UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXII

Real-time management of kinematic and safety constraints in motion planning systems

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutor:
Chiar.mo Prof. Corrado Guarino Lo Bianco

Dottoranda: Marina Raineri

Anni 2016/2019

To my parents,
for being beside me
whatever happens.

Contents

List of Figures

List of Tables

Introduction

Nowadays, robotic systems are massively used in many industrial contexts because of their efficiency, robustness, and precision. Thanks to the collaboration between robots and humans, several industrial needs can be efficiently tackled. Indeed, robots can execute precise tasks requiring efforts which would be unfeasible for humans. Conversely, the use of autonomous robots in collaborative environments poses several concerns. For example, robots must promptly react to unpredictable events and/or handle feasibility issues caused by the assigned trajectories. An appropriate management of the human-robot interactions must be foreseen in order to avoid dangerous situations.

Trajectories must be planned by assuming an optimal criterion, which is chosen depending on the application requirements. Usually, in industrial plant, the total traveling time is minimized in order to improve the productivity performances. Generated trajectories must comply with the kinematic limits of mechatronic systems. For example, for which concerns autonomous vehicles, the longitudinal velocity or the norm of the lateral acceleration should be bounded. Moreover, one may desire obtaining smooth trajectories, in order to limit the systems solicitations; a smart planning technique, which satisfies constraints on jerk as well as on velocity and acceleration, should be adopted. The trajectory planner should also be able to face critical situations by modifying, for example, the assigned trajectories in order to accomplish a given task. For example, this is the action to be taken for the avoidance of kinematic singularities occurring in anthropomorphic manipulators.

This thesis will face two problems that must be tackled while working with au-

tonomous systems in an industrial environment. The first one deals with anthropomorphic manipulators and some critical situations which could occur when trajectories are planned in the operational space. The second topic is relative to laser guided autonomous vehicles. In details, it concerns the design of a smart velocity planner which can also assure the motion safety.

As early anticipated, in industrial plants, autonomous systems share their workspace with human operators. Such working conditions give rise to several issues concerning safety aspects. The safeness of coworkers and of the overall plant must be assured, despite they are antithetic with respect to productivity. The target pursued in this thesis is to assure both objectives are simultaneously satisfied, i.e. the plant productivity is maintained and improved through a smart trajectory planner which also guarantees safe working conditions.

Different trajectory planners could be adopted for the management of such problems. Theoretically, the trajectory can be computed offline, before the motion begins, by assuming the knowledge of the whole system, the robot characteristics and the environment in which it moves. In actual warehouses such assumption is not realistic, since the environment is only partially structured and the robot must instantly react to unforeseen situations, for example a human worker invading the vehicle route. Due to this necessity, planners must evaluate trajectories in real time, while the robot is already moving.

In this work, trajectories are always planned according to the path-velocity decomposition concept, i.e., at first the geometric path is defined and, then, a time-law is assigned to such path. The combination between path and time-law must be feasible and the resulting trajectory must satisfy the assigned constraints. Additionally, evaluation times of the planning algorithms must be compatible with sample rate of the control systems.

This thesis is divided in two parts:

I. The first one presents a trajectory planner specifically conceived for non-redundant anthropomorphic manipulators dealing with trajectories planned in the operational space. As known, such trajectories could present kinematic singularities, which can be avoided by introducing small errors in the Cartesian path or in

the time-law. Another approach, which is the one adopted in the system proposed, suggests avoiding singular configurations through minor modifications of the end-effector orientation. This method is valid when the robot is equipped with a spherical wrist and it can be used for industrial applications in which the Cartesian path and the time-law are mandatory but, conversely, small orientation changes of the tool-frame can be admitted since they do not worsen the final result. The first work on the singularity avoidance system, proposed in this thesis, has appeared in [1]. Then the whole strategy has been modified and presented in [2]; finally some theoretical results and a tuning procedure are proposed in [3]. The idea of introducing minor changes to the end-effector orientation, in order to obtain a degree of freedom to be exploited for the singularity avoidance, is already present in the literature [4–6]; however such works are conceived for manually operated manipulators. Conversely the proposed method is able to operate at standard working speeds, hence it can be adopted in real industrial contexts, and the singular configuration can even be crossed. More details on the existent literature and the novelties introduced can be found in section 1.1.

