d < t_f$ .
- $\beta_d$ ,  $\dot{\beta}_d$  and  $\ddot{\beta}_d$  are respectively the position, velocity and acceleration of the variable  $\beta_i(t)$  at  $t_d$ .

### 2.2.2 A Novel Approach of a Jerk Optimal Control

To optimize the Jerk for gait pattern generation in the three dimensional space during the swing phase, we propose in this section a new optimal jerk approach different from the Point-to-Point and Via-Point methods. Indeed, the Point-to-Point method implies constraints on positions, velocities and accelerations on boundary conditions

whereas the Via-Point method imposes the same boundary constraints in addition to other constraints related to some intermediate desired positions at the instants at which these specific positions have to be reached.

Actually, the proposed approach is based on the generation of reference trajectories specifying at each time iteration not only constraints at boundary conditions or intermediate points but also constraints on the current positions, velocities and accelerations. Furthermore, the proposed method induces three-dimensional reference trajectories involving both trigonometric and polynomial functions. This particular choice provides at the same time benefits of trigonometric functions requiring fewer resources for real time implementation and benefits of polynomial functions giving smoother dynamics and fewer vibrations. Finally, in order to realize an efficient control law, easy to implement, trajectories will be planned in the Cartesian space while the control law is depending on angular variables. Such control design is very close to the human brain cognitive [39].

Considering the swing foot of the bipedal robot, the differential kinematic model of the bipedal robot is given by:

$$\dot{X}_f(t) = J(\theta)\dot{\theta}(t) \quad (10)$$

where  $\dot{X}_f \in R^3$  is the Cartesian velocity vector for the swing foot and  $J(\theta) \in R^{3 \times n}$  is the Jacobian matrix defined by:

$$J(\theta) = \left. \frac{\partial X_f(\theta)}{\partial \theta} \right|_{\theta_d} \quad (11)$$

where  $\theta_d \in R^n$  is the desired joint position vector. The inverse kinematic model is then given by:

$$\theta(t) = J(\theta)^+ (X_f(t) - X_{f,d}(t)) + \theta_d(t) \quad (12)$$

where  $J(\theta)^+$  is the pseudo-inverse of the Jacobian matrix.  $X_{f,d} \in R^3$  is the desired Cartesian position vector for the swing foot. The joint velocity vector  $\dot{\theta}(t)$  is obtained using the following equation:

$$\dot{\theta}(t) = J(\theta)^+ \dot{X}_f(t) \quad (13)$$

where  $\dot{X}_f$  is the Cartesian velocity. To get the joint acceleration vector  $\ddot{\theta}(t)$ , one just has to determine the first time derivative of the previous equation so that:

$$\ddot{\theta}(t) = J(\theta)^+ (\ddot{X}_f(t) - \dot{J}(\theta)\dot{\theta}(t)) \quad (14)$$

where  $\dot{J}(\theta)$  is the first time derivative of the Jacobian matrix  $J(\theta)$ .  $\ddot{X}_f$  is the Cartesian acceleration. In order to produce a walking gait closed to a human one, the toe of the biped robot has to follow a path similar to the one generated by the human foot when performing a walking step. To reach this goal, we impose to the toe

end effector a semi-elliptical trajectory in the sagittal plane such that the three-dimensional Cartesian desired trajectory is described by:

$$\dot{X}_{f,d} = [g + v \cos(\alpha \beta(t) + \pi) \quad w \quad h + y \sin(\alpha \beta(t) + \pi)]^T \quad (15)$$

where, in this case,  $\beta(t) \in R$  designs the ankle joint trajectory in the sagittal plane defined by the five order polynomial (5) and where its corresponding velocity and acceleration functions are defined by (6) and (7), respectively. Furthermore, the different constant coefficients of the polynomial function (5) are computed using the relation (8) related to the Point-to-point approach. The pair  $(g, h)$  represents the initial coordinates of the center of the ellipse in the sagittal plane. Parameter  $v$  represents a half step length whereas  $w$  defines the distance between the two legs and  $y$  represents the maximum height of the step. Parameter  $\alpha$  is a multiplier used in order to increase the variation range of the variable  $\beta(t)$ . The first and second derivatives of (15) with respect to time are respectively given by the two following expressions:

$$\dot{X}_{f,d} = [-v\alpha \dot{\beta}(t) \sin(\alpha \beta(t) + \pi) \quad 0 \quad y\alpha \dot{\beta}(t) \cos(\alpha \beta(t) + \pi)]^T \quad (16)$$

$$\ddot{X}_{f,d} = [\ddot{x}_{f,d} \quad \ddot{y}_{f,d} \quad \ddot{z}_{f,d}]^T \quad (17)$$

where:

$$\begin{cases} \ddot{x}_{f,d} = -v (\alpha \dot{\beta}(t))^2 \cos(\alpha \beta(t) + \pi) - v \alpha \ddot{\beta}(t) \sin(\alpha \beta(t) + \pi) \\ \ddot{y}_{f,d} = 0 \\ \ddot{z}_{f,d} = -y (\alpha \dot{\beta}(t))^2 \sin(\alpha \beta(t) + \pi) + y \alpha \ddot{\beta}(t) \cos(\alpha \beta(t) + \pi) \end{cases}$$

During the SSP, the Minimum Jerk criterion is applied to reduce abrupt displacements of the bipedal robot which is controlled via a linearizing control law. Actually, we impose to the bipedal robotic model (1) to follow the following second order linear input-output behavior [40]:

$$(\ddot{\theta}(t) - \ddot{\theta}_d(t)) + K_{v,1} (\dot{\theta}(t) - \dot{\theta}_d(t)) + K_{p,1} (\theta(t) - \theta_d(t)) = 0$$

where  $K_{v,1}$  and  $K_{p,1} \in R^{n \times n}$  are two positive definite diagonal matrices computed offline to ensure global stability, decoupling properties and desired performances [41]. If  $\lambda$  is the desired bandwidth, then to obtain a critically damped closed-loop performance, we must select [16]:

$$\begin{aligned} K_{v,1} &= \text{diag}[2 \lambda] \\ K_{p,1} &= \text{diag}[\lambda^2] \end{aligned} \quad (18)$$

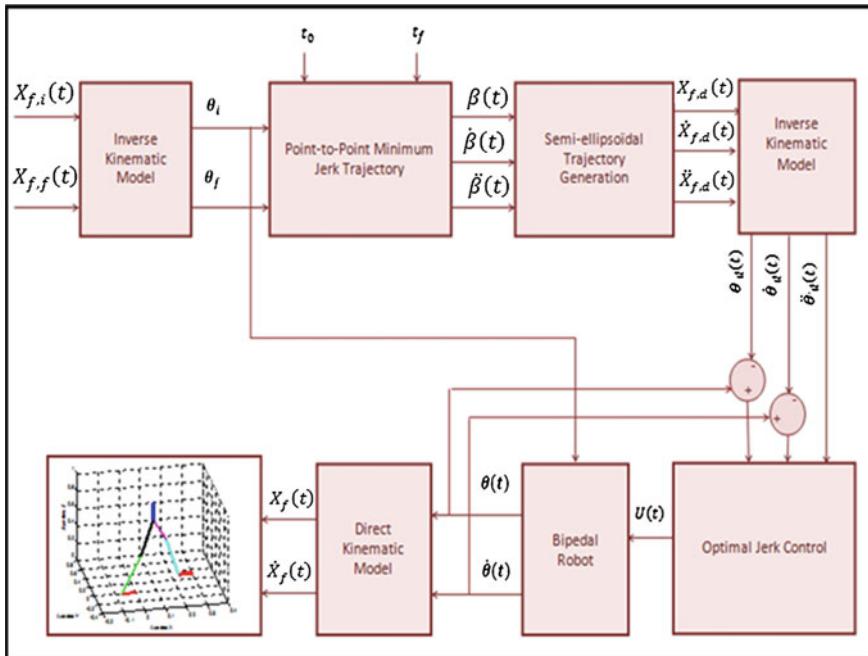
Using (1) and (18), the control law of the bipedal robot when achieving the swing phase is deduced as:

$$U(t) = D^{-1} \left( M(\theta) \left[ \ddot{\theta}_d(t) - K_{v,1} (\dot{\theta}(t) - \dot{\theta}_d(t)) - K_{p,1} (\theta(t) - \theta_d(t)) \right] + H(\theta, \dot{\theta}) + G(\theta) \right) \quad (19)$$

Figure 2 explains all required steps for the achievement of the novel Jerk optimal control approach.

The calculus of a control law based on the proposed approach of the Jerk optimal control includes the following steps:

- i. Computing the initial joint position vector  $\theta_i$  and the final joint position vector  $\theta_f$  using the inverse kinematic model (12).
- ii. Generation of  $\beta(t)$ ,  $\dot{\beta}(t)$ ,  $\ddot{\beta}(t)$  trajectories based on the Point-to-Point method according to Eqs. (5)–(7) applied to the initial and final joint constraints and taking also account of time data  $t_0$  and  $t_f$ .
- iii. Generation of semi-ellipsoidal reference trajectories in the Cartesian space by computing at each time iteration the reference Cartesian positions, velocities and accelerations  $X_{f,d}(t)$ ,  $\dot{X}_{f,d}(t)$  and  $\ddot{X}_{f,d}(t)$  according to Eqs. (15)–(17), respectively.



**Fig. 2** The novel approach of Minimum Jerk based control during the swing phase

- iv. Generation of the desired joint trajectories  $\theta_d(t)$ ,  $\dot{\theta}_d(t)$  and  $\ddot{\theta}_d(t)$  using (12)–(14).
- v. Computing the *Jerk* optimal control law  $U(t)$  using (19).
- vi. Implementation of the control law  $U(t)$  to the free robotic system (1) in order to generate joint position and velocity vectors  $\theta(t)$  and  $\dot{\theta}(t)$  using (13) and (14).  $\theta(t)$  and  $\dot{\theta}(t)$  are supposed to be measured via online sensors.
- vii. Generation of Cartesian trajectories  $X_f(t)$  and  $\dot{X}_f(t)$  by applying the direct kinematic model (10).

### 2.3 Impedance Based Control for the Constrained Phases

To control the bipedal robot during the impact and double support stages, we have chosen to implement the impedance based control proposed in [25]. Indeed, the impedance control represents an efficient control law to overcome difficulties raised in the impact phase. Moreover, it guarantees a stable and safe elastic contact with the ground as it takes into consideration environmental parameters related to the nature of the ground and the contact type. Regarding the control law implementation to the bipedal robot, it is simple since the same control expression is used in IP and DSP. During the ground contact, the free end of the biped, at the completion of the step, comes into contact with the ground. This phase is assumed to take place in an infinitesimal time interval [32]. For the constrained model (2), the ground reaction force is given by [25]:

$$F = F_d - K_d(X_{f,d} - X_f) - B_d(\dot{X}_{f,d} - \dot{X}_f) - M_d(\ddot{X}_{f,d} - \ddot{X}_f) \quad (20)$$

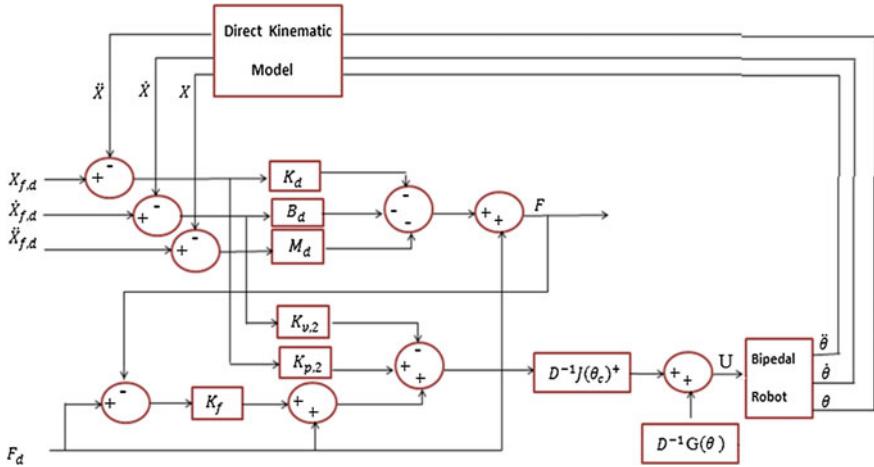
where  $F_d \in R^3$  is a desired reaction force and  $K_d$ ,  $B_d$ ,  $M_d \in R^{3 \times 3}$  are the stiffness, the damping and the inertia matrices, respectively. These three parameters characterize the contact type and the environment nature where the contact takes place, under the following control law:

$$\begin{aligned} U = D^{-1} J(\theta_c)^+ &[K_{p,2}(X_{f,d} - X_f) - K_{v,2}(\dot{X}_{f,d} - \dot{X}_f) + K_f(F_d - F) + F_d] \\ &+ D^{-1}G(\theta) \end{aligned} \quad (21)$$

where  $J(\theta_c)^+$  is the pseudo-inverse of the Jacobian matrix of the bipedal robotic system at the contact point  $c$  and where  $K_{p,2}$ ,  $K_{v,2}$  and  $K_f \in R^{3 \times 3}$  are diagonal matrices representing respectively the position, velocity and force gains related to the impedance control law. Using a Lyapunov approach, the asymptotic stability of the robotic system (2) is guaranteed if the following stability conditions are satisfied [25]:

$$\begin{aligned} K_{p,2} &> 0 \\ K_{v,2} &> 0 \\ k_f &= -I \end{aligned} \quad (22)$$

where  $I \in R^{3 \times 3}$  is the identity matrix.



**Fig. 3** The impedance based control implemented during the constrained phases

Figure 3 shows all involved parameters in the impedance control implementation. Parameters  $K_{p,2}$ ,  $K_{v,2}$  and  $K_f$  are computed offline to satisfy the Lyapunov asymptotic conditions (22). However and even if a trajectory inside a sagittal plane is imposed, some step parameters like the reference Cartesian positions, velocities and accelerations  $X_{f,d}(t)$ ,  $\dot{X}_{f,d}(t)$  and  $\ddot{X}_{f,d}(t)$  and the desired joint trajectories  $\theta_d(t)$ ,  $\dot{\theta}_d(t)$  and  $\ddot{\theta}_d(t)$  are calculated online at each time iteration. Online sensors are used to measure real joint trajectories  $\theta(t)$ ,  $\dot{\theta}(t)$  and  $\ddot{\theta}(t)$  and the contact forces with the ground  $F$ .

### 3 Application to a Humanoid Robot Prototype

In this section the control laws proposed in the last section will be applied to a Humanoid robot prototype which a particular morphology very close to a human one [41].

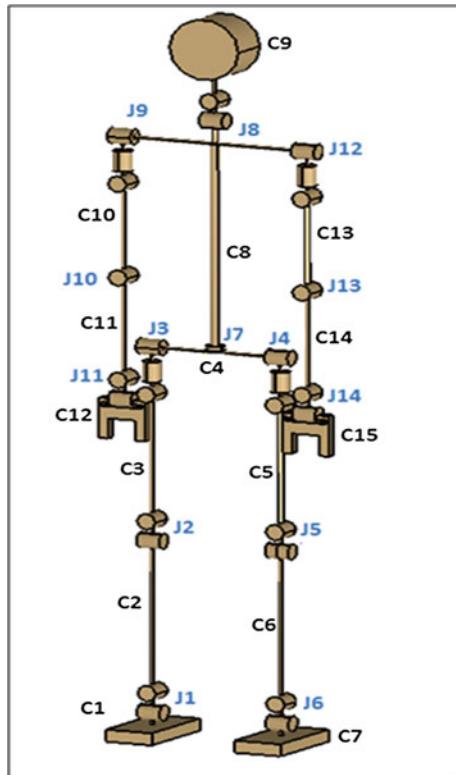
#### 3.1 The Anthropomorphic Model of the Humanoid Robot

The Humanoid robot prototype [41] considered in this paper is composed of fifteen links associated to twenty-six degrees of freedom (DOF). The morphological constitution of the humanoid robot corresponds to a human being's whole anatomy with a weight of 70 kg and a height of 1.73 m. Furthermore, we take account on the following assumptions:

- Assumption 1:** The connection between upper and lower bodies is made by a passive joint. Indeed, trunk and pelvis rigid bodies are assumed not to be subject to rotations however their mere presence allows the rigid bodies to move in a correct way. Hence, the upper body's mass is considered and taken into account during the gait calculation and simulations of the lower body.
- Assumption 2:** The vectors of joint position  $\theta$ , joint velocity  $\dot{\theta}$ , joint acceleration  $\ddot{\theta}$  and contact forces with the ground  $F$  are measured via online sensors.
- Assumption 3:** During the gait cycle, the humanoid robot is supposed to evolve in a well known environment where no obstacles are encountered. Therefore, no adaptive controls are required.
- Assumption 4:** The ground contact surface is assumed to be smooth and regular.

Figure 4 and Table 1 show the involved rotations for each link. The whole humanoid robot is composed of two independent robotic systems: the upper body and the lower body.

**Fig. 4** The Humanoid robot prototype



**Table 1** Rigid bodies and articulations

Link	Link description	Joint	Joint description	Degrees of freedom
C1	Right foot	J1	Right ankle	$\xi_1 = [0 \ \theta_1 \ \theta_2]$
C2	Right leg	J2	Right knee	$\xi_2 = [0 \ \theta_1 \ \theta_3]$
C3	Right thigh	J3	Right hip	$\xi_3 = [\theta_4 \ \theta_5 \ \theta_6]$
C4	Pelvis	J7	Passive joint	$\xi_4 = [0 \ 0 \ 0]$
C5	Left thigh	J4	Left hip	$\xi_5 = [\theta_7 \ \theta_8 \ \theta_9]$
C6	Left leg	J5	Left knee	$\xi_6 = [0 \ \theta_{10} \ \theta_{11}]$
C7	Left foot	J6	Left ankle	$\xi_7 = [0 \ \theta_{10} \ \theta_{12}]$
C8	Trunk	J7	Passive joint	$\xi_8 = [0 \ 0 \ 0]$
C9	Head and neck	J8	Neck	$\xi_9 = [0 \ \theta_{13} \ \theta_{14}]$
C10	Right arm	J9	Right shoulder	$\xi_{10} = [\theta_{15} \ \theta_{16} \ \theta_{17}]$
C11	Right forearm	J10	Right elbow	$\xi_{11} = [0 \ \theta_{18} \ 0]$
C12	Right hand	J11	Right wrist	$\xi_{12} = [0 \ \theta_{19} \ \theta_{20}]$
C13	Left arm	J12	Left shoulder	$\xi_{13} = [\theta_{21} \ \theta_{22} \ \theta_{23}]$
C14	Left forearm	J13	Left elbow	$\xi_{14} = [0 \ \theta_{24} \ 0]$
C15	Left hand	J14	Left wrist	$\xi_{15} = [0 \ \theta_{25} \ \theta_{26}]$

Winter statistical model [42] was used to determine all physical parameters corresponding to each link  $C_i$ . Each rigid body  $C_i$  of the humanoid robot is characterized by the following physical parameters (see Fig. 5):

- $k_i \in \Re$ : Proximal distance defined as the distance from the center of gravity to the connect joint of the previous link  $C_{i-1}$ .
- $l_i \in \Re$ : Distal distance defined as the distance from the center of gravity to the connect joint of the next link  $C_{i+1}$ .

Since the kinematic model is elaborated in the three dimensional space, we define  $K_i \in \Re^{3 \times 1}$  and  $L_i \in \Re^{3 \times 1}$  as respectively the proximal and distal distance vectors of the link  $C_i$  given by:

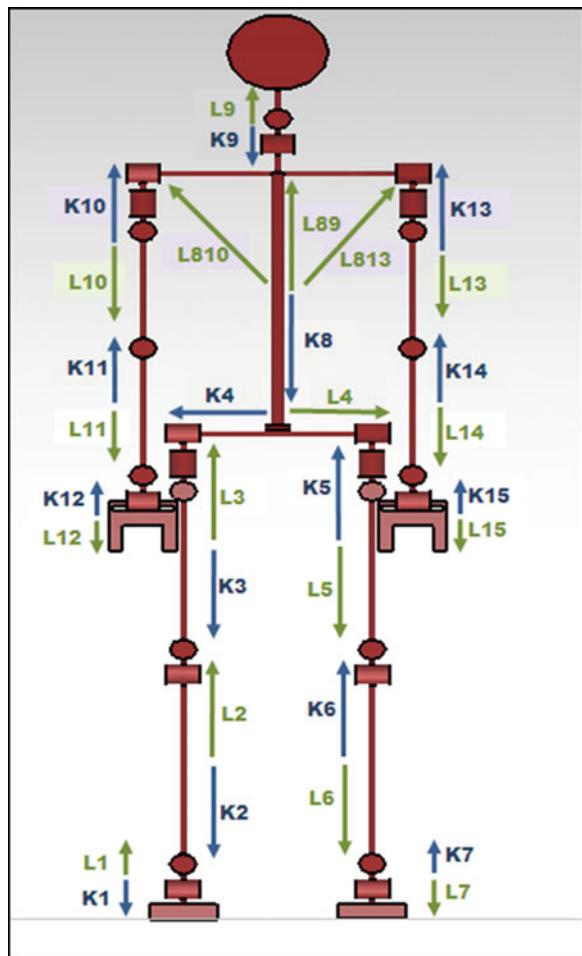
$$K_i = [0 \ 0 \ K_i]^T \quad \text{and} \quad L_i = [0 \ 0 \ l_i]^T$$

Each rigid body  $C_i$  of the bipedal robot is characterized by the previous physical parameters plus the following physical parameters:

- $m_i \in \Re$ : Mass of the link  $C_i$
- $i_i \in \Re$ : Inertia about the center of mass of the link  $C_i$

Since the dynamic model is elaborated in the three dimensional space, the following three dimensional parameters are considered:

**Fig. 5** Segmental proportions of rigid bodies



–  $M_i \in \Re^{3 \times 3}$ : mass matrix of the link  $C_i$  given by:

$$M_i = \begin{pmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{pmatrix}$$

–  $I_i \in \Re^{3 \times 3}$ : Inertia matrix about the center of mass of link  $C_i$  described by:

$$I_i = \begin{pmatrix} i_{ix} & 0 & 0 \\ 0 & i_{iy} & 0 \\ 0 & 0 & i_{iz} \end{pmatrix}$$

Table 2 gives all segmental physical parameters.

**Table 2** Physical parameters obtained via winter model

Link	$k_i$ (m)	$l_i$ (m)	$m_i$ (kg)	Inertia about center of mass ( $\text{kg m}^{-2}$ )		
				$i_{ix}$	$i_{ix}$	$i_{ix}$
Right foot	0.034	0.034	1.015	0.001	0.001	0.001
Right leg	0.184	0.241	3.255	0.051	0.051	0.051
Right thigh	0.184	0.240	7.000	0.113	0.113	0.113
Pelvis	0.021	0.178	9.940	0.112	0.112	0.112
Left thigh	0.240	0.184	7.000	0.113	0.113	0.113
Left leg	0.241	0.184	3.255	0.051	0.051	0.051
Left foot	0.034	0.034	1.015	0.001	0.001	0.001

### 3.2 Kinematic and Dynamic Modelling of the Bipedal Robot

All along this chapter, we focus only on the bipedal part of the humanoid robot presented in the last section. The direct and inverse kinematics of the humanoid robot are obtained using Euler's transformation principle [43]. Let  $X = [X_1, \dots, X_7]^T$  be the vector of Cartesian positions for links center of gravity and let  $A_i$  be the transformation matrix of link  $C_i$  from the body coordinate system to the inertial coordinate one. The robot implicit kinematic model is then described by:

$$X_i = A_i(L_i - K_i) + X_{i-1}, \quad i = 1, \dots, 15 \quad (23)$$

where  $X_i$  is the vector of Cartesian positions of the link  $C_i$ . An explicit expression of the bipedal robot kinematic model is given here under:

$$\begin{aligned} X_7 &= A_7(L_7 - K_7) + A_6(L_6 - K_6) + A_5(L_5 - K_5) + A_4(L_4 - K_4) \\ &\quad + A_3(L_3 - K_3) + A_2(L_2 - K_2) + A_1(L_1 - K_1) \end{aligned}$$

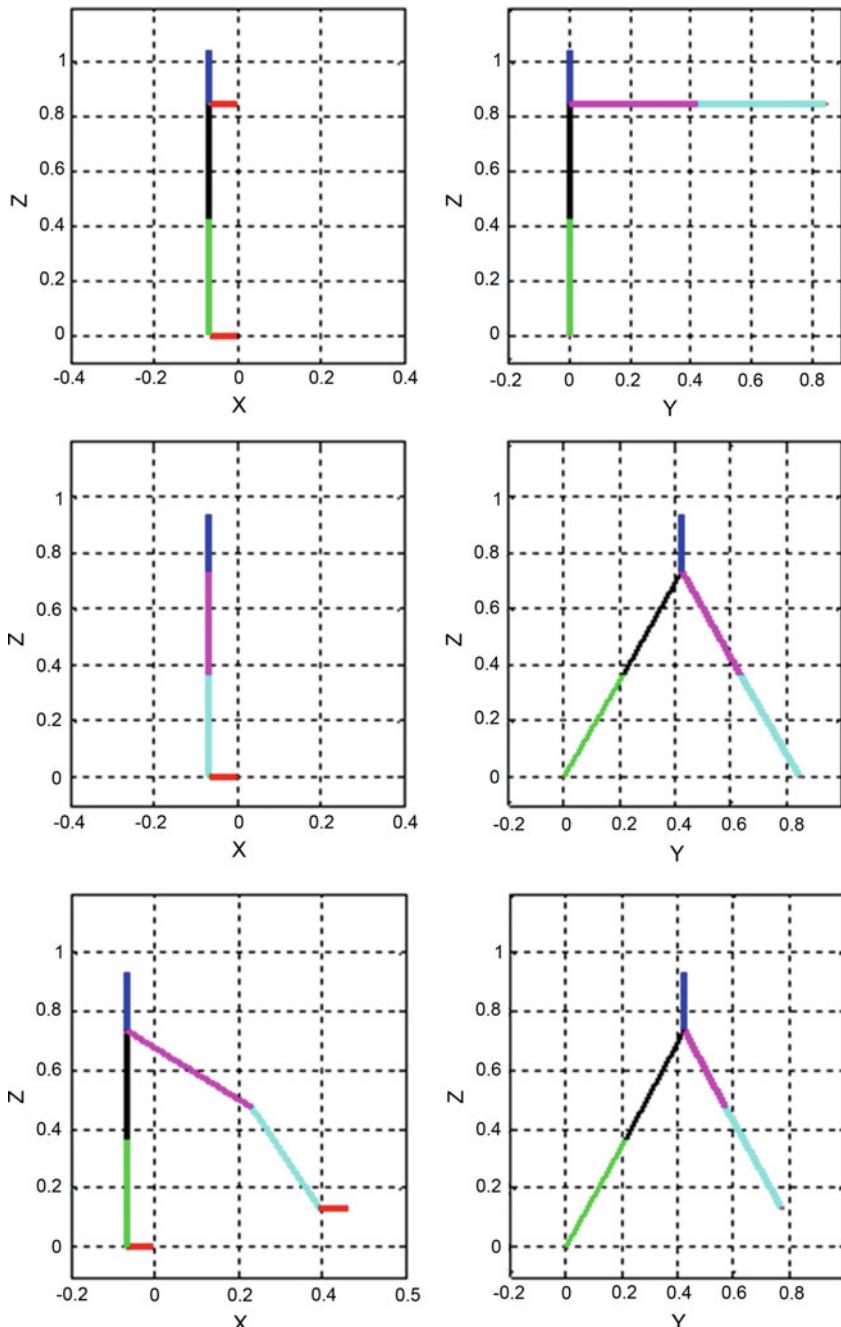
We consider at this level a number of standard scenarios [44–46] to validate the lower body kinematic model. Figure 6 shows validation of standard scenarios for the Humanoid robot lower body. The different postures are represented in both the sagittal and frontal planes.

The three dimensional dynamic modeling of the seven linked bipedal robot has been accurately developed using the Newton-Euler formalism [47] for the SSP, the IP and DSP.

In order to reach the dynamic model we use Hemami's works [29, 43] and we suggest a generalized motion equation for the translation as in (24) and the rotation as in (25) of each link  $C_i$ :

$$M_i \ddot{X}_i = M_i g + \Gamma_i - \Gamma_{i+1}. \quad (24)$$

$$I_i \dot{\omega}_i = f_i + F_i + F_{i+1} + G_i + G_{i+1} + \tau_i + \tau_{i+1} \quad (25)$$



**Fig. 6** Standard scenarios for the Humanoid robot lower body

where:

$W_i, \dot{W}_i$ : Angular position and acceleration of the link  $C_i$ .

$X_i, \dot{X}_i$ : Linear position and acceleration of the link  $C_i$

$F_i, \dot{F}_{i+1}$ : Torques due to the holonom force applied respectively to the proximal and distal articulation of the link  $C_i$  expressed in the body coordinate system

$G_i, \dot{G}_{i+1}$ : Non-holonom torques applied respectively to the proximal and distal articulation of the link  $C_i$  expressed in the body coordinate system

$\tau_i, \dot{\tau}_{i+1}$ : Muscular torques applied respectively to the proximal and distal articulation of the link  $C_i$  expressed in the body coordinate system

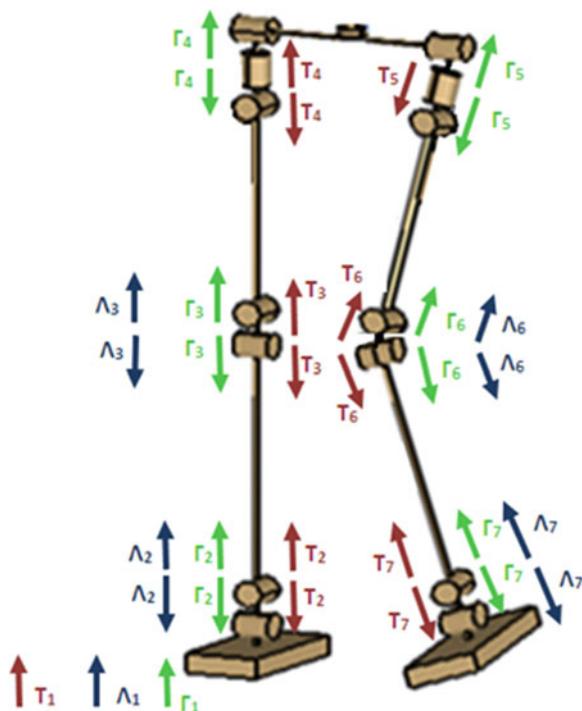
$\Gamma_i, \dot{\Gamma}_{i+1}$ : Holonom forces applied respectively to the proximal and distal articulation of the link  $C_i$  expressed in the inertial coordinate system

$f_i$ : Intrinsic torque of the link  $C_i$  expressed in the body coordinates system ( $x_i, y_i, z_i$ ) and relating angular velocity to the link inertia.

Human body's balance of forces and torques reveals that humanoid limbs are subject to three kinds of forces: holonom, non holonom and mechanic forces [43]. Figure 7 shows the applied forces and torques to the humanoid lower body.

A clear and sequential methodology to follow in order to establish a reduced and expendable dynamic model is rigorously explained in [47]. Using this method, the dynamic model of the Humanoid robot lower body has been reduced such moving from 42 initial DOFs to only 12 state variables.

**Fig. 7** Applied forces and torques to the Humanoid robot lower body



### 3.3 Jerk Optimal Control

Obviously initial conditions have a great influence on the bipedal robot's trajectory and equilibrium as the robotic system has an inherent nonlinear and very complex dynamical model due to the high number of degrees of freedom involved. To the best of our knowledge, there is no specific methodology that could be used to obtain optimal initial conditions. Therefore, based on previous standard validation scenarios, initial angular conditions are inspired on the one hand by the usual and common posture of a Human being when starting a walking step with a swinging left foot and on the other hand by very intensive simulations. The combination of angular corresponding positions is validated with Fig. 8 and is given by:

$$\theta_i = \left[ -\frac{\pi}{11} \ 0 \ 0 \ 0 \ -\frac{\pi}{9} \ 0 \ 0 \ \frac{\pi}{9} \ 0 \ \frac{\pi}{11} \ -0.0089 \ 0 \right]^T$$

Indeed, Fig. 8 shows the bipedal robot initial posture in the 3D space.

Using direct kinematic modelling given, initial Cartesian conditions are given by:

$$X_{f,i} = [0 \ 0.53 \ 0]^T$$

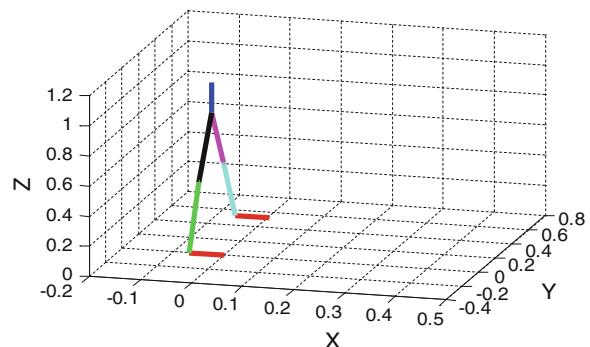
A semi-elliptical trajectory for a walking step of 0.5 s duration is generated using the parameters given in Table 3. It corresponds to a velocity of  $0.6 \text{ m s}^{-1}$  which is an acceptable Cartesian velocity when compared to current state of the art gait velocities rates for walking Humanoid robots [48, 49].

The fifth order angular trajectory is given then by:

$$\beta(t) = -3.18t^5 + 6.36t^4 - 3.39t^3 - 0.0089$$

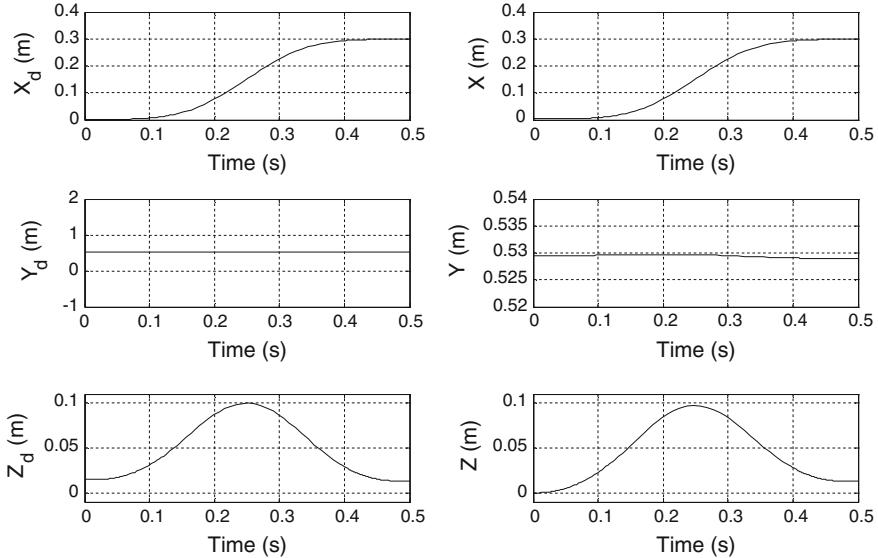
Position and velocity gains related to the Minimum Jerk control law are chosen so that global stability conditions (18) are satisfied for a desired bandwidth  $\lambda = 12$  such that:

**Fig. 8** Initial posture for the bipedal robot



**Table 3** Parameters of the Cartesian desired trajectory

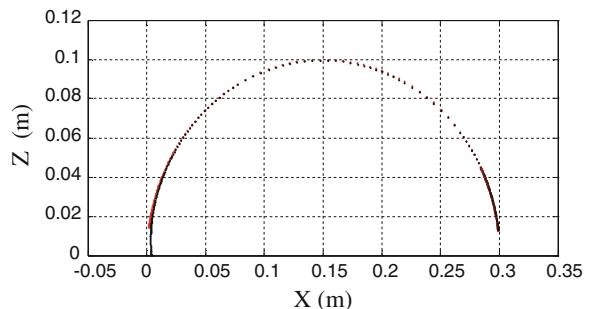
v (m)	w (m)	y (m)	$\alpha$ (m)	$(g, h)$ (m)
0.15	0.53	0.1	16.5	(0.15, 0)

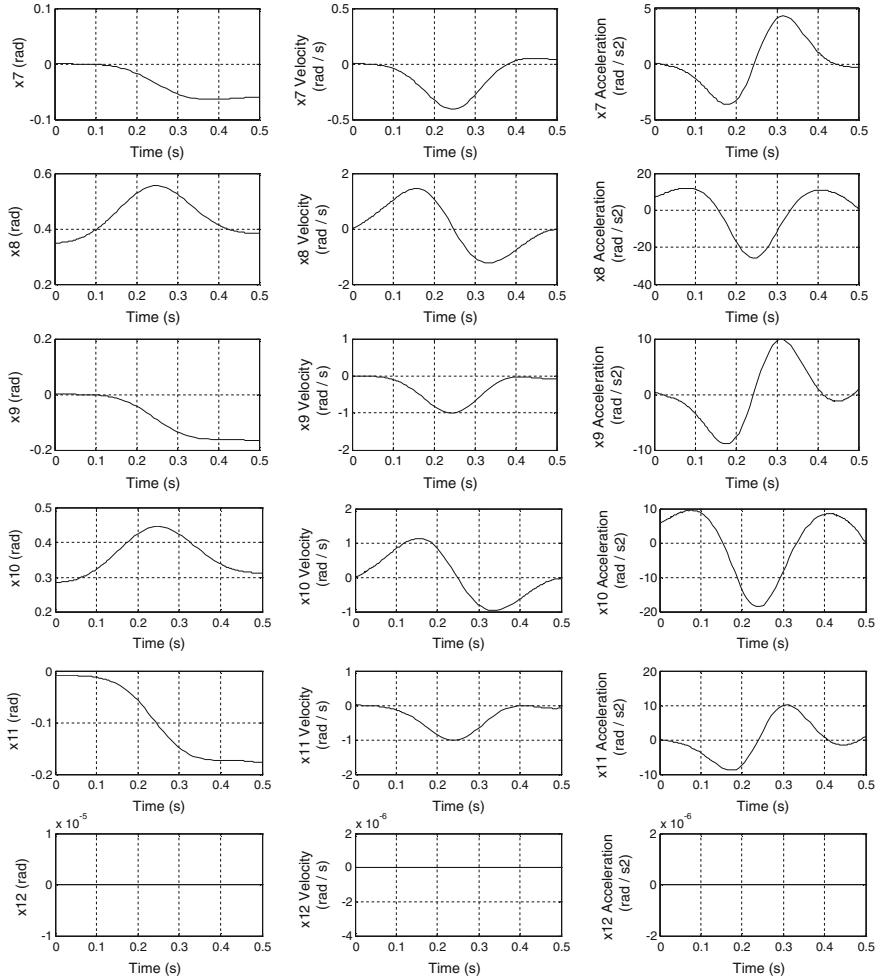
**Fig. 9** Desired and real Cartesian positions of the swing foot for a walking step

$$K_{p,1} = \text{diag}[24 \quad 24 \quad 24]$$

$$K_{v,1} = \text{diag}[144 \quad 144 \quad 144]$$

Figure 9 shows the evolution of the desired and real positions of the swing foot in the 3D Cartesian space while Fig. 10 represents the real Cartesian trajectory of the swing foot during the realization of a walking step.

**Fig. 10** Real Cartesian trajectory of the swing foot during a walking step

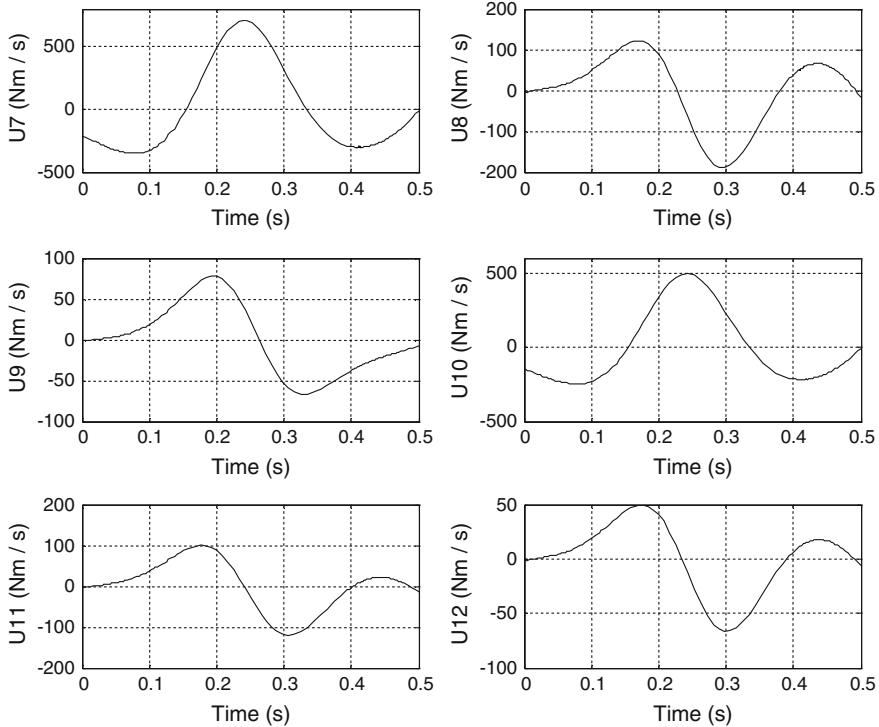


**Fig. 11** Evolution of the joint positions, velocities and accelerations of the swing foot during the swing phase

Figure 11 shows the evolution in time of the position, velocity and acceleration of the swing foot joints. All angular position reach after the step duration their desired values, this is emphasized by the very low values of velocities and accelerations at the step completion.

Figure 12 emphasizes the control laws involved in the swing foot motion for a walking step of 0.5 s duration.

Simulation results show the efficiency of the Minimum jerk control. Minimum Jerk observed benefits are the smoothness of the foot trajectory and the absence of sudden movements.



**Fig. 12** Evolution of the control laws of the swing foot during the swing phase

### 3.4 Impedance Based Control

Simulation of the bipedal robot during the IP and DSP is established using the constrained model (2) for the control law (21) and the ground reaction force (20). Environmental parameters used to describe the contact environment are inspired by the Park's research work [50] as:

$$\begin{aligned} K_d &= \text{diag}[10^4 \quad 10^4 \quad 10^4] \\ B_d &= \text{diag}[630 \quad 630 \quad 630] \\ M_d &= \text{diag}[10^{-2} \quad 10^{-2} \quad 10^{-2}] \end{aligned}$$

Regarding the reference ground reaction force, its role is to allow the free leg located at a height of 0.01 m from the ground to land on the contact area and at the same time to enable a low foot sliding along the  $x$ -axis. The resulting ground reaction force is composed of a vertical component and a normal one. According to [21], the vertical component of the reference external force has to emphasize the weight transfer of the bipedal robot from the right supporting foot to the left free foot.

The normal component of the reference ground reaction force is usually a low value but high enough to enable a short translation of the left foot along the  $x$ -axis. Thus,  $F_d$  is chosen as follows:

$$F_d = [30 \ 0 \ 300]^T$$

Position, velocity and force gains related to the impedance control law are computed offline so that the Lyapunov asymptotic stability conditions (22) are satisfied:

$$\begin{aligned} K_{p,2} &= \text{diag} [10^4 \ 10^4 \ 10^4] \\ K_{v,2} &= \text{diag} [600 \ 600 \ 600] \\ K_f &= -\text{diag} [1 \ 1 \ 1] \end{aligned}$$

Actually, the final position reached by the swinging foot while applying a Minimum Jerk control law during the SSP represents the initial position of the robotic system when starting the IP. The duration of the free phase is 0.5 s and the constrained phases' duration is 0.2 s. Consequently, the swinging foot initial Cartesian position for the impact and double support phases is given by:

$$X_{f,i} = [0.3 \ 0.53 \ 0.01]^T$$

The final desired Cartesian position of the active foot at the step completion is:

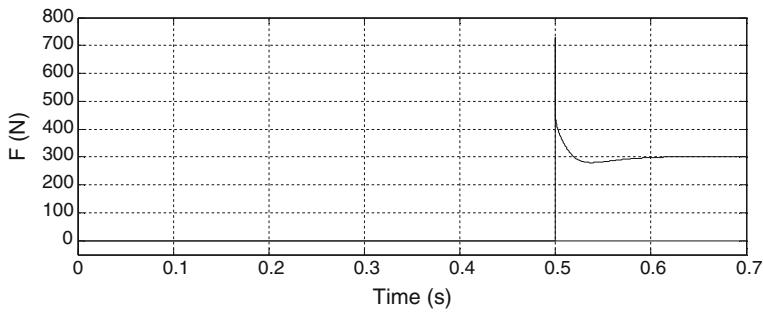
$$X_{f,f} = [0.35 \ 0.53 \ 0]^T$$

To explore the relevance of the proposed approach, we consider two case studies for simulation.

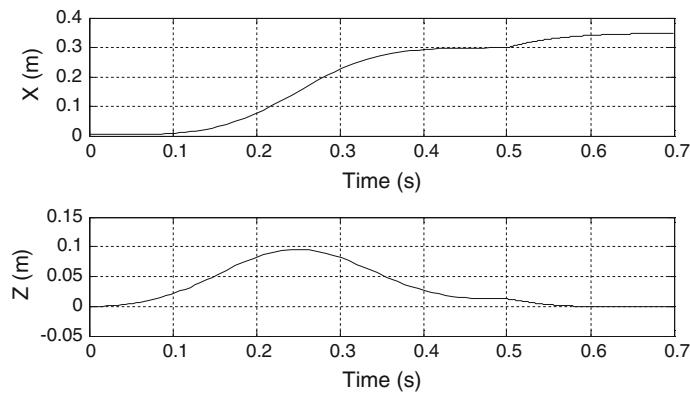
### 3.4.1 Case Study 1: Smooth and Regular Ground Surface and No Sensory Noise

This case study validates the proposed methodology of Minimum Jerk control for gait pattern generation under the stated assumptions. The Humanoid robot is supposed then to evolve in a smooth and regular ground contact surface. Furthermore, no sensory noise occurs here. Simulation results show the evolution of the active leg variables when the robot end-effector achieves a complete walking step. The ground reaction force is shown in Fig. 13. The active foot Cartesian components along  $x$  and  $z$  axis are represented in Fig. 14 while the real Cartesian trajectory of the humanoid robot free foot is shown in Fig. 15.

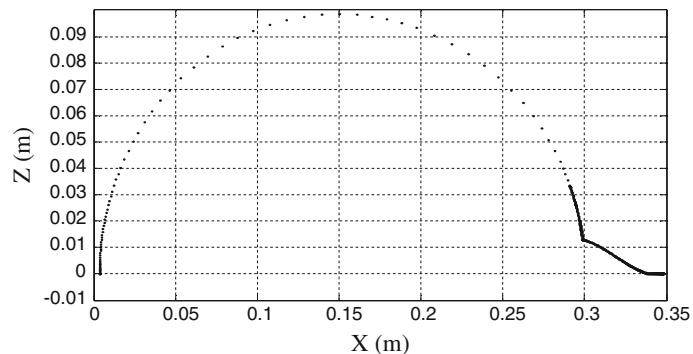
During the SSP, the ground reaction force is null. Then at the impact, it takes the shape of a delta impulse such reaching a maximal value of 750 N. Finally after the impact, the ground reaction force gradually decreases and gets stabilized at the value of 300 N.



**Fig. 13** Evolution of the ground reaction force during a walking step



**Fig. 14** Real Cartesian components of the active foot during a walking step



**Fig. 15** Real Cartesian trajectory of the active foot during a walking

Figure 14 emphasizes the real Cartesian positions evolution for the swinging foot during the achievement of a whole walking step. At the moment of impact, it clearly appears that Cartesian positions along  $x$  and  $z$  axis observe a decrease of their velocity convergence in order to avoid that the foot gets pushed across the floor. Desired Cartesian positions of the free foot are reached  $t_f = 0.7$  s which is the specified time for step completion. The Cartesian trajectory of the moving leg is represented in Fig. 15. During the free stage, the swinging leg follows a semi-ellipsoidal trajectory. This shape is quite close to a human leg motion while achieving a walking step. At the contact point, the trajectory taken by the bipedal robot is similar to a line segment with a negative slope and at the end of the double support phase, the Humanoid robot leg slides along the contact area.

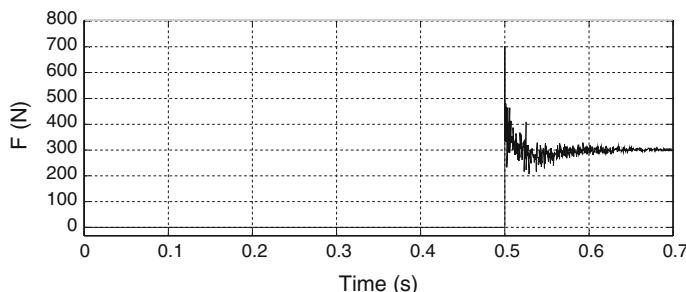
### 3.4.2 Case Study 2: Uncertain Stiffness of the Ground Surface and Sensory Noise

In this case, the objective is to analyze the robustness of the proposed approach to the environment uncertainties and sensory noise. Actually, the Humanoid robot evolves in an unknown environment as we consider an uncertainty on stiffness in the environment model. Furthermore, sensory noise is introduced. Indeed we consider a Gaussian noise of  $0.01^\circ$  mean and  $0.01^\circ$  standard deviation for the joint position measurements and  $0.05^\circ \text{ s}^{-1}$  mean and  $0.05^\circ \text{ s}^{-1}$  standard deviation for the joint velocity measurements. Damping and inertia environmental parameters previously given are used while the stiffness parameter is subject to uncertainties such that:

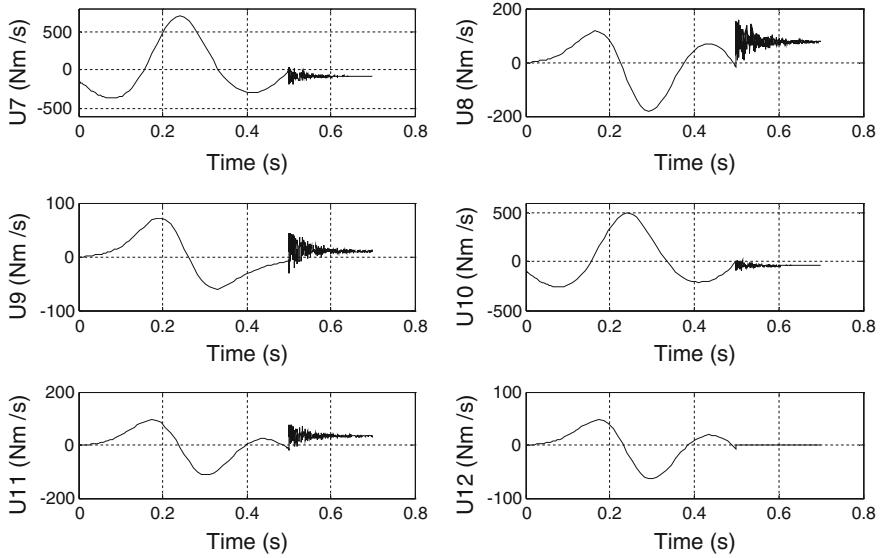
$$K_d = [1 + 2000 * \sin(t)] * \text{diag}[10^4 \quad 10^4 \quad 10^4]$$

Simulation results show the effect of sensory noise and stiffness uncertainties on the contact force and torques during the impact and double support phases. Hence, the ground reaction force is represented in Fig. 16 while Fig. 17 shows the control laws involved in the swing foot motion.

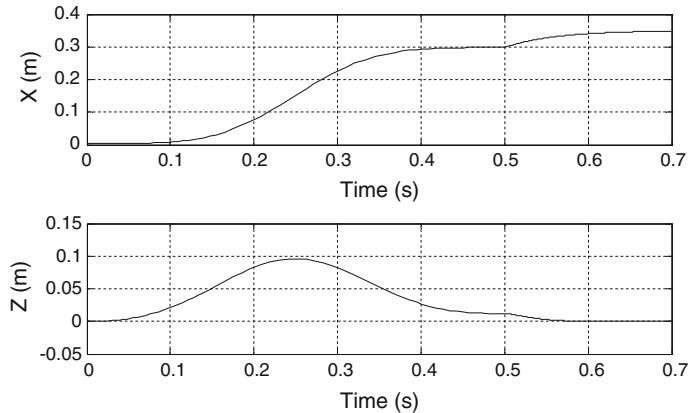
The active foot Cartesian components along  $x$  and  $z$  axis are represented in Fig. 18.



**Fig. 16** Evolution of the ground reaction force during a walking step

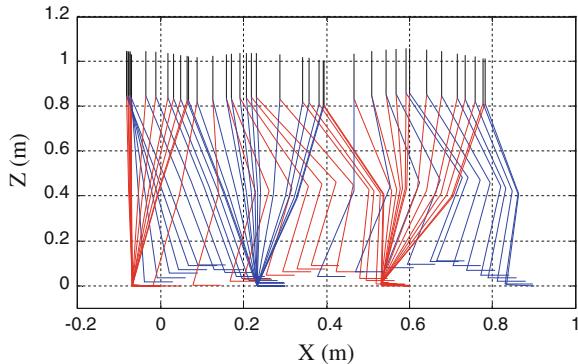


**Fig. 17** Evolution of the control laws of the swing foot during the swing phase



**Fig. 18** Real Cartesian components of the active foot during a walking step

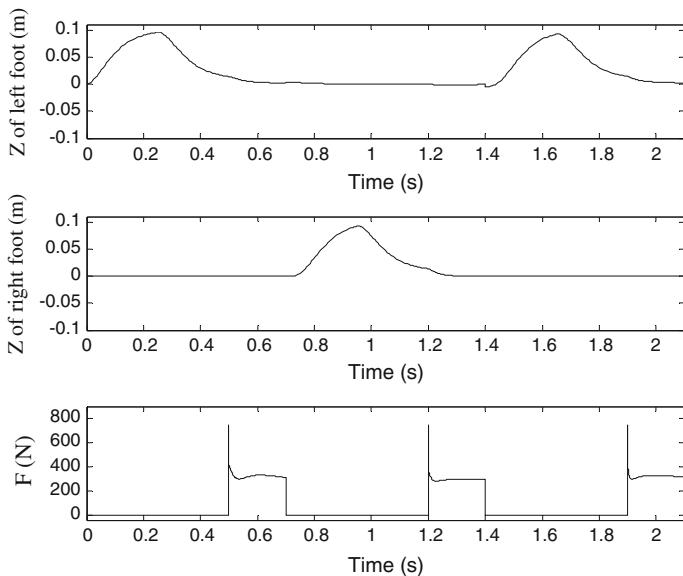
Even if the noise and environment uncertainties are not modeled in the design of the controller, simulation results emphasize the efficiency of the control law and the robustness of the proposed method as perturbations have no effect on the bipedal robot trajectory. We must however note that the effects of vibration are observed in control laws.



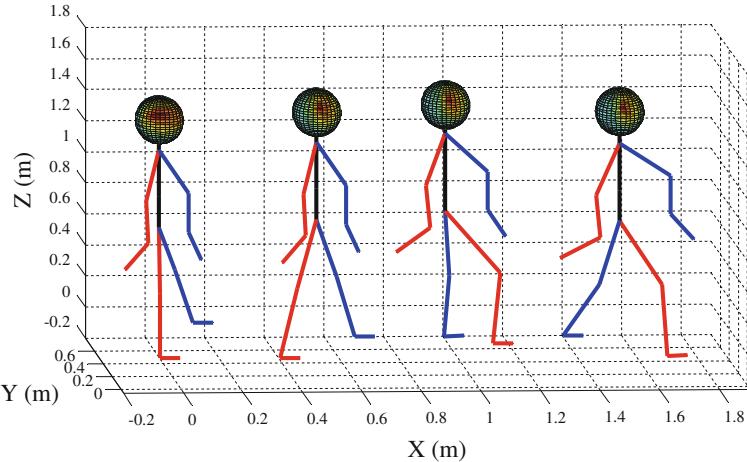
**Fig. 19** The 3D gait pattern generation during a walking cycle

### 3.5 Walking Cycle Generation

To show the walking gait pattern generation during a whole walking cycle, the different rigid bodies composing the bipedal robot are represented in the Cartesian three dimensional space in Fig. 19. Figure 20 gives the evolution in time of the z coordinate of the two feet and the ground reaction force during the achievement of three alternate walking steps.



**Fig. 20** The two feet z coordinate and the contact force during a walking cycle



**Fig. 21** Intermediate positions of the whole Humanoid robot during a walking cycle

In Fig. 19, the several postures taken by the bipedal robot emphasize its motion along the time and shows well the achievement of three alternate footsteps starting with a step initiated by the left foot (in blue). Two steps initiated by the left foot are alternate with one step by the right foot. Thus, the alternation of the Minimum Jerk and impedance based control laws represents an efficient control combination for the achievement of a stable walking cycle. Actually, it combines benefits of both control laws a smooth motion during the swing phase and a stable and safe elastic contact with the ground in the constrained phases.

In order to realize the whole Humanoid motion, we adopt two separate controls for the upper and lower bodies where position, velocity and time objectives allow the synchronization of movements between upper and lower limbs during gait. Thus, using the same strategy of linearizing control law (19), we impose to the arms a movement from the inside to the outside according to the walking cycle phase considered. Figure 21 represents in the Cartesian 3D space the several postures taken by the robot during the achievement of two alternate footsteps. We notice that only the two arms, forearms and hands are moving from the inside to the outside in a motion that is alike a human being's one.

## 4 Future Works

Even if stability conditions are guaranteed, tuning control parameters remains a difficult issue considering the complexity of models (1) and (2). The optimization of the controller parameters using multi-objective criteria and biological-inspired optimization techniques known for their global convergence and good performances and

also using results of [51] will be addressed in future works. On the other hand, safety is mostly ensured through control software level during the SSP and DSP. Actually, the current control system is designed in order to authorize corrective forces and torques to the ground characteristics through suitable choice of stiffness, damping and inertia parameters for the impedance control approach. In future works, a quantitative index must be established to evaluate the safety criterion [52]. Furthermore, robust safe motion under unknown environment and in presence of obstacles will be addressed based on [53, 54].

## 5 Conclusion

In this chapter, satisfying control laws have been proposed to provide good dynamic performances in terms of stability, smoothness and safety for bipedal robots during a gait cycle resembling as much as possible to a Human being one. Indeed, during the swing phase a novel Minimum Jerk control strategy is produced to generate a stable semi-ellipsoidal motion trajectory without dynamic vibrations and control shakings while an appropriate impedance control law is proposed to ensure stable and safe elastic contact balance at foot landings on the ground even in presence of sensory noise and uncertainties on the environment stiffness. Simulation results performed on a 15 link/26 DOF Humanoid robot with a weight of 70 kg and a height of 1.73 m walking at a velocity of  $0.6 \text{ m s}^{-1}$  show the effectiveness of the proposed strategy and better performances compared to related approaches. Future works will focus on optimizing tuning controller parameters to ensure faster speed for the SSP and better safety performances and robustness for the IP and DSP.

## References

1. Carbone G, Ceccarelli M (2005) Legged robotic systems. Cutting edge robotics ARS scientific book. Wien, pp 553–576
2. Harada K, Yoshida E, Yokoi K (2010) Motion planning for humanoid robots. Springer, New York
3. Harada K, Yoshida E, Yokoi K (2004) Motion planning for humanoid robots. Springer, New York
4. Piazz A, Vissoli A (2000) Global minimum-jerk trajectory planning of robot manipulators. *IEEE Trans Ind Electron* 47(1):140–149
5. Hogan N (1984) An organizing principle for a class of voluntary movements. *J Neurosci* 4:2745–2754
6. Flash T, Hogan N (1985) The coordination of arm movements: an experimentally confirmed mathematical model. *J Neurosci* 5:1688–1703
7. Hogan N (1984) Adaptive control of mechanical impedance by co-activation of antagonist muscles. *IEEE Trans Autom Control* AC-29:681–690
8. Kyriakopoulos KJ, Saridis GN (1988) Minimum jerk path generation. In: Proceedings of the IEEE international conference on robotics and automation, pp 364–369
9. Simon D, Isik C (1991) Optimal trigonometric robot joint trajectories. *Robotica* 9(4):379–386

10. Simon D (1993) The application of neural networks to optimal robot trajectory planning. *Robot Auton Syst* 11(1):23–34
11. Doang Nguyen K, Ng TC, Chen IM (2008) On algorithms for planning S-curve motion profiles. *Int J Adv Robot Syst* 5:99–106
12. Amirabdollahian F, Loureiro R, Harwin W (2002) Minimum jerk trajectory control for rehabilitation and haptic applications. In: Proceedings of the IEEE international conference on robotics and automation, pp 3380–3385
13. Alfredo R, Riosa O, Romero-Troncoso RJ, Herrera-Ruiza G, Castaneda-Miranda R (2009) FPGA implementation of higher degree polynomial acceleration profiles for peak jerk reduction in servomotors. *Robot Comput-Integr Manuf* 25:379–392
14. Flash T, Henis E (1991) Arm trajectory modification during reaching toward visual targets. *J Cogn Neurosci* 3(3):220–230 (1991)
15. Boonpratong A, Malisuan S, Degenaar P, Veeraklaew T (2008) A minimum-jerk design of active artificial foot. *Mechatron Embed Syst Appl* 10:443–448
16. Aloulou A, Boubaker O (2011) Minimum jerk-based control for a three dimensional bipedal robot. In: Jeschke S, Liuet H, Schilberg D (eds) Intelligent robotics and applications, Lecture notes in computer science. vol 7120. Springer, Heidelberg, pp 251–262
17. Sung C, Kagawa T, Uno Y (2013) Whole-body motion planning for humanoid robots by specifying via-points. *Int J Adv Robot Syst* 10:1
18. Hogan N (1984) Impedance control: an approach to manipulation. American Control Conference, pp 304–313
19. Yoshikawa T (2000) Force control of robot manipulators. *IEEE Int Conf Robot Autom* 1: 220–226
20. Arevalo JC, Garcia E (2012) Impedance control for legged robots: an insight into the concepts involved. *IEEE Trans Syst, Man, Cybern, Part C: Appl Rev* 42(6):1400–1411
21. Park JH, Chung H (1999) Hybrid control for biped robots using impedance control and computed-torque control. *IEEE Int Conf Robot Autom* 2:1365–1370
22. Lim HO, Setiawan SA, Takanishi A (2004) Position-based impedance control of a biped humanoid robot. *Adv Robot* 18(4):415–435
23. Carbone G, Lim HK, Takanishi A, Ceccarelli M (2006) Stiffness analysis of biped humanoid robot WABIAN-RIV. *Mech Mach Theory* 41(1):17–40
24. Kwon O, Park JH (2009) Asymmetric trajectory generation and impedance control for running of biped robots. *Auton Robot* 26(1):47–78
25. Mehdi H, Boubaker O (2011) Stiffness and impedance control using Lyapunov theory for robot-aided rehabilitation. *Int J Soc Robot* 4:107–119
26. Katić DM, Rodić AD, Vukobratović MK (2008) Hybrid dynamic control algorithm for humanoid robots based on reinforcement learning. *J Intell Robot Syst* 51(1):3–30
27. Darmane R (2004) Le Cycle de la Marche Normale. La lettre de l'Observatoire du mouvement, 11:2
28. Viton JM, Bensoussan L, de Bovis Milhe V, Collado H, Delarque A (2006) Marche Normale et Marche Pathologique, Faculté de Médecine, Université de la Méditerranée, Fédération de Médecine Physique et de Réadaptation, CHU Timone, Marseille
29. Hemami H, Zheng YF (1984) Dynamics and control of motion on the ground and in the air with application to biped robots. *J Robot Syst* 1(1):101–116
30. Chemori A, Alimir M (2004) Multi-step limit cycle generation for rabbit's walking based on a nonlinear low dimensional predictive control scheme. *Mechtronics* 16(5):259–277
31. Cho BK, Park IW, Oh JH (2009) Running pattern generation of humanoid biped with a fixed point and its realization. *Int J Humanoid Robot* 6(4):631–656
32. Westervelt ER, Grizzle JW, Koditschek DE (2003) Hybrid zero dynamics of planar biped walkers. *IEEE Trans Autom Control* 48(1):42–56
33. Hemami H, Utkin VI (2002) On the dynamics and Lyapunov stability of constrained and embedded rigid bodies. *Int J Control* 75(6):408–420
34. Shadmehr R, Wise SP (2005) The computational neurobiology of reaching and pointing: a foundation for motor learning. MIT Press, Cambridge

35. Gasparetto A, Zanotto V (2010) Optimal trajectory planning for industrial robots. *J Adv Eng Softw* 41(4):548–556
36. Morimoto J, Atkeson CG (2007) Learning biped locomotion: application of Poincaré-map-based reinforcement learning. *IEEE Robot Autom Mag* 14(2):41–51
37. Miyamoto H, Morimoto J, Doya K, Kawato M (2004) Reinforcement learning with via-point representation. *Neural Netw* 17:299–305
38. Piazzoli A, Visioli A (1997) An interval algorithm for minimum-jerk trajectory planning of robot manipulators. Conference on Decision and Control, vol 2, pp 1924–1927
39. Saini A, Malouin F, Tousignant B, Jackson PL (2012) The influence of body configuration on motor imagery of walking in younger and older adults. *Neuroscience* 222(11):49–57
40. Arous Y, Boubaker O (2012) Gait trajectory generation for a five link bipedal robot based on a reduced dynamical model. 16th IEEE mediterranean electro-technical conference, Yasmine Hammamet
41. Aloulou A, Boubaker O (2010) Control of a step walking combined to arms swinging for a three dimensional humanoid prototype. *J Comput Sci* 6(8):886–895
42. Winter DA (2009) Biomechanics and motor control of human movement, 4th edn. Wiley, New York
43. Hemami H (1982) Some aspects of Euler-Newton equations of motion. *Arch Appl Mech* 52 (3–4):167–176
44. Shoushtari AL, Abedi P (2012) Realistic dynamic posture prediction of humanoid robot: manual lifting task simulation. *Lect Notes Comput Sci* 7506:565–578
45. Atkeson CG, Hale JG, Riley M, Kotosaka S, Schaul S, Shibata T, Tevatia G, Ude A, Vijayakumar S, Kawato E, Kawato M (2000) Using humanoid robots to study human behavior. *J Intell Syst Appl* 15(4):46–56
46. Aloulou A, Boubaker O (2013) Model validation of a humanoid robot via standard scenarios. 14th international conference on sciences and techniques of automatic control and computer engineering, Sousse, Tunisia
47. Aloulou A, Boubaker O (2012) A relevant reduction method for dynamic modeling of a seven-linked humanoid robot in the three-dimensional space. *Procedia Eng* 41:1277–1284
48. Castley D, Oh P (2011) Development and application of a gel actuator for the design of a humanoid robotic finger. IEEE conference on technologies for practical robot applications, pp 105–108
49. <http://world.honda.com/ASIMO/technology/2011/specification>
50. Park JH (2001) Impedance control for biped robot locomotion. *IEEE Trans Robot Autom* 17(6):870–882
51. Mehdi H, Boubaker O (2011) Impedance controller tuned by particle swarm optimization for robotic arms. *Int J Adv Robot Syst* 8:93–103
52. Echávarri J, Ceccarelli M, Carbone G, Alén C, Muñoz JL, Díaz A, Muñoz-Guajosa JM (2013) Towards a safety index for assessing head injury potential in service robotics. *Adv Robot* 61:1–14
53. Khan SG, Herrmann G, Pipe T, Melhuish C, Spiers A (2010) Safe adaptive compliance control of a humanoid robotic arm with anti-windup compensation and posture control. *Int J Soc Robot* 2:305–319
54. Mehdi H, Boubaker O (2012) New robust tracking control for safe constrained robots under unknown impedance environment. *Lect Notes Comput Sci* 7429:313–323

# Online Walking Pattern Generation Using FFT for Humanoid Robots

Kenji Hashimoto, Hideki Kondo, Hun-Ok Lim and Atsuo Takanishi

**Abstract** An online walking pattern generation is important for a biped humanoid robot to move in dynamic environments. This chapter describes a novel online walking pattern generation using Fast Fourier Transform (FFT). Most previous studies about online walking pattern generation use an inverted pendulum model, which requires other methods to compensate for errors between the model and a real robot. Because our method uses a multi-body robot model, a biped robot can dynamically change its motions online by utilizing only the proposed method. If a robot has external sensors such as a stereo vision system to recognize dynamic environments, the robot can follow a moving target. Verification of the proposed method is conducted through experiments with the human-sized humanoid robots WABIAN-2R and KOBIAN.

**Keywords** Biped humanoid robot · Fast Fourier transform · Walking pattern

## 1 Introduction

We have developed a biped humanoid robot named WABIAN-2R (WAseda BIpedal humANoid-No. 2 Refined) as a human motion simulator to mimic human's motions and mechanisms [4, 5, 16] (see Fig. 1a). It is 1,480 mm tall, weighs 63.8 kg, and has

---

K. Hashimoto (✉)

Research Institute for Science and Engineering, Waseda University, #41-304,  
17 Kikui-cho, Shinjuku-ku, Tokyo 162-0044, Japan  
e-mail: k-hashimoto@takanishi.mech.waseda.ac.jp

H. Kondo · A. Takanishi

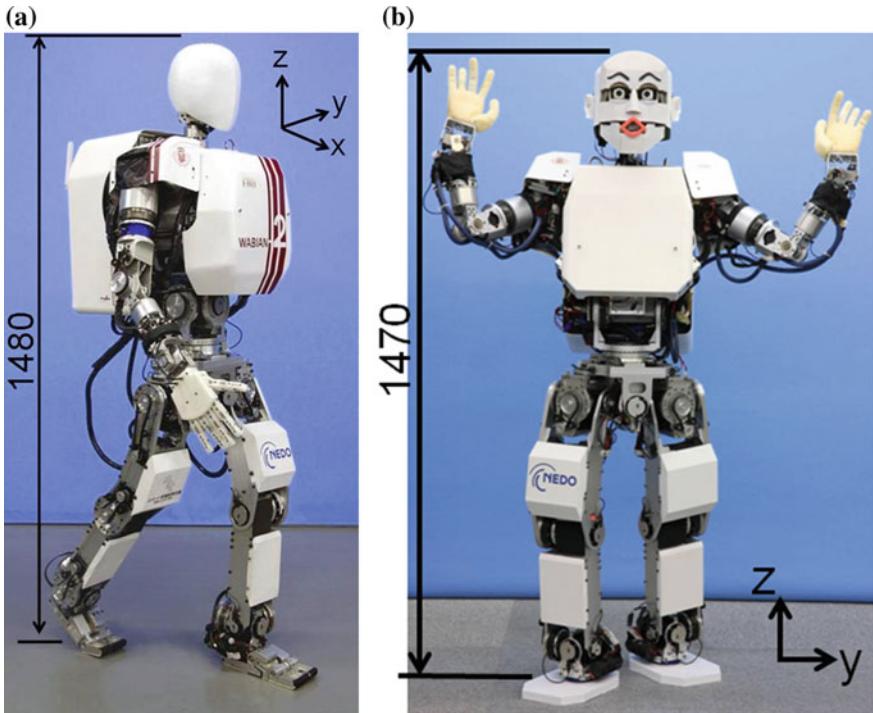
Department of Modern Mechanical Engineering, Waseda University,  
2-2 Wakamatsu-cho, Shinjuku-ku, Tokyo 162-8480, Japan

H.-O. Lim

Faculty of Engineering, Kanagawa University, 3-27-1 Kanagawa-ku,  
Yokohama 221-8686, Rokkakubashi, Japan

H.-O. Lim · A. Takanishi

Humanoid Robotics Institute (HRI), Waseda University, 2-2 Wakamatsu-cho,  
Shinjuku-ku, Tokyo 162-8480, Japan



**Fig. 1** Human-sized humanoid robots used in this research. **a** WABIAN-2R. **b** KOBIAN

41 degrees of freedom (DoF) (two 6-DoF legs, a 2-DoF waist, a 2-DoF trunk, two 7-DoF arms, two 3-DoF hands, a 3-DoF neck and two 1-DoF feet having passive toe joint). We have also developed a whole body emotion expression humanoid robot named KOBIAN [2] (see Fig. 1b), which is based on WABIAN-2R and an emotion expression humanoid robot WE-4RII [10]. KOBIAN is 1,470 mm tall, weighs 62.0 kg, and has 48-DoF. Two CMOS cameras are mounted on each eyeball.

Humanoid robots are expected as partners with us and to accomplish tasks. In order to work in a human living environment, what is important for biped robots is to realize stable walking in the real environment. Until now, many studies about walking pattern generation have been researched based on the Zero Moment Point (ZMP) criteria [6, 11, 13]. ZMP is defined as a point where the total forces and moments acting on a robot are zero [19, 22–24]. If the ZMP exists inside the support polygon formed by the support points between the feet and the ground, a biped robot can walk stably without falling down.

The authors have proposed walking pattern generation based on the ZMP criteria by using FFT [8]. In this method, we can obtain a waist compensatory trajectory as periodic solution in the frequency domain. However, it's necessary to regard the compensated moment from the start to the end of walking as a periodic function and

to compute the whole walking pattern at once. Therefore, it's difficult to generate a part of walking pattern and change walking pattern during walking.

However, in order to work in a human living environment, humanoids need to recognize changing environment by using external sensors such as laser range finder, stereo camera, etc. and move in dynamic and crowded environments such as office spaces, homes, shopping malls, and so on. To realize autonomous locomotion in dynamic environments, many researchers have worked on an online walking pattern generation [1, 3, 7, 9, 12, 14, 15, 17, 18, 20, 21, 26]. Most methods use an analytical derivation for ZMP equations based on an inverted pendulum model and compensate for errors between the model and a real robot with a dynamic filter [25].

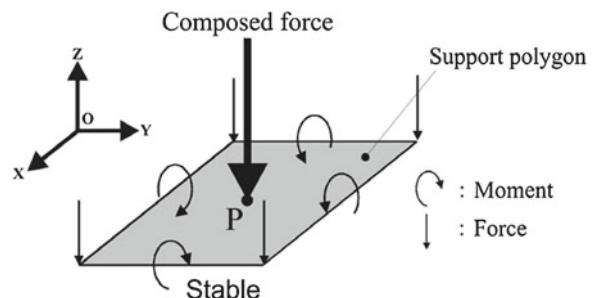
Meanwhile, we have researched on an online walking pattern generation which is based on a multi-body robot model. This method has the advantage that it does not require a dynamic filter because errors between the model and a real robot are small thanks to using the multi-body robot model. This chapter describes a novel online walking pattern generation using Fast Fourier Transform (FFT).

This chapter is organized as follows. Sections 2 and 3 describes a FFT-based offline and online walking pattern generation, respectively, and Sect. 4 describes simulation results. In Sect. 5, experimental results are shown. Section 6 provides conclusions and future work.

## 2 FFT-Based Offline Walking Pattern Generation

Zero moment Point (ZMP) is defined as a point where the total forces and moments acting on a robot are zero [23] (see Fig. 2). If the ZMP exists inside the support polygon formed by the support points between the feet and the ground, a biped robot can walk stably without falling down. One complete walking cycle is divided into two phases: single support phase and double support phase. During the single support phase, one foot is on the ground and the other foot is in the air. The biped robot is in the double support phase as soon as the swing foot reaches the ground. The ZMP should be changed smoothly according to two support phases for dynamic stability.

**Fig. 2** Support polygon and composed force



In this study, the ZMP trajectory is set arbitrarily and smoothly within the support polygon before the robot begins biped walking.

A basic complete motion pattern is generated offline based on ZMP criteria. Our offline pattern generator generates a continuous walking pattern as follows:

1. Feet, waist, and ZMP trajectories are determined as walking parameters. The ZMP trajectory should be planned within the support polygon composed of contact points of the feet.
2. The compensated moments generated by the movements of the robot are computed, which should be compensated for by a horizontal waist motion.
3. We solve ZMP equations to calculate the compensatory waist motion in the frequency domain by FFT on the linearized biped robot model.
4. We transform the waist motion from the frequency domain into the time domain by IFFT. The walking pattern of the robot based on the compensatory waist motion is computed by solving inverse kinematics.
5. We calculate the moment errors ( $M_{error}$ ), by substituting the compensatory motion into the strict robot model. This calculation is repeated until the moment errors decrease to an acceptable moment ( $M_{tolerance}$ ).

## 2.1 Coordinate Frames

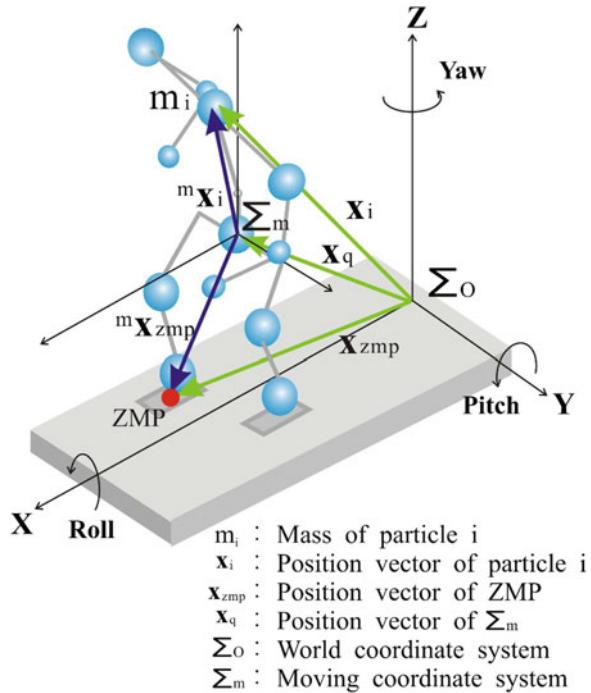
A biped model is shown in Fig. 3. To define mathematical quantities, a world coordinate frame  $\sum_O$  is fixed on the floor where the biped robot can walk. A moving coordinate frame  $\sum_m$  is attached on the center of the waist to consider the relative motion of each particle. The moving coordinate frame  $\sum_m$  is parallel to the world coordinate frame  $\sum_O$ . The following five assumptions are defined to model the biped robot.

- (i) The biped robot consists of a set of particles.
- (ii) The foothold is rigid and not moved by any force and moment.
- (iii) The contact region between the foot and the ground surface is a set of contact points.
- (iv) The coefficients of friction for rotation around the X, Y and Z-axes are nearly zero at the contact point between the feet and the ground surface.
- (v) The feet of the biped robot do not slide on the contact surface.

## 2.2 Approximate Waist and Trunk Motion

Under the modelling assumptions, the moment balance around the ZMP with respect to the world coordinate frame is described as follows:

**Fig. 3** Multi-body model and definition of coordinate systems and vectors

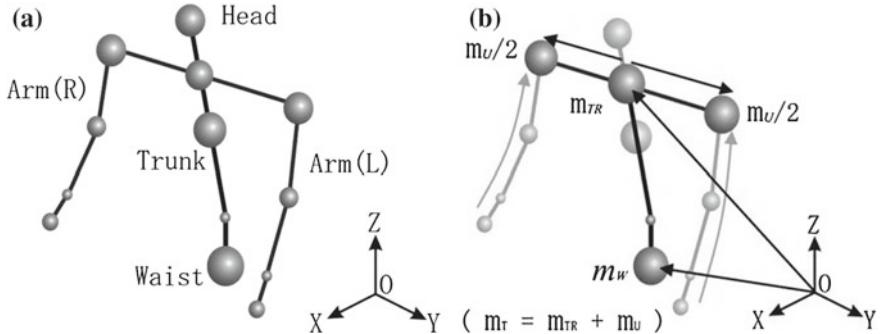


$$\sum_{i}^{All\ Particles} m_i (\mathbf{x}_i - \mathbf{x}_{zmp}) \times (\ddot{\mathbf{x}}_i + \mathbf{G}) - \sum_{k}^{All\ Points} (\mathbf{x}_{Fk} - \mathbf{x}_{zmp}) \times \mathbf{F}_k - \sum_{j}^{All\ Points} \mathbf{M}_j + \mathbf{T}_0 = \mathbf{0} \quad (1)$$

where  $m_i$  is the mass of the particle  $i$ .  $\mathbf{x}_i = [x_i, y_i, z_i]^T$  is the position vector of the particle  $i$ .  $\mathbf{x}_{zmp} = [x_{zmp}, y_{zmp}, z_{zmp}]^T$  is the position vector of reference ZMP.  $\mathbf{G} = [0, 0, g_z]^T$  is the gravitational acceleration vector.  $\mathbf{x}_{Fk} = [x_{Fk}, y_{Fk}, z_{Fk}]^T$  is the position vector where external force  $k$  is applied.  $\mathbf{F}_k = [F_{kx}, F_{ky}, F_{kz}]^T$  is the external force  $k$ .  $\mathbf{M}_j = [M_{jx}, M_{jy}, M_{jz}]^T$  is the external moment  $j$ .  $\mathbf{T}_0 = [0, 0, T_z]^T$  is the total moment acts on a reference ZMP.

To consider the relative motion of each part, Eq.(1) can be modified as follows:

$$\begin{aligned} & \sum_{i}^{All\ Particles} m_i ({}^m\mathbf{x}_i - {}^m\mathbf{x}_{zmp}) \times ({}^m\ddot{\mathbf{x}}_i + {}^m\ddot{\mathbf{x}}_q + {}^m\mathbf{G} + {}^m\dot{\boldsymbol{\omega}} \times {}^m\mathbf{x}_i \\ & \quad + 2{}^m\boldsymbol{\omega} \times {}^m\dot{\mathbf{x}}_i + {}^m\boldsymbol{\omega} \times ({}^m\boldsymbol{\omega} \times {}^m\mathbf{x}_i)) \\ & - \sum_{k}^{All\ Points} ({}^m\mathbf{x}_{Fk} - {}^m\mathbf{x}_{zmp}) \times {}^m\mathbf{F}_k - \sum_{j}^{All\ Points} {}^m\mathbf{M}_j + {}^m\mathbf{T}_0 = \mathbf{0} \end{aligned} \quad (2)$$



**Fig. 4** Upper body model as a four-particle model. **a** Strict model. **b** Four-particle model

where  $\mathbf{x}_q = [x_q, y_q, z_q]^T$  is the position vector of the origin of the frame  $\sum_m$  from the origin of the frame  $\sum_o$ .  ${}^m\omega$  and  ${}^m\dot{\omega}$  are the angular velocity and acceleration vectors, respectively.

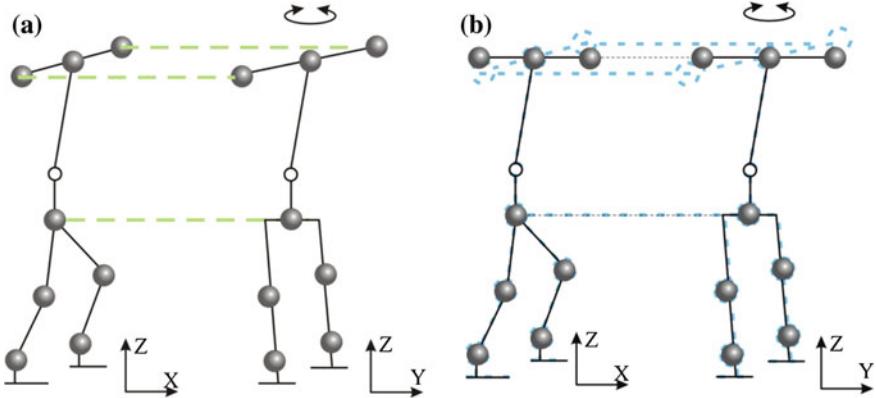
This equation is non-linear because the three-axis motion of the trunk is inter-fifferential each other. Therefore, it is difficult to derive analytically the compensatory motion of the trunk and the waist from Eq. (2). To obtain the approximate solution analytically, we assume the followings:

- The external forces are not considered in the approximate model.
- The upper body is modelled as a four-particle model (see Fig. 4).
- The moving coordinate frame  $\sum_m$  does not rotate.
- The trunk and the waist do not move vertically.

The moment generated by the motion of the lower-limb particles,  $\mathbf{M} = [M_x, M_y, M_z]^T$ , can be obtained as follows:

$$\begin{aligned}
 & m_u {}^m \mathbf{x}_u \times ({}^m \ddot{\mathbf{x}}_u + {}^m \dot{\omega} \times {}^m \mathbf{x}_u + 2 {}^m \omega \times {}^m \dot{\mathbf{x}}_u + {}^m \omega \times ({}^m \omega \times {}^m \mathbf{x}_u)) \\
 & + m_t ({}^m \mathbf{x}_t - {}^m \mathbf{x}_{zmp}) \times ({}^m \ddot{\mathbf{x}}_t + \ddot{\mathbf{x}}_q + {}^m \mathbf{G} + {}^m \dot{\omega} \times {}^m \mathbf{x}_t \\
 & \quad + 2 {}^m \omega \times {}^m \dot{\mathbf{x}}_t + {}^m \omega \times ({}^m \omega \times {}^m \mathbf{x}_t)) \\
 & + m_w ({}^m \mathbf{x}_w - {}^m \mathbf{x}_{zmp}) \times ({}^m \ddot{\mathbf{x}}_w + \ddot{\mathbf{x}}_q + {}^m \mathbf{G} + {}^m \dot{\omega} \times {}^m \mathbf{x}_w \\
 & \quad + 2 {}^m \omega \times {}^m \dot{\mathbf{x}}_w + {}^m \omega \times ({}^m \omega \times {}^m \mathbf{x}_w)) = -\mathbf{M}
 \end{aligned} \tag{3}$$

where  $m_u$  is the mass of both shoulders including the mass of arms.  $m_t$  is the mass of the torso including the head, shoulders and arms.  $m_w$  is the mass of the waist.  ${}^m \mathbf{x}_u$  is the position vector of the shoulder with respect to the neck frame.  ${}^m \mathbf{x}_t$  and  ${}^m \mathbf{x}_w$  are the position vectors of the neck and the waist with respect to the moving coordinate frame  $\sum_m$ , respectively.