II. The second part is relative to a velocity planner conceived for laser guided vehicles. In such industrial context, the velocity planning problem must be solved online, since the path is composed by many path-segments whose sequence is assigned on-the-fly, while the vehicle is moving. In warehouses, autonomous vehicles are equipped with certified laser sensors which are used to guarantee the safeness of the human workers and of the whole plant. The information provided by such laser sensors is used to obtain an analytical representation of the safety constraint, which then becomes part of the optimization problem which is solved for the generation of the optimal trajectory. The velocity planner proposed is also able to limit the jerk signal in order to reduce vehicle stresses. In the literature, when the jerk is limited, constant bounds on velocities and accelerations are considered, as in [7–9]. Variable velocity bounds strongly complicate the jerk-limited optimization problem, this part of the thesis proposes a velocity planner which overcomes these limitations; a first version has

appeared in [10]. Then, experimental results are presented in [11] while [12] draws some considerations on the optimal path to be followed by vehicles. Finally, the proposed algorithm has been improved in [13] in order to obtain a solution similar to the one found by nonlinear solvers. More details on the existent literature and the novelties introduced can be found in section 4.1.

Part I

Real-time management of wrist singularities in non-redundant anthropomorphic manipulators

Introduction to Part I

The automatic management of kinematic singularities, which are typical for trajectories planned in the operational space, is arousing a renewed interest among the scientific community because the most recent strategies are suited for real-time implementations. The proposed approach is conceived for non-redundant anthropomorphic manipulator equipped with a spherical wrist and it allows handling wrist singularities, which may appear everywhere in the workspace.

In non-redundant systems, there are not degrees of freedom to be exploited for the singularity avoidance, so that kinematic singularities must be handled by introducing small trajectory errors. For example, it could be managed by admitting small tracking errors in the assigned Cartesian path or, alternatively, by reducing the tool-frame velocity, thus altering the assigned time-law. Still, several applications do not allow deviations from the nominal path or the assigned time-law. In such cases singular points can be managed by slightly modifying the tool-frame orientation. In fact, in many industrial processes small orientation changes w.r.t. the nominal trajectory have a minimal impact on the quality of the final product. The singularity avoidance system described in this thesis is based on such concept. The degree of freedom, acquired through the change of the tool-frame orientation, is used to avoid the singular condition by preserving, simultaneously, both the assigned path and the time-law. Additionally, the proposed methodology allows one specifying the maximum orientation displacement: other works proposed in the literature minimize the amplitude of the introduced errors, but do not explicitly bound them. A further property of the singularity avoidance system here proposed is represented by its capability to man-

age trajectories which are planned on-the-fly. In fact, modern industrial applications impose developing system which can react to unforeseen events, so that trajectories are often planned on the basis of data acquired by perceptual sensors. Finally, the approach is effective also for manipulators moving at standard operative speeds and it explicitly handles given limits on joint velocities and accelerations. Conversely, existent methods do not constrain velocities and accelerations between specific bounds or consider very slow trajectories generated through a teach pendant device.

The first part of the thesis is organized as follows. Chapter 1 illustrates the methods proposed in the literature for the management of the kinematic singularities. In addition, an earlier version of the singularity avoidance system is summarized in order to describe its basic principles. At the beginning of chapter 2, the singularity avoidance problem is theoretically analyzed and a necessary condition to be fulfilled for the existence of a solution is devised. Subsequently, the novelties characterizing the new singularity avoidance system are summarized, and each component of the latter is analyzed. In details, a procedural method is proposed for the tuning of the algorithm, so as to make it deterministic and to increase the success rate. Finally, chapter 3 proposes the simulated results and, additionally, the results experimentally obtained through an anthropomorphic industrial manipulator.

Chapter 1

State of the Art

Kinematic singularities must be faced when trajectories are planned in the operational space of a non-redundant anthropomorphic manipulator. Moreover such critical configurations must be managed in real time while simultaneously limiting the joint velocities and accelerations. In this context, different methods have been proposed in the literature and they are illustrated in sections 1.1. Then, section 1.2 briefly recalls the first version proposed of the singularity avoidance system, subsequently improved in chapter 2.

1.1 Related Works

One of the major problems that must be tackled when trajectories are planned in the operational space is associated to the management of the so-called kinematic singularities, i.e., configurations in which bounded Cartesian speeds lead to endless joint speeds and bounded joint torques lead to unbounded end-effector forces. Anthropomorphic manipulators admit three types of singularities: shoulder singularities are only significant for hanging robots and appear when the wrist crosses the first axis; elbow singularities are scarcely relevant since they occur at the border of the workspace, i.e., in areas which are seldom used; wrist singularities appear when the 4th and the 6th joint axes are aligned. This work focuses on the management of wrist

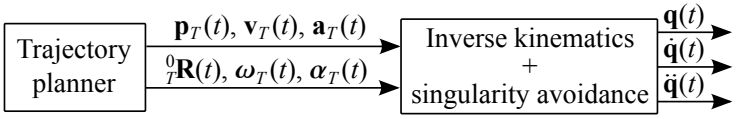


Figure 1.1: Typical planner scheme for trajectories in the operational space.

singularities since they may occur in any point of the workspace and, consequently, they are relevant in many practical applications.