**Fig. 5** Linearization of the robot model. **a** Strict model. **b** Linearized model

In case the moments are not generated by the fictitious forces, the moment  $\mathbf{M}$  can be divided as three moment components such as pitch, roll and yaw moments. We assume that both the waist and the trunk particles do not move vertically ( ${}^m\ddot{z}_w = 0$ ,  ${}^m\ddot{z}_q = 0$ ), and the trunk arm rotates on only the horizontal plane as shown in Fig. 5. We put the terms relating to the motion of the upper-body particles on the left-hand side as unknown variables, and the terms relating to the moments generated by the lower-limb particles on the right-hand side as known parameters. The decoupled and linearized ZMP equations can be obtained as follows:

$$\left. \begin{array}{l} \hat{M}_{yt} + \hat{M}_{yw} = \hat{M}_y(t) \\ \hat{M}_{xt} + \hat{M}_{xw} = \hat{M}_x(t) \\ m_t l^2 \ddot{\theta}_t = \hat{M}_z(t) \end{array} \right\} \quad (4)$$

where

$$\left. \begin{array}{l} \hat{M}_{yt} = m_t ({}^m z_t - {}^m z_{zmp}) {}^m \ddot{x}_t - m_t g_z {}^m x_t \\ \hat{M}_{yw} = m_w ({}^m z_w - {}^m z_{zmp}) {}^m \ddot{x}_w - m_w g_z {}^m x_w \\ \hat{M}_y(t) = -M_y - (m_t ({}^m z_t - {}^m z_{zmp}) \ddot{x}_q + m_t g_z {}^m x_{zmp} \\ \quad + m_w ({}^m z_w - {}^m z_{zmp}) \ddot{x}_q + m_w g_z {}^m x_{zmp}) \\ \hat{M}_{xt} = -m_t ({}^m z_t - {}^m z_{zmp}) {}^m \ddot{y}_t + m_t g_z {}^m y_t \\ \hat{M}_{xw} = -m_w ({}^m z_w - {}^m z_{zmp}) {}^m \ddot{y}_w + m_w g_z {}^m y_w \\ \hat{M}_x(t) = -M_x - (-m_t ({}^m z_t - {}^m z_{zmp}) \ddot{y}_q - m_t g_z {}^m y_{zmp} \\ \quad - m_w ({}^m z_w - {}^m z_{zmp}) \ddot{y}_q - m_w g_z {}^m y_{zmp}) \\ \hat{M}_z(t) = -M_z \\ \quad - \left( m_t ({}^m x_t - {}^m x_{zmp}) ({}^m \ddot{y}_t + \ddot{y}_q) - m_t ({}^m y_t - {}^m y_{zmp}) ({}^m \ddot{x}_t + \ddot{x}_q) \right. \\ \quad \left. + m_w ({}^m x_w - {}^m x_{zmp}) ({}^m \ddot{y}_w + \ddot{y}_q) - m_w ({}^m y_w - {}^m y_{zmp}) ({}^m \ddot{x}_w + \ddot{x}_q) \right) \end{array} \right\} \quad (5)$$

where  $l$  is the length between the neck and the shoulder.  $\theta_t$  is the vertical angle of the trunk.  $\hat{M}_{xt}$  and  $\hat{M}_{yt}$  are the roll and the pitch trunk components of the moments, respectively.  $\hat{M}_{xw}$  and  $\hat{M}_{yw}$  are the roll and the pitch waist components of the moments, respectively.

The compensatory motion of the trunk and the waist can be easily computed by Fourier transforms.  $\hat{M}_{xt}$ ,  $\hat{M}_{yt}$ ,  $\hat{M}_{xw}$  and  $\hat{M}_{yw}$  become the known functions because they are calculated by the motion of the lower limbs and the time trajectory of ZMP. In steady walking, they are periodic functions because each particle of the biped robot and ZMP move periodically with respect to the moving coordinate frame  $\sum_m$ . Comparing the Fourier transform coefficients of both sides of each equation, the approximate periodic solutions of the pitch and roll of the trunk and waist,  ${}^m x_t$ ,  ${}^m y_t$ ,  ${}^m x_w$ ,  ${}^m y_w$  and  $\theta_t$ , can be obtained.

### 2.3 Expansion into Complete Walking

This method is applicable to a complete walking. By regarding the whole complete walking as one periodic walking motion, the approximate solutions for the complete walking can be derived. However, the biped robot must have a long period of standing time before starting and after stopping. Here, we simply discuss the waist motion. The transfer function in the frequency domain of (4) can be expressed as follows:

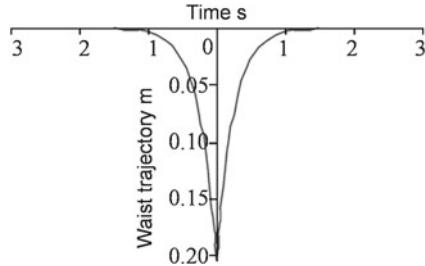
$${}^m x_w(\omega) = \frac{2a}{\omega^2 + a^2} b \quad (6)$$

$$a = \sqrt{\frac{{}^m g_z}{{}^m z_w - {}^m z_{zmp}}}, \quad b = -\frac{1}{2\sqrt{{}^m g_z ({}^m z_w - {}^m z_{zmp})}}$$

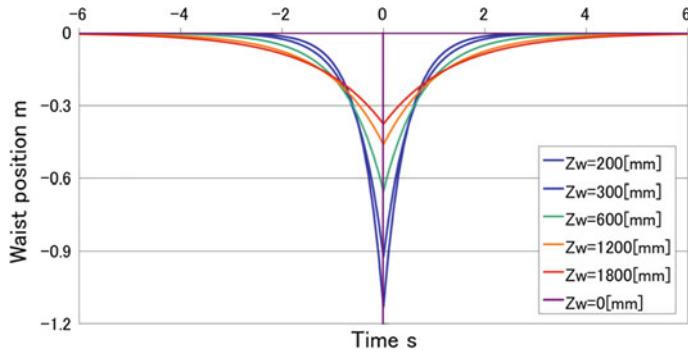
Equation (6) is generally known as Lorenz function. The original function in time domain of (6) is as follows:

$${}^m x_w(t) = b e^{-a|t|} \quad (7)$$

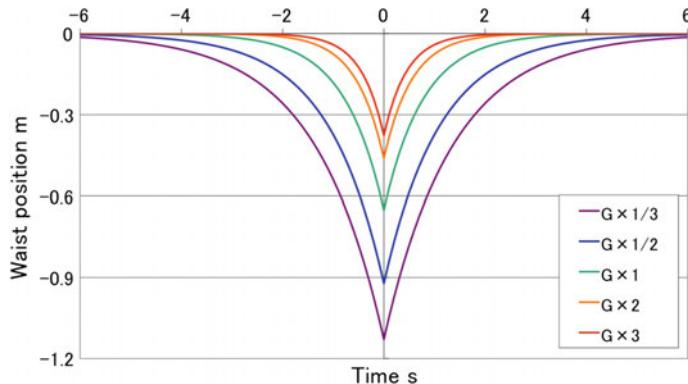
From (7), we can imagine that the causal law may not be applied. If the walking speed of the biped robot increases more, it goes without saying that  ${}^m x_w$  affects stability more badly. Therefore, the waist should be in motion earlier than the shift of ZMP on the ground to cancel the effect of the produced moments. The relationship between the waist motion and the applied force has been simulated. When the biped robot is not in motion, an impulse moment is applied to the waist. Figure 6 shows the waist motion in case of  ${}^m z_w = 600$  mm which is the waist height of WABIAN-2R. In this simulation, we can see that the compensatory motion of the waist should be begun with a view of balancing before and after the impulse moment is applied to the biped robot.



**Fig. 6** Compensatory waist trajectory for an impulse moment along the front-back direction



**Fig. 7** Compensatory motion of the waist when changing the waist height



**Fig. 8** Compensatory motion of the waist when changing the acceleration of gravity

Figure 7 shows the compensatory motion of the waist when changing the waist height of the biped robot. We can see that the higher the waist height is, the earlier the compensatory motion of the waist must be begun. Figure 8 shows the compensatory motion of the waist when changing the acceleration of gravity. The smaller the acceleration of gravity is, the earlier the compensatory motion of the waist must be begun.

## 2.4 Recursive Calculation

A recursive method is used to obtain the strict solutions of the trunk and the waist motions. First, the approximate periodic solutions of the linearized equation (5) are calculated. Second, the approximate periodic solutions are substituted into the moment equation (2) of the strict biped model, and the errors of moments generated by the trunk and waist motions are calculated according to the planned ZMP. These errors are accumulated in the right-hand side of Eq. (4). The approximate solutions are computed again. Finally, these computations are repeated until the errors fall below a certain tolerance level. As a result, the strict periodic solutions of the nonlinear equations are obtained by a convergent regularity. The limit value of an accumulated moment error on each axis,  $E_n$ , is estimated as follows:

$$E_n = \frac{2E_{n-1} + e_{n-1}}{2} \quad (n = 3, 4, 5, \dots) \quad (8)$$

where  $e_n$  is the calculated moment error after  $n$  times of iterations.  $E_1$  is zero, and  $E_2$  is equal to  $e_1$ .

Using this method, we have realized about a 90 % decrease in the number of iteration times. Figure 9 shows how to obtain the compensatory motion for stability.

## 3 FFT-Based Online Walking Pattern Generation

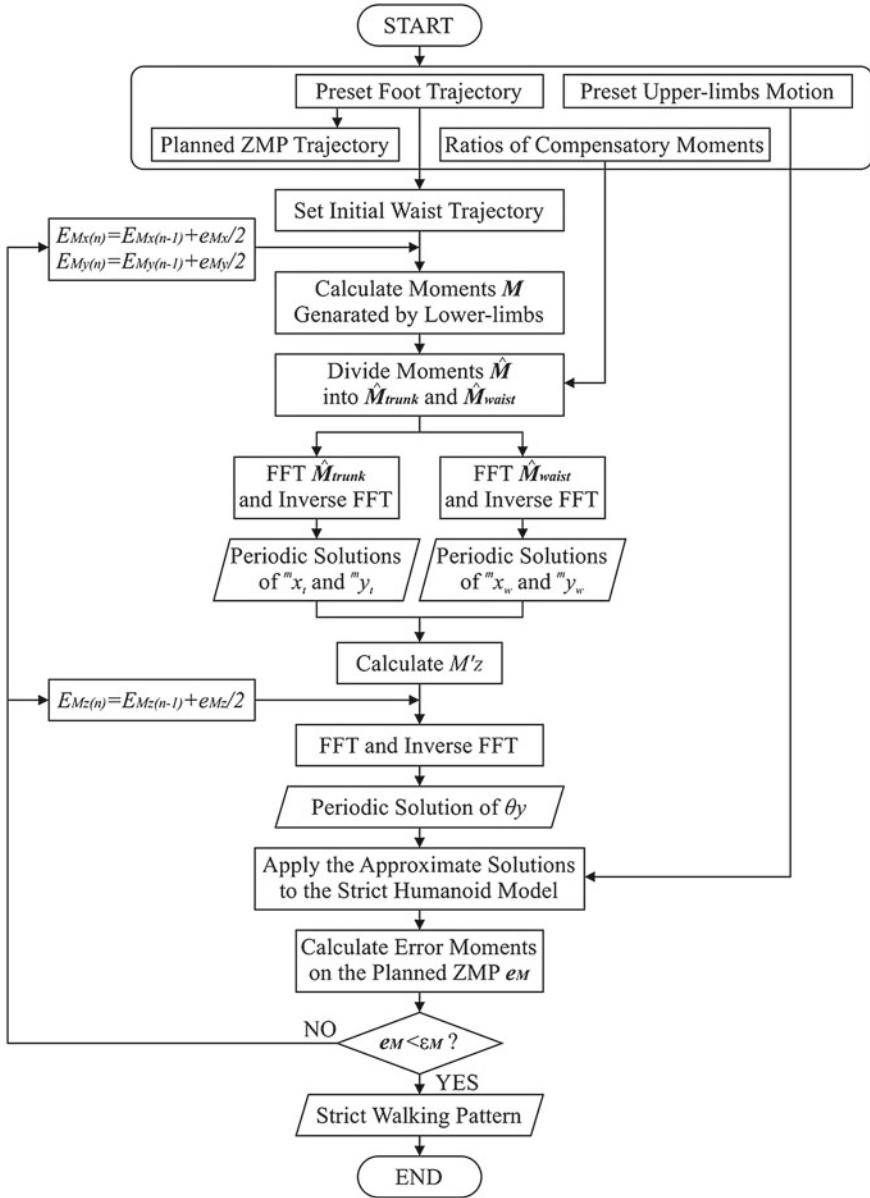
Our online walking pattern generation is based on the FFT-based offline pattern generation. Hereinafter, we will explain about a FFT-based online walking pattern generation.

According to Fig. 6, the compensatory waist trajectory occurs for a few seconds before and after the impulse moment is applied, and there are almost no effects on the other periods. Therefore, if a moment to be compensated is regarded as a collection of impulse moments, the compensatory waist trajectory is represented as a superposition of compensatory waist trajectories for the impulse moments. Thus, the compensatory waist trajectory at a given time can be computed if the moment to be compensated is known for a few seconds before and after the given time.

We focused on the compensatory waist trajectory for the moment to be compensated which is cut out with a window of predetermined duration as shown in (9). The moment to be compensated is regarded as zero outside the window.

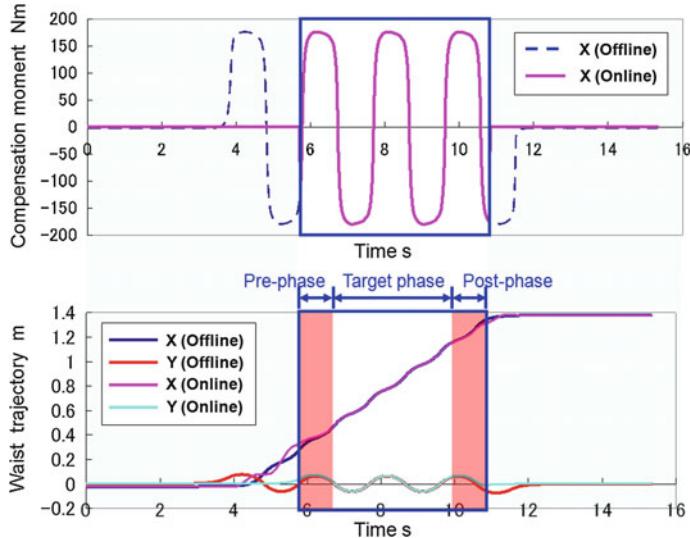
$$M'_y = \begin{cases} M_y^* & (\text{within the window}) \\ 0 & (\text{otherwise}) \end{cases} \quad (9)$$

The compensatory waist trajectory is shown as Fig. 10. Compared with the compensatory waist trajectory generated by the conventional offline pattern generation for



**Fig. 9** Flowchart of the offline pattern generation method based on Fourier transform

the whole moment to be compensated, the compensatory trajectories around both ends diverge and are not proper. On the other hand, the compensatory trajectory during the other period is proper.



**Fig. 10** Compensatory waist trajectory and compensated moment which is cut out with a window of predetermined duration

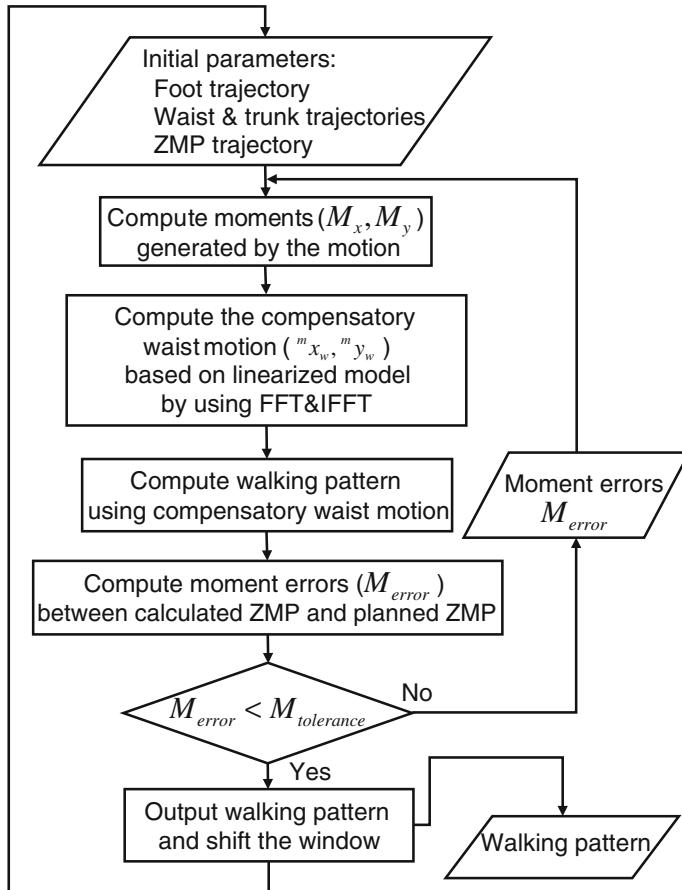
As mentioned above, it is necessary that the moment to be compensated is known during a few seconds in the past and future on the given time to compute the proper trajectory around both ends. However, it's impossible to compute the proper trajectories around both ends in this case due to no consideration of the compensated moment outside the window.

As a solution, we divide the window into three phases as following (see Fig. 10):

- pre-phase              (duration:  $t_{pre}$  s)
- target phase            (duration:  $t_{tar}$  s)
- post-phase             (duration:  $t_{pst}$  s)

The durations of each part of both  $t_{pre}$  and  $t_{pst}$  were predetermined as 1.8 seconds according to the result of experimental trials and errors. As shown in Fig. 7, appropriate values of both  $t_{pre}$  and  $t_{pst}$  depend on dynamics of robots such as its weight, its waist height and so on. The duration  $t_{tar}$  is regarded as the time span of the compensatory trajectory which is generated within a window and the shorter  $t_{tar}$  is, the better the pattern generation is, because it's possible to modify walking pattern in short periods.  $t_{tar}$  is determined based on the length of calculation time.

If both  $t_{pre}$  and  $t_{pst}$  are long enough (more than 1.8 s), the compensatory trajectories during the target phase are computed properly. Then, the trajectory is available for making a robot's motion stable. A biped robot can change walking patterns online by shifting the window to next target phase to generate a compensatory trajectory. Figure 11 shows flowchart of the sequential pattern generation.



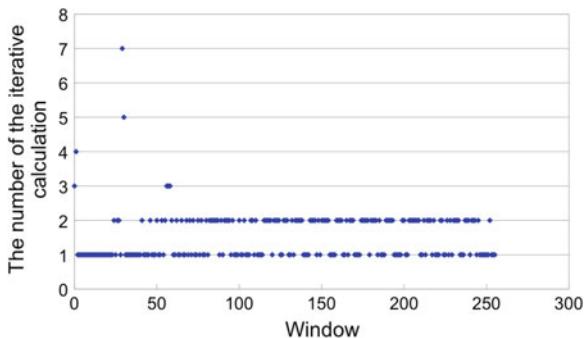
**Fig. 11** Flowchart of the FFT-based online walking pattern generation

If the sequential pattern generation is realized while a robot is walking, we can generate a walking pattern online. In that event, the walking parameters such as the feet trajectories and ZMP trajectory are modifiable outside the predetermined window.

## 4 Walking Simulations

First, we evaluated the calculation time for generating walking patterns online by using the computer mounted on the robot (Pentium(R) M 1.8 GHz, QNX(R) Neutrino(R) RTOS 6.3 as an operating system). Figure 12 shows the number of the iterative calculations until error moments become negligibly small at each window.

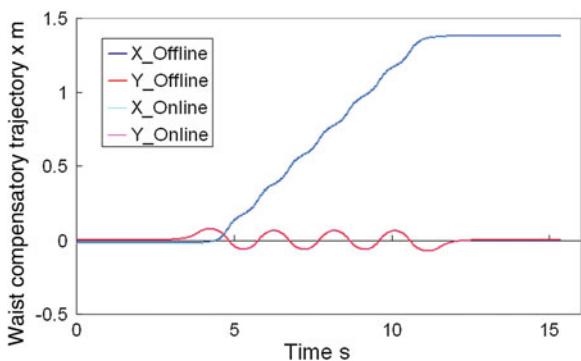
**Fig. 12** Number of the iterative calculation at each window



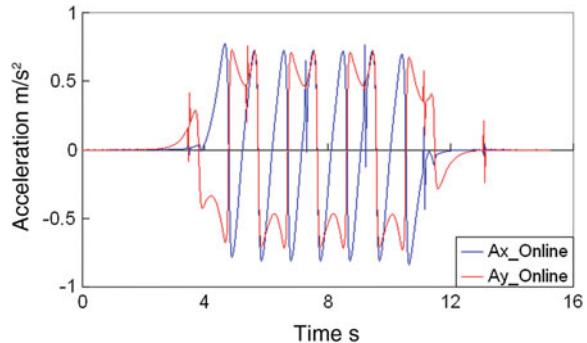
In this simulation, a walking cycle was 0.96 s/step, a step length was 200 mm/step, both  $t_{pre}$  and  $t_{pst}$  were 1.80 s, and  $t_{tar}$  was 0.06 s. According to Fig. 12, it's confirmed that the average number of iterative calculations is about 1.2 times, and 7 times iterative calculations are needed at worst. Because a calculation time per each calculation is 6.8 ms, we can update walking patterns sequentially and fast enough.

Second, we compared the compensatory trajectory generated by the proposed online walking pattern generation with that generated by the conventional offline pattern generation. Figure 13 shows the comparison of two conditions. In this simulation, a walking cycle was 0.96 s/step, a step length was 200 mm/step, both  $t_{pre}$  and  $t_{pst}$  were 1.80 s, and  $t_{tar}$  was 1.92 s. From Fig. 13 we can see that the trajectory generated by the proposed method is almost the same as the conventional one. However, in this method, the continuity of the position and velocity of the trajectory is not guaranteed at the connecting points between windows because the trajectory consists of the series of trajectories which come from each window. If the accelerations at the connecting points are very large, there is a possibility that the biped robot falls down during walking. Figures 14 and 15 show the acceleration of the trajectory and the difference value of the acceleration compared with the conventional one, respectively. As the result, the acceleration at the connecting points is comparable with

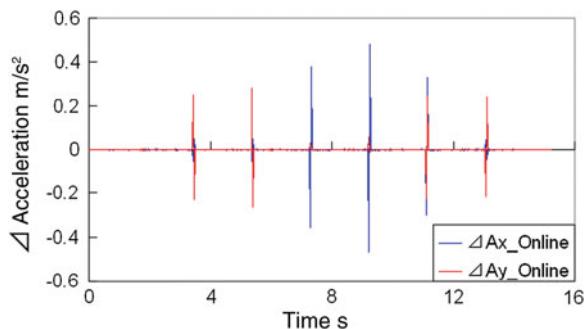
**Fig. 13** Comparison of the compensatory trajectories between offline and online walking pattern generation



**Fig. 14** Acceleration of the trajectory



**Fig. 15** Difference value of the acceleration

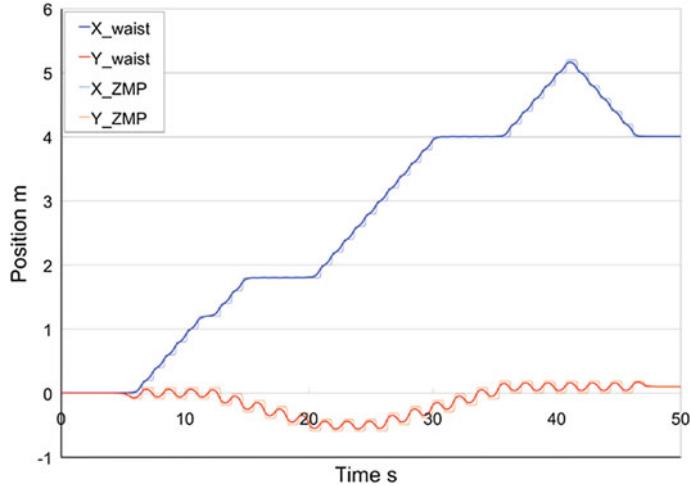


acceleration during walking and a stable walking was realized through the walking experiment as described below.

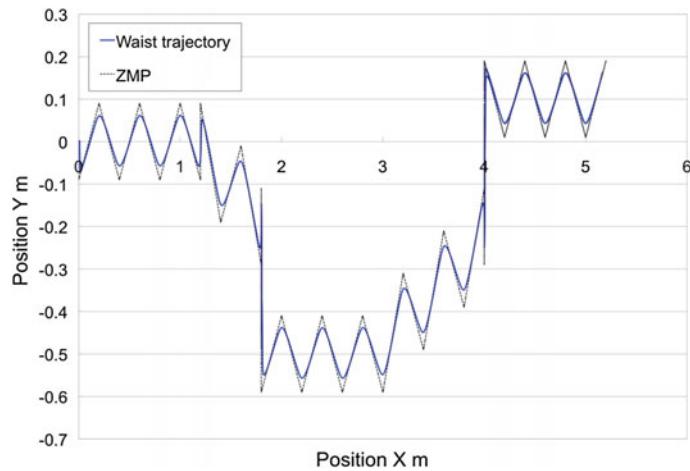
Finally, we developed a simulation program to confirm the basic function of the proposed online walking pattern generation. In the simulation, both  $t_{pre}$  and  $t_{pst}$  were set to 1.80 s, and  $t_{tar}$  was set to 0.96 s. The moving direction of a robot is controlled with a joystick. We made the robot move forward from 5 to 11 s, diagonally forward-right from 11 to 15 s, rightward 15 to 20 s, forward 20 to 25 s, diagonally forward-left from 25 to 31 s, leftward 31 to 36 s, forward 36 to 41 s, and backward 41 to 46 s. Figures 16 and 17 depict planned ZMP trajectories and waist compensatory motion. Through the simulations we confirmed that the robot can generate walking patterns online according to the input from the joystick.

## 5 Experimental Tests and Considerations

The effectiveness of the proposed method was confirmed through simulations. Therefore, next, we implemented the proposed online walking pattern generation on WABIAN-2R and carried out walking experiments. Because WABIAN-2R doesn't have external sensors to recognize its surroundings, we use a joystick to move the



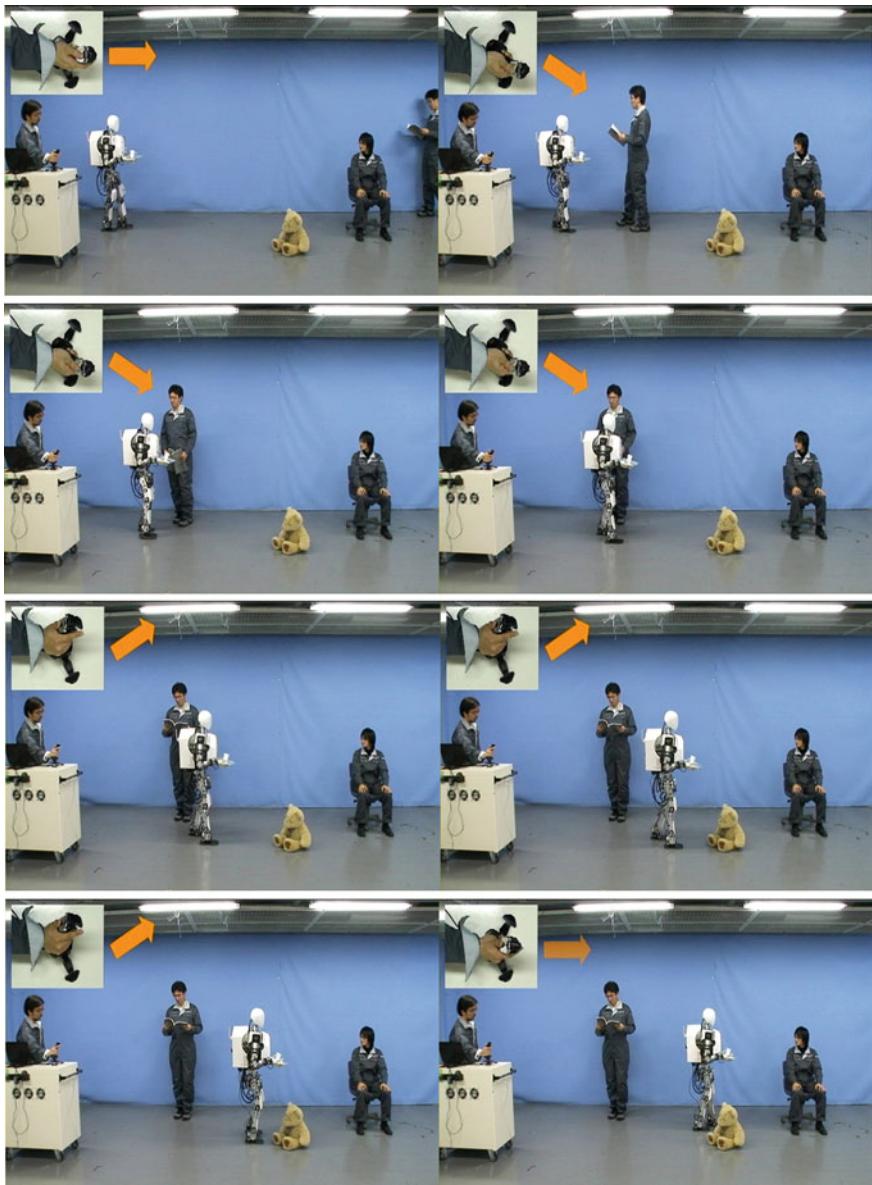
**Fig. 16** Planned ZMP trajectory and waist compensatory motion controlled with a joystick



**Fig. 17** Planned ZMP trajectory and waist compensatory motion on the horizontal plane

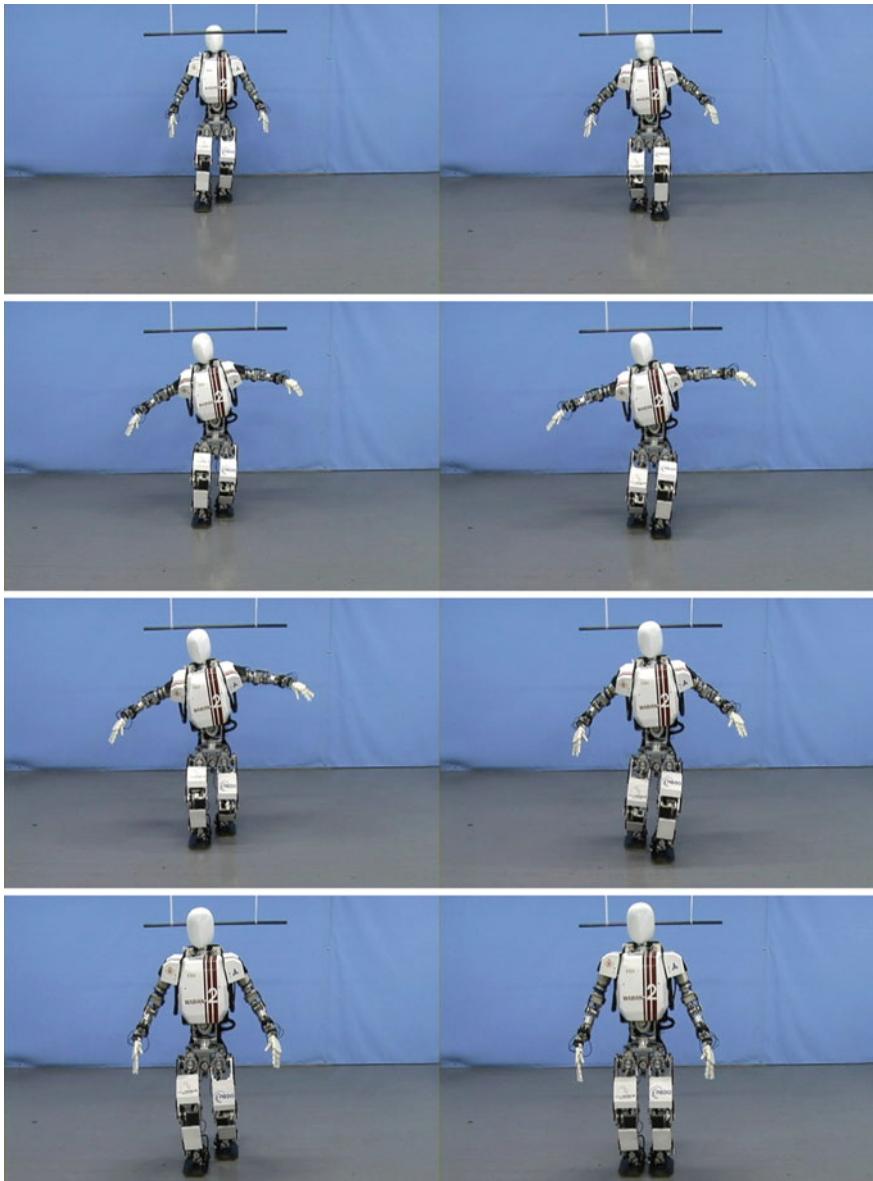
robot. The joystick is connected to a laptop PC which communicates with robot's control computer through LAN. In this experiment, a walking cycle is 0.96 s/step, both  $t_{pre}$  and  $t_{pst}$  are 1.80 s, and  $t_{tar}$  is 0.96 s. As a result of this experiment, WABIAN-2R could continue walking according to inputs from the joystick. Figure 18 shows snapshots of the walking experiment.

Second, we conducted experiments so that a robot changes CoM height and its upper body posture dynamically. Our proposed online walking pattern generation can deal with such dynamic motions because our proposed method uses not an inverted pendulum model but a multi-body robot model. In this experiment, the



**Fig. 18** Walking experiment using joystick inputs

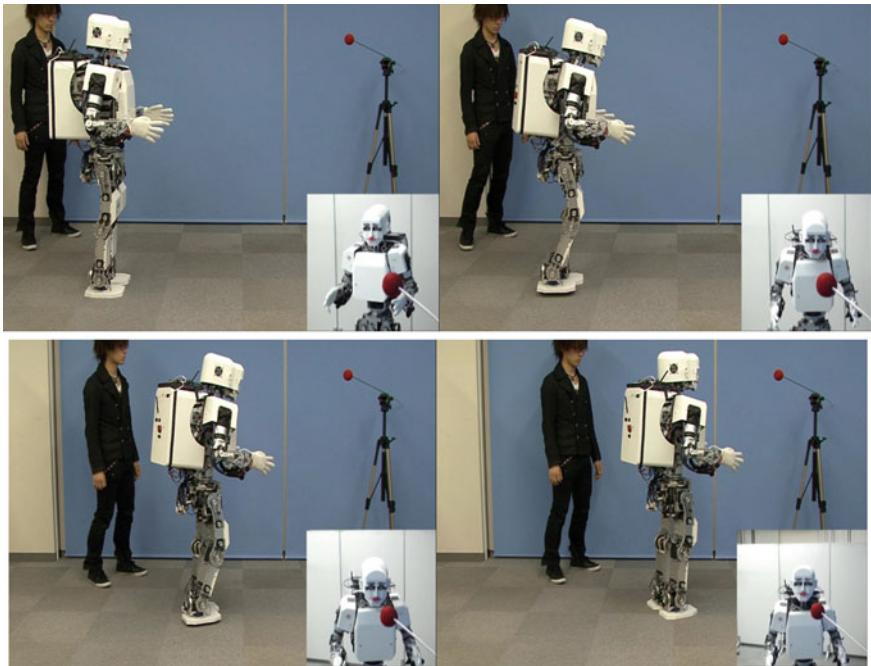
robot gradually decreases CoM height by 100 mm, bends the trunk forward by 10 degrees, and returns both the CoM height and the trunk angle to the original position during walking. A walking cycle is 0.96 s/step, both  $t_{pre}$  and  $t_{pst}$  are 1.80 s, and  $t_{tar}$  is



**Fig. 19** Walking experiment changing CoM height and its upper body posture

0.96 s. As a result, WABIAN-2R realized a stable walk even if the robot significantly changed its posture while walking as shown in Fig. 19.

Finally, we implemented the proposed online walking pattern generation on KOBIAN. Because KOBIAN has two CMOS cameras on each eye, KOBIAN should

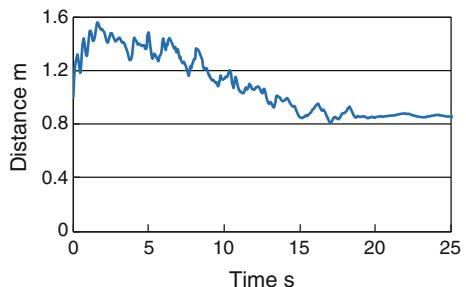


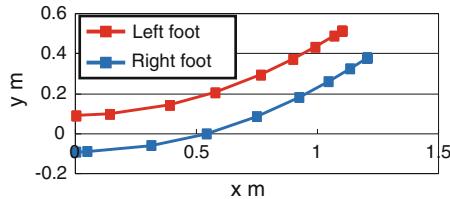
**Fig. 20** Walking experiment while following a stationary target

be able to follow a visual target. First, the target was put 1,000 mm ahead and 750 mm left of the robot, and the height is the same as that of robot's eyes. In this experiment, the robot approaches 800 mm ahead of the target. Figure 20 shows the snapshots of object tracking locomotion. Figure 21 shows the distance between the robot and the target, and Fig. 22 shows feet trajectories. From these figures, we can find that the robot succeeded in coming close about 800 mm ahead of the target with 16 steps.

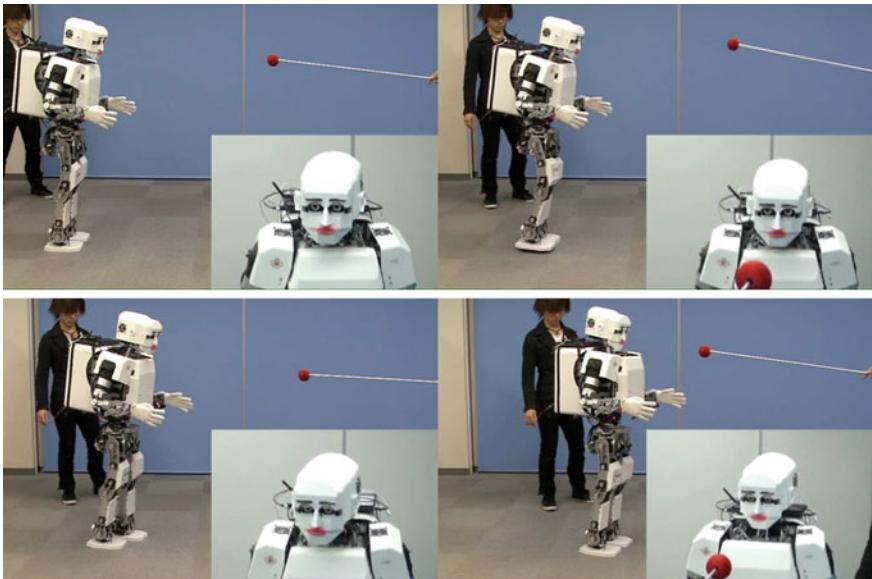
In addition, we conducted an experiment of object tracking locomotion when a human moves a target freely (see Fig. 23). The robot could follow a moving target by coordinating eye, head and leg movements. We confirmed the effectiveness of

**Fig. 21** Distance between the robot and the target





**Fig. 22** Feet trajectories while following a stationary target



**Fig. 23** Walking experiment while following a moving target

the proposed online walking pattern generation through experiments. Although the robot followed an object in this experiment, we can apply this method to move in dynamic environments with obstacles avoidance.

## 6 Conclusions and Future Work

We described a FFT-based online walking pattern generation for biped robots. Our proposed online pattern generation can deal with dynamic motions such as changing the CoM height, bending its upper body and so on because our method uses not an inverted pendulum model but a multi-body robot model. Verification of the proposed method was conducted through experiments with a biped humanoid robot WABIAN-2R and KOBIAN. Using WABIAN-2R which doesn't have external sensors, we

confirmed fundamental effectiveness of the proposed method. WABIAN-2R could change its moving direction online according to joystick inputs and also realized a stable walk even if the robot significantly changed its posture while walking. With KOBIAN which has two CMOS cameras on each eye, we confirmed that the robot could track and follow a moving target by using stereo vision.

Our next goal is to develop a new algorithm to change walking parameters based on changes of dynamic environments and realize adaptive walking in the real environment.

**Acknowledgments** This study was conducted as part of the Research Institute for Science and Engineering, Waseda University, and as part of the humanoid project at the Humanoid Robotics Institute, Waseda University. It was also supported in part by MEXT/JSPS KAKENHI (Grant Number: 24360099 and 25220005), Grants for Excellent Graduate Schools, MEXT, Japan, SolidWorks Japan K.K., DYDEN Corporation, and Cybernet Systems Co., Ltd whom we thank for their financial and technical support.

## References

1. Chestnutt J, Kuffner J (2004) A tiered planning strategy for biped navigation. In: Proceedings of the 2004 IEEE-RAS/RSJ international conference on humanoid robots, pp 422–436
2. Endo N, Momoki S, Zecca M, Saito M, Mizoguchi Y, Itoh K, Takanishi A (2008) Development of whole-body emotion expression humanoid robot. In: Proceedings of the 2008 IEEE international conference on robotics and automation, pp 2140–2145
3. Harada K, Kajita S, Kaneko K, Hirukawa H (2006) An analytical method on real-time gait planning for humanoid robots. *Int J Humanoid Robot* 3(1):1–19
4. Hashimoto K, Takezaki Y, Hattori K, Kondo H, Takashima T, Lim HO, Takanishi A (2010) A study of function of the human's foot arch structure using biped humanoid robot. In: Proceedings of the 2010 IEEE/RSJ international conference on intelligent robots and systems, pp 2206–2211
5. Hashimoto K, Takezaki Y, Lim HO, Takanishi A (2013) Walking stabilization based on gait analysis for biped humanoid robot. *Adv Robot* 27(7):541–551
6. Huang Q, Yokoi K, Kajita S, Kaneko K, Arai H, Koyachi N, Tanie K (2001) Planning walking patterns for a biped robot. *IEEE Trans Robot Autom* 17(3):280–289
7. Kajita S, Kanehiro F, Kaneko K, Fujiwara K, Harada K, Yokoi K, Hirukawa H (2003) Biped walking pattern generation by using preview control of zero-moment point. In: Proceedings of the 2003 international conference on robotics and automation, pp 1620–1626
8. Lim HO, Takanishi A (2005) Compensatory motion control for a biped walking robot. *Robotica* 23(1):1–11
9. Löffler K, Gienger M, Pfeiffer F (2003) Sensor and control design of a dynamically stable biped robot. In: Proceedings of the 2003 IEEE international conference on robotics and automation, pp 484–490
10. Miwa H, Itoh K, Matsumoto M, Zecca M, Takanobu H, Roccella S, Carrozza MC, Dario P, Takanishi A (2004) Effective emotional expressions with emotion expression humanoid robot WE-4RII. In: Proceedings of the 2004 IEEE/RSJ international conference on intelligent robots and systems, pp 2203–2208
11. Nagasaka K, Inoue H, Inaba M (1999) Dynamic walking pattern generation for a humanoid robot based on optimal gradient method. In: Proceedings of the 1999 IEEE international conference on systems, man, and cybernetics, pp 908–913
12. Nishiwaki K, Kagami S (2006) High frequency walking pattern generation based on preview control of ZMP. In: Proceedings of the 2006 IEEE international conference on robotics and automation, pp 2667–2672

13. Nishiwaki K, Nagasaka K, Inaba M, Inoue H (1999) Generation of reactive stepping motion for a humanoid by dynamically stable mixture of pre-designed motions. In: Proceedings of the 1999 IEEE international conference on systems, man, and cybernetics, pp 902–907
14. Nishiwaki K, Sugihara T, Kagami S, Inaba M, Inoue H (2001) Online mixture and connection of basic motions for humanoid walking control by footprint specification. In: Proceedings of the 2001 IEEE international conference on robotics and automation, pp 4110–4115
15. Nishiwaki K, Kagami S, Kuniyoshi Y, Inaba M, Inoue H (2002) Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired ZMP. In: Proceedings of the 2002 IEEE/RSJ international conference on intelligent robots and systems, pp 2684–2689
16. Ogura Y, Aikawa H, Shimomura K, Kondo H, Morishima A, Lim HO, Takanishi A (2006) Development of a humanoid robot WABIAN-2. In: Proceedings of the 2006 IEEE international conference on robotics and automation, pp 76–81
17. Park IW, Kim JY, Lee J, Oh JH (2006) Online free walking trajectory generation for biped humanoid robot KHR-3 (HUBO). In: Proceedings of the 2006 IEEE international conference on robotics and automation, pp 2667–2672
18. Sakagami Y, Watanabe R, Aoyama C, Matsunaga S, Higaki N, Fujimura K (2002) The intelligent ASIMO: system overview and integration. In: Proceedings of the 2002 IEEE/RSJ international conference on intelligent robots and systems, pp 2478–2483
19. Sardain P, Bessonnet G (2004) Forces acting on a biped robot. Center of pressure-zero moment point. *IEEE Trans Syst, Man, Cybern—Part A Syst Hum* 34(5):630–637
20. Sugihara T, Nakamura Y (2005) A fast online gait planning with boundary condition relaxation for humanoid robots. In: Proceedings of the 2005 IEEE international conference on robotics and automation, pp 306–311
21. Sugihara T, Nakamura Y (2009) Boundary condition relaxation method for stepwise pedipulation planning of biped robots. *IEEE Trans Robot* 25(3):658–669
22. Vukobratovic M, Stepanenko J (1972) On the stability of anthropomorphic systems. *Math Biosci* 15:1–37
23. Vukobratovic M, Frank AA, Juricic D (1970) On the stability of biped locomotion. *IEEE Trans Biomed Eng* 17(1):25–36
24. Vukobratovic M, Borovac B, Surla D, Stokic D (1990) Biped locomotion: dynamics, stability, control and application. *Scientific fundamentals of robotics 7*. Springer, New York
25. Yamane K, Nakamura Y (2003) Dynamics filter—concept and implementation of online motion generator for human figures. *IEEE Trans Robot Autom* 19(3):421–432
26. Yokoi K, Kanehiro F, Kaneko K, Fujiwara K, Kajita S, Hirukawa H (2001) A Honda humanoid robot controlled by AIST software. In: Proceedings of the IEEE-RAS international conference on humanoid robots, pp 259–264

# Hexapod Walking Robot Locomotion

Franco Tedeschi and Giuseppe Carbone

**Abstract** Path planning, gait planning and trajectory planning are assuming an increasing significance for Hexapod Walking robots and more generally, in legged robotics. Indeed, the trend for walking robots is to improve the speed, the stability, the navigation autonomy and the energy efficiency. In this Chapter it will be addressed the problem of Hexapod Walking Robots (HWR) locomotion. An overview of the State of the Art is carried out with references to the numerous contributions to this field. It will be provided a background on the topics of path planning, gait and trajectory planning for HWR locomotion. Special attention will be given to the hexapod gaits, starting from their classification together with a detailed description of most common ones. A case of study is described as referring to previous experiences at LARM in Cassino. Examples of a path planning, gait and trajectory planning are provided through kinematic and dynamic features of Cassino Hexapod leg operation.