Kinematic singularities can be handled in several ways. The most commonly used approaches react to singularities by marginally modifying the assigned paths and time-laws. Planners for the operational space are typically based on the functional scheme in figure 1.1: the Cartesian trajectory planner is immediately followed by an inverse kinematics block, which is also in charge for the management of possible kinematic singularities. All works proposed during 80th and 90th were practically based on such conceptual scheme.

Many of the techniques in the literature derive from the original approach proposed in [14] for the solution of the inverse kinematics of redundant manipulators: the generated joint reference signals guarantee that the trajectory in the operational space is exactly executed, while available degrees of freedom are used to accomplish secondary tasks. The strategy was later revised and better formalized in [15]. In that paper, the inverse kinematics was solved through a Moore-Penrose inverse of the Jacobian matrix used jointly with a projection operator in the Jacobian null-space. For the first time, it was explicitly remarked that such technique is potentially suited for the management of kinematic singularities. The methodology was later extended in [16, 17] in order to manage constraints through a task priority approach. In details, the task priority strategy was revised in [18] by explicitly considering its use for the management of kinematic singularities. Practically, system bounds were considered as low-priority tasks to be accomplished, while the generation of the Cartesian trajectory had the highest priority. Such approach induces a nice behavior: cyclic trajectories in the operational space always lead to cyclic trajectories in the joint space. In the same years other alternative methods were proposed for the solution of the

inverse kinematic problem. Some of them were based on a damped least-square approach [19, 20] while others were based on closed loop schemes and were able to manage constrained problems by means of a proper augmentation of the task space dimension [21, 22].

Researches based on the above methods prosecuted up to nowadays as proved by several works in the literature [23–26]. It is important to remark that, despite many algorithms are conceived for redundant manipulators, they may also be adopted for non-redundant ones. To this purpose, it is sufficient to introduce some degrees of redundancy by admitting small deviations from the assigned trajectory. All mentioned techniques, when extended to non-redundant manipulators, show some common characteristics:

1. singularities are managed by introducing small position and orientation errors;
2. the amplitude of such errors is kept small through proper tunings, but explicit bounds are not imposed;
3. velocities and accelerations are generically limited, but they are not forced within given bounds.

Such characteristics may or may not be appropriate depending on the application at hand. For applications in which the precision requirement can be relaxed, critical configurations are managed by admitting minor path and orientation deviations [23–27]. Conversely, there exist applications which do not allow deviations from the assigned Cartesian path, so that the problem must be tackled through alternative methods. If the path does not exactly cross singular configurations and the time law is not assigned, control methods based on predictive controllers can be used for the generation of efficient trajectories [28–32]. Alternatively, if the time law is assigned, the trajectory can be slowed down so as to preserve both path and orientation of the end-effector [33–39].

The situation becomes more critical if the assigned path crosses a kinematic singularity and the time-law is given and unmodifiable. In that case, singular points can be managed by slightly modifying the nominal orientation of the end-effector.

In many industrial processes, indeed, small orientation changes have a minimal impact on the product quality, while speed and/or path changes may worsen the final result. Among them it is worth mentioning applications like painting [40,41], gluing, or welding [42], for which, conversely, minor changes of the end-effector orientation have a minimal impact on the quality of the final result [27, 43, 44]. The acquired degrees of freedom can be used to avoid the singular configurations by preserving, simultaneously, both the assigned Cartesian path and the time-law.

The mentioned problem may be alternatively handled by means of offline planners, but nowadays applications require trajectories generated on-the-fly on the basis of data acquired by perceptual sensors. Some real-time planners, able to preserve the Cartesian path, have been already proposed. They are typically conceived for the generation of slow motions like the ones deriving, for example, from the use of teaching devices handled by human operators [4–6]. Conversely, in an industrial environment a method which is effective also at standard operative speeds should be preferred.

1.2 Singularity Avoidance System (SAS): state of the art

A first method able to guarantee all the properties cited in the previous section has been proposed in [45], small changes of the end-effector orientation allow dealing with singularities even when the motion is executed at normal operative speeds. In fact, the Singularity Avoidance Systems (SAS) represents a step ahead in the field of the management of trajectories passing close to wrist singularities.