## 1 Introduction

Hexapod Walking robots (HWR) are programmable mobile platforms on which six legs mechanisms are attached to the robot body [1]. HWR are controlled with a degree of autonomy that allows a robot to move within its environment, to perform intended tasks [2]. According to this definition, a degree of autonomy is required for HWR ranging from partial autonomy, including human robot interaction, to full autonomy without active human intervention [3].

HWR can include important features such as omnidirectional locomotion, variable geometry, good stability, lower impact on the terrain, great mobility in natural surroundings, fault tolerant locomotion [4]. HWR can overcome obstacles that are comparable with the size of the robot leg [5]. Operation features of HWR, such as example the locomotion with discrete contact points are especially important in

---

F. Tedeschi · G. Carbone (✉)

DICEM—Department of Civil and Mechanical Engineering,  
University of Cassino and South Lazio, Via Di Biasio, 46-03043 Cassino, FR, Italy  
e-mail: carbone@unicas.it

dangerous environments like mine fields [6], or scenarios where it is essential to keep the terrain undisturbed for scientific reasons as proposed for example in [7]. HWR have been used in exploration of remote locations and hostile environments such as seabed [8], in space environments or on planets exploration [9] in nuclear power stations [10], in search and rescue operations [11]. HWR have been also used in a wide variety of tasks such as forests harvesting [12], transport of cargo [13], welding [14] and climbing on vertical surface [15]. Despite the above referenced advantages and applications many challenges remain in the field of hexapodal locomotion. In fact HWR are still complex and slow machines, consisting of many actuators, sensors, transmissions and supporting hardware [16]. They are also often expensive and very difficult to operate by non-expert users [17].

This chapter gives an overview of the state of the art on hexapodal locomotion in order to identify the most significant features for a proper operation of a hexapod robot. Then, it provides insight on the topics of path planning, gait and trajectory planning for HWR locomotion. Special attention will be given to the hexapod gaits starting from their classification, and then providing a description of most commons ones. A case of study is described as referring to previous experiences at LARM in Cassino. Examples of a path planning, gait and trajectory planning are provided through kinematic and dynamic features of Cassino Hexapod leg operation.

## 2 State of Art Overview

As early 1899, Mybridge used successive photographs to study the locomotion of animals [18]. His work is regarded as classical in the study of walking gaits. The first hexapods walking robots can be identified as robots based on a rigidly predetermined motion so that an adaptation to the ground was not possible. Early researches in fifties were focused to assign the locomotion control completely by a human operator manually [19]. In [20] the theory of finite states was applied to legged systems and mathematical methods were used to analyse legged locomotion. In [21] it was developed the concept of a gait formula to describe symmetric gaits. It was also introduced the gait diagram to describe the gait of horses. In [22] it was defined the basic terminology for legged locomotion including the definitions of the duty factor, phase and stride length.

Oscillating systems generating the so called fixed gait have been used on a number of early hexapod robots. The tripod gait has been the first, and sometimes the only implemented gait. One of the first successful hexapod robot in which early gait planning approach was applied, was constructed at University of Rome in 1972 as a computer-controlled walking machine with electric drives [23]. This walking machine was able to do the straight-line movement only. In [24] it was numerically demonstrated that a regular and symmetric hexapod gait maximizes the longitudinal stability margin of all periodic gaits of a hexapod when using specific phase relationships.

In [25] it was developed a computer program to calculate the longitudinal stability margin of six-legged regular symmetric gaits. In 1976 Masha hexapod was designed at Moscow State University. Masha had a tubular axial chassis and articulated legs with three DoFs [26] and was able to negotiate obstacles using contact on the feet and a proximity sensor. Ohio State University in 1977 developed a six-legged insect-like robot system called OSU Hexapod [27]. The OSU hexapod was kept tethered and was made to walk short distances. A non periodic gait, called free gait was introduced in [28]. The free gait was suitable for locomotion over terrains which included regions not suitable for weight bearing. These regions must be avoided by the control system in deciding when and where to successively place the feet [29]. In 1983 Carnegie-Mellon University developed the first man-carrying hexapod capable of navigating rough terrain using different types of gaits [30]. This hexapod used a combination of hydraulic feedback, computer control and human control. In 1984, Odetic Inc., California, developed Odex I, a six-legged radially symmetric hexapod robot, which used an on board computer to play back pre-programmed motions [31]. Using remote human control or the pre-recorded motions, the hexapod could climb obstacles such as stairs or a pickup truck.

The Adaptive Suspension Vehicle (ASV) developed at the Ohio State University, USA, used wave and free gaits for omnidirectional walking and navigating over rough terrain [32]. This six legged robot, used hydraulic actuation powered by an internal combustion engine. A human was able to operate it through a joystick while the individual control of each leg was assured by a central computer. A gait named “follow the leader” was implemented and tested into the ASV Hexapod. In a “follow the leader” mode, the front legs steps on a selected foothold, the middle leg steps in the footprints of the front leg. The rear leg steps in the footprint of the middle leg. The selection of a foothold for the front leg was made by the human operator using a laser beam. By using this gait, the load on the human operator to specify foot placement when moving over difficult terrain was drastically reduced. A hexapod walking robot named Aquarobots was constructed in 1989 and used for underwater measurements of ground profiles for the construction of harbours. A specific gait planning was developed for seabed environments [33]. A small hexapod robot named Genghis was the first distributed control system with semiautonomous legs for a hexapod robot [34]. The behaviour and the locomotion of Genghis was not explicitly controlled, but was built by adding layers of control on top of existing simpler layer. This approach was different to the more traditional method of task decomposition [34]. A bio inspired approach of hexapod locomotion was developed in 1991 by TUM Walking Machine. The robot was similar to stick insect morphology [35]. The control system was realized as a neural structure. The system achieved a stable tripod gait for forward walking after a few cycles and was robust in the face of disturbances.

A novel type of free gait was studied for AMBLER (Autonomous MoBiLe Exploration Robot). This hexapod robot was developed by the Jet Propulsion Laboratory during the middle 1990s for operating under the particular constraints of planetary terrain. AMBLER had a circular body architecture in which the rear legs rotated through the body to become the front legs for the next step. Ambler’s legs remain

vertical, while they swing horizontally, adopting a telescope like displacement to touch the ground. A novel locomotion approach, most computationally intensive, was to search amongst possible future movement combinations for the one that will achieve optimal performance. This approach requires terrain information given by a laser scanner system [9].

The two last decades have been characterized by a rapid development of control systems technology. Hexapod robots were equipped with various sensing systems. Artificial Intelligence systems were widely applied to the analysis of environment and motion of robots on a complex surface. A series of bio inspired robots was developed at Case Western Reserve University (USA) at the end the 90s, such as for example Robot III that had a total of 24 DoFs. Robot III architecture was based on the structure of cockroach trying to imitate their behaviour and gait [36]. Several hexapod and interesting gait algorithms have been developed at University of Bruxelles. As example, MAX was a small hexapod robot with rectangular body, built for gait studies [38]. The Mechanical Engineering Laboratory in Osaka, Japan constructed MELMANTIS-2 [37] a hexapod robot adopting fixed gait with the ability to use their feet as hands. In [39] was addressed the foot-step planning problem by using a fault-tolerant gait with the capability to avoid forbidden regions, or regions where the robot cannot step safely. In 2001 it was developed a robot named RHex [40]. The RHex design consists of a rigid body with six compliant legs, each with one DoF. Thus, RHex had only six motors that rotate the legs such as a wheel. This solution is also known as whegs since this leg behaviour is very similar to a wheel. In [41] it was developed the hexapod robot named BILL-Ant-p. The robot was based on insect's behaviour. In [42] it was showed that some networks of six generic non-linear oscillators could exhibit the common walking gaits of insects, if each oscillator was connected to a leg. Some models named CPGs (Central pattern generators) have been implemented using the paradigm of neural networks or systems of coupled oscillators for controlling the locomotion of articulated hexapod robots. For example in [43] the hexapod Gregor I reproduce the cockroach's agility and the locomotion control was based on the theory of the Central Pattern Generator.

Hamlet was a hexapod robot constructed at the University of Canterbury, New Zealand [44] in order to study hexapod gait planning, force and position control on uneven terrain. A series of hexapod named LEMUR (Limbed Excursion Mechanical Utility Robots robot) was developed by Jet Propulsion Laboratory with the goals of using robots for repair and maintenance in near-zero gravity on the surface of spacecraft [45]. In 2004 a six-legged robot called ATHLETE was developed by the Jet Propulsion Laboratory [13]. The robot had the ability to roll rapidly on rotating wheels over flat smooth terrain and walk on fixed wheels over irregular and steep terrain. An interesting study of ATHLETE gait optimization can be found in [4]. The hexapod robot called RiSE was able to climb on a variety of vertical surfaces [46]. COMET hexapods are a series of robots designed to operate on extremely unstructured terrain [47]. Lauron V hexapod robot was the result of about ten years of progressive improvement on the previous configurations. LAURON hexapod was biologically-inspired by the stick insect [48]. The control architecture was based on neural networks and it allows the robot to climb stairs and ditches.

### 3 Basic Locomotion Issues

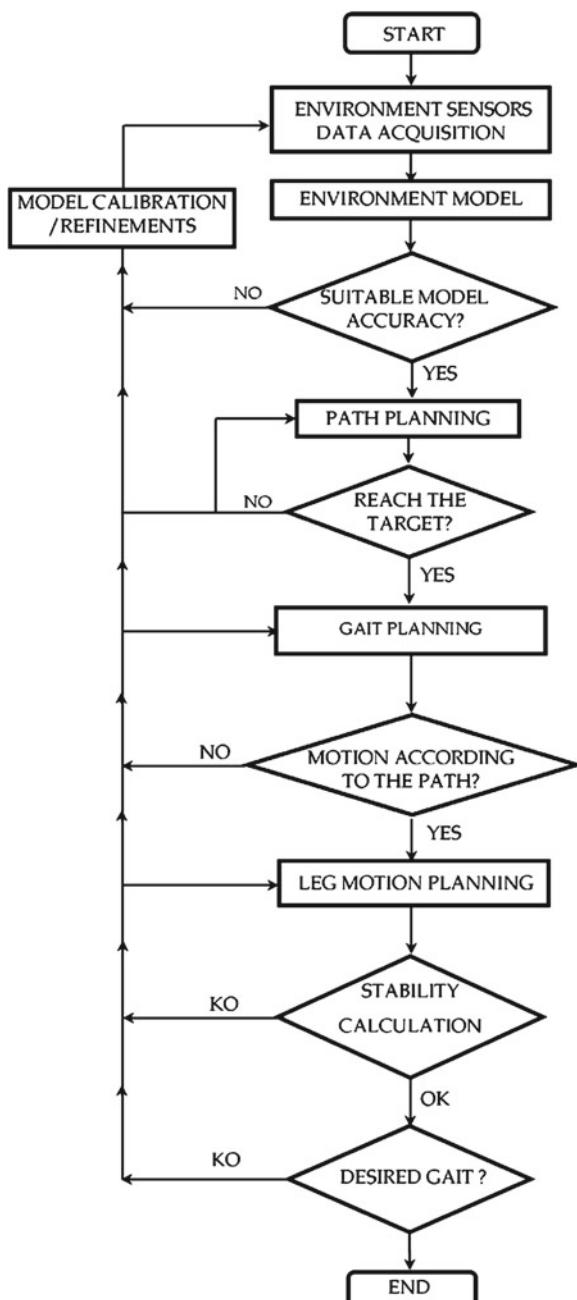
The above overview shows the significance of HWR and their locomotion features in the literature. A proper design and operation of the locomotion is getting even an increasing significance in legged robotics, since the trend for walking robots is to improve the speed, the stability, the navigation autonomy and the energy efficiency [49]. A proper design and operation of the locomotion requires to address several key aspects such as:

- path planning
- gait planning
- leg motion planning.

In order to introduce basic locomotion issues, Fig. 1 shows the control hierarchy that has been implemented in a wide number of HWR, such as outlined in [50]. As first step a proper environment model has to be built. One may range from a fully structured to a fully unstructured model with limited or significant need or sensory feedback. Then a path planning layer generates a geometric path, from an initial point A to a final point B, passing through pre-defined via-points, or respecting certain constraints [51]. The path planning layer supervises the lower gait planning level by providing information such as the desired direction, speed, posture. The degree of autonomy of path planning layer is ranging from partial autonomy, including human robot interaction, to full autonomy without active human robot intervention by means of artificial intelligence. Several sensors such as example laser range finder, sonar or cameras can be used in order to allow the environment analysis. Then, it is necessary to define a proper gait sequence in order to achieve the desired path and properly reach its final point B. The gait planning is a sequence of leg motions that is coordinated with a sequence of body motions for the purpose of transporting the body of the legged system from one place to another [32]. The gait planning layer provides movement commands for each leg to position them in a coordinated manner. Sensors such as example inertial measurements units, and force sensors can be used in order to allow the posture and stability calculation. The gait planning may be able to feedback to the task controller in case it cannot achieve the intended movement, for example due to an obstacle. In this case, task planner should attempt a different movement.

Leg motion planning has the responsibility of achieving the desired motion of each foot in terms of positions, velocities, and accelerations. Of course, a proper coordination is required among the legs to fulfil the intended gait. Thus, trajectory planning algorithms take a given geometric path and endow it with the time information. Trajectory planning algorithms influence not only the kinematic properties of the motion, but also the dynamic properties. Namely, the inertial forces and torques, to which the robot is subjected, depend on the accelerations along the trajectory.

**Fig. 1** Main sequence for achieving hexapod locomotion



### 3.1 Path Planning

The goal of the path planning problem is to find a collision-free motion between an initial (start) and a final configuration (goal) within a specified environment. The literature on robot path planning is very wide see for example [51–53]. Path planning is a general robotic problem. Thus, one can refer to well established robotic path planning approaches also in dealing with HWR. In the HWR locomotion, the problem of path planning can become very complex since robots are characterized by a large degree of autonomy and often must operate in hostile environments, such as example space, underwater, nuclear, demining, search and rescue. The complexity of HWR locomotion comes from the high DoFs that need to be controlled using limited knowledge of the environmental interaction during the robots gait [54].

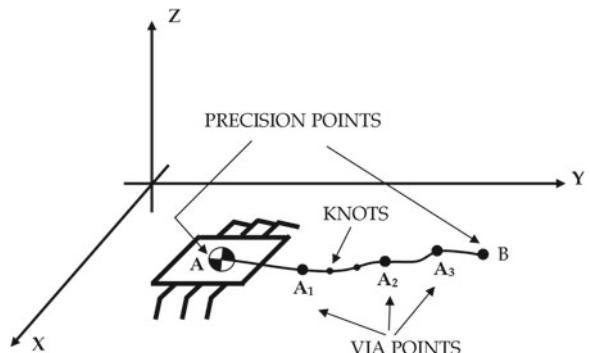
A general approach to path planning can be based on the definition of precision points and via points for the HWR centre of mass (COM), as reported in the scheme of Fig. 2, see for example [51]. The path between via points in Cartesian space can be further decomposed, for example, by using polynomials equations. In order to determine joint trajectories, it can be convenient to use interpolation functions such as with B-splines. One can use the given initial and final points in the Cartesian coordinates. B-splines are often used as interpolating functions to represent a trajectory of mechanical systems. An important characteristic is that they allow to control the degree of continuity between two adjacent segments. This fact is important because smooth transition is required for path planning. Another important characteristic of B-splines is that they satisfy the convex hull property, which allows the refinement of a trajectory.

Thus, each trajectory function  $\alpha_k(t)$  can be modelled by a uniform B-Spline in the form

$$\alpha_k(t) = \sum_{i=0}^m p_i^k B_{i,d}^k(t) \quad (M \geq 3, k = 1, 2, n) \quad (1)$$

where  $p_i^k$  (with  $i = 0, \dots, m$ ) are the  $m + 1$  control parameters of the knot points corresponding to the trajectory function  $\alpha_k(t)$ ;  $B_{i,d}^k(t)$  are the functions that can

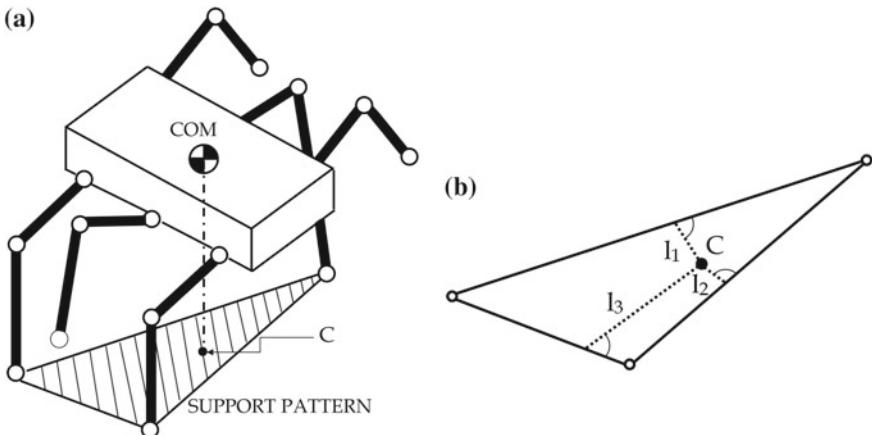
**Fig. 2** Basic path planning terms for HWR



be defined by using the Cox de Boor recurrence formulas [55]. B-spline can be conveniently formulated within optimization algorithms to achieve specific optimal behaviours, as proposed for example in [56].

The path planning task for a hexapod leg with  $n$  DoFs can be described using  $m$  knots in the trajectory of each  $k$ th joint of a manipulator. The prescribed task can be given by the initial and final points A and B of the trajectory. The movement of the leg can be obtained by the simultaneous motion of the  $n$  joints in order to perform the prescribed task. Among the many available optimality criteria, one can assume for HWR the energy consumptions as one of the most significant performance in order to optimize the manipulator operation, since the energy formulation can consider simultaneously dynamic and kinematic characteristics of the performing motion.

Another key issue to be carefully verified and optimized is the stability. This aspect is critical for a proper HWR locomotion. (Due to the space limits this chapter just mentions the main issues while readers should refer to the wide literature for further insight, see for example [49]). HWR locomotion can be classified into dynamic locomotion, such as running and hopping and statically stable locomotion as walking. Statically stable locomotion has the constraint that the moving body is stable at all times. The vertical projection of the centre of gravity of the robot must be at all times within the support pattern of the legs which have ground contact, see Fig. 3a. Dynamic stability is needed when the CoM is outside or on the border of the support pattern. When the Centre of Mass is outside the support pattern the robot will fall over when no additional forces and movement are made with the legs. There are some important definitions that are related to stability. First definition is the stability margin  $S_m$ . For an arbitrary support pattern, the stability margin is the shortest distance from the vertical projection of the centre of gravity to any point on the boundary of the support pattern in the horizontal plane [29], see Fig. 3b.



**Fig. 3** Stability in hexapod robot. **a** General view. **b** Stability margin and support pattern

$$S_m = \min(l_1, l_2, l_3) \quad (2)$$

The front stability margin and the rear stability margin describe distances from the vertical projection of the centre of gravity to the forward and rearward boundaries of the support pattern, respectively [29]. Finally, the longitudinal stability margin  $S_1$ , is defined as the shortest distance from the body's centre of mass to the boundary of the support pattern, measured in the direction of travel [29].

Several approaches have been proposed in order to define a suitable criterion for dynamic tip-over stability evaluation, such as the Zero Moment Point [57], Energy-Based measure [58], the force-angle margin [59], the Moment-Height Stability measure [60]. An interesting description of a number of stability criteria for HWR, can be found in [61]. In trajectory-based control, one major criterion in the generation of joint trajectories is the position of the Zero-Moment Point (ZMP). ZMP is defined as the point on the ground where the net moment of the inertial forces and the gravity forces has no component along the horizontal axes. For dynamic stable locomotion, the necessary and sufficient condition is to have the ZMP in the support polygon at all stages of the locomotion gait [62].

Usually HWR legs are having 3 DoFs each so that it is necessary to control 18 joints ( $3 \times 6$ ). The state space,  $\bar{q}$  for the hexapod then lies in 18 dimensions.  $\bar{q}$  state space consists of configurations which satisfy the joint limits, without considering collisions of the vehicle with itself or the environment. Of course not all configurations in the state space  $\bar{q}$  are valid for stable navigation.

Cell decomposition, potential field method and roadmap planning are some of the most widely used methods for path planning [63]. The cell decomposition methods [64], subdivides the free space of the robot into several regions, called cells, in such a way that a path between any two configurations lying in the same cell is straightforward to generate. The so-called connectivity graph, represents the adjacency relations between cells. Namely, the nodes of the graph represent the cells extracted from the free space, and there is an arch between two nodes is connected if and only if the corresponding cells are adjacent. The path planning problem is turned into a graph searching problem, and can therefore be solved using graph-searching techniques.

The basic idea of artificial potential methodologies [65, 66] is to consider the robot in the configuration space as a moving point subject to a potential field generated by the goal configuration and the obstacles in the  $\bar{q}$ -space: namely, the target configuration produces an attractive potential, while the obstacles generate a repulsive potential. The sum of these two contributions is the total potential, which can be seen as an artificial force applied to the robot, aimed at approaching the goal and avoiding the obstacles. Thus, given any configuration during the robot motion, the next configuration can be determined by the direction of the artificial force to which the robot is subjected. This normally represents the most promising direction of motion in terms of free path.

The roadmap techniques are based upon the reduction of the N-dimensional configuration space to a set of one-dimensional paths to search, possibly on a graph. The graph can then be searched in order to get the optimal solution to the path planning problem (in most cases, this is represented by the shortest path). Examples of path

planning algorithms may be found in [67, 68]. Another approach called probabilistic roadmap method is often suggested and applicable with high degrees of freedom, but is computationally expensive as the degrees of freedom increase [69]. Additionally, many other strategies exist for motion and path planning that are based on sampling within a configuration space. Path planning using a rapidly exploring random tree (RRT) algorithms is an example. An implementation of a RRT algorithms in HWR can be found for example in [70].

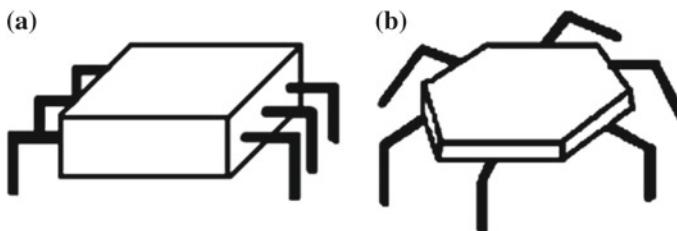
### 3.2 Gait Planning

For HWR, developing walking patterns is a challenging task, because there are a large number of degrees of freedom and therefore the solution must be found in a large, multidimensional search/state space [71]. Gait planning for hexapod walking robots is a very complex activity in which kinematic, dynamic and control architecture are playing an important role together with the HWR body architecture characteristics. HWR may have two basic body shapes: rectangular and hexagonal as in Fig. 4. The first one has six legs distributed symmetrically along two sides, each side having three legs. The second has legs distributed axi-symmetrically around the body, in a hexagonal or circular shape. Bilateral symmetry may be better suited than radial symmetry to move along a straight line. Rectangular architectures require a special gait for a turning action. Generally, they need four steps in order to achieve a turning action [72]. True radial symmetry implies that all legs are equal and the body has no ‘front’ or ‘rear’ thus there is no preferential direction for the motion [73].

There are many types of gaits for hexapod robot, since the possible combinations to move a single leg or pair of legs are large.

In [22] it was calculated that for a machine with  $k$  legs, the number of distinct permutations corresponding to different combinations of placement of the legs on the ground and the possible sequences of lifting the legs is equal to

$$N(k) = (2k - 1)! \quad (3)$$



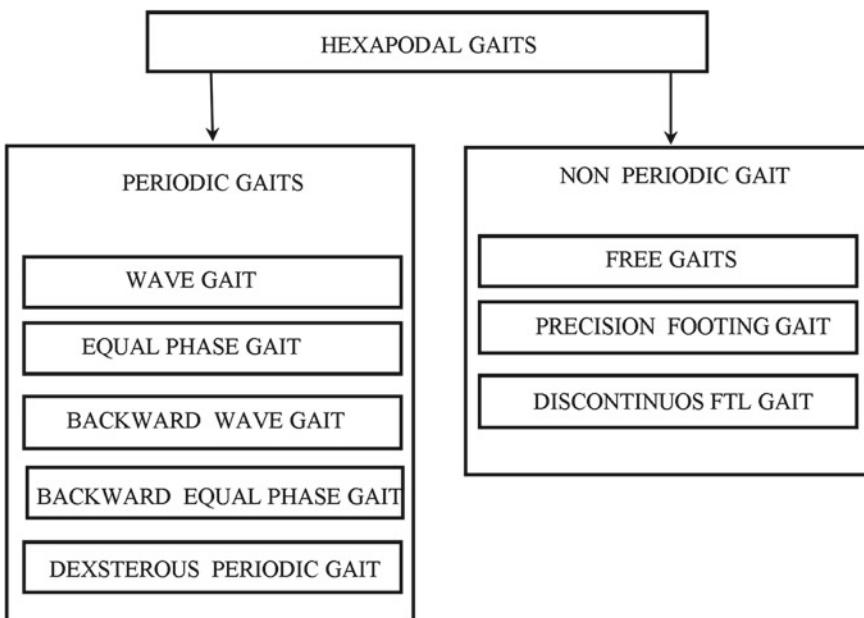
**Fig. 4** Shape of hexapod walking robots. **a** Rectangular. **b** Hexagonal

Therefore for a hexapod,  $N(6) = 39.916.800$ . Most of these permutations or sequences of theoretical events are not really feasible to implement walks or gaits. This is because they require a number of legs not sufficient to ensure adequate stability or because involving sequences that have not significant differences or advantages compared to the regular and symmetrical gaits. In practice, over of approximately 40 million possible gaits for a hexapod, only about thirty of them are valid gaits [74].

The gait selection is a very complex problem that depends on several factors such as example, terrain type, stability requirements, speed requirements, mobility requirements, and power requirements. Figure 5 reports a scheme of gait classification such as proposed for example in [29, 49]. Gaits are classified in two main types: periodic and non-periodic. A gait is periodic if the legs always have the same mode at the same point in their cycle and all the leg cycles have the same length. Periodic gaits repeat the same sequence of steps every cycle [22]. A periodic gait is one in which every limb operates with the same cycle time. Otherwise, the gait is non-periodic.

In regular gaits, all the legs spend the same amount of time per cycle on the ground (they have the same duty cycle  $\beta$ ). A gait is considered regular if all legs have the same duty factor, as when

$$\beta_i = \beta_j = \beta \quad \text{where } i, j = 1, 2, \dots, n \text{ is the leg number} \quad (4)$$



**Fig. 5** Main types of HWR gaits

A gait is symmetric if the motion of the legs of any right-left pair is exactly half a cycle out of phase.

Typical gaits observed in stick insects are based on five concepts [75]:

- a wave of protractions runs from posterior to anterior;
- contralateral legs of the same segment alternate in phase;
- protraction time is constant;
- retraction time decreases as walking frequency increases;
- the intervals between steps of ipsilateral adjacent legs are constant, while the interval between the foreleg and hind leg steps varies inversely with frequency.

Gaits of this family are called wave-gaits and show a periodic behaviour with a  $\beta \geq 0.5$  equal for all the legs. The most important characteristic of these patterns is their capability to ensure a statically balanced motion on a wide speed range. Another important feature is the possibility to adapt the gait to speed variations continuously. The general form of wave gait for a 2n-legged robot was described for example, in [29]:

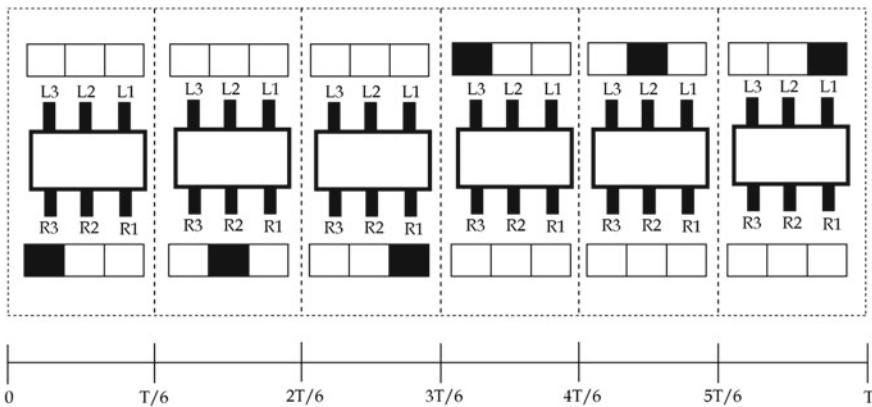
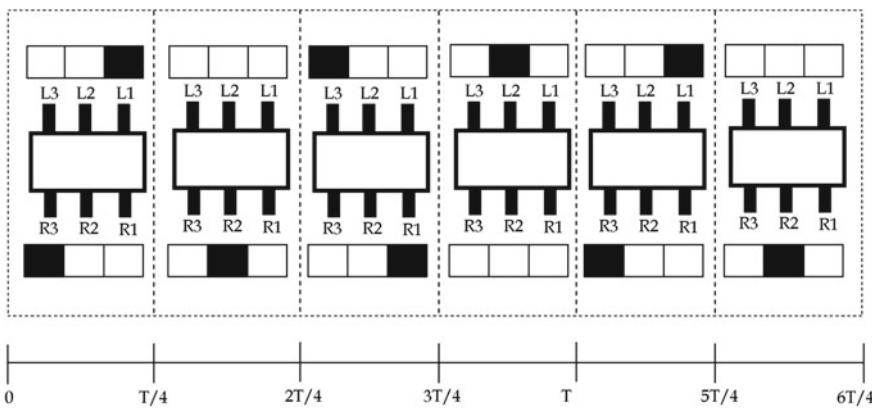
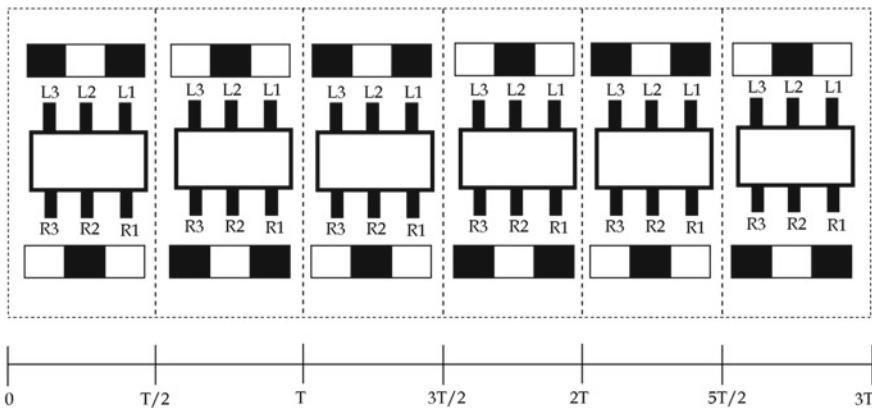
$$\Phi_{2m+1} = F(m\beta), m = 1, 2, \dots, n-1, \frac{3}{2n} \leq \beta < 1 \quad (5)$$

where  $F(X)$  is the fractional part of the real number  $X$ , and  $m$  denotes successive legs on the left side numbered from the back. In [25] it was discovered that the gait stability margin of a 2n legged regular symmetric gait is maximized by a wave gait. An interesting application of equal phase gait was developed in the ASV project in order to distribute power consumption as evenly as possible over a cycle. The stepping sequences and the general form of backward gait and dexterous periodic gait were defined for example in [29].

Over uneven ground the time of footfall becomes unpredictable breaking the rigorously defined leg phasing and compromising stability (Fielding 2002). In non-periodic gaits, so-called free gaits, there is no predetermined stepping sequence. Algorithms or rules determine the most appropriate stepping sequence as the situation requires for maximizing stability and/or motion. Free gaits are more suitable than periodic gaits when the movement command changes often, when walking over rough ground, and when the application demands precision stepping.

Figures 6, 7 and 8 show the gait diagrams for the most common HWR wave gaits [76]: metachronal, ripple and tripod. In the figures the white colour indicates that the foot is in ground contact and the black colour otherwise.

The metachronal gait, illustrated in Fig. 6 is adopted by the hexapod when it moves slowly. This gait can be described as a back to front propagating wave, first moving the limbs on the right side and then the limbs on the left side. In the metachronal gait, all legs on one side are moved forward in succession, starting with the rear-most leg. This is then repeated on the other side. Since only 1 leg is ever lifted at a time, with the other 5 being down, the robot is always in a highly-stable posture. The metachronal gait will be most stable gait, since it keeps the most legs on the ground at all phases of the stride. It will also be the easiest to adjust during movement over uneven terrain.

**Fig. 6** Metachronal gait**Fig. 7** Ripple gait**Fig. 8** Tripod gait

The ripple gait is used by the hexapod to move with a medium speed where each foot is on the ground according to the gait diagram shown in Fig. 7. During this gait L1 and R3 start together the movement, then after one quarter of a period moves R2, more a quarter of a cycle begin their movement R1 and L3 and finally after more quarter of a period moves L2. The ripple gait is next most-stable. At most, only 2 legs are ever off the ground at the same time.

The Tripod Gait (Fig. 8) is the best-known hexapod gait. A tripod consists of the front-back legs on one side and the middle leg on the opposite side. For each tripod, the legs are lifted, lowered, and moved forwards and backwards synchronously. During walking, a hexapod uses its 2 tripods like a biped stepping from one foot to the other: the weight is simply shifted alternately from one tripod to the other. Since 3 legs are on the ground at all times, this gait is both statically and dynamically stable. Tripod gait is suitable for high speed walking over relatively flat ground. Of course periodic gaits such as the tripod gait, require synchronization between the expected and actual times that legs make contact with the ground.

### 3.2.1 Leg Motion Planning

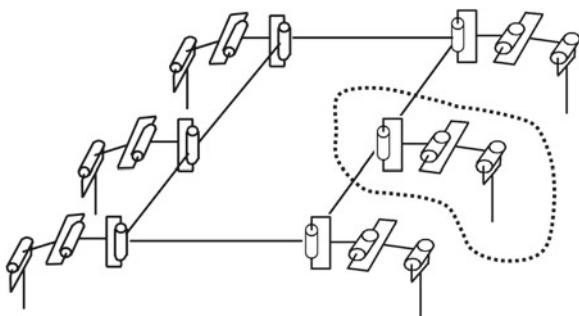
The leg motion planning requires the definition of a proper kinematic model for the whole hexapod (Fig. 9). Accordingly, some basic definitions are needed to introduce the leg motion planning.

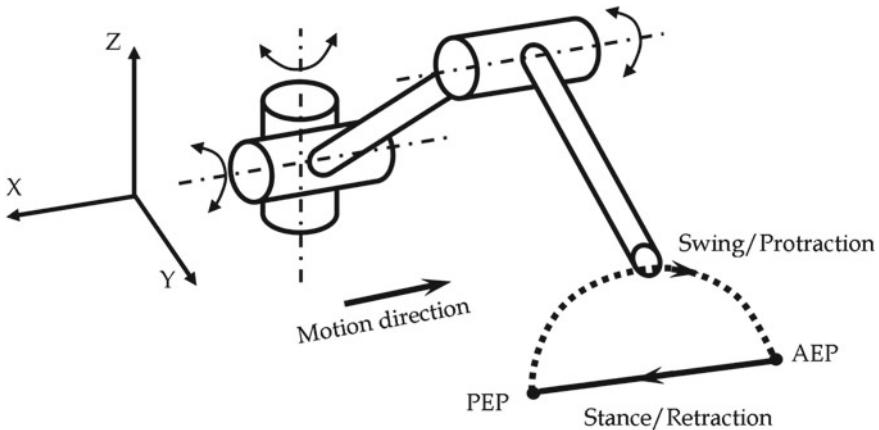
In particular it is necessary to define:

- leg cycle;
- stride;
- leg stroke;
- gait parameters;
- relative phase.

The leg cycle is the sequence of lifting and placing the legs. A gait is cyclic when the sequence of lifting and placing the legs is repeated within a given amount of time that is called time cycle ( $T$ ). A leg cycle can be divided into two phases: the stance phase (also called support phase or retraction) in which the leg is on the ground;

**Fig. 9** A kinematic model for leg motion planning





**Fig. 10** Leg cycle basic definitions (The arrows on the trajectory show the direction of foot movement through the leg cycle)

the swing phase (also called transfer phase or protraction) in which the leg is in the air [2]. The stance and swing phase have independent durations and their sum gives the time cycle  $T$ . Basically, the stance phase corresponds to the time interval in which the limb is in ground contact and the body is propelled. During the transfer phase, the leg is moved from one foothold to the next. Figure 10 shows a foot kinematic scheme while a dotted line shows the foot trajectory during a basic leg cycle. The transition between the phases happens at the anterior extreme position (AEP) and posterior extreme position (PEP) [77].

The leg stride is a complete cycle of leg movements from a particular leg movement to the next occurrence of the same leg movement; the stride length is the distance the centre of gravity of the body travels during one stride [78].

The leg stroke is the distance the foot travels, relative to the body, during the support phase [29].

A gait can be characterized by the concepts of cycle time ( $T$ ), duty factor ( $\beta$ ) and relative phase  $\theta_{ji}$ . The cycle time  $T$  is the time required for a complete cycle of leg locomotion of a periodic gait. The duty factor  $\beta$ , is the fraction of the cycle time in which the leg  $i$  is in the support phase [29]. In other word the duty factor  $\beta$  is defined as the fraction of the duration of the step cycle for which a foot is on the ground.

$$\beta = \frac{\text{time of support phase of leg } i}{\text{cycle time of leg } i} = \frac{t_{st}}{T_i} \quad (6)$$

The duration of the stance phase  $T_{st}$  is defined as

$$T_{st} = \beta T \quad (7)$$

In swing phase the leg is moved to the starting point of the next standing phase. The duration of the swing phase  $T_{sw}$  can be calculated as

$$T_{sw} = (1 - \beta)T \quad (8)$$

The relationship between the stance swing phase duration and the cycle time  $T$  is

$$T = T_{st} + T_{sw} \quad (9)$$

accordingly, can be calculated as the duty factor  $\beta \in [0, 1]$ ,

$$\beta = \frac{T_{st}}{T_{st} + T_{sw}} \quad (10)$$

In order to define the relative phase, we follow usual limb conventions [76], the limbs of the left (L) and right (R) sides of the hexapod are numbered from front to back. The sub index stands for the limb number: 1 is the front leg, 2 is the middle leg and 3 is the rear leg (Fig. 11).

It is considered that the first event, and the start of the stride, is chosen as the reference event when an arbitrary chosen reference limb is set down. The convention used here is that the reference limb is the right rear leg (R3). The relative phase of leg  $\theta_i$  is defined as the time elapsed from the setting down of a chosen reference foot until the foot of leg  $i$  is set down, given as the fraction of the cycle time. Thus, consider as reference the right rear limb (R3), the relative phase for all the limbs is given by

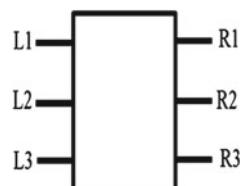
$$\theta_i = \frac{\Delta t_i}{T} \quad (11)$$

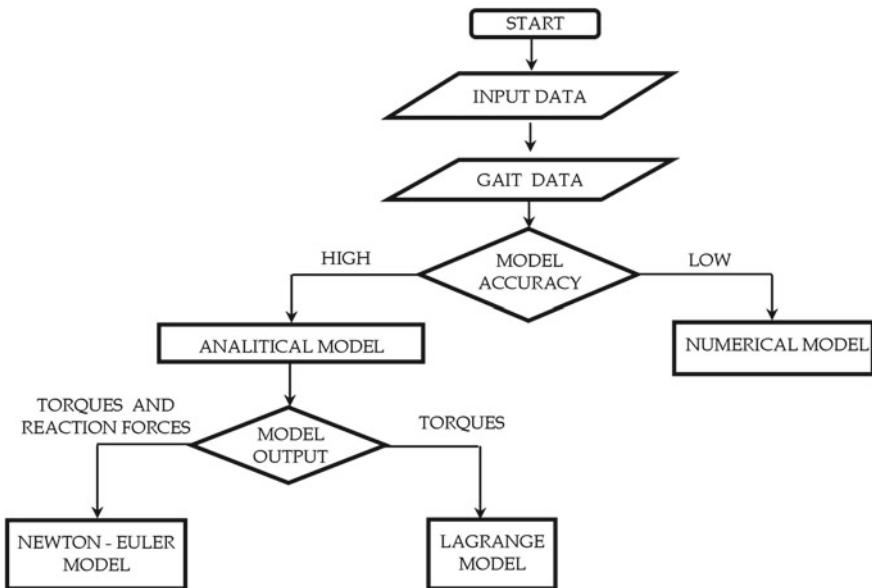
where  $\theta_i \in [0, 1]$ , because  $\Delta t_i \leq T$  is the time delay between the placing events of the right rear leg and leg  $i$ .

Following the above definitions one should develop proper kinematic and dynamic models of a full hexapod robot and specifically of a single leg. Further discussion on forward and inverse kinematics, for several configuration of HWR such as mammalian, reptile or arachnid can be found, for example, in [1].

Figure 12 shows a flow-chart in order to choose a proper dynamic model of HWR. At first step input data such as sizes, masses, inertias, payload, initial configuration and gait data will be defined. The solution of the dynamic model can be

**Fig. 11** Leg convention





**Fig. 12** Flow-chart for HWR modelling

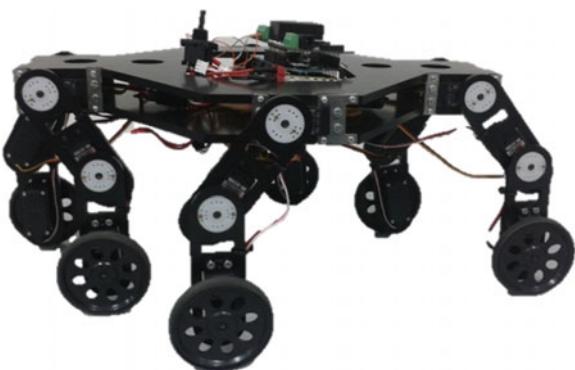
achieved either analytically or numerically. The choice of model depend of the needed model accuracy. The analytical solution can be obtained both by means of Lagrange or Newton-Euler methods. The Lagrange Methods require the computation of the Lagrangian  $L$  as proposed for example in [51]. These methods cannot be used for the calculation of reaction forces and usually neglects friction. The dynamic behaviour of HWR trough analytical approach can be modelled as proposed for example in [80].

Numerical methods are usually approximate approaches. They allow calculation of reaction forces with reduced computational costs. Numerical methods can be also implemented through commercial software that can be very useful at design stage. SolidWorks and MSC ADAMS environments can be used due to their convenient features for 3D CAD drawing, structure analysis and operation analysis of multi-body systems in order to check the feasibility of a prototype in a virtual environment. In particular, a model in MSC ADAMS can take into account several aspects such as, external forces, gravity, contacts constraints, friction, and inertia properties.

## 4 A Case of Study

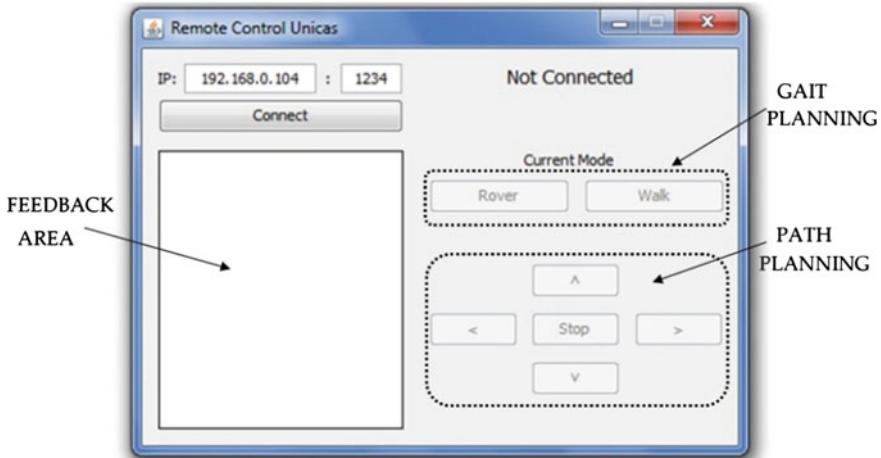
In a recent past, research activities have been undergoing at LARM, Laboratory of Robotics and Mechatronics of Cassino and Southern Lazio University, for developing six-legged robots within the so called “Cassino Hexapod” series. The main features

**Fig. 13** The Cassino Hexapod II



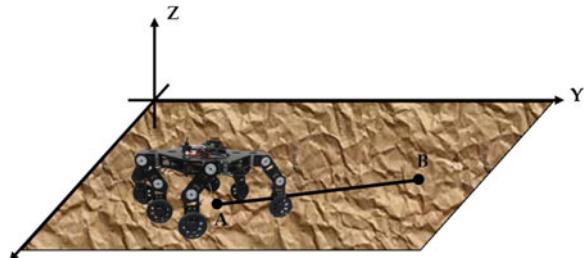
of the proposed design solutions have been the use of low-cost mechanism architectures and user friendly operation features. Cassino Hexapod is a mobile hybrid leg-wheel robot, whose intended main application task is the inspection and analysis of historical sites (Fig. 13). In particular, the robot should be able to move inside archaeological and/or architectural sites by carrying surveying devices and by avoiding damage to the delicate surfaces or historical items of the site. Additionally, the robot should be able to operate also in environments that cannot be reached or that are unsafe for human operators. The Cassino Hexapod is equipped with a wireless inertial measurement unit sensors LPMS-B OEM [81] in order to control the stability. This unit can measure orientation about all three global axes by using three different sensing units: a 3-axis gyroscope, a 3-axis accelerometer and a 3-axis magnetometer. The LPMS-B unit combines the orientation information from the three sensing units using a complementary filter in conjunction with an extended Kalman filter. A commercial Wi-Fi camera with a resolution of  $640 \times 480$  and weight of 0.1 kg has been assembled under the robot main body in order to acquire image and videos of the operating scenarios. Further details of sensors, actuators and control architecture can be found in [82–84].

Path and gait planning of Cassino Hexapod includes human robot interaction by means of a Wi-Fi network. Figure 14 shows a scheme of user interface that provides the path planning, gait planning and feedback information of the robot. In particular the path planning layer of Cassino Hexapod robot finds a collision-free motion between an initial and a final configuration within a specified environment. Figure 15 shows an example of linear path-planning between the initial (A) and final position (B). The camera allows to collect environments data in order to identify if there are obstacles for a proper path planning. The gait planning layer allows to select a walking strategy by selecting, for example, a tripod gait or moving by using wheels in order to improve the speed on a flat terrain. In the walking mode, the tripod gait has been adopted since it is suitable for high speed walking over relatively flat ground. The gait controller selects which legs to lift and at what time to lift them; plans trajectories for all the legs; maintains an appropriate body orientation and clearance with respect to the ground. Figure 16 report three frames by referring to the first tripod in



**Fig. 14** User interface of Cassino Hexapod

**Fig. 15** Path planning of Cassino Hexapod



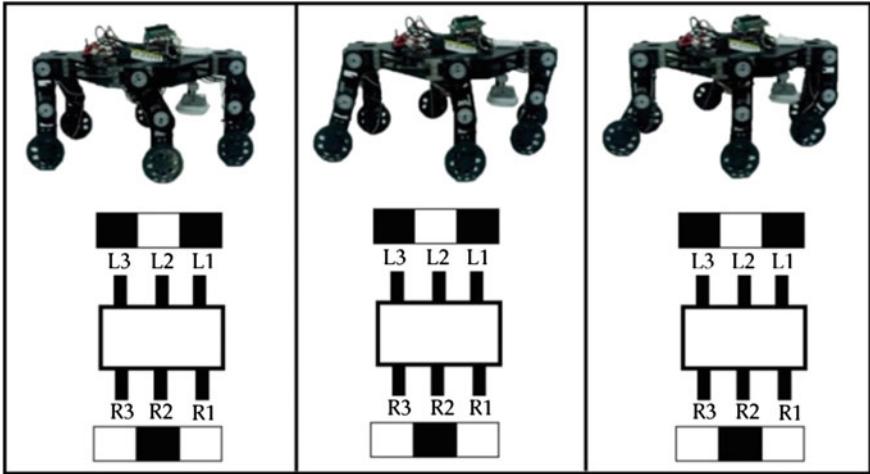
a regular periodic tripod gait of Cassino Hexapod II. The anterior and posterior legs on one side lift synchronously with the contralateral middle leg, forming alternating tripods. Figure 17 shows the next three frames by referring to the second tripod.

In order to describe the leg motion planning of Cassino Hexapod now we show the physical modelling of robotics legs. We identify path planning forces which cause a prescribed motion of the effector. The task is to investigate the necessary movement in kinematic pairs which would ensure the relocation of the end point  $p$  of two links from position  $t = 0$  to position  $t = t_f$  (Fig. 18). At the beginning and at the end of the movement the velocity and acceleration of point  $p$  should be zero [85]. We find the time diagram of angular rotations and in the form of 5th degree polynomials, such as outlined in [86]:

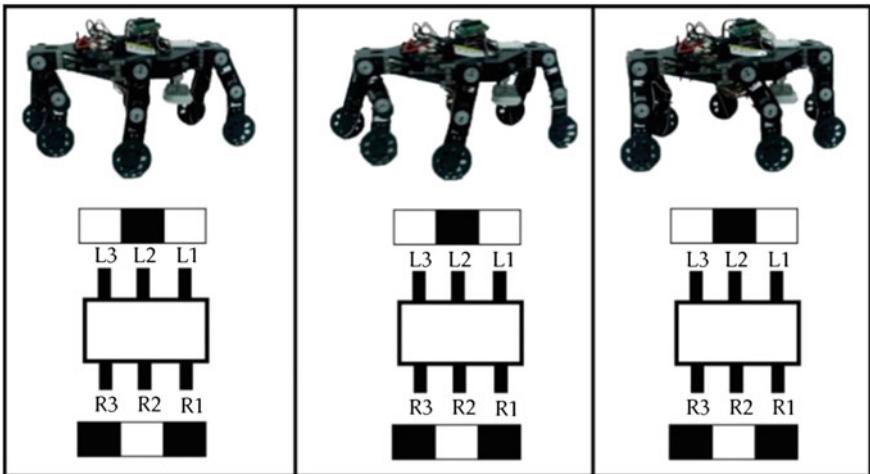
$$\theta_1(t) = a_1 t^5 + a_2 t^4 + a_3 t^3 + a_4 t^2 + a_5 t + a_6 \quad (12)$$

$$\theta_2(t) = b_1 t^5 + b_2 t^4 + b_3 t^3 + b_4 t^2 + b_5 t + b_6 \quad (13)$$

The constants in the polynomials are determined by the initial and end conditions of the endpoint  $p$  for  $l_1 = 70$  [mm],  $l_2 = 70$  [mm],  $m_1 = 0.05$  [kg];  $m_2 = 0.05$  [kg]



**Fig. 16** Waking strategy in a tripod gait on Cassino Hexapod: frames by the first tripod



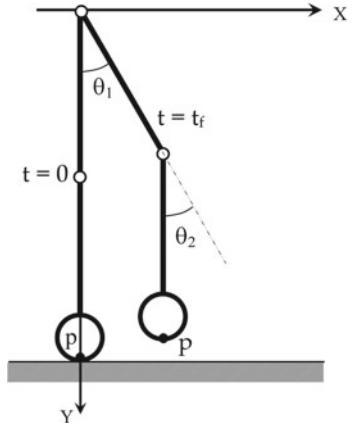
**Fig. 17** Waking strategy in a tripod gait on Cassino Hexapod: frames by the second tripod.

where  $l_i$  is the length of  $i$ th link and  $m_i$  is the mass of the  $i$ th actuator. The degree of these polynomials is chosen so that they are able to express the desired conditions on the initial and final acceleration [87]. After substituting the initial and final conditions we obtain the following values of their respective constants in the form

$$\theta_1(t = 0) = 0 \rightarrow a_6 = 0 \quad (14)$$

$$\theta_2(t = 0) = 0 \rightarrow b_6 = 0 \quad (15)$$

**Fig. 18** Scheme of a leg motion



$$\dot{\theta}_1(t=0) = 0 \rightarrow a_5 = 0 \quad (16)$$

$$\dot{\theta}_2(t=0) = 0 \rightarrow b_5 = 0 \quad (17)$$

$$\ddot{\theta}_1(t=0) = 0 \rightarrow a_4 = 0 \quad (18)$$

$$\ddot{\theta}_2(t=0) = 0 \rightarrow b_4 = 0 \quad (19)$$

By referring to Fig. 18 we are considering for example, the displacement from the initial straight leg configuration at time ( $t = 0$ ) to final configuration at time ( $t_f = 1\text{s}$ ) having leg joints coordinates  $\theta_1 = 30^\circ$  and  $\theta_2 = -30^\circ$ . Accordingly, we obtain the constants  $a_6$ ,  $a_5$ ,  $a_4$  for  $\theta_1$  and  $b_6$ ,  $b_5$ ,  $b_4$  for  $\theta_2$  motion such as

$$\theta_1(t_f) - \theta_1(t=0) = \theta_1(t_f) = a_1 t_f^5 + a_2 t_f^4 + a_3 t_f^3 = \frac{30 \cdot \pi}{180} \quad (20)$$

$$\dot{\theta}_1(t_f) = 5a_1 t_f^4 + 4a_2 t_f^3 + 3a_3 t_f^2 = 0 \quad (21)$$

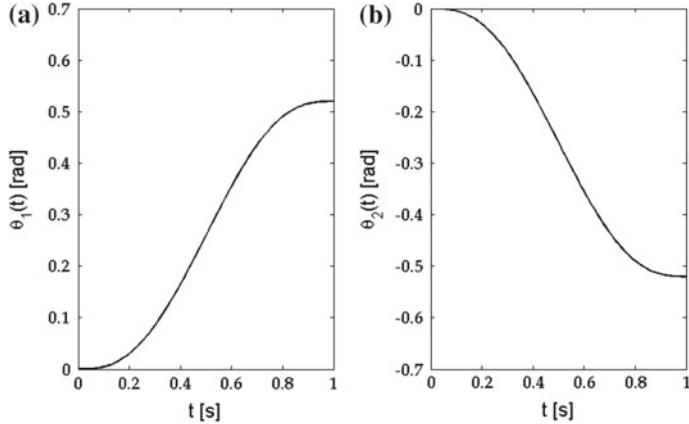
$$\ddot{\theta}_1(t_f) = 20a_1 t_f^3 + 12a_2 t_f^2 + 6a_3 t_f = 0 \quad (22)$$

Then, in the matrix form one can write

$$\begin{bmatrix} t_f^5 & t_f^4 & t_f^3 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 \\ 20t_f^3 & 12t_f^2 & 6t_f \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \frac{30\pi}{180} \\ 0 \\ 0 \end{bmatrix} \quad (23)$$

In a similar way, one can write

$$\theta_2(t_f) - \theta_2(t=0) = \theta_2(t_f) = b_1 t_f^5 + b_2 t_f^4 + b_3 t_f^3 = -\frac{30 \cdot \pi}{180} \quad (24)$$



**Fig. 19** Time diagram of angular rotation. **a**  $\theta_1(t)$ . **b**  $\theta_2(t)$

$$\dot{\theta}_2(t_f) = 5b_1 t_f^4 + 4b_2 t_f^3 + 3b_3 t_f^2 = 0 \quad (25)$$

$$\ddot{\theta}_2(t_f) = 20b_1 t_f^3 + 12b_2 t_f^2 + 6b_3 t_f = 0 \quad (26)$$

Then, in matrix form

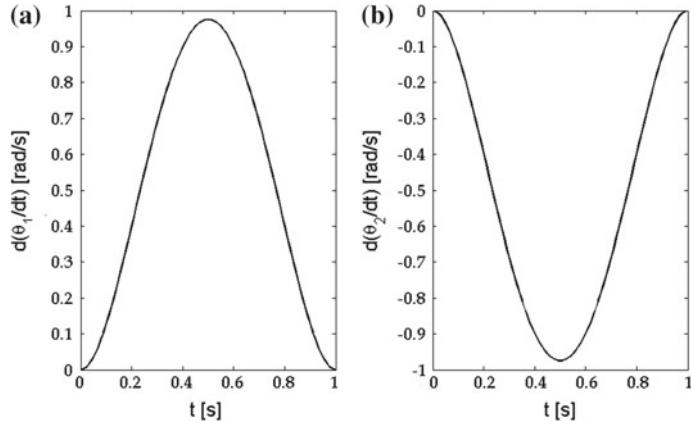
$$\begin{bmatrix} t_f^5 & t_f^4 & t_f^3 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 \\ 20t_f^3 & 12t_f^2 & 6t_f \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \frac{30\pi}{180} \\ 0 \\ 0 \end{bmatrix} \quad (27)$$

Solving the systems provide the values of  $a_1, a_2, a_3, b_1, b_2, b_3$

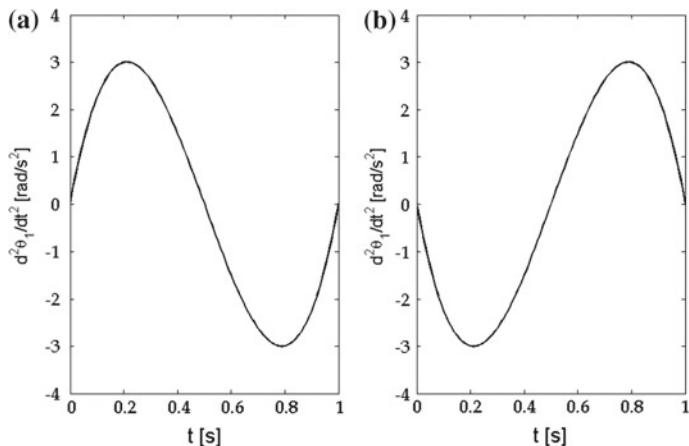
$$\begin{aligned} a_1 &= 3.12 & b_1 &= -3.12 \\ a_2 &= -7.8 & b_2 &= 7.8 \\ a_3 &= 5.2 & b_3 &= -5.2 \\ a_4 &= 0 & b_4 &= 0 \\ a_5 &= 0 & b_5 &= 0 \\ a_6 &= 0 & b_6 &= 0 \end{aligned} \quad (28)$$

Figure 19 shows a time diagrams of angular rotation  $\theta_1(t)$  and  $\theta_2(t)$ . Figure 20 reports the time diagram of velocities  $\dot{\theta}_1(t)$  and  $\dot{\theta}_2(t)$ . Figure 21 shows the time diagram of accelerations  $\ddot{\theta}_1(t)$  and  $\ddot{\theta}_2(t)$ . In Fig. 22 is the leg trajectory on the xy plane.

After the definition of the desired path we identify the forces which cause a prescribed motion of the foot. For this purpose we define a model of double pendulum as proposed in Fig. 23, with the main objective to develop a dynamic model of a single robotic leg. The method used is based on the formulation of the Euler-Lagrange [51]:



**Fig. 20** Time diagram of velocity. **a**  $\dot{\theta}_1(t)$ . **b**  $\dot{\theta}_2(t)$



**Fig. 21** Time diagram of acceleration. **a**  $\ddot{\theta}_1(t)$ . **b**  $\ddot{\theta}_2(t)$

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = \tau_i \quad i = 1, 2, \dots, n \quad (29)$$

in which

$L$  = Lagrangian =  $T - U$

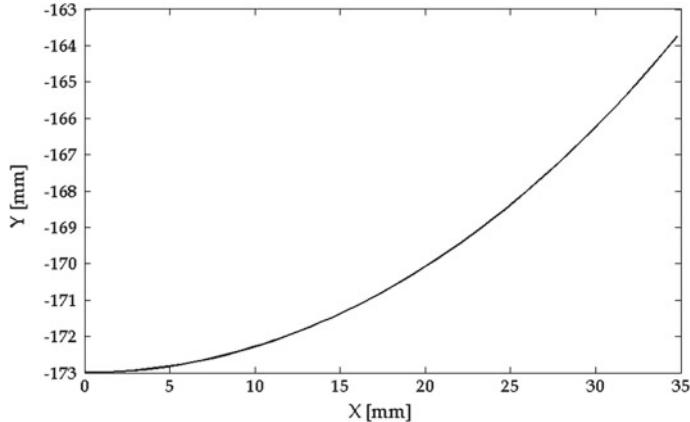
$T$  = total kinetic energy of the system

$U$  = potential energy of the system

$q_i$  = generalized coordinates

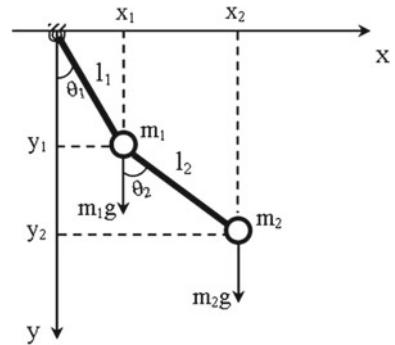
$\dot{q}_i$  = time derivatives of the generalized coordinates

$\tau_i$  = generalized force (or moment) applied by the system to joint  $i$  in order to move the link  $i$ .



**Fig. 22** Leg trajectory

**Fig. 23** A leg with mass distribution in a swing phase



A significant advantage of the Lagrangian approach to developing equations of motion for complex systems comes as we leave the cartesian  $x_i$  coordinate system and move into a general coordinate system. However, the Lagrange approach cannot be used for the calculation of reaction forces and usually neglects friction.

The general formulation in Eq. (29) can be used to determine the torques that are needed to achieve a the prescribed movement of the leg. Inputs are the prescribed movements of its individual members that moved from initial position to desired position along the trajectory that has been determined in the path planning. The solution of Eq. (29) can be obtained by defining the kinematic joint coordinates as function of the generalized coordinates such as in Eqs. (30) and (31).

We consider the double pendulum shown in Fig. 23 consisting of two links of length  $l_1$  and  $l_2$  with mass points of motors  $m_1$  and  $m_2$ . It is to note that the proposed model refers to the swing phase. This system has two degrees of freedom:  $\theta_1$  and  $\theta_2$  defined by the generalized coordinates.

In order to evaluate the Lagrangian, we must obtain the kinetic and potential energies in terms of the generalized coordinates  $\{\theta_1, \theta_2\}$  and their corresponding

velocities  $\{\dot{\theta}_1, \dot{\theta}_2\}$ . We therefore express the Cartesian coordinates  $\{x_1, y_1, x_2, y_2\}$  in terms of the generalized coordinates  $\{\theta_1, \theta_2\}$ :

$$x_1 = l_1 \sin \theta_1 \quad y_1 = -l_1 \cos \theta_1 \quad (30)$$

$$x_2 = l_1 \sin \theta_1 + l_2 \sin \theta_2 \quad y_2 = -l_1 \cos \theta_1 - l_2 \cos \theta_2 \quad (31)$$

Similarly, the components of velocity can be calculated by computing the time derivatives of Eqs. (30) and (31).

The potential energy is

$$U = m_1 g y_1 + m_2 g y_2 = m_1 g l_1 \cos \theta_1 - m_1 g (l_1 \cos \theta_1 + l_2 \cos \theta_2) \quad (32)$$

For the kinetic energy  $T$  we know that:

$$T = \frac{1}{2} m v^2 = \frac{1}{2} m (\dot{x}^2 + \dot{y}^2) = \frac{1}{2} m_1 (\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2} m_2 (\dot{x}_2^2 + \dot{y}_2^2) \quad (33)$$

Then one can substitute Eqs. (30) and (31) into Eq. (33) to obtain

$$T = \frac{1}{2} m_1 l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + \frac{1}{2} m_2 (2 \dot{\theta}_1 l_1 \dot{\theta}_2 l_2 \cos(\theta_1 - \theta_2)) \quad (34)$$

The Lagrangian will be computed by using Eqs. (32) and (34) in the form

$$\begin{aligned} L = T - U = & \frac{1}{2} (m_1 + m_2) l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \\ & + (m_1 + m_2) g l_1 \cos \theta_1 + m_2 l_2 \cos \theta_2 \end{aligned} \quad (35)$$

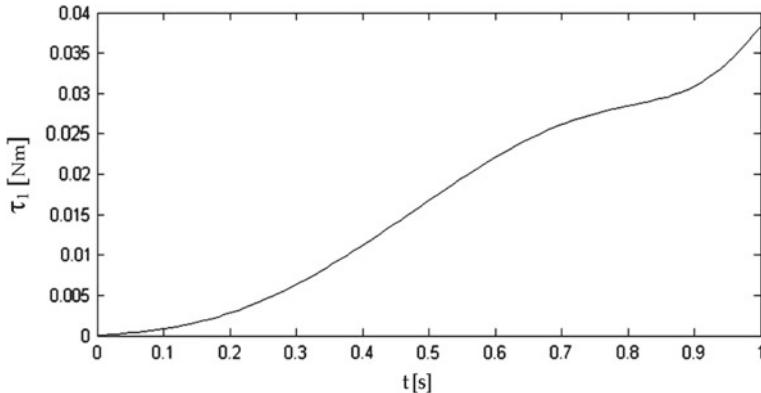
Then the Lagrangian in Eq. (35) can be substituted in Eq. (29) in order to compute the torques  $\tau_i$  in the form

$$\tau_i = \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} \quad (36)$$

Finally one can compute the required torques for the model in Fig. 23 such as

$$\begin{aligned} \tau_1 = & (m_1 + m_2) l_1^2 \ddot{\theta}_1 + m_2 l_1 l_2 \ddot{\theta} \cos(\theta_1 - \theta_2) \\ & + m_2 l_1 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) + g l_1 (m_1 + m_2) \sin \theta_1 \end{aligned} \quad (37)$$

$$\tau_2 = m_2 l_2^2 \ddot{\theta}_2 + m_2 l_1 l_2 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) - m_2 l_1 l_2 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + l_2 m_2 g \sin \theta_2 \quad (38)$$



**Fig. 24** Time diagram of the computed torque  $\tau_1$

Equations (37) and (38) can be computed for the desired configurations of the leg so that the required torques can be computed as function of time as shown for example, in Fig. 24 that reports the time diagram of the computed torque  $\tau_1$ , by referring to the most loaded actuator in a swing phase. The computed torques can be used at design stage for a proper choice of the required actuators. For example Fig. 24 shows a maximum torque of 0.038 Nm by referring to a leg trajectory as defined in Fig. 22. Additionally one can use the computed torques for control purposes in order to fulfil the desired path as initially planned.

## 5 Conclusions

This chapter gives a detailed overview of the literature on Hexapod Walking Robots. Then, the basic hexapod locomotion issues are systematically described. A general procedure is proposed for properly performing hexapod locomotion as starting from environments data. Key aspects are described in details by referring to the steps of path planning, gait planning and leg locomotion planning. Each specific step has been described also by referring to the wide literature on the topic. A case of study is described by referring to a built prototype of Cassino Hexapod II at LARM in Cassino in order to show the engineering details and the practical feasibility of the proposed procedure.

## References

1. Genta G (2012) Introduction to the mechanics of space robots. Springer, New York
2. Nonami K, Barai RK, Irawan A, Daud MR (2014) Hydraulically actuated hexapod robots. Springer, Japan

3. IFR home page (2014) <http://www.ifr.org/>. Accessed 18 Oct 2014
4. Chàvez-Clemente D (2011) Gait optimization for multi-legged walking robots, with application to a lunar hexapod. Ph.D. Thesis, Stanford University, Stanford
5. Carbone G, Ceccarelli M (2005) Legged robotic systems. In: Kordic V, Lazinica A, Merdan M (eds) Cutting edge robotics. InTech, Vienna, pp 553–576
6. Gonzalez de Santos P, Garcia E, Estremera J (2006) Quadrupedal locomotion: an introduction to the control of four-legged robots. Springer, London
7. Cigola M, Pelliccio A, Salotto O, Carbone G, Ottaviano E, Ceccarelli M (2005) Application of robots for inspection and restoration of historical sites. In: Proceeding of international symposium on automation and robotics in construction (paper 37), Ferrara
8. Jun BH, Shim H, Kim B, Park JY, Baek H, Yoo S, Lee PM (2013) Development of seabed walking robot CR200. In: Proceedings of the OCEANS'13 MTS/IEEE Conference, San Diego, CA, USA, 23–26 September 2013, pp 1–5
9. Bares J, Hebert M, Kande T, Krotkov E, Mitchell T, Simmons R, Whittaker W (1989) Ambler an autonomous rover for planetary exploration. IEEE Comput 22(6):18–26
10. Bartholet T, Crawson R (1985) Robot applications for nuclear power plant maintenance; EPRI Report-NP-3941, Research report center, Palo Alto 1985
11. Oku M, Yang H, Paio G, Harada Y, Adachi K, Barai R, Nonami K (2007) Development of hydraulically actuated hexapod robot COMET-IV-The 1st report: system design and configuration. In: Proceedings of the 2007 JSME conference on robotics and mechatronics 2A2-G01
12. Silva MF, Tenreiro Machado JA (2007) A historical perspective of legged robots. J Vib Control 13:1447–1486
13. Hauser T, Bretl K, Latombe JC, Harada W (2008) Motion planning for legged robots on varied terrain. Int J Robot Res 27:1325
14. Armada M, Gonzalez de Santos P (1997) Climbing, walking and intervention robots. Ind Robot 24(2):158–163
15. Autumn K, Buehler M, Cutkosky M, Fearing R, Full RJ, Goldman D, Groff R, Provancher W, Rizzi AA, Saranli U, Saunders A, Koditschek DE (2005) Robotics in scansorial environments. In: Gerhart GR, Shoemaker CM, Gage DW (eds) 5804(1):291–302. SPIE, 2005
16. Gregorio P, Ahmadi M, Buehler M (1997) Design, control, and energetics of an electrically actuated legged robot. IEEE Trans Syst Man Cybern B 27:626–634
17. Carbone G, Shrot A, Ceccarelli M (2007) Operation strategy for a low-cost easy-operation Cassino hexapod. Appl Bionics Biomech 4:149–156
18. Muybridge E (1887) Animal locomotion. University of Pennsylvania
19. Schneider A, Schmucker U (2006) Force Sensing for multi-legged walking robots: theory and experiments part 1: overview and force sensing. In: Buchli J (ed) Mobile robotics, moving intelligence
20. Tomovic R, Karplus WJ (1961) Land locomotion simulation and control. In: Proceedings of third international analogue computation, Opatija, Yugoslavia, pp 385–390
21. Hildebrand M (1967) Symmetrical gaits of horse. Science 150:701–708
22. McGhee R (1968) Some finite state aspects of legged locomotion. Math Biosci 2:67–84
23. Peternella M, Salinari S (1973) Simulation by digital computer of walking machine control system. In: Proceeding of 5th IFAC symposium on automatic control in space, Genova
24. Bessonov A, Umnov N (1973) The analysis of gaits in six-legged vehicles according to their static stability. In: Proceedings of the 1st CISM-IFToMM conference, Udine, pp 1–9
25. Sun SS (1974) A theoretical study of gaits for legged locomotion systems. Ph.D. Thesis, The Ohio State University, Columbus
26. Gurfinkel V, Gurfinkel E, Devjanin E, Efremov E, Zhicharev D, Lensky A, Schneider A, Shtilman L (1982) Investigation of robotics. In: Six-legged walking model of vehicle with supervisory control. Nauka Press, Moscow
27. McGhee R (1977) Control of legged locomotion systems. In: Proceeding of 18th automatic control conference, San Francisco, pp 205–215
28. McGhee RB, Iswandhi GI (1979) Adaptive locomotion of a multi legged robot over rouch terrain. IEEE Trans Syst Man Cybern, SMC-9(4):176–182

29. Song SM, Waldron K (1989) Machines that walk: the adaptive suspension vehicle. MIT Press, Cambridge
30. Raibert M (1986) Legged robots that balance. MIT Press, Cambridge
31. Byrd J, De Vries K (1990) A six-legged telerobot for nuclear applications development. *Int J Robot Res* 9(2):43–52
32. Song SM, Waldron KJ (1987) An analytical approach for gait study and its applications on wave gaits. *Int J Robot Res* 6:60–71
33. Akizono J, Iwasaki M, Asakura O (1989) Development on a walking robot for underwater inspection. In: Proceedings of ICAR'89. Columbus, pp 652–663
34. Brooks RA (1989) A robot that walks; emergent behaviors from a carefully evolved network. *Neural Comput* 1:253–262
35. Pfeiffer F, Eltze J, Weidermann H (1995) Six-legged technical walking considering biological principles. *Robot Autom* 22–23
36. Nelson GM, Quinn RD, Bachmann RJ, Flannigan WC, Ritzmann RE, Watson JT (1997) Design and simulation of a cockroach-like hexapod robot. In: Proceedings of the 1997 IEEE international conference on robotics and automation, pp 1106–1111
37. Koyachi N, Adachi H, Izumi M, Hirose T, Senjo N, Murata R, Arai T (2002) Multimodal control of hexapod mobile manipulator MELMANTIS-1. In: Proceedings of 5th International Conference on Climbing Walking Robots, pp 471–478
38. Alexandre P, Preumont A (1995) On the gait control of a six-legged walking machine. In: Proceedings of 2nd IFAC workshop on intelligent autonomous vehicles, Espoo
39. Yang J, Kim J (2000) A fault tolerant gait for a hexapod robot over uneven terrain. *IEEE Trans Syst, Man, Cybern, Part B* 30(1):172–180
40. Saranli U, Buehler M, Koditschek DE (2001) RHex-A simple and highly mobile hexapod robot. *Int J Robot Res* 20(7):616–631
41. Lewinger WA, Branicky MS, Quinn RD (2005) Insect-inspired, actively compliant hexapod capable of object manipulation. In: Proceedings of the 8th international conference on climbing and walking robots (CLAWAR'2005), London, UK, 13–15, September 2005, pp 65–72
42. Collins JJ, Stewart I (2004) Hexapodal gaits and coupled nonlinear oscillator models. *Biol Cybern*
43. Arena P, Fortuna L, Frasca M, Patanè L, Pavone M (2006) Realization of a CNN driven cockroach-inspired robot. In: Proceedings of IEEE international symposium on circuits and systems. Kos. doi:[10.1109/ISCAS.2006.1693168](https://doi.org/10.1109/ISCAS.2006.1693168)
44. Fielding MR, Dunlop GR (2004) Omnidirectional hexapod walking and efficient gaits using restrictedness. *Int J Robot Res* 23(10):1105–1110
45. Kennedy B, Koon A, Aghazarian H, Garrett M, Huntsberger T, Magnone L, Robinson M, Townsend J (2005) The Lemur II-class robots for inspection and maintenance of orbital structures: a system description. In: Proceedings of CLAWAR'2005-8th international conference on climbing and walking robots, pp 1069–1076
46. Asbeck AT, Kim S, McClung A, Parness A, Cutkosky MR (2006) Climbing walls with microspines. In: Proceedings of IEEE international conference robotics and automation, Orlando, pp 4315–4317
47. Nonami K (2001) Humanitarian mine detection six-legged walking robot Comet-II with two manipulators. In: CLAWAR 2001—climbing and walking robots and the support technologies for mobile machines
48. Roennau A, Heppner G, Pfozter L, Dillman R (2013) Lauron V: optimized leg configuration for the design of a bio-inspired walking robot. In: Proceedings of the 16th international conference on climbing and walking robots and the support technologies for mobile machines, Sydney, pp 563–571
49. Siciliano B, Khatib O (2008) Springer handbook of robotics. Springer, New York
50. Fielding MR, Dunlop R, Damaren, Hamlet CJ (2001) Force/position controlled hexapod walker—design and systems. In: Proceeding of IEEE conference on control applications, Mexico City, pp 984–989

51. Ceccarelli M (2010) Fundamentals of mechanics of robotic manipulation. Springer, Dordrecht
52. Kazemi M, Gupta K, Mehrandezh M (2010) Path-planning for visual servoing: a review and issues. Visual servoing via advanced numerical methods. Springer, London, pp 189–207
53. Latombe JC (1991) Robot motion planning. Kluwer, Boston
54. Hoerger M, Kottege N, Bandyopadhyay T, Elfes A, Moghadam P (2014) Real-time stabilisation for hexapod robots. In: Proceedings of the international symposium on experimental robotics (ISER 2014), 15–18 June 2014. Marrakech and Essaouira, Morocco
55. Foley JD, Van Dam A, Feiner SK, Hughes JF (1990) Computer graphics: principles and practice, 2nd edn. Addison-Wesley, Reading
56. Carbone G, Ceccarelli M, Oliveira PJ, Saramago SF, Carvalho JCM (2008) An optimum path planning for Cassino parallel manipulator by using inverse dynamics. *Robotica* 26(2):229–239
57. Vukobratovic M, Borovac B (2004) Zero moment point: thirty five years of its life. *Int J Hum Robot* 1(1):157–173
58. Ghasempoor A, Sepehri N (1998) A measure of stability for mobile manipulators with application to heavy-duty hydraulic machines. *ASME J Dyn Syst, Meas Control* 120:360–370
59. Abo-Shanab RF, Sepehri N (2005) Tip-over stability of manipulator-like mobile hydraulic machines. *ASME J Dyn Syst, Meas, Control* 127:295–301
60. Papadopoulos E, Rey DA (2000) The force-angle measure of tip-over stability margin for mobile manipulators. *J Veh Syst Dyn* 33:29–48
61. Hirose S, Tsukagoshi H, Yoneda K (2001) Normalized energy stability margin and its contour of walking vehicles on rough terrain. In: Proceedings IEEE international conference on robotics and automation, Seoul, pp 181–186
62. Moosavian SAA, Dabiri A (2010) Dynamics and planning for stable motion of a hexapod robot. In: Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Montreal 6–9 July 2010, pp 818–823
63. Lingelbach F (2004) Path planning using Probabilistic cell decomposition. In: Proceedings of IEEE international conference on robotics and automation ICRA'04
64. Chazelle B (1987) Advances in robotics. In: Schwartz JT, Yap CK (eds) Approximation and decomposition of shapes. Erlbaum Hillsdale, Hillsdale, pp 145–185
65. Khatib O (1985) Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings of the 1985 IEEE international conference on robotics and automation, pp 500–505
66. Volpe RA (1990) Real and artificial forces in the control of manipulators: theory and experiments. Carnegie Mellon University, Pittsburgh, The Robotics Institute
67. Takahashi O, Schilling RJ (1989) Motion planning in a plane using generalized Voronoi diagrams. *IEEE Trans Robot Autom* 5(2):143–150
68. Garrido S, Moreno L, Lima PU (2011) Robot formation motion planning using fast marching. *Robot Auton Syst* 59(9):675–683
69. Ahmad A, Nirjhar D (2008) Probabilistic roadmap method and real time gait changing technique implementation for travel time optimization on a designed six-legged robot . In: Proceedings of the 39nd ISR(International Symposium on Robotics), 15–17 October 2008
70. Gómez-Bravo F, Carbone G, Fortes JC (2012) Collision free trajectory planning for hybrid manipulators. *Mechatronics* 22:836–851. ISSN: 0957–4158, doi:[10.1016/j.mechatronics.2012.05.001](https://doi.org/10.1016/j.mechatronics.2012.05.001)
71. Belter D, Skrzypczynski P (2011) Integrated motion planning for a hexapod robot walking on rough terrain. In: Proceedings of 18th IFAC World Congress Milano (Italy) August 28–September 2, 2011
72. Ding X, Wang Z, Rovetta A, Zhu JM (2010) Locomotion analysis of hexapod robot. In: Miripour B (ed) Climbing and walking robots
73. Preumont A, Alexandre P, Ghys D (1991) Gait analysis and implementation of a six leg walking machine. In: Proceedings of the fifth international conference on advanced robotics. Robots in unstructured environments (ICAR'91), vol 2. pp 941–945. Pisa, 19–22 June 1991
74. Ozguner F, Tsai SJ, McGhee RB (1984) An approach to the use of terrain-preview information in rough terrain locomotion by a hexapod walking machine. *Int J Robot Res* 3(2):134–146
75. Wilson DM (1966) Insect walking. *Annu Rev Entomol* 11:103–122

76. Collins JJ, Stewart I (1993) Hexapodal gaits and coupled nonlinear oscillator models. *Biol Cybern* 68: 287–298
77. Berns K, Dillman R (2001) From biology to industrial applications. In: 4th CLAWAR international conference of climbing and walking robots, Karlsruhe, pp 80–82
78. Todd DJ (1985) Walking machines—an introduction to legged robots. Anchor Press, Essex
79. Carbone G, Ceccarelli M (2004) A mechanical design of a low-cost easy-operation anthropomorphic wheeled leg for walking machines. *Int J Robot Manag* 9(2):3–8
80. Silva MF, Tenreiro Machado JA, Lopes AM (2003) Comparison of fractional and integer order control of an hexapod robot. In: Proceedings of DETC 03 ASME 2003 design engineering technical conference and computers information in engineering conference, Chicago, USA, 2–6 September 2003
81. Life Performance Research home page (2014)<http://www.lp-research.com/>. Accessed 18 Oct 2014
82. Carbone G, Tedeschi F (2013) A low cost control architecture for Cassino Hexapod II. *Int J Mech Control* 14(01):19–24
83. Tedeschi F, Cafolla D, Carbone G (2014) Design and operation of Cassino hexapod. *Int J Mech Control* 15(01):19–25
84. Tedeschi F, Carbone G (2014) Design issues for hexapod walking robots. *Robotics* 3(2):181–206
85. Julis K, Brepta R (1987) Mechanics II. Dynamics, SNTL, Praha
86. Frankovsky P, Hroncova D, Delyova I, Hudak P (2012) Inverse and forward dynamic analysis of two link manipulator. *Procedia Eng* 48:158–163
87. Ogatha K (1978) System dynamics. Prentice Hall Inc, Englewood Cliffs

**Part V**

**Robot Cooperation and Interaction**

# Distributed Cooperation of Multiple UAVs for Area Monitoring Missions

José J. Acevedo, Begoña C. Arrue, Iván Maza and Aníbal Ollero

**Abstract** Monitoring and surveillance is a very relevant issue in recent years where the use of multiple Unmanned Aerial Vehicles (UAVs) offers interesting advantages. In this kind of mission, assuming that there is no “a priori” information about the location or time of the events or intruders to detect, a frequency-based criterion seems to be an interesting approach to solve the problem. This chapter describes a frequency-based approach applied to a cooperative area monitoring problem using a team of UAVs. Three different cooperative patrolling strategies (cyclic, path partitioning and area partitioning) are analyzed and compared with respect to refresh and latency times criteria. Finally, assuming communication constraints, a distributed implementation is required and convergence to the centralized cooperative patrolling strategy should be ensured. Two different distributed techniques are described: one-to-one coordination and a method based on coordination variables. Both techniques are compared from a convergence complexity criterion.

**Keywords** Multi-UAV · Distributed coordination · Cooperative patrolling · Surveillance and monitoring

---

J.J. Acevedo (✉) · B.C. Arrue · I. Maza · A. Ollero  
Grupo de Robótica, Visión y Control, Escuela Superior de Ingenieros,  
Universidad de Sevilla. Camino de Los Descubrimientos s/n, 41092 Sevilla, Spain  
e-mail: jacevedo@us.es

B.C. Arrue  
e-mail: barrue@us.es

I. Maza  
e-mail: imaza@us.es

A. Ollero  
e-mail: aollero@us.es

## 1 Introduction

Monitoring is a relevant feature in different contexts: automated inspection, search and rescue missions, planetary explorations, etc., as can be found in [12, 21]. The use of the multiple cooperating UAVs offers very interesting advantages to accomplish this kind of mission: robustness against failures, higher spatial coverage and an efficient deployment [8, 18].

From a *frequency-based approach*, minimizing the time to detect an object of interest in the area is the same as minimizing the elapsed time between two consecutive visits to any position or *refresh time*. This approach has been used by many authors, obtaining solutions to guarantee an uniform frequency of visits as in [13], or the maximal minimum frequency as in [7]. The obtained solution is a deterministic motion plan for each vehicle. Some authors, as in [6], address the patrolling problem in adversarial settings applying a probabilistic approach because with a deterministic solution, intelligent intruders could learn the strategy. A frequency-based approach is also followed in [10], which defines and compares different partitioning and cyclic patrolling strategies. Authors of [20] analyze the *refresh time* and *latency* in area coverage problems with multiple robots using different approaches.