The strategy proposed in [45] for the management of the kinematic singularities was based on the assumption that a supervisory planner would provide real-time trajectories for the joint actuators of an n-link manipulator. Such trajectories, being generated in real time, might be unfeasible, i.e., they might not satisfy the following kinematic constraint equations

$$\dot{\mathbf{q}}^{-} \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}^{+}, \tag{1.1}$$

$$\ddot{\mathbf{q}}^{-} \leq \ddot{\mathbf{q}} \leq \ddot{\mathbf{q}}^{+}, \tag{1.2}$$

where $\dot{\mathbf{q}} \in \mathbb{R}^n$ and $\ddot{\mathbf{q}} \in \mathbb{R}^n$ indicate, respectively, the joint velocities and accelerations,

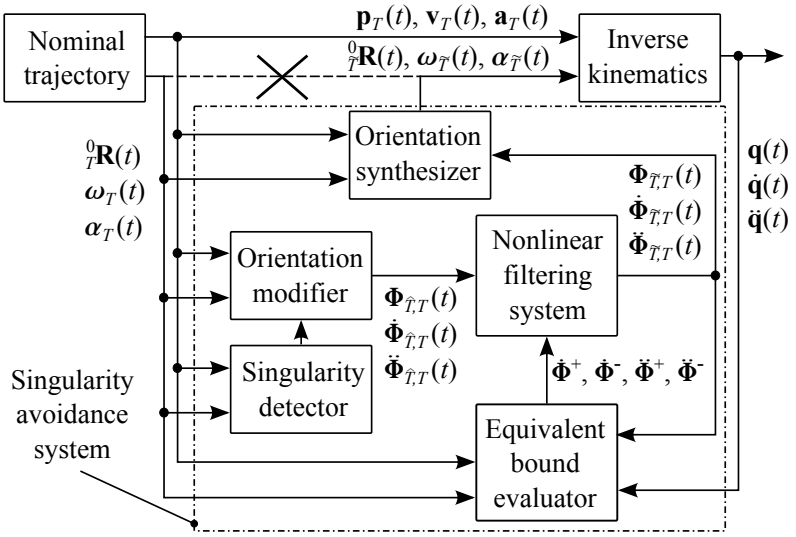


Figure 1.2: A schematic representation of an earlier version of the Singularity Avoidance System.

while $\dot{\mathbf{q}}^-, \ddot{\mathbf{q}}^- \in (\mathbb{R}^-)^n$ and $\dot{\mathbf{q}}^+, \ddot{\mathbf{q}}^+ \in (\mathbb{R}^+)^n$ represent the given bounds.

Unfeasible trajectories cannot be correctly followed by feedback controllers, so that an algorithm was proposed for their real-time management. Let us briefly recall its behavior with the aid of figure 1.2. The nominal trajectory of the tool-frame T associated to the end-effector is planned on-the-fly, for example, by a supervisory task. Its Cartesian components – $\mathbf{p}_T$, $\mathbf{v}_T$, and $\mathbf{a}_T$, i.e., its position, its linear velocity, and its linear acceleration – are directly sent to the inverse kinematics block which evaluates the joints reference signals. Conversely, its orientation components – ${}^0_T\mathbf{R}$, $\boldsymbol{\omega}_T$, and $\boldsymbol{\alpha}_T$, i.e., its orientation, its angular velocity, and its angular acceleration – are sent to the SAS which introduces slight orientation changes in order to fulfill the constraints. Practically, the Cartesian trajectory is preserved by the SAS, while any singular condition is avoided through minimal orientation changes.

The SAS structure is shown in figure 1.2. It is composed by several logic blocks. The singularity detector is used to reveal if the trajectory is approaching a singular configuration and, in that case, it activates the orientation modifier which proposes

a new candidate direction for the tool frame T. The new orientation, which should guarantee the singularity avoidance, is expressed in terms of displacement between the nominal T and a new frame $\hat{T}$, and it is represented by means of a Roll-Pitch-Yaw (RPY) vector $\mathbf{\Phi}_{\hat{T},T} \in \mathbb{R}^3$. Unfortunately, the trajectory modification occurs close to a critical region and could potentially worsen an already degenerated situation: to avoid problems, the dynamics of $\mathbf{\Phi}_{\hat{T},T}$, represented by its first and second time derivatives, must be bounded within proper limits, that are estimated by the equivalent bound evaluator. The constraint fulfillment is achieved by means of a nonlinear filtering system which generates, starting from $\mathbf{\Phi}_{\hat{T},T}$, a new feasible orientation displacement $\mathbf{\Phi}_{\tilde{T},T} := [\alpha\ \beta\ \gamma]^T \in \mathbb{R}^3$. The orientation synthesizer combines $\mathbf{\Phi}_{\tilde{T},T}$ with the nominal trajectory, in order to generate the final feasible one. Details on the whole algorithm can be found in [45], while in the following the discussion will focus on the equivalent bound evaluator in order to better understand its importance and the relevance of the modifications which will be described in chapter 2.