In [2], an area partitioning strategy is proposed to monitor a rectangular area with a team of homogeneous UAVs. The area is divided into non overlapped sub-areas and each UAV covers a different sub-area using an efficient path, minimizing the total path length. On other hand, in [5], the problem with irregular areas and heterogeneous UAVs is solved using a path partitioning strategy. A single coverage path is created to monitor the whole area and the path is divided in segments that are allocated to the different UAVs. Other authors, as [19], propose cyclic strategies where all the robots patrol the same closed coverage path in the same direction and equally spaced through it. This strategy offers theoretically optimal results from a frequency-based approach with homogeneous robots. However, in scenarios with constrained communications, the robots could not share the required information.

On the other hand, a coverage path planning algorithm is required for monitoring missions. Reference [14] proposes an on-line algorithm, assuming that area to cover is initially unknown, that solves the problem for multi-robot systems using Voronoi spatial partitioning. An off-line algorithm, where the area to cover is known a priori, is proposed in [15]. The authors create a spanning tree and generate a coverage path around it. The most well known off-line coverage path planning is called *Boustrophedon Cellular Decomposition* and was presented in [11]. It proposes to divide the whole area into smaller sub-areas which can be covered with a simple *back and forth* method. A variation of this method has been used in this chapter to obtain a *pseudo-symmetrical* path [1].

Assuming communications constraints, a decentralized algorithm has to be proposed to coordinate the motions of the UAVs in a distributed manner. In [17], the authors propose behavioral control to perform perimeter patrolling mission with multiple robots, even under communications constraints, but it does not optimize the solution from a refresh time criterion. Authors of [4, 16] use the technique of

*coordination variables* to ensure cooperation between a team of UAVs to accomplish a perimeter surveillance mission using a path partitioning strategy. *Coordination variables* are the minimum global information required by each robot to solve the problem in a coherent manner. The selection of those variables can be difficult for complex problems. In [2], the technique called *one-to-one coordination* is presented to implement an area partitioning strategy from a distributed manner with a team of homogeneous UAVs. This technique implies that each pair of UAVs solves a coordination problem including only their own information. A *one-to-one* coordination technique allows the system to obtain the whole coordination from local decision and information. The resulting system is scalable, because each UAV only requires information from nearby neighbors. In [9], the authors use a similar technique to coordinate a team of video-cameras in surveillance missions.

A large set of validation tests assuming different scenarios has been performed using different coordination techniques and patrolling strategies. Results will be focused on analyzing the convergence complexity and the performance of the different algorithms and strategies in the different scenarios from a frequency-based approach and according to the refresh time and latency criteria.

The rest of the chapter is structured as follows. Section 2 describes the *frequency-based approach* related to the monitoring missions. The area monitoring mission with a team of UAVs is stated in Sect. 3. In Sect. 4, three different cooperative patrolling strategies are proposed to solve the proposed problem. This section analyzes the strategies from a *refresh time* and a *latency time* criterion. Section 5 describes and analyzes from a convergence complexity criterion two different distributed coordination techniques. Section 6 closes the chapter with the conclusions.

## 2 Frequency-Based Criterion for Monitoring Missions

Monitoring and surveillance missions with mobile robots implies to patrol the area of interest according to a defined criterion. When there is no information about the events or intruders to detect, it can be assumed that the events can appear in any position with the same probability.

Then, the goal is to optimize the frequency of visits of any position into the area. Different optimization criteria can be defined from a frequency-based approach:

- Uniformity criterion. The objective is to obtain the same frequency of visits for all locations in the area.
- Average frequency. The objective is to maximize the minimum average frequency of visits for all positions in the area.
- Maximal minimum frequency. The objective is to maximize the minimum frequency of visits for any location into the area.

## 2.1 Refresh Time

Maximizing the frequency of visits of a position  $r$  in the area  $S$  is equivalent to minimizing the elapsed time between two consecutive visits to that position, which is also known as the *refresh time*  $RT$ . The *refresh time* at a position  $r$  increases while that position is not monitored. Given an updating time  $dT$ , the refresh time can be periodically recalculated as

$$RT(r, t) = \begin{cases} 0, & \text{if } r \text{ is being monitored at time } t \\ RT(r, t - dT) + dT, & \text{in other case} \end{cases}, \quad (1)$$

where  $RT(r, 0) = 0, \forall r \in S$ .

Therefore, the previous optimization criteria can be redefined as follows:

- Uniformity criterion. The objective is to obtain the same refresh time for all locations in the area.
- Average refresh time. The objective is to minimize the average refresh time for all the positions in the area.
- Minimal maximum refresh time. The objective is to minimize the maximum refresh time for any location in the area. In this case, the objective function  $J$  to minimize can be formally defined as

$$J = \max_{r,t} RT(r, t), \quad (2)$$

and this problem is addressed in [10] (min idleness problem).

If there is additional information about the distribution of the objects of interest in the area, the scenario can be divided in different zones with different probabilities. Then, the frequency of visits can be normalized by using priorities and defining the *urgency* along the whole area, such as in [3].

## 3 Multi-UAV Systems for Area Monitoring Missions

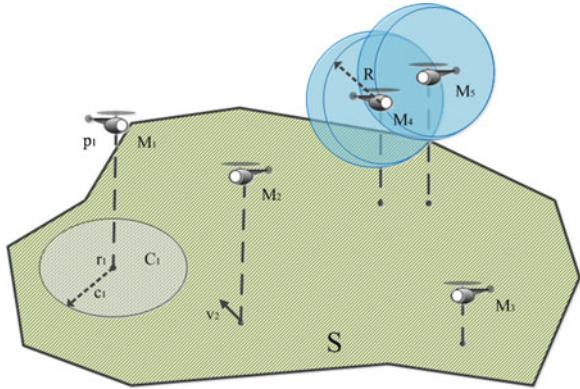
Let us consider an irregular area  $S$  that has to be cooperatively monitored by a team of  $N$  UAVs  $M := \{M_1, M_2, \dots, M_N\}$  controlled in position and velocity, see Fig. 1.

Let us define  $p_i(t) \in \mathbb{R}^3$  as the position of  $M_i$  at time  $t$  in the space and  $r_i(t) \in \mathbb{R}^2$  as its projection on the ground. A maximum motion speed can be defined for each UAV  $M_i$ , such that the velocity  $v_i(t)$  of each UAV on the ground at any time  $t$  is limited by  $v_i^{\max}$  as

$$v_i(t) = \left| \frac{dr_i(t)}{dt} \right| \leq v_i^{\max}. \quad (3)$$

It is assumed that each UAV can use its velocity  $v_i(t)$  as input to control its motion.

**Fig. 1** A set of five UAVs monitoring an area  $S$



On other hand, assuming a constant altitude and a circular coverage pattern, the covering capabilities of each UAV  $M_i$  can be computed from the coverage range  $c_i$ , which defines how large is the area  $C_i(t)$  that the UAV can instantaneously monitor as

$$C_i(t) := \{r \in \mathbb{R}^2 : |r - r_i(t)| < c_i\}. \quad (4)$$

The UAVs are considered heterogeneous since they can have different motion speeds and coverage ranges. But, what is better? A larger coverage range or a greater motion speed? Assuming that both UAVs have a maximum motion speeds greater than 0, both the motion speed and the coverage range can be normalized in a single one: the *coverage speed*  $a_i^{\max}$  that can be defined as the maximum area covered per second by each UAV.

It can be computed as follows. Considering a UAV moving in straight line at constant speed  $v_i^{\max}$  during an interval of  $\Delta t$  s and assuming a circular coverage pattern with a radio  $c_i$ , it would have covered a size of  $A m^2$

$$A = \pi c_i^2 + 2c_i v_i^{\max} \Delta t. \quad (5)$$

and deriving it, the coverage speed is as follows

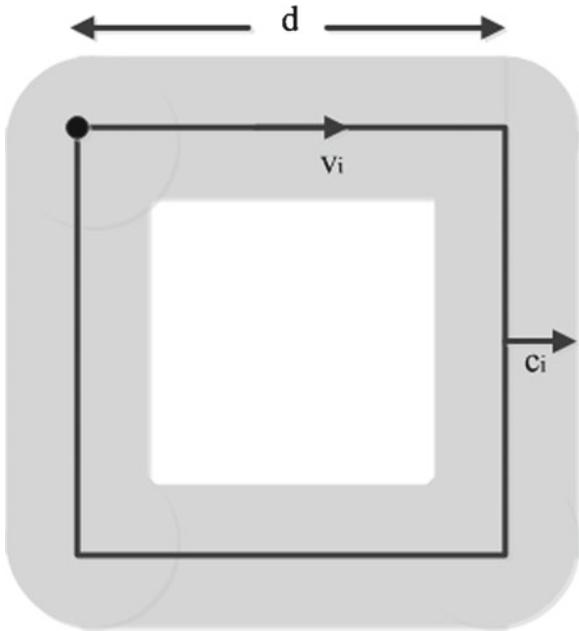
$$a_i^{\max} = \frac{dA}{dt} = 2c_i v_i^{\max}. \quad (6)$$

In Fig. 2, a UAV  $M_i$  takes  $\frac{4d}{v_i}$  seconds to cover an area of  $8dc_i m^2$ .

The goal is to define cooperatively the UAVs motions to minimize the cost function defined in (2), which means minimizing the *maximum refresh time* along the whole area  $S$ .

On the other hand, communication constraints can be modeled considering a limited communication range  $R$  for the UAVs

**Fig. 2** Area covered by a UAV  $M_i$  with a coverage range  $c_i$  moving around a square of side  $d$



$$\|p_i - p_j\| \leq R , \quad (7)$$

so that only close enough UAVs can communicate and interchange information. Dealing with communication constraints implies to achieve the above mentioned goal in a distributed manner (considering a non continuously opened communications channel between all the UAVs).

Finally, in distributed systems and assuming communication constraints, for monitoring and surveillance missions it is interesting to estimate how long does the information detected by a single UAV take to reach a central control station, such that it can decide the following actions with respect to this detected information. This time is upper bounded by the *latency time*  $LT$  or time to share any information between all the UAVs. Therefore, another relevant objective for this type of missions is to minimize the theoretical maximum *latency time*  $LT$ .

## 4 Cooperative Patrolling Strategies

Solving the monitoring mission in a cooperative manner from a frequency-based approach implies to define a cooperative patrolling strategy for the multiple UAVs. The challenge is to coordinate the UAVs motions such that the *refresh time* and the *latency time* can be minimized along the whole area.

## 4.1 Cyclic Strategy

A single closed coverage path  $P$  of length  $L$  for all the UAVs is usually assumed in this type of strategies. The path  $P$  can be defined by a curve  $\lambda(x) : \mathbb{R} \rightarrow \mathbb{R}^2$ , where  $x$  is the distance from the location  $\lambda(x)$  to the initial path position  $\lambda(0)$  along the path  $P$ . In [19], the authors describe this strategy. All the vehicles move along the path  $P$  in the same direction with the same velocity  $v^{\max}$  and equally spaced, as it is shown in Fig. 3. Therefore, at any time  $t$ , the position  $\lambda(x_i(t))$  of a UAV  $M_i$  is related to the locations of its neighbors ( $\lambda(x_{i-1}(t))$ ) by

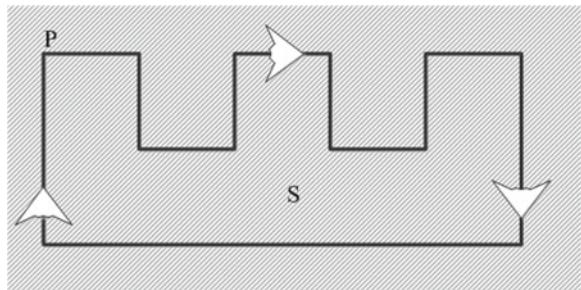
$$x_i(t) = x_{i-1}(t) + \frac{L}{N}, \quad (8)$$

where  $N$  is the total number of UAVs.

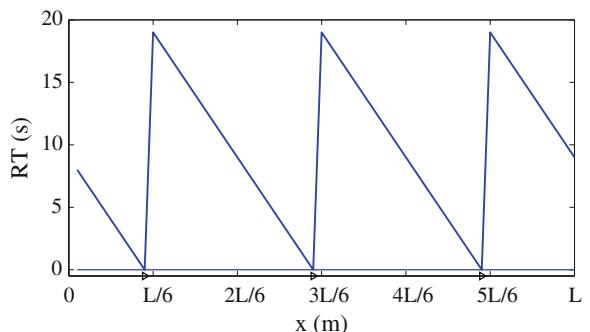
Assuming that all the UAVs have the same capabilities (maximum speed and coverage range), this strategy obtains theoretically the best performance from a refresh time criterion. Figure 4 plots the refresh time for the setup shown in Fig. 3. The theoretical *maximum refresh time* using the cyclic strategy can be computed as

$$RT^{\max} = \frac{L}{Nv^{\max}}, \quad (9)$$

**Fig. 3** A team of three UAVs running a cyclic strategy along a path  $P$  to monitor a rectangular area  $S$



**Fig. 4** Refresh time computed along a path of length  $L$  while the three UAVs shown in Fig. 3 run a cyclic strategy



and the theoretical *average refresh time* is a constant value

$$RT^{ave} = \frac{L}{2Nv^{\max}} . \quad (10)$$

Within the cyclic strategy, additional aspects are considered in the following.

#### 4.1.1 Open Paths

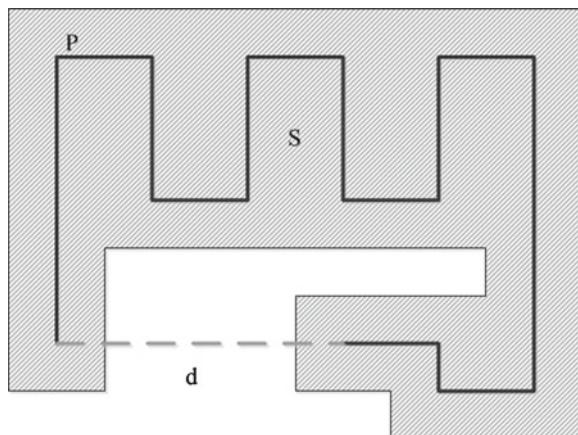
Assuming an open coverage path ( $\lambda(0) \neq \lambda(L)$ ), the UAVs can not be continuously patrolling while keeping the same motion direction. Then, in order to implement a cyclic strategy, the UAVs should use a larger closed path  $P'$  which includes the coverage open path  $P$  and a segment between the final  $\lambda(L)$  and the initial  $\lambda(0)$  path positions. Defining the distance between  $\lambda(0)$  and  $\lambda(L)$  as  $d > 0$ , the length of  $P'$  is  $L + d$  (see Fig. 5) and the previous values for the maximum and average refresh times are degraded to

$$RT^{\max} = \frac{L + d}{Nv^{\max}} \quad (11)$$

and

$$RT^{ave} = \frac{L + d}{2Nv^{\max}} . \quad (12)$$

**Fig. 5** When the path  $P$  is not closed, the UAVs should go from the initial to the final position of  $P$  to implement a cyclic strategy



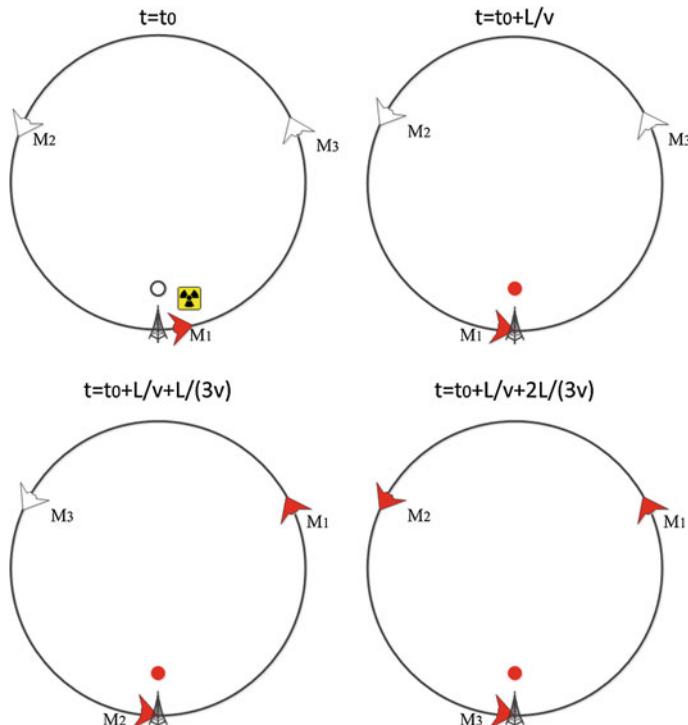
#### 4.1.2 Latency Under Communication Constraints

Considering a limited communication range  $R$  for the UAVs, two neighbor UAVs can not communicate if

$$R > \|\lambda(x_i) - \lambda(x_{i-1})\| , \quad (13)$$

for any  $i \in [1, 2, \dots, N]$ .

Even assuming that the UAVs can continue the mission without communication executing the plan previously computed, they can not share the information about the detected objects of interest with the rest of the team and the *latency*  $LT$  minimization can not be addressed. A possible solution could be to consider a fixed station in a position into the coverage path, such that it can retrieve the information from the UAVs. Theoretically, this solution could keep the best performance from a *refresh time* criterion while allowing information propagation among the UAVs. The question is: how long the information about an event detected by a single UAV takes to be propagated to the rest of the team? In the worst case scenario, the UAV  $M_i$  takes a time  $\frac{L}{v^{\max}}$  to visit the fixed station after it has detected an object of interest (see Fig. 6), and each period  $\frac{L}{Nv^{\max}}$ , a new UAV passes close to the fixed station to receive



**Fig. 6** a UAV  $M_1$  detected an object of interest whose information has to be shared with  $M_2$  and  $M_3$ . As they are running a cyclic strategy,  $M_1$  uses a fixed station as relay to share the information

information about the detected object. Then, after a period  $\frac{(N-1)L}{Nv^{\max}}$ , the information received at the fixed station from  $M_i$  has been shared among the other other  $N - 1$  UAVs and the *latency* can be upper-bounded as

$$LT \leq \frac{L}{v^{\max}} + \frac{(N-1)L}{Nv^{\max}}. \quad (14)$$

And for  $N \rightarrow \infty$ , the upper bounds of the latency can be approximated

$$LT \approx \frac{2L}{v^{\max}}. \quad (15)$$

#### 4.1.3 Heterogeneous UAVs

Assuming heterogeneous UAVs, the cyclic strategy can not exploit their different capabilities and can not obtain the best performance from a frequency-based criterion.

First, if any UAV has a coverage range different to the rest, its optimal coverage path would be different. However, according to the cyclic strategy, all the UAV must patrol the same path. If they follow the path computed according to the largest coverage range, the UAVs with lower coverage ranges will not be able to monitor the whole area and the maximum *refresh time* will depend only on the UAV with the greater coverage range. If they follow the path computed according to the lowest coverage range, the maximum *refresh time* will be also increased with respect to the theoretically best one but it will depend on all the UAVs.

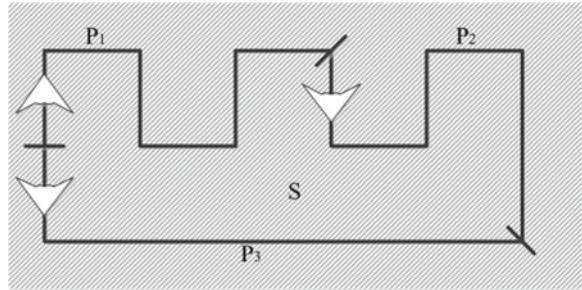
Also, assuming that all the UAVs have the same coverage range but some of them have a different maximum speed, there are different possibilities. The first one implies that the slowest UAVs do not participate in the mission; then, a cyclic strategy with  $N' < N$  UAVs will obtain a *refresh time* greater than the best one. The second one assumes that the faster UAVs limit their speed, such as a cyclic strategy with  $N$  UAVs moving at the lowest speed is implemented obtaining a *refresh time* greater than the best one. Finally, a mixed option, choosing a velocity such that some UAVs can not participate in the mission and others have to limit their speeds, would obtain better results than the previous ones but it will be also worse than the theoretical best solution.

Assuming  $v_1^{\max} \geq v_2^{\max} \geq \dots \geq v_N^{\max}$ , the theoretical maximum refresh time can be computed as follows, depending on how many UAVs participate in the mission.

$$RT^{\max} = \frac{L}{nv_n^{\max}}. \quad (16)$$

Therefore, the challenge would be to choose the value  $n$  which minimizes the expression (16).

**Fig. 7** A team of three UAVs running a path partitioning strategy to patrol a path  $P$ . An area  $S$  is monitored by dividing the path in three segments  $P_1$ ,  $P_2$  and  $P_3$



#### 4.2 Path Partitioning Strategy

Another usual approach to solve monitoring missions in a cooperative manner is to divide the whole task among the available UAVs. The path partitioning strategy assumes that there is a coverage path  $P$  of length  $L$  for the whole area  $S$  that is divided in  $N$  non-overlapping segments  $P_i$  of length  $L_i$  (see Fig. 7) according to the UAVs maximum speeds  $v_i^{\max}$  such that

$$\begin{aligned} P_1 \cup P_2 \cup \dots \cup P_N &= P \\ P_1 \cap P_2 \cap \dots \cap P_N &= \emptyset \end{aligned} \quad (17)$$

and

$$L_i = v_i^{\max} \frac{L}{\sum_{j=1}^N v_j^{\max}} \quad \forall i = 1, \dots, N . \quad (18)$$

Each UAV patrols its own segment by using a *back and forth* motion so that all the UAVs take the same time

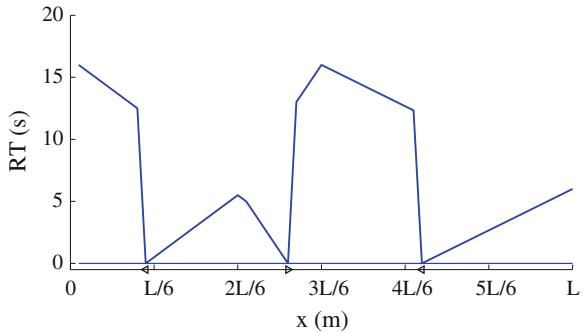
$$T' = \frac{L_i}{v_i^{\max}} = \frac{L}{\sum_{j=1}^N v_j^{\max}} \quad (19)$$

to cover their own segments. Figure 8 shows the *refresh time* for three UAVs running a path partitioning strategy. The *maximum refresh time* can be computed when a UAV almost reaches one of the endpoints of its segments as

$$RT^{\max} = 2 \frac{L}{\sum_{j=1}^N v_j^{\max}} , \quad (20)$$

whereas the *maximum average refresh time*, in the worst case scenario when all the UAVs are in the middle of their own segments, can be computed as

**Fig. 8** Refresh time computed along a path of length  $L$  while three UAVs run a path partitioning strategy



$$RT^{ave} \leq \frac{3}{4} \frac{L}{\sum_{j=1}^N v_j^{\max}} . \quad (21)$$

According to these results, the path partitioning strategy is worse than the cyclic strategy from a frequency-based criterion if all the UAVs are homogeneous and the path is closed. However, this strategy is useful with open and closed paths because robots move using a back and forth motion and can exploit the different maximum speeds of the UAVs. And assuming different coverage ranges, the path partitioning strategy has the same disadvantages as the cyclic one.

#### 4.2.1 Latency Analysis

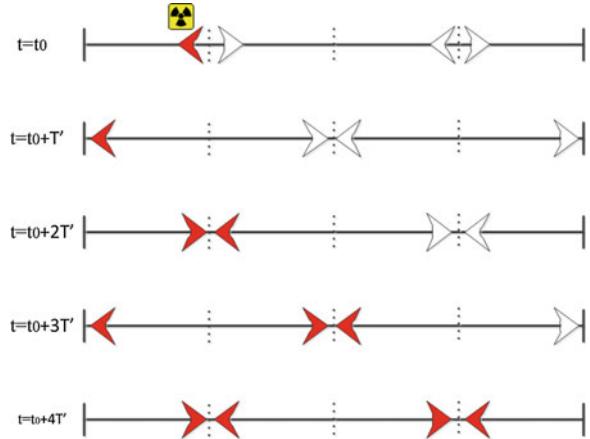
On the other hand, as all the UAVs take the same time to cover their own segments, it is possible to coordinate their motion such that each pair of neighbors can meet periodically in the common end of their segments. This strategy can ensure that any detected information is shared between all the UAVs even under communication constraints.

The *latency time* can be computed following the idea in Fig. 9. In a steady state and in the worst case scenario, a UAV with only a single neighbor detects an object of interest just after meeting its neighbor. In a period  $2T'$  the UAV meets its neighbor again and share the detected information. Now, every  $T'$  another UAV receives that information and after a period  $(N - 2)T'$  the last UAV receives it. Therefore, the *latency* is upper-bounded by

$$LT \leq 2T' + (N - 2)T' = NT' . \quad (22)$$

The path partitioning strategy obtains a good performance from a frequency-based criterion because it can exploit the different speed capabilities and it can be applied with open paths. It also ensures information propagation even under communication constraints.

**Fig. 9** An UAV at the endpoint of a segment detects an object of interest and has to share this information with the rest of the team. Neighbor UAVs meet periodically and share the information



### 4.3 Area Partitioning Strategy

The path partitioning strategy ensures periodical communication and exploits the different speeds. However, it can not obtain the best performance from a frequency-based criterion mainly because the UAVs have to move with a *back and forth* motion along their segments. In the area partitioning strategy, the UAVs implement a cyclic motion to patrol their segments in order to improve the previous solution.

In this strategy the whole area  $S$  to monitor is divided in  $N$  non-overlapping sub-areas  $S_i$  of sizes  $A_i$  which depend on the UAVs capabilities ( $a_i = 2c_i v_i^{\max}$ ) such that

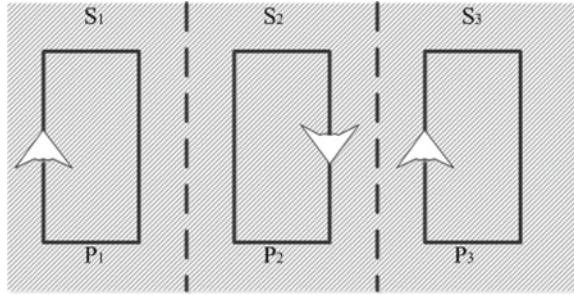
$$\begin{aligned} S_1 \cup S_2 \cup \dots \cup S_N &= S \\ S_1 \cap S_2 \cap \dots \cap S_N &= \emptyset \end{aligned} \quad (23)$$

and

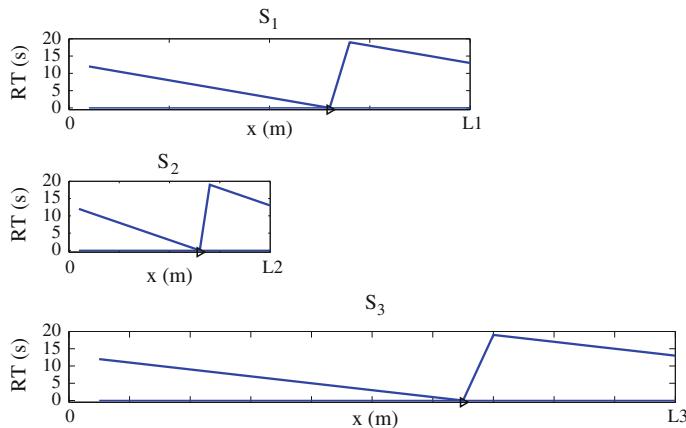
$$A_i = a_i^{\max} \frac{A}{\sum_{j=1}^N a_j^{\max}} \forall i = 1, \dots, N, \quad (24)$$

as it is shown in Fig. 10. Each UAV  $M_i$  can generate its own closed coverage path  $P_i$  to patrol its own sub-area  $S_i$ . It is assumed that it is possible to generate a closed coverage path  $P_i$  for any area such that its length  $L_i$  depends directly on the coverage range  $c_i$  of each UAV. Each UAV patrols continuously its own coverage path in a cyclic manner Fig. 11

As the sub-area sizes match the UAVs capabilities, all the UAVs take the same time to patrol its own sub-area and the *maximum refresh time* is theoretically the best one from a frequency-based criterion



**Fig. 10** A team of three UAVs an area partitioning strategy to monitor an area  $S$  by dividing it into three sub-areas  $S_1$ ,  $S_2$  and  $S_3$  and generating three different paths  $P_1$ ,  $P_2$  and  $P_3$ . Figure 11 shows the *refresh time* computed along the different paths generated



**Fig. 11** Refresh time computed along the three paths while three UAVs run an area partitioning strategy

$$RT^{\max} = T' = \frac{A_i}{a_i^{\max}} = \frac{A}{\sum_{j=1}^N a_j^{\max}}, \quad (25)$$

as well as the *average refresh time*

$$RT^{ave} = \frac{A}{2 \sum_{j=1}^N a_j^{\max}}. \quad (26)$$

Then, this strategy can obtain theoretically the best performance from a *frequency-based approach* exploiting the UAVs different capabilities (maximum speed, coverage range, etc.).

### 4.3.1 Latency Analysis

As all the UAVs take the same time to cover their sub-areas and assuming that each UAV has at least one neighbors, it is possible to coordinate their motions such that each pair of neighbors can meet periodically. A particular configuration is stated to organize the different UAVs in an area partitioning strategy.

Assuming that the coverage path of adjacent sub-areas have at least a pair of points close enough to allow communication, it should be possible to ensure information propagation among all the UAVs. These pair of points are defined as *link positions*. Then, each UAV has so many neighbors as *link positions* are in its coverage path.

Defining the coverage paths such that the distances between each pair of consecutive *link positions* are always the same and that the length of the paths are related to the UAVs motion speeds, it is possible to synchronize the UAVs' motions to upper-bound the *latency time* [1]. This *latency time* is related to the total number of link positions in the system (as the number of *link positions* increases, the upper-bound of the *latency time* decreases). Let us consider two configurations as examples:

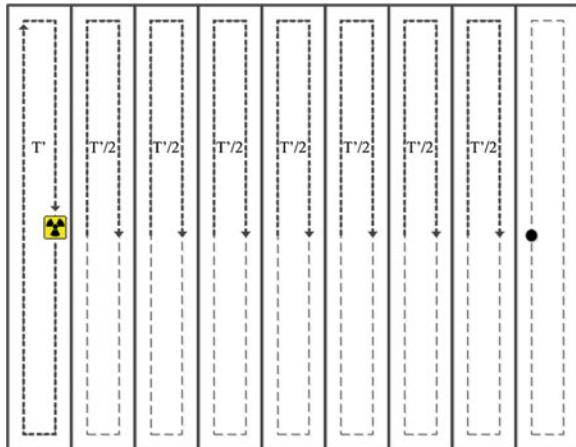
- Vector configuration: This is the configuration with the minimum number of *link positions* as it can be inferred from Fig. 12. Each UAV has a maximum of two neighbors and a minimum (in the sub-areas of the extremities) of one neighbor. Then, the total number of *link positions* is  $N - 1$ . In this case, the worst case scenario happens when a UAV in one of the sub-areas of the limits detects an object of interest just after meeting its neighbor.  $T'$  seconds later, it meets its neighbor again and shares the detected information. Now, each period  $T'/2$ , a UAV which received the information from one neighbor meets another one. At the end, in a period of  $(N - 2)T'/2$  the other  $N - 2$  UAVs will receive the detected information and

$$LT \leq 2 \frac{T'}{2} + (N - 2) \frac{T'}{2} = N \frac{T'}{2}. \quad (27)$$

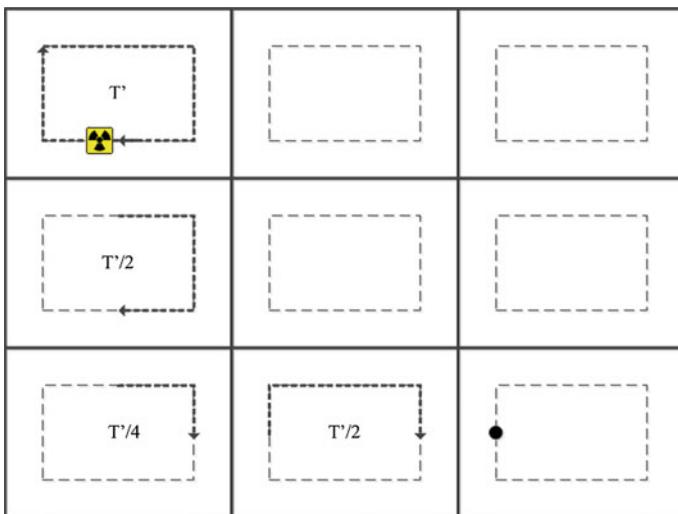
- Grid-shape configuration: in this case with  $N = m \times n$  UAVs, each one has at most four *link positions* as can be seen in Fig. 13 and the *maximum latency time* is reduced. The total number of *link positions* is  $m \cdot (n - 1) + n \cdot (m - 1)$  and the *maximum latency time* is defined by the distance in number of *link positions* between the two farthest coverage paths:  $(n - 1) + (m - 1)$  *link positions*. Then, in the worst case scenario and assuming a single direction of motion for all the UAVs, the *latency* is upper-bounded by

$$LT \leq T' + (n - 2)T'/2 + T'/4 + (m - 2)T'/2 = (2n + 2m + 1)T'/4. \quad (28)$$

It can be seen that the *latency time* depends on the configuration, but in general the area partitioning strategy usually offers better results than the path partitioning strategy from a latency criterion.



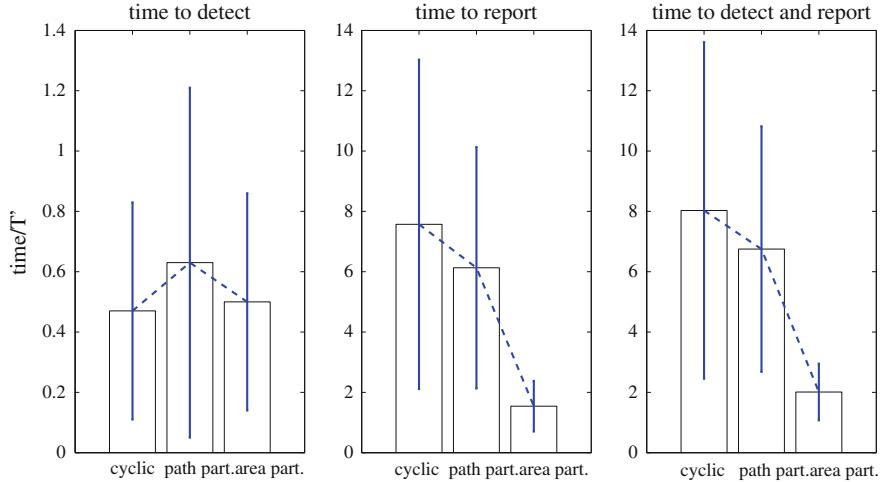
**Fig. 12** Maximum time since an object of interest is detected until this information reaches the farthest sub-area in a setup with nine UAVs running an area partitioning strategy using a vector-shape area division



**Fig. 13** Maximum time since an object of interest is detected until this information reaches the farthest sub-area in a setup with nine UAVs running an area partitioning strategy using a  $3 \times 3$  grid-shape area division

#### 4.4 Comparison Between Patrolling Strategies

The three proposed strategies have been implemented, simulated and compared for the same missions: a team of UAVs has to detect objects of interest in a given scenario and has to report the collected information to a central control station. A large set of



**Fig. 14** Relation between average detection and information propagation times computed during the simulations for the three cooperative patrolling strategies with respect to the optimal maximum refresh time

more than 80 objects of interest are launched in random positions across the area. Two different scenarios have been considered: an urban scenario with buildings and 16 UAVs and a natural opened scenario with 20 UAVs. All the UAVs are defined homogeneous, such that the three strategies can be compared assuming their best case. In order to normalize the results from the different scenarios, the *optimal maximum refresh time* is defined. Assuming that there are  $N$  UAVs, that all the UAVs have a maximum coverage speed  $a^{\max}$  and that the area has a size  $A$ , this parameter can be defined as

$$T' = \frac{A}{Na^{\max}} . \quad (29)$$

Figure 14 shows the average times ( $\pm$  their standard deviations) computed during the simulations divided by the *optimal maximum refresh time*.

Results confirm that cyclic and area partitioning strategies obtains the lowest detection times (related to the *refresh times*). On the other hand, the area partitioning strategy propagates the information faster than the others (related to the *latency times*).

## 5 Distributed Coordination Techniques

Assuming a limited communications range, the UAVs can not keep always direct communication in order to coordinate their motions in a centralized manner. In dynamic scenarios where the problem conditions can change during the mission,

the relevance of this issue is more critical. The objective is that the UAVs motions converge to implement a cooperative patrolling strategy in a distributed manner. Each UAV should decide its actions by using local information and limited communication with others. There is not any UAV who rules the rest and all the UAVs are in the same hierarchical level. The distributed algorithm should be based on the communication exchanges between UAVs that allow to update their local information. The system has to be totally decentralized and should be able to adapt to UAV faults and changes in the initial conditions. Two different approaches are described in the following.

## 5.1 One-to-one Coordination

The *one-to-one coordination* technique assumes that, solving the coordination problem between each pair of UAVs within communication, allows to solve the whole coordination problem between the  $N$  UAVs. For instance, an area partitioning strategy implies that each UAV has to cover a sub-area with a surface related to its own capabilities. Using *one-to-one coordination*, when two UAVs communicate, they apply the area partitioning strategy only for them (not for the  $N$  UAVs) considering the sub-areas they are currently covering as

$$A_i = a_i \frac{A(S_i \cup S_j)}{a_i + a_j} . \quad (30)$$

As it can be seen in Fig. 15, they join their currently patrolled areas and divide the resulting area between both according to their capabilities.

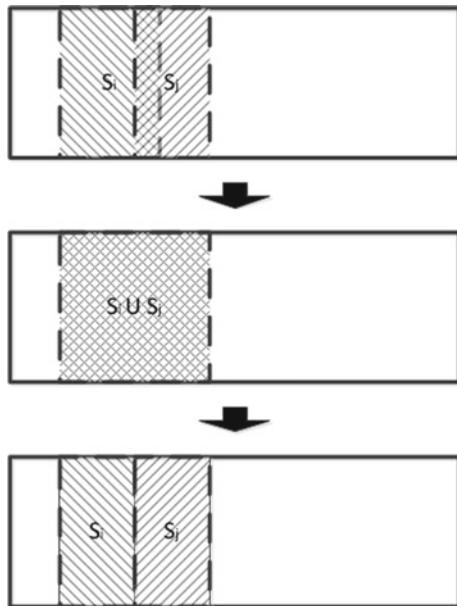
Assuming the area partitioning strategy, the UAVs communicate periodically with all their neighbors. Then, each UAV merges sequentially the areas of its neighbors obtaining an *any time solution* which converges to the centralized area division for the  $N$  UAVs in a finite amount of iterations (number of communication exchanges).

In [9], authors use a similar technique to coordinate a set of  $N$  cameras to converge to a partitioning strategy. Authors probe that the convergence time complexity is increasing quadratically with the number of agents in the problem and also depends inversely on a parameter  $\varepsilon > 0$  which defines how precise have to be the actual division according to the optimal one.

## 5.2 Coordination Variables

These methods assume that all the UAVs can obtain the same solution to the whole problem if they have the same minimum information encoded in the so-called *coordination variables*. In this case, when two UAVs meet they interchange information about the whole problem, update their own *coordination variables* and try to solve, not a reduced version of the problem, but the whole coordination problem. As both

**Fig. 15** Two neighbor UAVs  $M_i$  and  $M_j$  are covering the sub-areas  $S_i$  and  $S_j$ . When they meet, they join both and divide the resulting area between them, obtaining their new sub-areas



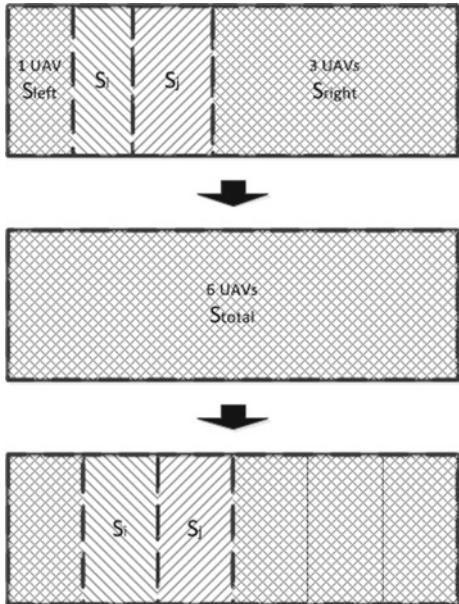
UAVs have the same information, they have the same *coordination variables*, address the same problem and obtain the same solution. Then, each UAV uses its current information to compute its current solution of the problem: for example, expressions (18) for the path partitioning strategy or (24) for the area partitioning strategy. In the latter case, when two UAVs communicate, they exchange information about the whole problem: number and capabilities of the UAVs into the area and size and shape of the area. Both UAVs share the same information about the problem and each one can address the area division problem to compute the size and shape of the sub-area to cover (not in a reduced version of the problem with two UAVs, but in the whole problem with the whole area and the  $N$  UAVs). See Fig. 16.

In [4] a similar technique is used to divide a perimeter between  $N$  mobile robots in a distributed manner, converging to a path partitioning strategy. The authors show that the convergence time complexity increases linearly with the number of robots.

### 5.3 Comparison Between Distributed Coordination Techniques

A large set of more than 200 simulations has been executed to compare these two different distributed coordination techniques: *one-to-one coordination* and *coordination variables*. Heterogeneous UAVs, different number of UAVs and different lengths of the paths have been considered and the goal was to evaluate the convergence to

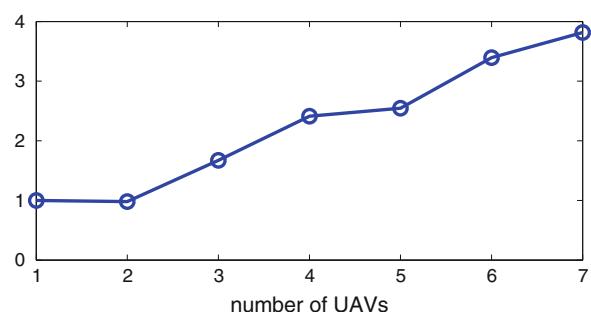
**Fig. 16** Two neighbor UAVs  $M_i$  and  $M_j$  are covering the sub-areas  $S_i$  and  $S_j$ . They have also information about the total area and UAVs on their *left* or *right sides*. When they communicate, they share the information not only about their current patrolled areas, but also about the number of UAVs and areas on their right and left sides. Then, both UAVs have information about the whole problem (the  $N$  UAVs and area  $S$  to monitor) and can solve it



a centralized path partitioning strategy. More details about these simulations can be found in [4]. Each scenario was simulated with both distributed coordination techniques and the convergence times were computed. It was considered that the system had converged when the difference between the current segment and the segment that theoretically should cover (according to expression (18)) was less than 5 %. In order to normalize the results, the relation between the convergence times computed for both techniques was calculated for each scenario. Figure 17 shows the average value for this parameter depending on the number of UAVs.

These results show that as the number of UAVs increases the technique based on the coordination variables obtains a quicker convergence than the technique based on one-to-one coordination.

**Fig. 17** Relation between the convergence times using the algorithms based on one-to-one coordination and the coordination variables





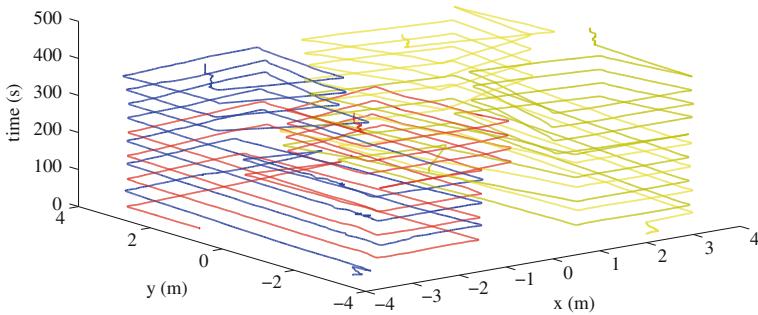
**Fig. 18** Snapshot from the experiment where 4 quad-rotors implement an area partitioning strategy from a distributed manner using an algorithm based on the one-to-one coordination. The white solid lines defines the area to monitor

#### **5.4 Experimental Test: Area Monitoring Missions Using the One-to-one Coordination**

Physical experiments have been performed to validate the one-to-one coordination applied to area monitoring missions. They have been carried out in the indoor testbed of the Spanish Center for Advanced Aerospace Technologies (CATEC) in Seville. It offers localization information about the objects with centimeter accuracy in real time based on 20 VICON cameras.

The aerial robots used were 4 Hummingbird quad-rotors by Ascending Technologies with 200 g payload and up to 20 min of flight autonomy. Their maximum flight altitude was fixed by software to 2.5 m. The area to cover is a rectangular area of  $9 \times 9 \text{ m}^2$  in the center of the testbed (see Fig. 18). The aerial robots were considered homogeneous since they all move at a maximum motion speed of 1 m/s. Communications are constrained via software to 2 m and the coverage path are assumed rectangular in any case.

The 4 quad-rotors implement an algorithm based on the one-to-one coordination to converge in a distributed manner to the area partitioning strategy. As all the robots are considered homogeneous, all of them should cover the same area. Quad-rotors do not know initially nothing about the rest of the team, so, they try to patrol the whole area. When two quad-rotors meet they share the area that they have presently assigned and divide the union between both. Figure 19 shows the actual xy-position in the area of the quad-rotors during the experiment. It shows as the robots can converge to the area partitioning strategy from a distributed manner using a one-to-one coordination technique.



**Fig. 19** This graph shows the positions ( $x$ ,  $y$ ) of the quad-rotors during the experiments. Each color represents a different quad-rotor. A video from the experiment can be viewed in over the time (*vertical axis*) when they use a dividing area strategy. <https://www.youtube.com/watch?v=K8L4vzx7toc>

## 6 Conclusions

A patrolling strategy is required for monitoring mission with multiple UAVs optimizing a refresh time criterion and ensuring the information propagation among the UAVs even under communications constraints.

The cyclic strategy can theoretically obtains the best performance from a refresh time criterion, but it is not useful when the team of UAVs is heterogeneous. Also, it can not ensure that an information is propagated among the UAVs in a finite time. The path partitioning strategy can not obtains the minimum refresh time, but it obtains an acceptable performance both from a latency and a refresh time criterion.

The area partitioning strategy can obtain theoretically the minimum maximum refresh time, even with heterogeneous UAVs. It can also ensure periodical communication between neighbor UAVs, upper-bounding the maximum latency time. This latency time can be minimized depending on the area partition configuration or division. Results show a better performance of this strategy compared to the previous ones.

Finally, two different distributed coordination techniques have been analyzed to evaluate if they allow to achieve convergence to the centralized patrolling strategy. Techniques based on the coordination variables converge quicker than the techniques based on the one-to-one coordination algorithm.

**Acknowledgments** This work has been carried out in the framework of the CLEAR (DPI2011-28937-C02-01) Spanish National Research project and the MUAC-IREN (FP7-PEOPLE-295300) and EC-SAFEMOBIL (FP7-ICT-288082) EU-funded projects. We would also like to thank Miguel Angel Trujillo and Felix Robles for their support with the experiments.

## References

1. Acevedo JJ, Arrue BC, Diaz-Baez JM, Ventura I, Maza I, Ollero A (2013) One-to-one coordination algorithm for decentralized area partition in surveillance missions with a team of aerial robots. *J Intell Robot Syst* 1–17
2. Acevedo JJ, Arrue BC, Maza I, Ollero A (2013) Cooperative large area surveillance with a team of aerial mobile robots for long endurance missions. *J Intell Robot Syst* 70:329–345
3. Acevedo JJ, Arrue BC, Maza I, Ollero A (2013) Cooperative perimeter surveillance using aerial robots and fixed ground stations. In: 2nd RED-UAS 2013 workshop on research, education and development of unmanned aerial systems, Nov 2013
4. Acevedo JJ, Arrue BC, Maza I, Ollero A (2013) Cooperative perimeter surveillance with a team of mobile robots under communication constraints. In: International conference on intelligent robots and systems, Nov 2013
5. Acevedo JJ, Arrue BC, Maza I, Ollero A (2013) Distributed approach for coverage and patrolling missions with a team of heterogeneous aerial robots under communication constraints. *Int J Adv Rob Syst* 10(28):1–13
6. Agmon N, Kaminka GA, Kraus S (2011) Multi-robot adversarial patrolling: facing a full-knowledge opponent. *J Artif Int Res* 42(1):887–916
7. Baseggio M, Cenedese A, Merlo P, Pozzi M, Schenato L (2010) Distributed perimeter patrolling and tracking for camera networks. In: 49th IEEE conference on decision and control (CDC), Dec 2010, pp 2093–2098
8. Beard RW, McLain TW, Nelson DB, Kingston D, Johanson D (2006) Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. *Proc IEEE* 94(7): 1306–1324
9. Carli R, Cenedese A, Schenato L (2011) Distributed partitioning strategies for perimeter patrolling. *Am Control Conf (ACC)* 2011:4026–4031
10. Chevaleyre Y (2004) Theoretical analysis of the multi-agent patrolling problem. In: intelligent agent technology. (IAT 2004). Proceedings of the IEEE/WIC/ACM international conference on, Sept 2004, pp 302–308
11. Choset H, Pignon P (1997) Coverage path planning: the boustrophedon decomposition. In: Zelinsky A (ed) Field and service robotics. Springer, London, pp 203–209
12. Daniel K, Rohde S, Goddemeier N, Wietfeld C (2011) Cognitive agent mobility for aerial sensor networks. *Sens J IEEE* 11(11):2671–2682 Nov 2011
13. Elmaliach Y, Shiloni A, Kaminka GA (2008) A realistic model of frequency-based multi-robot polyline patrolling. In: Proceedings of the 7th international joint conference on autonomous agents and multiagent systems—volume 1, AAMAS’08, Richland, SC, 2008. international foundation for autonomous agents and multiagent systems, pp 63–70
14. Guruprasad KR, Wilson Z, Dasgupta P (2012) Complete coverage of an initially unknown environment by multiple robots using voronoi partition. In: Proc of 2nd international conference on advances in control and optimization in dynamical systems, Bengaluru, India, 16–18 Feb 2012
15. Hazon N, Kaminka GA (2008) On redundancy, efficiency, and robustness in coverage for multiple robots. *Robot Auton Syst* 56(12):1102–1114
16. Kingston D, Beard RW, Holt RS (2008) Decentralized perimeter surveillance using a team of UAVs. *Robot IEEE Trans* 24(6):1394–1404 Dec 2008
17. Marino A, Parker L, Antonelli G, Caccavale F (2009) Behavioral control for multi-robot perimeter patrol: a finite state automata approach. In: Robotics and automation, 2009. ICRA ’09. IEEE international conference on May 2009, pp 831–836
18. Merino L, Caballero F, Martinez de Dios JR, Maza I, Ollero A (2012) An unmanned aircraft system for automatic forest fire monitoring and measurement. *J Intell Robot Syst* 65(1): 533–548
19. Pasqualetti F, Durham JW, Bullo F (2012) Cooperative patrolling via weighted tours: performance analysis and distributed algorithms. *Robot IEEE Trans* 28(5):1181–1188 October 2012

20. Pasqualetti F, Franchi A, Bullo F (2012) On cooperative patrolling: optimal trajectories, complexity analysis, and approximation algorithms. *Robot IEEE Trans* 28(3):592–606
21. Wong CY, Seet G, Sim SK, Pang WC (2010) A framework for area coverage and the visual search for victims in usar with a mobile robot. In: Sustainable utilization and development in engineering and technology (STUDENT), 2010 IEEE conference on Nov 2010. pp 112–118

# Robotic Manipulation Within the Underwater Mission Planning Context

## A Use Case for Benchmarking

Javier Pérez, Jorge Sales, Antonio Peñalver, J. Javier Fernández,  
Pedro J. Sanz, Juan C. García, Jose V. Martí, Raul Marín  
and David Fornas

**Abstract** Nowadays, there is an increasing demand for underwater intervention systems around the world in several application domains. The commercially available systems are far from what is demanded in many aspects, justifying the need of more autonomous, cheap and easy-to-use solutions for underwater intervention missions. The chapter begins making a review of the most important research projects that have been able to demonstrate some results in sea conditions. Then, the expertise and know-how developed in the context of our research group in the last years is presented. Maybe, one of the main achieved results, from the methodological point of view, is a three-layer general system architecture based on the Robot Operating System (ROS), which allows an underwater vehicle to perform intervention missions with a high degree of autonomy, independently of the targeted scenario. Moreover, the use of an underwater simulator as a 3D simulation tool for benchmarking and Human Robot Interaction (HRI) is also discussed. In summary, a methodology has been developed for experimental validation, independently of the specific underwater intervention problem to solve. It consists on the use of the simulator, as a prior step before moving to any of the testbeds used for experimental validation. The reliability and feasibility of this methodology has been demonstrated for intervention missions in sea trial conditions.

**Keywords** Underwater robot · ROS · Mission planning · Grasping and manipulation planning

---

J. Pérez · J. Sales · A. Peñalver · J.J. Fernández · P.J. Sanz (✉) · J.C. García ·  
J.V. Martí · R. Marín · D. Fornas  
IRS Lab, Jaume I University, Castellon, Spain  
e-mail: pedrojose.sanz@icc.uji.es

J. Pérez  
e-mail: japerez@uji.es

## 1 Introduction

The need for intervention in underwater environments is significantly increasing in the last years. A large number of applications in marine environments need intervention capabilities. Potential areas include maintenance interventions in permanent observatories and offshore scenarios, and search and recovery for collecting objects of interest for different application domains like biology, geology, fishery, or marine rescue just to name a few.

Nowadays, these kind of tasks are usually solved with “work class” ROVs (i.e. Remote Operated Vehicles) that are launched from support vessels, and remotely operated by expert pilots through an umbilical communications cable and complex control interfaces. These solutions present several drawbacks. Firstly, ROVs are normally large and heavy vehicles that need significant logistics for its transportation and handling. Secondly, the complex user interfaces and control methods require skilled pilots for their use. These two facts significantly increase the cost of the applications. Moreover, the need of an umbilical cable introduces additional problems of control, or range limitation. Finally, the fatigue and high stress that users of remotely operated systems normally suffer supposes another serious drawback.

All the pointed questions justify the need of more autonomous, cheap and easy-to-use solutions for underwater intervention missions. With this aim, looking for higher autonomy levels in underwater intervention missions, a new concept, named “Autonomous Underwater Vehicles for Intervention” (I-AUV hereinafter), was born during the 90s. It is worth mentioning that this is a very new technology and, according to Gilmour et al. [18]: “Long-term AUV vision the technology for light intervention systems is still immature, but very promising. I-AUVs are currently in level 3 out of 9 (9 meaning routinely used) of the development cycle necessary to adopt this technology in the oil and gas industry, being expected to achieve up to level 7 by the end of 2018”.

However the progress becomes slow for this new technology. In fact, only very few I-AUV prototypes have been tested till date in real underwater scenarios. Among the reasons would be: complexities on required mechatronics (e.g. the vehicle, hand-arm, all kind of sensors, etc.); very hard communication problems; intelligent control architectures needed; letting apart the hostile environment inherent to underwater (e.g. poor visibility, currents, increasing pressure with depth, etc.).

After the pioneering works during the 90s (OTTER [51], ODIN [8] and UNION [40]), significant advances in this direction arrived during the last decade, when the first simple autonomous operations at sea were demonstrated. A dexterous subsea robot hand incorporating force and slip contact sensing, using fluid-filled tentacles for fingers, was developed in the mid 90s in the context of the AMADEUS project (Advanced MAnipulator for DEep Underwater Sampling) [21]. Fixed base manipulation (opening/closing a valve) was demonstrated in ALIVE [14]. Free-floating object manipulation was demonstrated in SAUVIM [25] and object search and recovery was demonstrated in TRIDENT [47]. In summary, to the best of author’s knowledge, only recent projects like SWIMMER [13], ALIVE, SAUVIM, RAUVI [45],

and TRIDENT have been able to demonstrate its performance in sea trials. It is noticeable that currently the only ongoing European project trying to demonstrate sea trials performance is the PANDORA project [17]. A summary of the most relevant international finished projects related to underwater intervention is given in Table 1.

Bearing in mind the aforementioned context, a three-year Spanish Coordinated Project, named TRITON (Multisensory Based Underwater Intervention through Cooperative Marine Robots) was launched in 2012. The TRITON marine robotics research project is focused on the development of autonomous intervention technologies really close to the real needs of the final user and, as such, it can facilitate the potential technological transfer of its results. This research project includes three sub-projects:

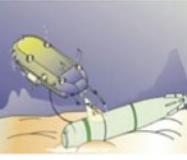
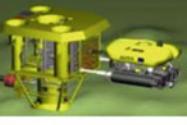
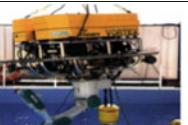
- COMAROB: “Cooperative Robotics”, under responsibility of Universitat de Girona (UdG)
- VISUAL2: “Multisensorial Perception”, under responsibility of Universitat de les Illes Baleares (UIB)
- GRASPER: “Autonomous Manipulation”, under responsibility of Universitat Jaume-I (UJI)

The project proposes two scenarios as a proof of concept to demonstrate the developed capabilities: (1) the search and recovery of an object of interest (e.g. a “black-box mockup” of a crashed airplane), and (2) the intervention on an underwater panel in a permanent observatory. In the area of search and recovery, previous projects like SAUVIM, RAUVI and more recently TRIDENT, have become milestone projects, progressively decreasing the operational costs and increasing the degree of autonomy. With respect to the intervention on an underwater panel, the ALIVE project demonstrated the capability of an underwater vehicle to dock autonomously with a ROV-friendly panel by using hydraulic grabs. Nevertheless, unlike in TRITON, a very simple automata-based manipulation strategy was used to open/close a valve. Finally, it is worth mentioning that currently only PANDORA has some similarities with TRITON, where a learning solution for autonomous robot valve turning, using Extended Kalman Filtering and Fuzzy Logic to learn manipulation trajectories via kinaesthetic teaching was recently proposed [1, 6].

The work presented in this chapter is mainly concerning GRASPER, focusing on one of its recent achievements: endowing an I-AUV with the ability to manipulate an underwater observatory panel in an autonomous way.

Moreover, the use of an underwater simulator as a 3D simulation tool for benchmarking and Human Robot Interaction (HRI) is also presented. Several definitions of the term benchmark have been proposed in the literature. In this chapter, the one stated in [11] is taken, in the sense that a benchmark adds numerical evaluation of results (performance metrics) as a key element. Moreover, the main aspects are repeatability, independency, and unambiguity. This objective, numerical evaluation (also known as performance metrics), will allow a fair comparison of algorithms from different origins.

**Table 1** Summary of the most relevant international finished projects related to underwater intervention

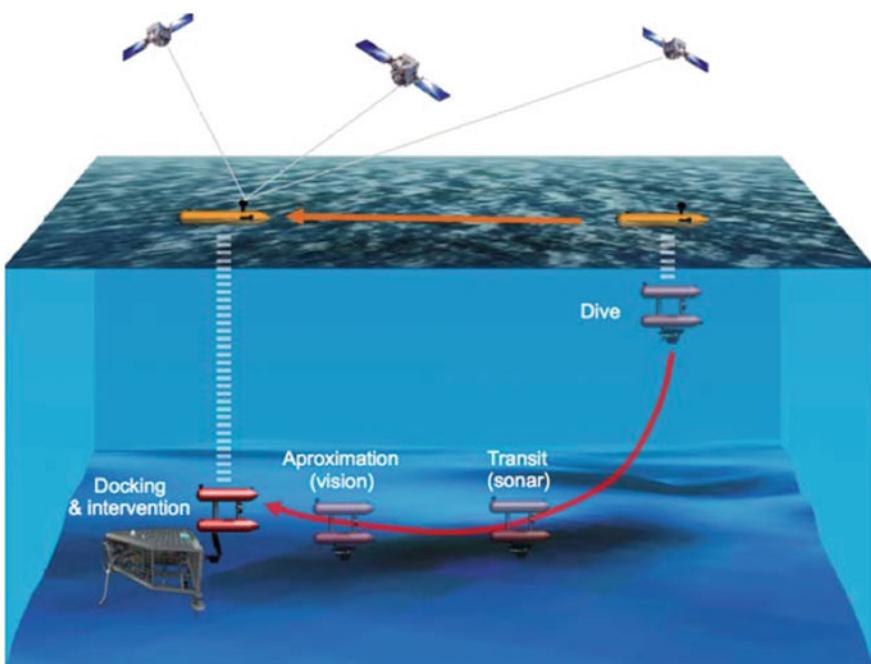
	Concept		Result		Description	
	TRIDENT	2010/13			TRIDENT was a EU funded project that proposed a methodology for multipurpose underwater intervention tasks using an I-AUV endowed with a dexterous manipulator and a 3-fingered hand. The intervention was based on two phases: Survey and Intervention. Final tests were demonstrated in a harbour environment in an uncoupled way: 1) The capability of both vehicles working in tandem during mapping and 2) the capability of the I-AUV to intervene over the target.	
	SAUVIM	1997-2009			SAUVIM is a project funded by the Office of Naval Research and carried out at the Autonomous System Laboratory of the University of Hawaii. It is conceived as an AUV with accurate navigation and station keeping capabilities to allow for the recovery of seafloor objects. In particular, SAUVIM is supposed to use its passive arm to localize itself with respect to the object of interest and use its 7 DOF electro-mechanical arm to carry out an intervention. SAUVIM was initially designed to recover test missiles from the seabed for the Pacific Missile test Centre in Hawaii.	ONR
	ALIVE	2001/4			The ALIVE vehicle is equipped with docking and 7 DOF manipulation arms to complete valve override and hot stab connections. It can also be used for the deployment and recovery of acoustic or seismic beacons at sea-bottom. During the final demo of the project, ALIVE proved its capability to autonomously navigate, dock and operate on an underwater panel similar to those of the oil industry.	EU
	SWIMMER	1999-2001			A hybrid AUV/ROV intervention system provides an efficient way to ensure permanent Inspection, Maintenance, and Reparation operations over deep-water oil production facilities. A ROV umbilical is integrated between the surface facility and the subsea site. The SWIMMER AUV transports the ROV to the subsea site and connects the ROV to the umbilical at the subsea location where it can be normally operated from the surface.	EU
UNION	AMADEUS	1993/96/99	EU	1999-2001	AMADEUS phase I (left) represents the first attempt in developing a dexterous gripper suitable for underwater applications. The 3-fingered gripper was hydraulically actuated and coordinately controlled by mimicking, within each finger, the motions of an artificial elephant trunk. AMADEUS phase II (right) instead studied the coordinated control of two underwater 7 DOF electro-mechanical arms. Each arm is equipped with an underwater JR3 force/torque wrist sensor.	EU
EU	1996/99				The main goal of the Union Esprit Basic Research Action was to develop methods for increasing the autonomy and intelligence of Underwater Remotely Operated Vehicles (ROVs). The project focused mainly on the development of coordinated control and sensing strategies for combined manipulator and vehicle systems. At the end, only experimental validation within simulation conditions was provided.	

According to our methodology, we always test the algorithms first in simulation and then in real conditions, with increasing degree of complexity: water tank, pool, harbour, shallow water, etc. The obtained results are related to simulations and also to water tank conditions, while we are currently working on the challenge of testing the manipulation capabilities in the sea.

## 2 Underwater Intervention Mission Planning

For better understanding of the required mission planning issues, the specific context of TRITON (2012–15) project will be used. The main goal of TRITON is the use of autonomous vehicles for the execution of complex underwater intervention tasks. The project is focused on the use of several vehicles (an ASC, *Autonomous Surface Craft*, and an I-AUV) running in a coordinated manner during the execution of a mission, and on the improvement of the manipulation capabilities required for intervention (i.e. opening/closing a valve, plugging/unplugging a connector, etc.).

The mission scenario that we are currently working on (the panel intervention in the context of underwater observatories), to be developed autonomously, is structured in 5 phases (see Fig. 1):



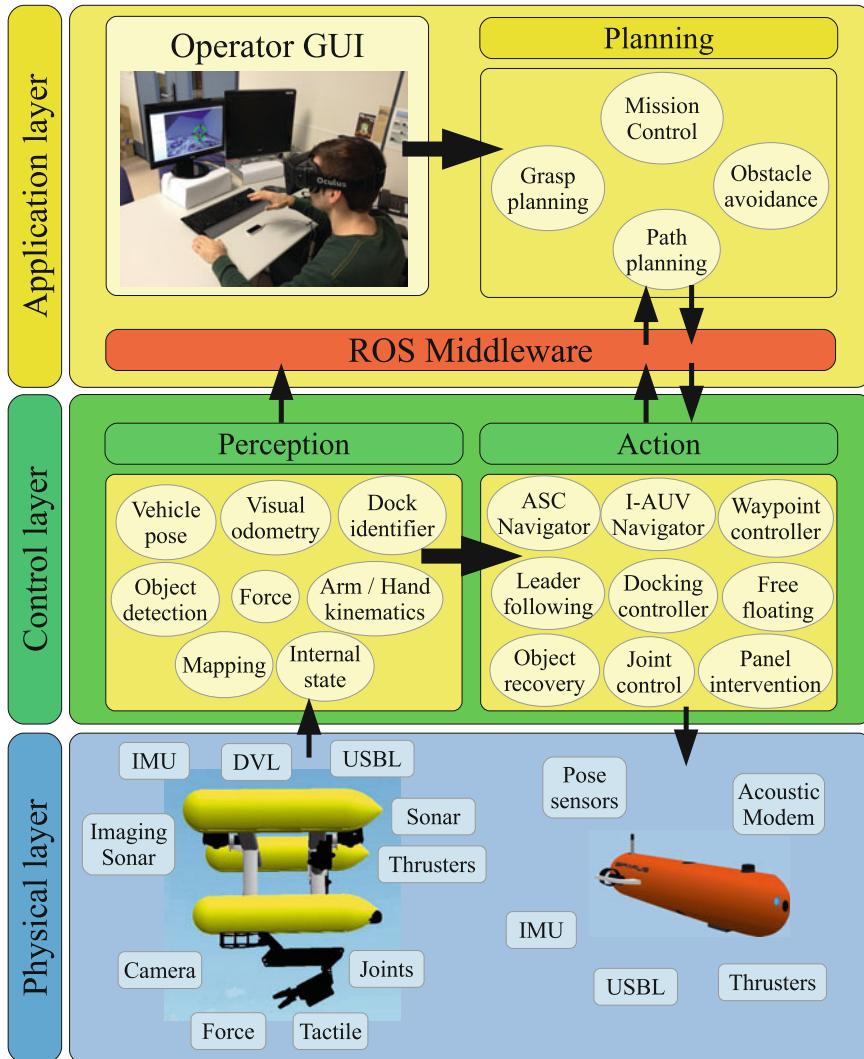
**Fig. 1** Underwater panel intervention scenario considered in TRITON project. The mission is structured in 5 phases: Dive, Transit, Approach, Docking and Intervention

1. **DIVE:** Both vehicles are sequentially deployed from the support boat. Then, the I-AUV dives a few meters until establishing an acoustic communication link with the surface vehicle. Next, the I-AUV descends to the bottom, while the ASC describes circles on the surface to better localize and geo-reference the underwater robot. At the bottom, the I-AUV performs station keeping while being geo-referenced by the ASC, which forwards an absolute position fix.
2. **TRANSIT:** The vehicle uses cooperative navigation with the surface AUV until reaching the acoustic area of coverage of the panel-mounted transponder. Then, the vehicle uses a transceiver to interrogate the transponder mounted on the permanent observatory. Using its dead reckoning navigation system combined with range only navigation techniques, the vehicle estimates the position of the observatory and transits towards it.
3. **APPROACH:** When the vehicle reaches the surroundings of the observatory, establishes visual contact, identifying the AUV-friendly intervention panel where it should dock. To achieve the robust accurate navigation requirements needed for docking, the vehicle switches to real-time vision based navigation relative to the panel.
4. **DOCKING:** Real time vision based localization techniques will be used to visually guide the vehicle during the docking. Three non-actuated mechanical bars will be used for docking to the panel using passive accommodation techniques. When the I-AUV docks, it becomes rigidly attached to the panel.
5. **INTERVENTION:** Once the vehicle is rigidly attached to the panel, the manipulation operation takes place. As proof of concept, two demonstrative applications have been designed with increasing complexity: (1) Opening/Closing a valve, and (2) Plugging/Unplugging a connector.

## 2.1 The System Architecture

Concerning the implemented architecture required for the current intervention system, the high level structure can be observed in Fig. 2. Obviously, general mission planning considerations are out of the scope of this work and, in the following, only “grasping and manipulation” aspects will be taken into account. In the figure, the real mechatronics used in the TRITON project are represented: the Girona500 I-AUV [39], equipped with the Light-Weight ARM5E 4 DOF underwater robotic arm [15], and the SPARUS AUV [24], used as surface vehicle in the mission.

The whole I-AUV control architecture is composed of two initially independent architectures: the underwater vehicle and the manipulator architectures. Concerning the manipulator architecture, the reactive actions are performed in the low-level control layer that communicates with the real or simulated I-AUV via an abstraction interface. The control layer also includes control strategies like *station keeping* or *free floating* to help in the manipulation actions. The station keeping approach allows to keep the position and orientation of the vehicle to facilitate the intervention. A combination of vision and inertial measurement systems are used to achieve this purpose.



**Fig. 2** Generic mission planning architecture for the intervention system

With this approach, it is possible to use the arm degrees of freedom to perform the desired manipulation [36]. The free floating approach uses all the available degrees of freedom, both from the vehicle and the arm, to increase the total amount of space configurations for a required task. In the TRIDENT project, a strategy based on the prioritization of tasks of equality and inequality type, once combined with Dynamic Programming techniques, was used for coordinately controlling the motion of the I-AUV [7]. In [49], real intervention experiments in sea conditions are described,

in which task priorities and a dynamic programming based approach is used for underwater floating manipulation.

At a higher level, the whole mission is supervised at a high level by a *Mission Control System (MCS)*, implemented using the Petri net formalism [36].

The *Robot Operating System (ROS)* [38], is used to integrate the heterogeneous computing hardware and software of all the system components, to allow for easy integration of additional mission-specific components, and to record all sensor input in a suitable playback format for simulation purposes.

The mission control system is the part of the control architecture in charge of defining the task execution flow to fulfill a mission. Each task can be executed by means of some manipulator action. The mission programmer must define how these actions/primitives are executed to fulfill each task and how the tasks are combined to fulfill the whole mission. The MCS was developed as generic as possible and it allows for an easily tailoring to different control architectures (refer to [36] for further details).

## 2.2 Planning Grasping and Manipulation for Intervention Missions

Planning a grasp is generally known to be a difficult problem due to the large search space resulting from all possible hand configurations, grasp types and object properties that occur in regular environments. The dominant approach to this problem has been the model-based paradigm, in which the object shape, contacts, and forces are modelled according to physical laws. Then, the research has been focused on grasp analysis (the study of the physical properties of a given grasp) and grasps synthesis (the computation of grasps that meet certain desirable properties) [48]. Unfortunately, these approaches have failed to deliver practical implementations, mainly because they rely on assumptions that are difficult to satisfy in complex and uncertain environments.

The current trend is to incorporate sensor information for grasp planning and synthesis, such as vision [9, 10, 19, 30, 46] or range sensors [41]. In this line, several approaches have also adopted machine learning techniques to determine the relevant features that indicate a successful grasp [10, 20, 29, 44]. Others make use human demonstrations for learning grasp tasks [12]. Most of these approaches commonly consider grasps as a fixed number of contact locations with no regard to hand geometry [4, 48]. Some recent work includes kinematics constraints of the hand in order to prune the search space [5, 27, 28]. Alternatively, the so-called knowledge-based approach tries to simplify the grasp planning problem by reasoning on a more symbolic level. Objects are often described using shape primitives [22, 50], grasp prototypes are defined in terms of purposeful hand preshapes [27, 28], and the planning and selection of grasps is made according to programmed decision rules [3].

Recently, the knowledge-based approach has been combined with vision-force-tactile feedback and task-related features that improve the robot performance in real scenarios [35].

Regarding autonomous manipulation in underwater environments, very few research has been carried out. So, the first fully autonomous intervention at sea, was demonstrated by the ALIVE project, where a hovering capable AUV was able to home to a subsea intervention panel using an imaging sonar, and then, docking into it with hydraulic grasps using visual feedback. Once attached to the panel, a very simple manipulation strategy (fixed base manipulation) was used to open/close a valve. First object manipulation from a floating vehicle (I-AUV) was achieved in 2009 within SAUVIM project. It was demonstrated the capability of searching for an object whose position was roughly known a priori. The object was endowed with artificial landmarks and the robot autonomously located it and hooked it with a recovery device while hovering.

Recently, the first multipurpose object search and recovery strategy was demonstrated in the TRIDENT project in 2012. First, the object was searched using a down-looking camera and photo-mosaicing techniques. Next, it was demonstrated how to autonomously “hook” the object in a water tank [36]. The experiment was repeated in a harbour environment using a 4 DOF arm [33] and later with a 7 DOF arm endowed with a 3 fingered hand [43, 47].

In summary, grasping and manipulation remain open research problems, and this situation becomes drastically worst in underwater scenarios. In the shallow water context, new complexities arise increasing the difficulty to control grasping and manipulation actions with agility capabilities. Under these very hostile conditions, only a few robot systems are endowed with semi-autonomous manipulation capabilities, mainly focused in specialized operations requiring an environment reasonably structured, like those devoted to the offshore industries.

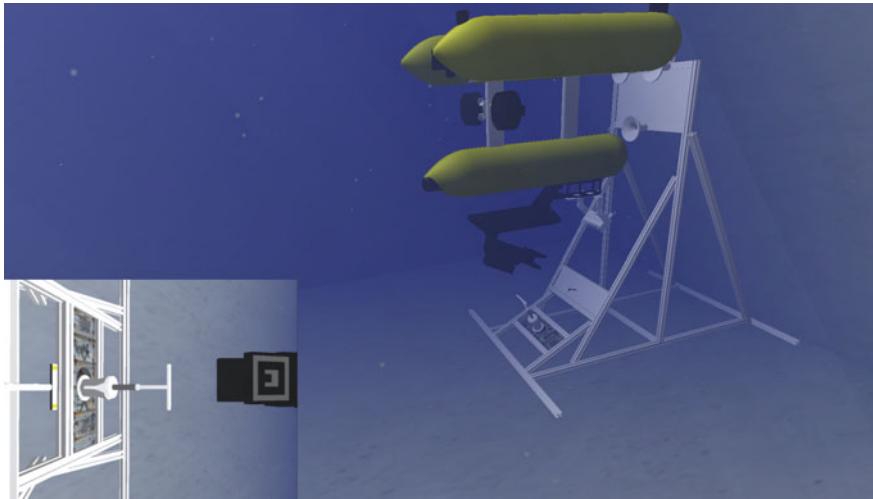
For further bibliography related to the motion control of I-AUVs and its manipulation systems, refer to [2], that addresses the main control aspects in underwater manipulation tasks; and [26], which provides an extensive tract on sensory-based autonomous manipulation for intervention tasks in unstructured environments.

### 3 UWSim: A 3D Simulation Tool for Benchmarking and HRI

UWSim<sup>1</sup> [34] is a software tool for visualization and simulation of underwater robotic missions (see Fig. 3). The software is able to visualize underwater virtual scenarios that can be configured using standard modeling software and can be connected to external control programs by using the *Robot Operating System (ROS)* [38] interfaces. UWSim is currently used in different ongoing projects funded by European Commission (MORPH [16] and PANDORA [17]) in order to perform HIL (Hardware

---

<sup>1</sup> Available on-line: <http://www.irs.uji.es/uwsim>.



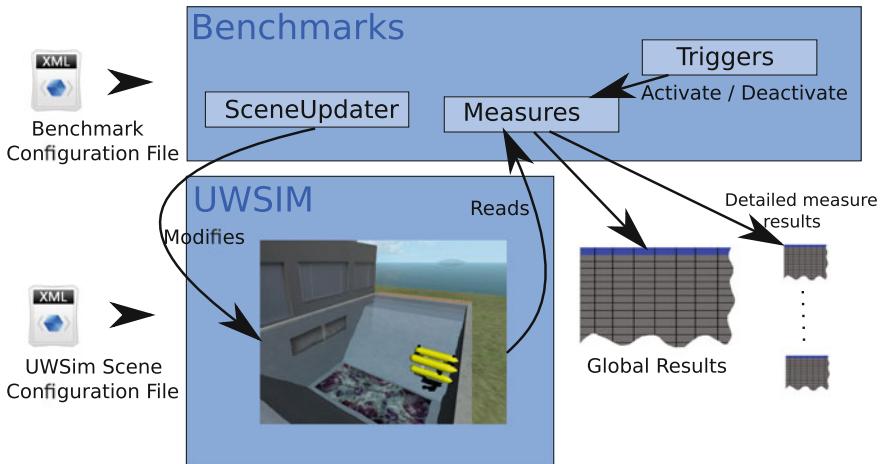
**Fig. 3** UWSim simulator, displaying the underwater panel intervention mission scenario proposed in TRITON and executing a vision *benchmark*. The I-AUV is docked to the observatory underwater panel structure and is ready to perform the intervention: open/close a valve

in the Loop) experiments and to reproduce real missions from the captured logs. UWSim is not only useful for software validation, but also for defining *benchmarking* mechanisms inside the simulator, so that control and vision algorithms can be easily compared in common scenarios. UWSim is also used as a *Graphical User Interface (GUI)* providing the necessary *Human Robot Interaction (HRI)* that is required to specify a task.

### 3.1 The Benchmarking Module for UWSim

A *benchmarking* module is available to be used with UWSim [37]. Like UWSim, this module uses ROS to interface with external software with which it can interact. The ROS interface allows the external program to be evaluated and can communicate both with the simulator (it can send commands to carry out a task) and the *benchmarking* module (it can send the results or data necessary to be evaluated).

*Benchmarks* are defined in XML (*eXtensible Markup Language*) files. Each file will define which measures are going to be used and how they will be evaluated. This allows the creation of standard *benchmarks* defined in a document to evaluate different aspects of underwater robotic algorithms, being able to compare algorithms from different origins. Each of these *benchmarks* will be associated with one or more UWSim scene configuration files, being the results of the *benchmark* dependent on the predefined scene. The whole process is depicted in Fig. 4. Detailed information on how to setup and execute a *benchmark* in UWSim can be found in our previous work [37].



**Fig. 4** Benchmarking module flow diagram: a benchmark configuration is loaded into the benchmark module, and a scene is loaded into the simulator. Then, the benchmark module produces some results that can be logged for posterior analysis

### 3.2 A User Interface for UWSim to Provide HRI

Traditionally, Remotely Operated Vehicles (ROVs), which are commercially available to develop all kind of intervention missions, are teleoperated by an expert user by means of a specific *Graphical User Interface (GUI)* (thus providing the necessary *Human Robot Interaction, HRI*) thanks to the tethered cable which connects the robot to the oceanographic vessel. The main drawback of this kind of systems, apart from the necessary expertise degree of the pilots, concerns the cognitive fatigue inherent to the master-slave control architectures. The evolution of this kind of robots (ROVs) are the Intervention Autonomous Underwater Vehicles (I-AUVs). These robots can perform some tasks autonomously, but the presence of the operator in the programming phase, is still required. Most of the GUI used in these robots use their own programming language, and the GUIs tend to be more complex, with lots of windows displaying information. So, this kind of GUIs are very suitable for expert users, but are very difficult to use for non-expert users.

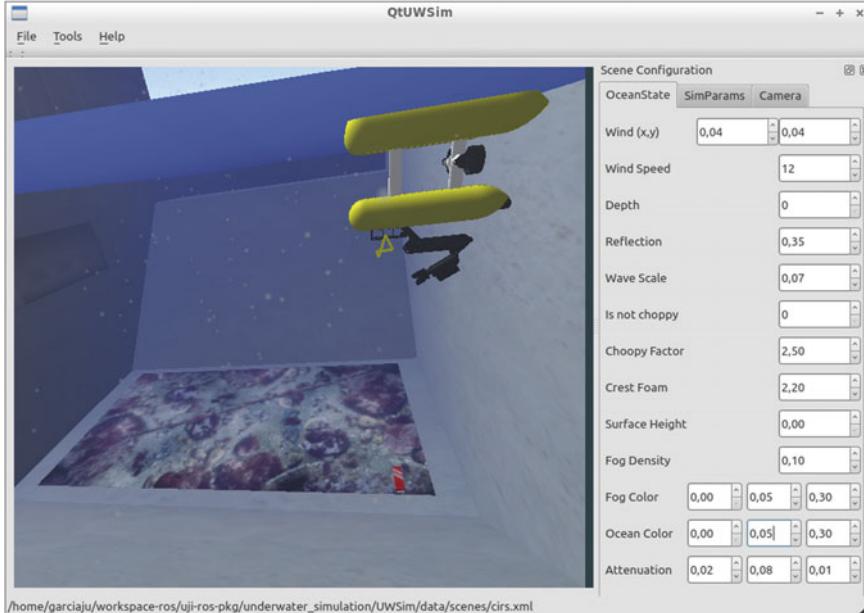
From our previous work and the know-how developed in the context of the aforementioned RAUVI and TRIDENT projects, a GUI is being developed by following a twofold strategy: (1) to guarantee the “intelligence” of the system and a good system performance, including the user in the control loop, and (2), not to require the user intervention in a continuous way like in ROV’s, just when it is strictly necessary. Despite the fact that we assume that the user has a minimum level of abilities related to the mission to be carried out and the robot to be used, the GUI is oriented to non-expert users. In order to include full 3D support at all the stages of the mission, the GUI is being integrated with the UWSim simulator. This will allow us to perform

realistic simulations and take advantages of visual aspects like 3D representation, Virtual and Augmented Reality (VR and AR), and general good system performance. In order to integrate the GUI in the whole project architecture, ROS is being used as a middleware.

The GUI will adapt the design and the information to show to the user, depending on the intervention to perform. Thus, when the user selects a “panel intervention” type of mission, the scenario configuration and the intervention panel CAD/VRML<sup>2</sup> files are loaded. Then, the user will be able to navigate through the scenario, looking for the target, and will get all the panel details and will select the predefined actions: plug-in a cable or valve operation. Nevertheless, some modification over these predefined actions could be done by using a specific menu.

Once the intervention is defined, it can be tested in the simulator or can be downloaded to the robot through the ROS communication module. In Fig. 5, the GUI integrated with UWSim (named *QtUWSim*) shows the panel to configure the scene environment.

Moreover, a 3D interface with a VR and AR layer is being developed, focusing on the human hand interaction (using a hand tracker device) and the vision (using a *Head-mounted Display, HMD*), allowing the user to interact with the scene with the support of interactive markers. An “interactive marker” is a marker that can be applied to an object in a 3D scene and allows the user to interact with it. Depending



**Fig. 5** *QtUWSim* graphical user interface showing the panel to configure the scene environment

<sup>2</sup>Computer Aided Design/Virtual Reality Modelling Software.

**Fig. 6** Oculus Rift VR provides a full immersion into the scene while the LeapMotion device facilitates the robot control with natural gestures



on the type of interactive marker, the user can perform either translations or rotations over the object, in one of the spatial axes. So, when the user selects the “grasp specification 3D” option, the end effector of the I-AUV defined in an URDF<sup>3</sup> file is loaded into the 3D scene. This end effector, which will be surrounded by 6 interactive markers (3 translational and 3 rotational), can be defined by a hand, a hook or a jaw. The user moves these interactive markers to indicate the end effector position and orientation to reach the target. These movements are currently done by the user with the mouse/trackpad, but a ROS package is being developed in order to allow the use of a hand tracker device. This will allow the user to interact with the GUI more fluently and in a more natural way (see Fig. 6).

The use of a *Head-mounted Display (HMD)* benefits the user, evolving him/her in a more realistic environment. One of the current development is to adapt this kind of device in order to get the most benefit to the VR and AR layer. Furthermore, if the HMD is endowed with sensors, these could be used to move the camera point-of-view, adding more realism to the scene.

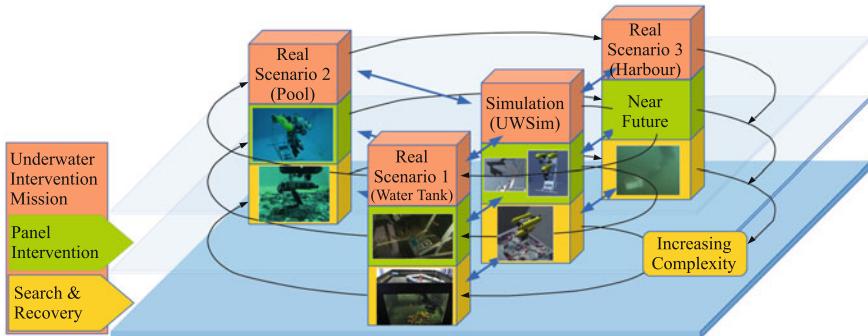
## 4 The Roadmap for Experimental Validation

Following the know-how generated through our recent projects (i.e. RAUVI, TRIDENT, TRITON), a methodology has been developed for experimental validation, independently of the specific underwater intervention problem to solve.

As can be seen in Fig. 7, the four generic steps designed for the experimental validation roadmap are defined (see red blocks), highlighting its instantiation for two different underwater intervention missions: (1) the search and recovery problem

---

<sup>3</sup>Unified Robot Description Format.



**Fig. 7** Developed methodology used to guarantee the success in the final sea trials of an intervention system, independently of the available mechatronics and specific scenarios. On the *top*, the generic approach is described, and two different instantiations are highlighted in the lower planes. Always in increasing complexity, the starting point will be the simulation test (UWSim block); followed by intervention trials, without the vehicle but with the real hand-arm mechatronics and sensors, under water tank conditions (Water Tank block, UJI); after succeed here the complete system, including now the vehicle, is tested in a pool (Pool block, UdG); finally, the sea trials will be carried out (Harbour block)

(under RAUVI and TRIDENT projects) (see yellow blocks) and (2) manipulation on a panel (under TRITON project) (see green blocks).

This methodology becomes very successful independently of the mechatronic system and the testbed used for experimental validation. As can be observed in Fig. 7 (red blocks), the idea is to start out with the performance test on the simulator (UWSim block), where the mechatronics, sensors, and scenario have been modelled in advance.

After succeeding in different current and visibility conditions, the next step will be the intervention trial, without the vehicle but with the real hand-arm system and sensors, and real devices to manipulate, on the water tank available in UJI (Water Tank block). An iterative process will follow here, between simulation and real tests, until complete succeed in the water tank conditions.