As anticipated, the feasibility is preserved by bounding $\dot{\mathbf{\Phi}}_{\tilde{T},T}$ and $\ddot{\mathbf{\Phi}}_{\tilde{T},T}$ within proper limits, i.e., by imposing the fulfillment of the following constraints

$$\dot{\mathbf{\Phi}}^- \leq \dot{\mathbf{\Phi}}_{\tilde{T},T} \leq \dot{\mathbf{\Phi}}^+, \tag{1.3}$$

$$\ddot{\mathbf{\Phi}}^- \leq \ddot{\mathbf{\Phi}}_{\tilde{T},T} \leq \ddot{\mathbf{\Phi}}^+, \tag{1.4}$$

where

$$\dot{\mathbf{\Phi}}^- := [\dot{\alpha}^-\ \dot{\beta}^-\ \dot{\gamma}^-]^T,\ \ddot{\mathbf{\Phi}}^- := [\ddot{\alpha}^-\ \ddot{\beta}^-\ \ddot{\gamma}^-]^T,$$
$$\dot{\mathbf{\Phi}}^+ := [\dot{\alpha}^+\ \dot{\beta}^+\ \dot{\gamma}^+]^T,\ \ddot{\mathbf{\Phi}}^+ := [\ddot{\alpha}^+\ \ddot{\beta}^+\ \ddot{\gamma}^+]^T$$

are a set of bounds which directly descend from (1.1) and (1.2): the final trajectory will be feasible if (1.3) and (1.4) are satisfied. In [45] it was shown that such bounds can be obtained through a two steps procedure. The first step is algebraic and allows rewriting (1.1) and (1.2) in the following form

$$\dot{\tilde{\mathbf{q}}}^- \leq \mathbf{A}(\mathbf{q})\dot{\mathbf{\Phi}}_{\tilde{T},T} \leq \dot{\tilde{\mathbf{q}}}^+, \tag{1.5}$$

$$\ddot{\tilde{\mathbf{q}}}^- \leq \mathbf{A}(\mathbf{q})\ddot{\mathbf{\Phi}}_{\tilde{T},T} \leq \ddot{\tilde{\mathbf{q}}}^+, \tag{1.6}$$

where $\mathbf{A}(\mathbf{q}) \in \mathbb{R}^{n\times 3}$ is a non-singular, known matrix, while $\dddot{\tilde{\mathbf{q}}}^{-}$, $\dddot{\tilde{\mathbf{q}}}^{+}$, $\ddddot{\tilde{\mathbf{q}}}^{-}$, and $\ddddot{\tilde{\mathbf{q}}}^{+}$ are equivalent limits derived from $\dot{\mathbf{q}}^{-}$, $\dot{\mathbf{q}}^{+}$, $\ddot{\mathbf{q}}^{-}$, and $\ddot{\mathbf{q}}^{+}$. The second step would be straightforward if $\mathbf{A}$ were a square matrix but, typically, $n > 3$, so that the conversion admits several degrees of freedom. For this reason, in [45] the equivalent bounds were evaluated by solving, at each sample time, an optimization problem whose cost index was given by the amplitude of the feasible intervals in (1.3) and (1.4): evidently, it is normally better to keep them as wide as possible. More in details, an optimal solution of the following optimization problem is searched for

$$\max_{\substack{\dddot{\mathbf{\Phi}}^{+},\ddddot{\mathbf{\Phi}}^{+} \in (\mathbb{R}^{+})^{3} \\ \dddot{\mathbf{\Phi}}^{-},\ddddot{\mathbf{\Phi}}^{-} \in (\mathbb{R}^{-})^{3}}} \ \min_{i=1,\ldots,6} (\Gamma_i^{+} - \Gamma_i^{-}) \tag{1.7}$$

subject to

$$\dddot{\tilde{\mathbf{q}}}^{-} \leq \mathbf{A}\dddot{\mathbf{\Phi}}_{\tilde{T},T} \leq \dddot{\tilde{\mathbf{q}}}^{+} \quad \forall \dddot{\mathbf{\Phi}}_{\tilde{T},T} \in [\dddot{\mathbf{\Phi}}^{-}, \dddot{\mathbf{\Phi}}^{+}], \tag{1.8}$$

$$\ddddot{\tilde{\mathbf{q}}}^{-} \leq \mathbf{A}\ddddot{\mathbf{\Phi}}_{\tilde{T},T} \leq \ddddot{\tilde{\mathbf{q}}}^{+} \quad \forall \ddddot{\mathbf{\Phi}}_{\tilde{T},T} \in [\ddddot{\mathbf{\Phi}}^{-}, \ddddot{\mathbf{\Phi}}^{+}], \tag{1.9}$$

where, Γ_i^{+} and Γ_i^{-} are, respectively, the components of vectors

$$\mathbf{\Gamma}^{+} := [k\,\dddot{\alpha}^{+}\ k\,\dddot{\beta}^{+}\ k\,\dddot{\gamma}^{+}\ \ddddot{\alpha}^{+}\ \ddddot{\beta}^{+}\ \ddddot{\gamma}^{+}]^{T} \tag{1.10}$$

$$\mathbf{\Gamma}^{-} := [k\,\dddot{\alpha}^{-}\ k\,\dddot{\beta}^{-}\ k\,\dddot{\gamma}^{-}\ \ddddot{\alpha}^{-}\ \ddddot{\beta}^{-}\ \ddddot{\gamma}^{-}]^{T} \tag{1.11}$$

which contain the desired bounds, while k is the weight factor used to properly scale the velocity bounds w.r.t. the acceleration ones. In [45] such scaling factor was posed equal to 10 in order to have acceleration limits 10 times larger than the velocity ones.

Problem (1.7)–(1.9) is semi-infinite but, since (1.8) and (1.9) are clearly linear, it can be converted into a finite one and it was solved by means of an efficient real-time algorithm appositely designed.

The SAS performance was tested in [45] through simulations and real experiments. In both cases, the tests consisted in the execution of a set of parallel trajectories approaching the singularity point from different directions (see also figure 1.3): in each test the scanning process was ended when feasibility was lost and the minimum distance d_i from the singularity of the last feasible trajectory was acquired. A

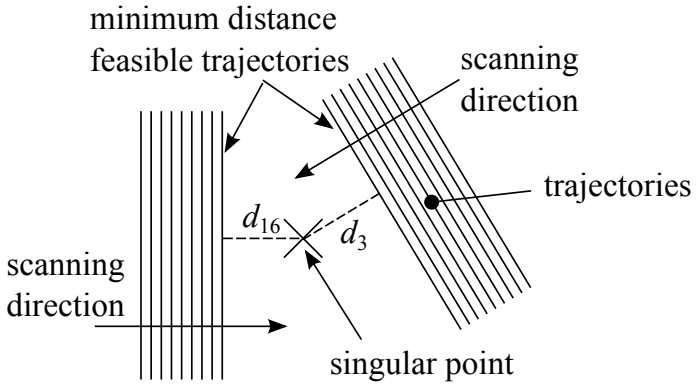


Figure 1.3: Two examples of the scanning procedure used to check the SAS performance.

total of 20 scanning directions and 4 Cartesian speeds were considered. The minimum distance points acquired in the tests are shown in figure 1.4 by means of dash-dotted magenta lines. The singular point is located in the center of each sub-figure. The smaller is d_i, $i = 1,2,\ldots,20$, the better the SAS performance. The solid-line area refers to the commercial controller. Improvements are evident, even if it is possible to notice that in some cases they are marginal.

In order to further improve the SAS performance, the planning strategy has been modified by introducing new degrees of freedom in the problem. In fact, (1.10) and (1.11) impose that $\ddot{\mathbf{\Phi}}^- = 10\,\dot{\mathbf{\Phi}}^-$ and $\ddot{\mathbf{\Phi}}^+ = 10\,\dot{\mathbf{\Phi}}^+$, i.e., acceleration bounds are always 10 times larger than velocity ones. Such heuristic choice is effective for many mechatronic systems, but other choices are possible and can be used to improve the SAS response. For example, by selecting $\mathbf{\Gamma}^+$ and $\mathbf{\Gamma}^-$ as follows