Later, the complete system integration, including now the vehicle, and real performance will be carried out in the pool available at UdG (Pool block). This is the last step before the sea trials (Harbour block). Obviously, the iterative process between UWSim and real tests will be always running, looking for success.

## 5 Simulation Results

By using the aforementioned *benchmarking* module for the UWSim simulator, we are able to setup many configurable options. Algorithms can be tested to their limits, to know under which conditions they can work, and which results can be obtained with

them. This way, resources can be optimized to provide the best results in each situation. In the following sections, two different *benchmarks* for UWSim are explained, followed by the experimental results. The first one is a visibility *benchmark*, where a visual tracking algorithm is evaluated under different visibility conditions. The second one, is a position error *benchmark*, where a pattern recognition algorithm is evaluated to see if it can be used to estimate the end effector position of a robot manipulator arm depending on the distance from the camera to the visual marker.

## 5.1 Benchmarking: Visibility Tracker

Below is an example of *benchmarking* done with UWSim. In this case, the goal is to evaluate how the underwater fog affects a visual ESM tracking algorithm [23], as done in our previous work with a black-box mockup [37]. Now, we will use the TRITON scenario, that includes the underwater panel and the Girona500 I-AUV, equipped with the Light-Weight ARM5E arm and a camera. We are considering now that the vehicle has already done the docking to the panel, but despite this fact, we are still interested in keeping track of it with the camera, as the intervention requires to manipulate the valve and connector installed on the panel. Thus, with the aid of this *benchmark* we will evaluate how the algorithm is able to keep track of the intervention panel while visibility conditions change.

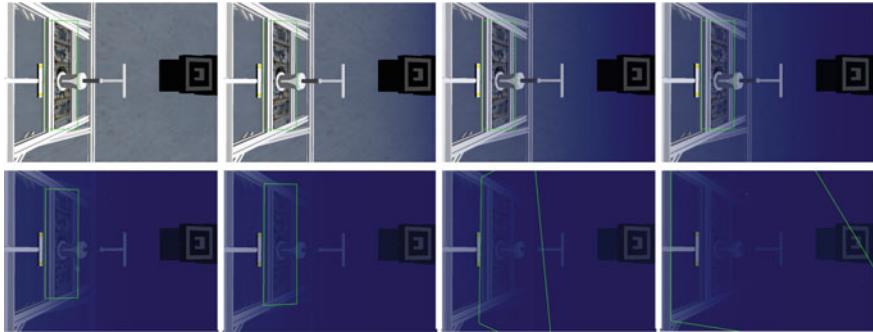
The configuration files for the scene and the *benchmark* are the same that were used for the black-box recovery example mentioned before [37]. It includes measure definitions needed to evaluate the performance of the tracking. Since the tracking algorithm returns the position of a four-corner object, an “euclideanNorm” measure is used, which measures the distance between the position returned by the tracking software and the real position on the simulator.

This measure is divided in two parts to get more information. On one hand, the distance between the actual corners with the ones that the tracking algorithm returns. On the other hand, the real distance from the centroid of the simulated object to the one calculated through vision.

For the final result, these two measurements are added, so that, the lower the result, the smaller the object recognition error is. In addition to these measurements, the scene updater “sceneFogUpdater” is configured varying the underwater scene visibility through time.

Finally, some triggers have been set up to make the evaluation task easier. The *benchmark* module will wait for a service call made by the tracking algorithm, and it will end when there are no more “sceneFogUpdater” iterations. The measurements will always be active, as it is taken as valid the last one received by the ROS “topic” that the vision system sends is taken.

Once the simulator and the *benchmark* are configured, a service call must be added in the tracking algorithm when it starts, and the estimated position of the box must be sent to the *benchmark* module. As shown on Fig. 8, the tracking algorithm



**Fig. 8** Tracking algorithm (represented as a *green line*) screenshots for decreasing visibility in the *benchmark*

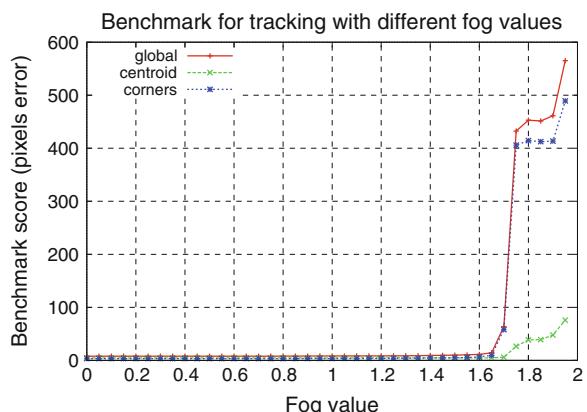
is able to find the manipulation panel while the fog is increasing in the *benchmark*, until finally it is completely lost when the visibility is very poor.

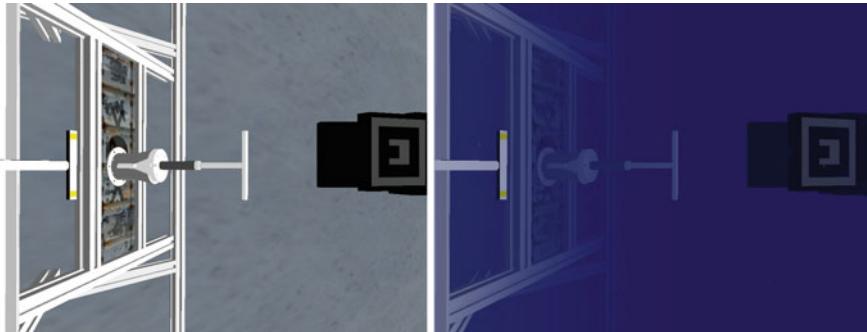
Once the *benchmark* is complete, the module stores the results in a file. These results are stored in a text file in table format. This file can be processed later with any statistical or graphical tool. For this case study, the results can be seen on Fig. 9. It can be observed how the tracking software error is very small throughout the experiment, less than 5 error pixels. When the fog level increases the value above 1.7, the error of the tracking algorithm increases drastically.

As we can see on the graph, the *benchmarking* module offers results for every measure, allowing the user to analyze the performance of the algorithm. In this case we can see how corners information is completely lost at 1.65 fog factor, centroid is still near the objective. So we can conclude that with fog factor bigger than 1.6 is not precise enough to do manipulation although it almost know where the target is.

According to the results provided, the vision system is reliable for fog levels below 1.6. Figure 10 shows a comparison between this levels of fog on UWSim

**Fig. 9** Tracking position error for corners and centroid with increasing fog





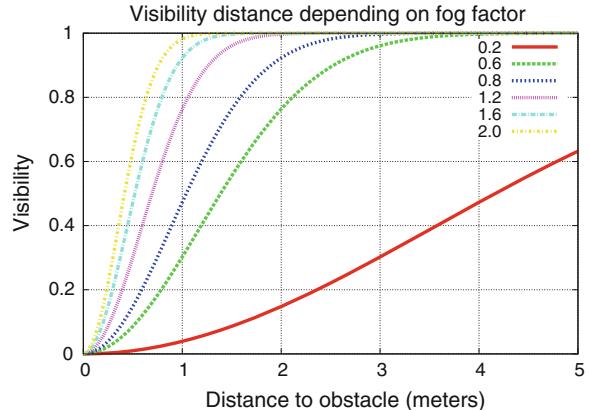
**Fig. 10** Comparison between fog levels 0 and 1.6 in the simulator's camera

simulator screenshots. The fog level is a value ranging from 0 to infinity and defines the visibility in the water depending on the distance. Visibility is a value between 0 and 1 where 0 represents a perfect visibility of the object and 1 represents no visibility at all. The visibility depends therefore on the water fog level and on the distance to the object, as it is represented by the following formula:

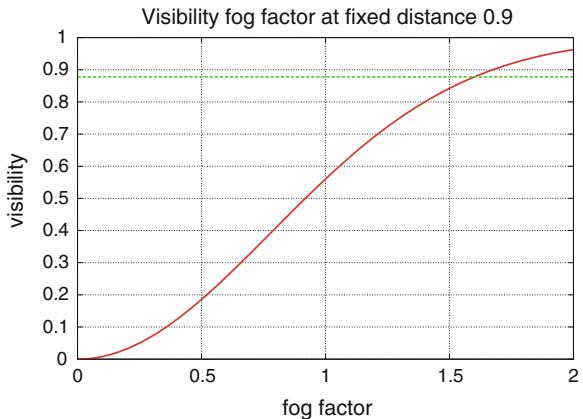
$$\text{visibility} = 1 - e^{-(\text{fog factor} * \text{distance})^2}$$

In Fig. 11, different values have been used to plot the relationship between visibility and the distance to the object. As can be seen, visibility drastically worsens with relatively small values of fog when the distance to the object increases. Under a 1.60 value of fog (represented in a cyan color line, which was the operating limit of the tracking software in this experiment), there is virtually no visibility for a distance greater than 1 m.

**Fig. 11** Visibility and distance relation for different fog levels



**Fig. 12** Visibility and fog factor relation for a fixed distance of 0.9 m



In Fig. 12, the distance to the object has been set to 0.9 m, which is actually the distance between the camera and the panel used in the *benchmark*, and it represents the visibility with respect to the fog factor. The value of visibility for a fog factor of 1.6 is depicted with a horizontal line. Thus the tracking algorithm is able to find an object when the degree of visibility is below 0.878, which is almost the same result as the one obtained using the same tracker in a different environment in the previous work [37].

## 5.2 Benchmarking: Position Error on End Effector Position

In this section, a position error *benchmark*, is defined to evaluate if a pattern recognition algorithm can be used to estimate the end effector position of a robot manipulator arm, compared to a kinematic solution, and thus, if the algorithm can be used for manipulation purposes. The results of the experiment will allow choosing the best way to estimate the end effector position when performing a manipulation.

The pattern recognition algorithm will estimate the position of a marker placed on the gripper of the Light-Weight ARM5E robotic arm. Using this method, some errors such as a bad initialization of the arm or miscalibration of the joints that affect to the kinematics of the arm, can be avoided.

The marker is detected using the ARTToolkit library (a software library for building Augmented Reality applications) that, among others, provides multiple methods for detecting and localizing the position and orientation of a marker. In order to do this, the arm moves in the camera field of view and the position error of the end effector is measured by the two different proposed systems.

The first approach, the direct kinematics, estimates the end effector position numerically using the known joints transforms from the base of the arm to the end effector. The advantage of this method is that it does not depend on the cameras, so it's immune to poor visibility. The main drawback is that some errors appear from bad initialization of the joints position and miscalibration of the joints, which depend on self-positioning sensors. To simulate the errors of the real arm, small offsets were applied to the joints, thus simulating this kind of errors in each joint.

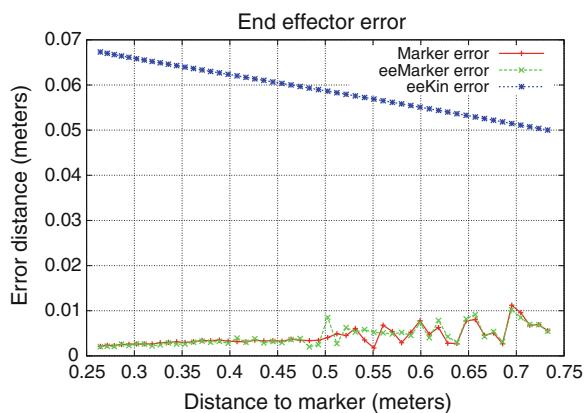
The second method uses the pattern recognition algorithm that finds the marker placed in the hand. Then, a transform from this position to the end effector is used. In this case, a low visibility can be an important disadvantage, but on the other hand, as most of the intervention missions require vision systems to find targets to manipulate, using this approach makes target and manipulator be referenced from the same origin, avoiding arm-camera calibration.

As can be seen on Fig. 13, marker error is significantly smaller than kinematic error. This is caused by small errors on joints, mainly the joints that are far in the kinematic chain from the end effector, the ones that produce big errors on kinematics. The marker approach allow us to avoid this kinematic chain errors driving the error to 0.003–0.01 m, which is a good error in order to manipulate the panel.

Another interesting result is that kinematic errors decrease when the target is far from the camera, while marker detection error increases and becomes unstable. The increase in the marker position error is caused by the fact that even small errors in the camera space produce appreciable precision errors. To avoid this, higher resolution cameras could be used. Instability is probably caused by light effects such as shadows, reflexes, etc.

To sum up, it seems that marker estimation is better than kinematics, although kinematic errors depend on each arm sensors and may be smaller depending on the arm used. In the particular context of the TRITON project, a hybrid solution was adopted, allowing changing the method depending on markers visibility because kinematic errors were too high to achieve a manipulation in a robust manner.

**Fig. 13** End effector position error evolution comparison: (1) using kinematics estimation (blue line), and (2) marker pose estimation (green line)



## 6 Real Scenarios Results

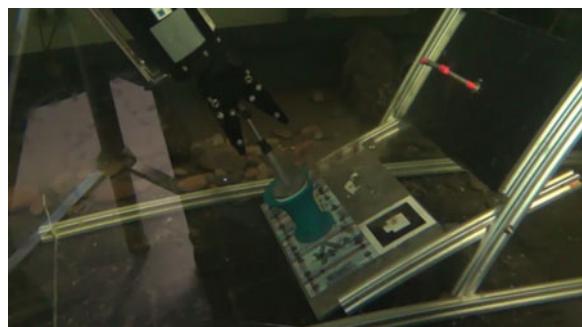
Once the algorithms to perform the proposed intervention (open and close a valve and plug and unplug a *hot-stab* connector) have been tested in simulated scenarios, the action moves to the real ones, according to our roadmap.

### 6.1 Intervention on a Panel in Water Tank Conditions

The first real scenario in our roadmap is the Water Tank at UJI (see Fig. 14), where an intervention panel mockup is installed inside the tank. In this scenario we test the manipulation actions, with the real hand-arm mechatronics and sensors. In this scenario, the complete AUV system is not used: at this point, it is only important the real hand-arm mechatronics and sensors (i.e. the Light-Weight ARM5E equipped with sensors).

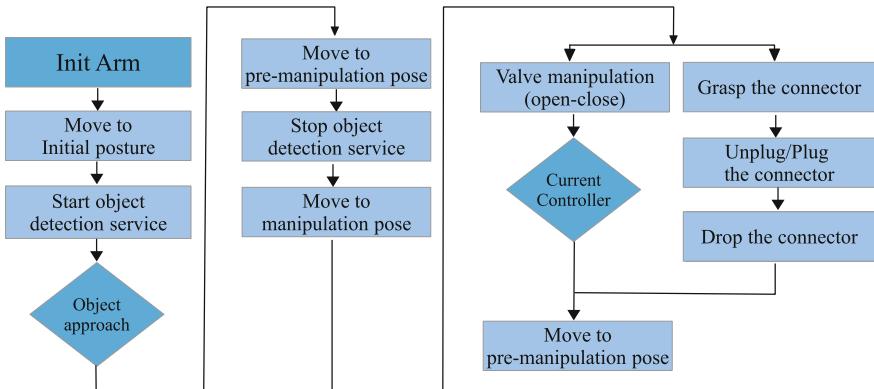
The detailed steps for the proposed dual operation (open and close a valve and plug and unplug a connector) are highlighted in a flow chart in Fig. 15. The first step is the system initialization, in this case, the arm. After the system has been completely initialized, the manipulation execution plan starts. In order to reach the positions to manipulate the objects in a correct way, some waypoints respect to the position of the object have been defined (see Fig. 16, where the frames are represented on a virtual visualization of the scene). To reach each waypoint, the system calculates the Cartesian distance between the end-effector and the waypoint, and using Cartesian velocities, the end-effector tries to reach the position of the waypoint. Now, depending of the intervention to perform (valve or connector) a series of steps are followed (refer to [32] for more details). Video sequences of the two interventions can be seen on-line.<sup>4</sup>

**Fig. 14** Intervention in water tank conditions at UJI: valve and connector (*hot-stab*) manipulation

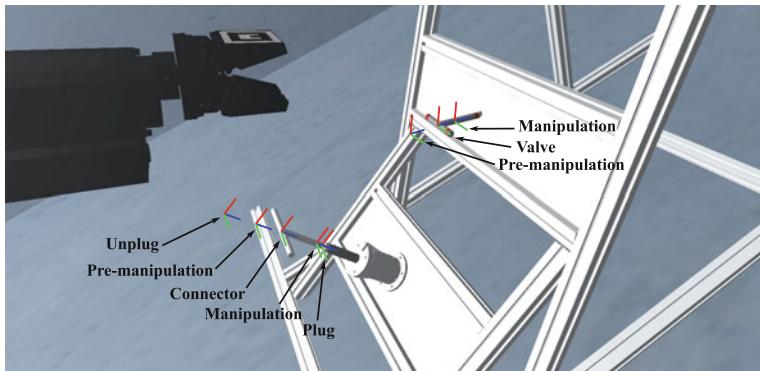



---

<sup>4</sup>Valve and connector autonomous intervention: (1) (side) <http://youtu.be/6pYBL-6Tw4c>, (2) (top) [http://youtu.be/\\_WkQYtcLsMU](http://youtu.be/_WkQYtcLsMU).

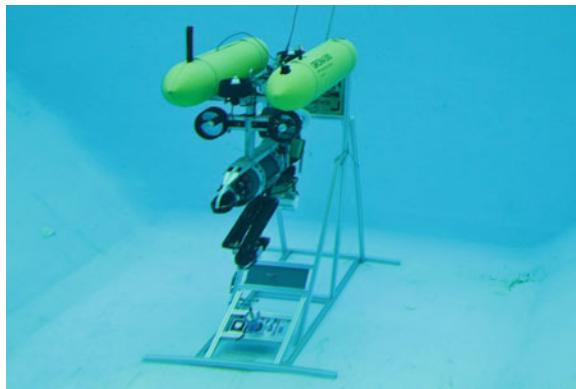


**Fig. 15** Flow chart of the intervention (open and close a valve and plug and unplug a connector)

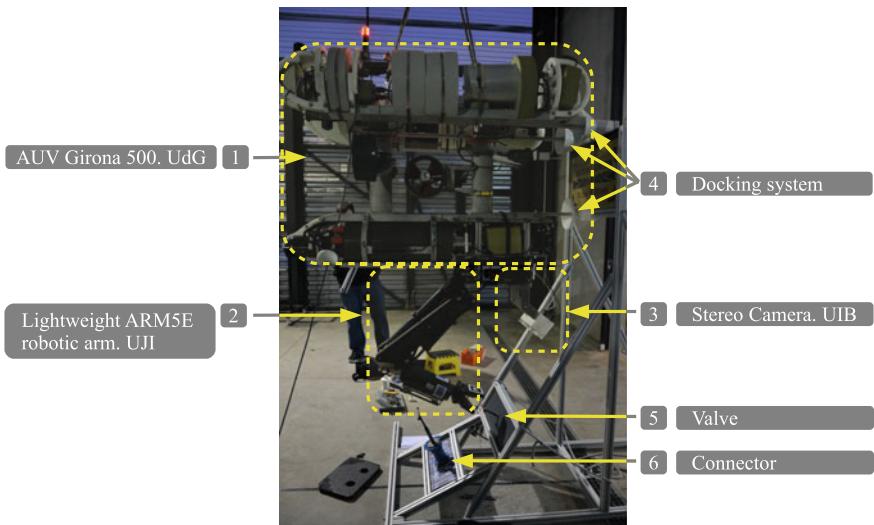


**Fig. 16** Designed waypoints for the intervention on the panel (open and close a valve and plug and unplug a connector)

Recently, and after two years of work (TRITON is a three year project), this envisioned concept concerning intervention on a panel become in a real system performing a successful intervention in pool conditions (see Fig. 17). The whole real system, once the mechatronics integration is complete, includes the Girona500 AUV with the docking devices assembled; the hand-arm system (the Light-Weight ARMSE), and different sensors; and the panel mockup (see Fig. 18). In this case, the intervention mission begins after the vehicle is rigidly attached to the panel after an autonomous docking [31]. In this case, the manipulation experiment takes place in a similar environment to the one described above, but this time including a more challenging scenario, taking into account the visibility issues. The details of this intervention are out of the scope of this chapter.



**Fig. 17** I-AUV used in TRITON, performing a panel intervention at UdG pool



**Fig. 18** TRITON hardware system with its components attached to an underwater panel mockup

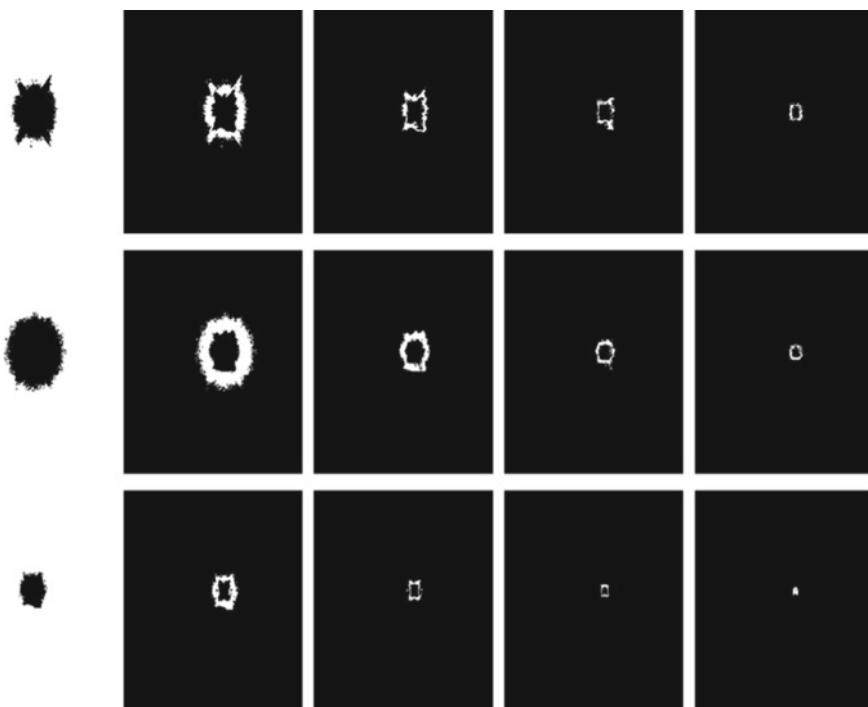
## 6.2 Ongoing Research on Autonomous Manipulation with Visibility Constraints

In the subsea context, the quality of the images captured by the camera mounted on the autonomous robots, can be strongly affected by the degree of the water turbidity. In unfavorable circumstances, the distance at which this device is usable (i.e. the range of visibility) is the required parameter in order to know how to make a proper use of it. On the other hand, when the image captured by the camera does not contain objects near the robot, it is not possible to determine whether this absence on the

image is due to the fact that there are really no objects near the vehicle or, contrarily, that water turbidity prevents their vision.

To have a metric to determine the maximum distance at which the camera is effective at each instant, a calibration experiment has been developed. Two high intensity LEDs (one red and one white), placed at a fixed distance from the camera, have been used. To reach this fixed distance the diodes can be placed in the submarine's robotic arm and then it can be moved properly until the LEDs reach the calibration location. On the other hand, a calibration image that is positioned at a distance of 1 m from the camera and is lightened by the built-in autonomous robot focus has been made.

To muddy the water, a special dye for decorative paintings has been used: a powder containing particles of different sizes. Thus, the water in the container in which the experiment has been developed, progressively blurred without having absolute measurements of turbidity. For each concentration of dye, in the absence of ambient light, the vehicle's built-in lights have been activated to illuminate the test image, and then, a screenshot of the captured image has been taken. After that, now with the lights turned off, both the red and white LEDs have been independently activated, taking screenshot of each of them.



**Fig. 19** Underwater visibility experimental results on increasing water turbidity. The three LED halos obtained for different turbidities have been binarized with different thresholds

The different images are the reference for the calibration of the degree of visibility of the focus-camera set for this particular conditions of turbidity. After that, LED halos are binarized for increasing water turbidity with different thresholds (see Fig. 19). Thus, the aspect of each of the LEDs makes it possible to determine the degree of visibility at 1 m of distance, and this can be used as a starting point for an estimation of the maximum distance that will have some degree of visibility.

## 7 Conclusions and Further Work

The field of underwater manipulation for intervention missions is an active research topic that still has many challenges to overcome. The most important research projects in this field, that have been able to demonstrate some results in sea conditions, are still far from what would be desirable for a fully autonomous underwater vehicle for intervention.

Nevertheless, the expertise and know-how developed in the context of our research group in the last years, in projects like RAUVI (09-11), TRIDENT (10-13), or TRITON (currently active), has resulted in a general system architecture that allows an underwater vehicle to perform intervention missions in different real scenarios with a high degree of autonomy. The results obtained in TRITON, and in particular, in the GRASPER subproject, in the field of autonomous underwater manipulation, represent the cutting edge of research in this area.

The use of UWSim as a 3D simulation tool for benchmarking and Human Robot Interaction (HRI) has also been presented. The simulator has demonstrated to be a useful tool in our roadmap, a methodology developed for experimental validation, where we first perform benchmarking and Hardware in the Loop (HIL) simulations as a prior stage before moving to the real testbeds, independently of the specific underwater intervention problem to solve. UWSim is also used as a *Graphical User Interface (GUI)*, providing the necessary *Human Robot Interaction (HRI)* that is required to specify a task.

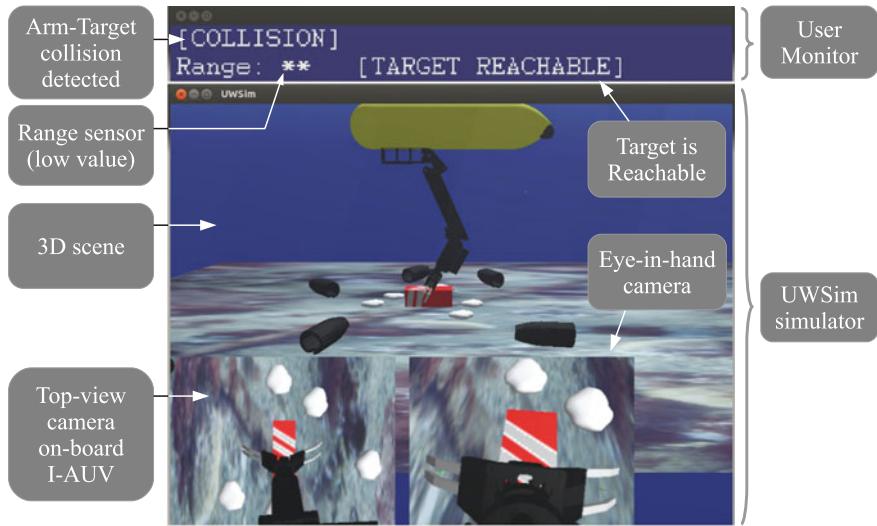
The *benchmarking* characteristics of UWSim allow the design of specific experiments on autonomous underwater interventions. More specifically, the simulator allows the integration, in a unique platform, of the data acquired from the sensors in a real submarine intervention, and define a dataset, in order to allow further experiments to work on the same scenario, permitting a better understanding of the results provided by previous experiments.

The usefulness of UWSim has been recently proven, as it is currently used in different ongoing projects funded by European Commission (MORPH and PANDORA). Moreover, it is available to the scientific community as live open source project,<sup>5</sup> and is also included as a module<sup>6</sup> within the ROS platform.

---

<sup>5</sup> Available on-line: <http://www.irs.ull.es/uwsim>.

<sup>6</sup> Available on-line: <http://wiki.ros.org/uwsim>.



**Fig. 20** User Interface connected to UWSim simulator providing feedback during the “learning by demonstration” stage in the “search and recovery” of a black-box mockup, the first scenario proposed in TRITON project (refer to [42] for additional details)

The underwater operation results on a permanent observatory panel in water tank conditions has also been presented, as a prior step to the next experiments that will take place in real sea conditions. The experiment consisted on opening and closing a valve and plugging and unplugging a connector. To perform the operation, the proposed general system architecture, that allows an underwater vehicle to perform intervention missions in different real scenarios, with a high degree of autonomy, has been used.

As future lines, it is worth mentioning that cooperation research actions with University of Coimbra (Portugal) are now open to explore other paradigms for improvements in manipulation like those based on “learning by demonstration” [42]. Experimental validation is being carried on UWSim with the aid of complementary modules to allow the user interaction for the learning process (see Fig. 20). It is expected that we incorporate those learning capabilities to the proposed system architecture, to be used in future interventions.

**Acknowledgments** This research was partly supported by Spanish Ministry of Research and Innovation DPI2011-27977-C03 (TRITON Project) and by Foundation Caixa Castelló-Bancaixa and Universitat Jaume I grant PI 1B2011-17.

## References

1. Ahmadzadeh S, Kormushev P, Caldwell D (2013) Autonomous robotic valve turning: a hierarchical learning approach. In: 2013 IEEE international conference on robotics and automation (ICRA), pp 4629–4634. doi:[10.1109/ICRA.2013.6631235](https://doi.org/10.1109/ICRA.2013.6631235)
2. Antonelli G (2014) Underwater robots. Springer tracts in advanced robotics, vol 96. Springer, Heidelberg
3. Bekey G, Liu H, Tomovic R, Karplus W (1993) Knowledge-based control of grasping in robot hands using heuristics from human motor skills. *IEEE Trans Robot Autom* 9(6):709–722. doi:[10.1109/70.265915](https://doi.org/10.1109/70.265915)
4. Bicchi A, Kumar V (2000) Robotic grasping and contact: a review. In: IEEE international conference on robotics and automation, ICRA'00, vol 1, pp 348–353. doi:[10.1109/ROBOT.2000.844081](https://doi.org/10.1109/ROBOT.2000.844081)
5. Borst C, Fischer M, Haidacher S, Liu H, Hirzinger G (2003) DLR hand II: experiments and experience with an anthropomorphic hand. In: IEEE international conference on robotics and automation, ICRA'03, vol 1, pp 702–707. doi:[10.1109/ROBOT.2003.1241676](https://doi.org/10.1109/ROBOT.2003.1241676)
6. Carrera A, Ahmadzadeh SR, Ajoudani A, Kormushev P, Carreras M, Caldwell DG (2012) Towards autonomous robotic valve turning. *J Cybern Inf Technol (CIT)* 12(3):17–26
7. Casalino G, Zereik E, Simetti E, Torelli S, Sperinde A, Turetta A (2012) Agility for underwater floating manipulation: task & subsystem priority based control strategy. In: 2012 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 1772–1779. doi:[10.1109/IROS.2012.6386127](https://doi.org/10.1109/IROS.2012.6386127)
8. Choi S, Takashige GY, Yuh J (1994) Experimental study on an underwater robotic vehicle: ODIN. In: Proceedings of 1994 symposium on autonomous underwater vehicle technology, AUV'94, pp 79–84. doi:[10.1109/AUV.1994.518610](https://doi.org/10.1109/AUV.1994.518610)
9. Cipolla R, Hollinghurst N (1997) Visually guided grasping in unstructured environments. *Robot Auton Syst* 19(3–4):337–346. doi:[10.1016/S0921-8890\(96\)00060-7](https://doi.org/10.1016/S0921-8890(96)00060-7)
10. Coelho J, Piater J, Grupen R (2001) Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. *Robot Auton Syst* 37(2–3):195–218. doi:[10.1016/S0921-8890\(01\)00158-0](https://doi.org/10.1016/S0921-8890(01)00158-0)
11. Dillmann R (2004) KA 1.10 Benchmarks for Robotics Research. Technical report
12. Ekwall S, Krägic D (2004) Interactive grasp learning based on human demonstration. In: 2004 IEEE international conference on robotics and automation, ICRA'04, vol 4, pp 3519–3524. doi:[10.1109/ROBOT.2004.1308798](https://doi.org/10.1109/ROBOT.2004.1308798)
13. Evans J, Keller K, Smith J, Marty P, Rigaud O (2001) Docking techniques and evaluation trials of the SWIMMER AUV: an autonomous deployment. In: AUV for work-class ROVs. OCEANS, 2001, pp 520–528
14. Evans J, Redmond P, Plakas C, Hamilton K, Lane D (2003) Autonomous docking for Intervention-AUVs using sonar and video-based real-time 3D pose estimation. In: OCEANS, vol 4, pp 2201–2210. doi:[10.1109/OCEANS.2003.178243](https://doi.org/10.1109/OCEANS.2003.178243)
15. Fernández J, Prats M, Sanz P, García J, Marín R, Robinson M, Ribas D, Ridao P (2013) Grasping for the seabed: developing a new underwater robot arm for shallow-water intervention. *IEEE Robot Autom Mag* 4(20):121–130. doi:[10.1109/MRA.2013.2248307](https://doi.org/10.1109/MRA.2013.2248307)
16. FP7-MORPH: Marine Robotic System of Self-Organizing, Logically Linked Physical Nodes (MORPH). <http://morph-project.eu/>
17. FP7-PANDORA: Persistent Autonomy through learnNing, aDaptation, Observation and Re-plAnning (PANDORA). <http://persistentautonomy.com/>
18. Gilmour B, Niccum G, O'Donnell T (2012) Field resident AUV systems: Chevron's long-term goal for AUV development. In: 2012 IEEE/OES autonomous underwater vehicles (AUV), pp 1–5. doi:[10.1109/AUV.2012.6380718](https://doi.org/10.1109/AUV.2012.6380718)
19. Hauck A, Ruttinger J, Sorg M, Farber G (1999) Visual determination of 3D grasping points on unknown objects with a binocular camera system. In: Proceedings of 1999 IEEE/RSJ international conference on intelligent robots and systems, IROS'99, vol 1, pp 272–278. doi:[10.1109/IROS.1999.813016](https://doi.org/10.1109/IROS.1999.813016)

20. Kamon I, Flash T, Edelman S (1998) Learning visually guided grasping: a test case in sensorimotor learning. *IEEE Trans Syst, Man Cybern Part A* 28(3):266–276
21. Lane D, Davies J, Casalino G, Bartolini G, Cannata G, Veruggio G, Canals M, Smith C, O'Brien D, Pickett M, Robinson G, Jones D, Scott E, Ferrara A, Angelletti D, Coccoli M, Bono R, Virgili P, Pallas R, Gracia E (1997) AMADEUS: advanced manipulation for deep underwater sampling. *IEEE Robot Autom Mag* 4(4):34–45. doi:[10.1109/100.637804](https://doi.org/10.1109/100.637804)
22. Liu H, Iberall T, Bekey G (1989) The multi-dimensional quality of task requirements for dexterous robot hand control. In: IEEE international conference on robotics and automation (ICRA'89), pp 452–457
23. Malis E (2004) Improving vision-based control using efficient second-order minimization techniques. In: 2004 IEEE international conference on robotics and automation, ICRA'04, vol 2, pp 1843–1848. doi:[10.1109/ROBOT.2004.1308092](https://doi.org/10.1109/ROBOT.2004.1308092)
24. Mallios A, Rida P, Carreras M, Hernandez E (2011) Navigating and mapping with the SPARUS AUV in a natural and unstructured underwater environment. In: OCEANS 2011, Waikoloa, Kona, Hawaii, Kona, pp 1–7
25. Marani G, Choi SK, Yuh J (2009) Underwater autonomous manipulation for intervention missions AUVs. *Ocean Eng* 36(1):15–23. doi:[10.1016/j.oceaneng.2008.08.007](https://doi.org/10.1016/j.oceaneng.2008.08.007)
26. Marani G, Yuh J (2014) Introduction to autonomous manipulation—case study with an underwater robot, SAUVIM. Springer tracts in advanced robotics, vol 102. Springer, Berlin
27. Miller AT, Knoop S, Christensen HI, Allen PK (2003) Automatic grasp planning using shape primitives. In: Proceedings of the IEEE international conference on robotics and automation (ICRA'03), Taipei, Taiwan, pp 1824–1829
28. Morales A, Asfour T, Azad P, Knoop S, Dillmann R (2006) Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In: IEEE/RSJ international conference on intelligent robots and systems, Beijing, China, pp 5663–5668
29. Morales A, Chinellato E, Fagg A, del Pobil A (2004) Experimental prediction of the performance of grasps tasks from visual features. *Int J Humanoid Robot* 10(1):671–691
30. Morales A, Recatalá G, Sanz P, del Pobil A (2001) Heuristic vision-based computation of planar antipodal grasps on unknown objects. In: IEEE international conference on robotics and automation (ICRA), vol 1, pp 583–588. doi:[10.1109/ROBOT.2001.932613](https://doi.org/10.1109/ROBOT.2001.932613)
31. Palomeras N, Ribas D, Vallicrosa G, Rida P, Carreras M (2014) Autonomous I-AUV docking for fixed-base manipulation. In: 19th world congress of the international federation of automatic control (IFAC), pp 12160–12165. doi:[10.3182/20140824-6-ZA-1003.01878](https://doi.org/10.3182/20140824-6-ZA-1003.01878)
32. Peñalver A, Pérez J, Fernández JJ, Sales J, Sanz PJ, García JC, Fornas D, Marín R (2014) Autonomous intervention on an underwater panel mockup by using visually-guided manipulation techniques. In: 19th world congress of the international federation of automatic control (IFAC), pp 5151–5156. doi:[10.3182/20140824-6-ZA-1003.02545](https://doi.org/10.3182/20140824-6-ZA-1003.02545)
33. Prats M, Garcia J, Wirth S, Ribas D, Sanz P, Rida P, Gracias N, Oliver G (2012) Multipurpose autonomous underwater intervention: a systems integration perspective. In: 2012 20th mediterranean conference on control automation (MED), pp 1379–1384. doi:[10.1109/MED.2012.6265831](https://doi.org/10.1109/MED.2012.6265831)
34. Prats M, Pérez J, Fernández J, Sanz P (2012) An open source tool for simulation and supervision of underwater intervention missions. In: 2012 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 2577–2582. doi:[10.1109/IROS.2012.6385788](https://doi.org/10.1109/IROS.2012.6385788)
35. Prats M, del Pobil AP, Sanz PJ (2013) Robot physical interaction through the combination of vision, tactile and force feedback. Applications to assistive robotics, Springer tracts in advanced robotics, vol 84. Springer, Berlin
36. Prats M, Ribas D, Palomeras N, García J, Nannen V, Wirth S, Fernández J, Beltrán J, Campos R, Rida P, Sanz P, Oliver G, Carreras M, Gracias N, Marín R, Ortiz A (2012) Reconfigurable AUV for intervention missions: a case study on underwater object recovery. *Intell Serv Robot* 5(1):19–31. doi:[10.1007/s11370-011-0101-z](https://doi.org/10.1007/s11370-011-0101-z)
37. Pérez J, Sales J, Prats M, Martí JV, Fornas D, Marín R, Sanz PJ (2013) The underwater simulator UWSim: benchmarking capabilities on autonomous grasping. In: 11th international conference on informatics in control, automation and robotics (ICINCO)

38. Quigley M, Conley K, Gerkey BP, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) ROS: an open-source robot operating system. In: ICRA workshop on open source software
39. Ribas D, Palomeras N, Ridao P, Carreras M (2012) Girona 500 AUV, from survey to intervention. *IEEE/ASME Trans Mechatron (Focused Section on Marine Mechatronic Systems)* 17(1):46–53
40. Rigaud V, Coste-Maniere E, Aldon M, Probert P, Perrier M, Rives P, Simon D, Lang D, Kiener J, Casal A, Amar J, Dauchez P, Chantler M (1998) Union: underwater intelligent operation and navigation. *IEEE Robot Autom Mag* 5(1):25–35. doi:[10.1109/100.667323](https://doi.org/10.1109/100.667323)
41. Rusu R, Holzbach A, Diankov R, Bradski G, Beetz M (2009) Perception for mobile manipulation and grasping using active stereo. In: 9th IEEE-RAS international conference on humanoid robots, humanoids 2009, pp 632–638. doi:[10.1109/ICHR.2009.5379597](https://doi.org/10.1109/ICHR.2009.5379597)
42. Sales J, Santos L, Sanz PJ, Dias J, García JC (2013) Increasing the autonomy levels for underwater intervention missions by using learning and probabilistic techniques. In: First Iberian robotics conference (ROBOT 2013), Madrid, Spain
43. Sanz PJ, Marín R, Sales J, Oliver G, Ridao P (2012) Recent advances in underwater robotics for intervention missions. Soller harbor experiments, Low-cost books
44. Sanz PJ, Marín R, Sánchez JS (2005) Including efficient object recognition capabilities in online robots: from a statistical to a neural-network classifier. *IEEE Trans Syst, Man, Cybern Part C: Appl Rev* 35(1):87–96. doi:[10.1109/TSMCC.2004.840055](https://doi.org/10.1109/TSMCC.2004.840055)
45. Sanz PJ, Prats M, Ridao P, Ribas D, Oliver G, Ortí A (2010) Recent progress in the RAUVI project a reconfigurable autonomous underwater vehicle for intervention. In: 52th international symposium ELMAR-2010. Zadar, Croatia, pp 471–474
46. Sanz PJ, Requena A, Iñesta JM, del Pobil AP (2005) Grasping the not-so-obvious: vision-based object handling for industrial applications. *IEEE Robot Autom Mag* 12(3):44–52. doi:[10.1109/MRA.2005.1511868](https://doi.org/10.1109/MRA.2005.1511868)
47. Sanz PJ, Ridao P, Oliver G, Casalino G, Petillot Y, Silvestre C, Melchiorri C, Turetta A (2013) TRIDENT: an European project targeted to increase the autonomy levels for underwater intervention missions. In: OCEANS'13 MTS/IEEE conference, San Diego
48. Shimoga KB (1996) Robot grasp synthesis algorithms: a survey. *Int J Robot Res* 15(3):230–266. doi:[10.1177/027836499601500302](https://doi.org/10.1177/027836499601500302)
49. Simetti E, Casalino G, Torelli S, Sperinde A, Turetta A (2013) Experimental results on task priority and dynamic programming based approach to underwater floating manipulation. In: OCEANS—Bergen, 2013 MTS/IEEE, pp 1–7. doi:[10.1109/OCEANS-Bergen.6608016](https://doi.org/10.1109/OCEANS-Bergen.6608016)
50. Stansfield S (1991) Robotic grasping of unknown objects: a knowledge-based approach. *Int J Robot Res* 10(4):314–326. doi:[10.1177/027836499101000402](https://doi.org/10.1177/027836499101000402)
51. Wang H, Rock S, Lee M (1995) Experiments in automatic retrieval of underwater objects with an AUV. In: OCEANS'95. Proceedings of conference on MTS/IEEE. Challenges of our changing global environment, vol 1, pp 366–373. doi:[10.1109/OCEANS.1995.526796](https://doi.org/10.1109/OCEANS.1995.526796)

# **Erratum to: Motion and Operation Planning of Robotic Systems**

**Giuseppe Carbone and Fernando Gomez-Bravo**

## **Erratum to:**

**G. Carbone and F. Gomez-Barvo (eds.),**  
*Motion and Operation Planning of Robotic Systems,*  
**Mechanisms and Machine Science 29,**  
**DOI [10.1007/978-3-319-14705-5](https://doi.org/10.1007/978-3-319-14705-5)**

The spelling of the editor name **Fernando Gomez-Barvo** was incorrect. The correct name should read as follows **Fernando Gomez-Bravo**.

---

The online version of the original book can be found under  
DOI [10.1007/978-3-319-14705-5](https://doi.org/10.1007/978-3-319-14705-5)

---

G. Carbone (✉)  
University of Cassino, Cassino, Frosinone, Italy  
e-mail: carbone@unicas.it

F. Gomez-Bravo  
Engineering School, University of Huelva, La Rábida, Huelva, Spain  
e-mail: fernando.gomez@riesia.uhu.es