$$\mathbf{\Gamma}^+ := \left[k_1\dot{\alpha}^+\ k_2\dot{\beta}^+\ k_3\dot{\gamma}^+\ k_7\ddot{\alpha}^+\ k_8\ddot{\beta}^+\ k_9\ddot{\gamma}^+\right];$$
$$\mathbf{\Gamma}^- := \left[k_4\dot{\alpha}^-\ k_5\dot{\beta}^-\ k_6\dot{\gamma}^-\ k_{10}\ddot{\alpha}^-\ k_{11}\ddot{\beta}^-\ k_{12}\ddot{\gamma}^-\right],$$

where $k_i \in \mathscr{H}_i \subset \mathbb{R}^+, i = 1,2,\ldots,12$. Values of $\dot{\mathbf{\Phi}}^-$, $\dot{\mathbf{\Phi}}^+$, $\ddot{\mathbf{\Phi}}^-$, $\ddot{\mathbf{\Phi}}^+$ become each other independent, i.e., the number of possible combinations enlarges, so that a better performance might be achieved with a proper choice of $k_i \in \mathscr{H}_i$. $\mathscr{H}_i$ are closed intervals

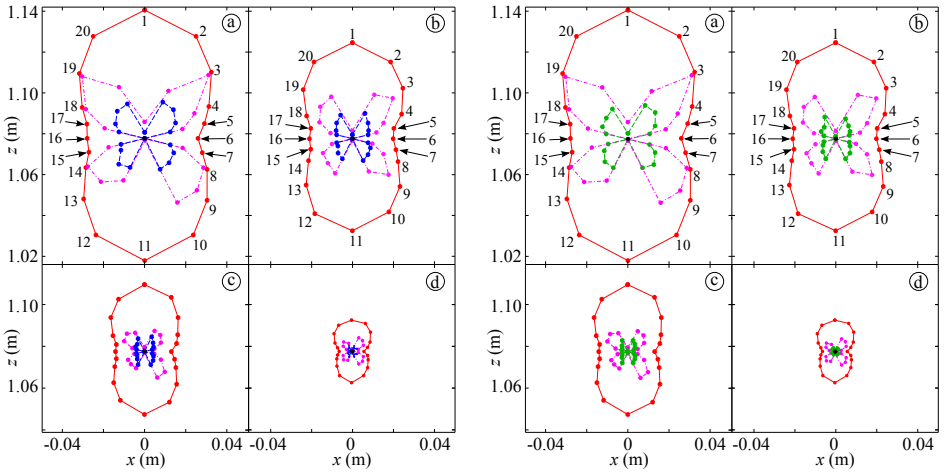


Figure 1.4: Distances from the singular point obtained through the commercial controller (solid red line) and the first SAS version (dash-dotted magenta line) compared to results obtained with the parameters optimization through RBF (dashed blue line) and DE (dashed green line). Results are reported at 4 different speeds, (a) 0.4 ms^{-1}, (b) 0.3 ms^{-1}, (c) 0.2 ms^{-1}, (d) 0.1 ms^{-1}

which contain the configuration used in the previous formulations of the problem, i.e., $k_1 = k_2 = \ldots = k_6 = 10$ and $k_7 = k_8 = \ldots = k_{12} = 1$. The new degrees of freedom have been exploited by selecting $\mathbf{k} := [k_1\ k_2\ \cdots k_{12}]^T$ through the solution of the following optimization problem

$$\min_{\mathbf{k}} \{J(\mathbf{k})\}\ . \tag{1.12}$$

Given $\mathbf{k}$, the following performance index is proposed in order to quantify the improvements

$$J(\mathbf{k}) := \max_{i=1,2,\ldots,20} \{d_i\}\ , \tag{1.13}$$

where distances d_i are measured for a Cartesian speed equal to 0.4 ms^{-1}.

$J(\mathbf{k})$ is a function that cannot be represented through closed form expressions; a preliminary analysis has shown that $J(\mathbf{k})$ is continuous w.r.t. its argument, but it is

highly nonlinear and multimodal. $J(\mathbf{k})$ is obtained by executing all the simulations which are required for the synthesis of figure 1.4(a) and by recording the distance where the feasibility is lost. Hence, the optimization problem to be solved is a black-box one. This rules out global optimization methods returning a certificate of optimality, namely branch-and-bound methods. For this reason the attention was restricted to two global approaches for poorly structured problems. The first one is based on the so called Radial Basis Functions (RBF), while the second one uses Differential Evolution concepts (DE), i.e., it is an evolutionary approach.

The solution proposed in [45] presents a performance index $J(\mathbf{k}) = 4.4 \cdot 10^{-2}$ m, while for the commercial controller $J(\mathbf{k}) = 6.3 \cdot 10^{-2}$ m. Through the optimization procedure the worst-case distance from the singularity has been roughly halved, the best minimizers proposed by RBF admit to achieve a performance index $J(\mathbf{k})$ equal to $2.0 \cdot 10^{-2}$ m, similar results have been obtained through the DE algorithm: $J(\mathbf{k}) = 1.9 \cdot 10^{-2}$ m. Results obtained for the 20 scanning directions are reported in figure 1.4, improvements introduced by the parameters optimization are evident.

Chapter 2

The Singularity Avoidance System

The system early proposed for the management of kinematic singularities has been modified in order to improve its performance. At the beginning of this chapter, in section 2.1, some theoretical considerations are drawn in order to analyze the critical situation. Then, section 2.2 explains the introduced novelties and the following sections 2.3, 2.4 and 2.5 describe in details the SAS blocks.

2.1 Preliminary considerations

The analysis in the following will consider a standard anthropomorphic manipulator equipped with a spherical wrist. Its structure is shown in figure 2.1. Frames have been assigned according to the modified Denavit-Hartenberg method [46] and the corresponding kinematic parameters are listed in table 2.1.

The SAS is activated, as introduced in section 1.2, only in proximity of kinematic singularities, i.e., it operates, for a short time, within small regions of the operational space. Inside such regions, trajectories can be reasonably approximated by their tangent and a constant tool-frame orientation can be assumed. Such simplifications allow drawing some considerations concerning the system behavior in the vicinity of singularities and suggest the strategy to be used for their avoidance.

According to the premise, the orientation of the $\hat{\mathbf{z}}_6$ axis is assumed constant in

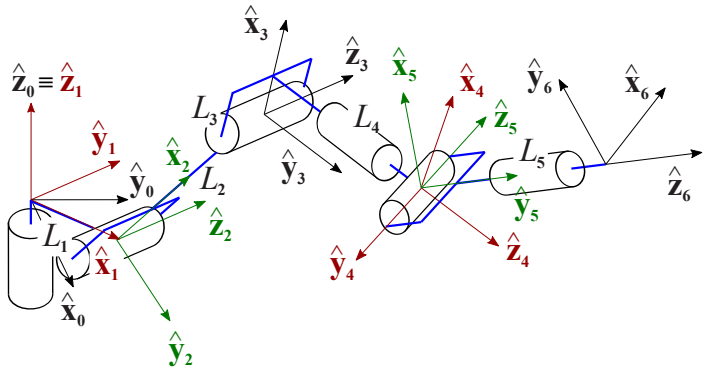


Figure 2.1: The manipulator frames assigned according to the modified Denavit-Hartenberg method.

Table 2.1: Kinematic parameters of a typical anthropomorphic manipulator.

i	α_{i-1}	a_{i-1}	θ_i	d_i
1	0	0	q_1	0
2	$-\pi/2$	L_1	q_2	0
3	0	L_2	q_3	0
4	$-\pi/2$	L_3	q_4	L_4
5	$\pi/2$	0	q_5	0
6	$-\pi/2$	0	q_6	L_5

the surroundings of the singularity, while no restrictions are posed on $\hat{\mathbf{x}}_6$ and $\hat{\mathbf{y}}_6$. The following proposition applies:

Proposition 1 *Any linear trajectory, executed with constant $\hat{\mathbf{z}}_6$, admits at most one wrist singular configuration if the variable associated to the first joint, i.e., q_1, changes during the motion, or two if q_1 is constant.*

Proof – According to the premises, unit vector $\hat{\mathbf{z}}_6 := [z_x\ z_y\ z_z]^T$ is supposed constant. A wrist singularity occurs every time $q_5 = 0$ or, equivalently, when $\hat{\mathbf{z}}_4 = \hat{\mathbf{z}}_6$ (see figure 2.2). Therefore, the first part of the proposition is verified if condition $\hat{\mathbf{z}}_4 = \hat{\mathbf{z}}_6$

applies for a single point of the trajectory.

Proper expressions for $\hat{\mathbf{z}}_4$ can be obtained by solving a direct kinematic problem. Given the parameters of table 2.1, it is possible to write

$$\hat{\mathbf{z}}_4 = \begin{bmatrix} c_1 c_{23} \\ s_1 c_{23} \\ s_{23} \end{bmatrix}, \tag{2.1}$$

where $c_1 = \cos(q_1)$, $s_1 = \sin(q_1)$, $c_{23} = \cos(q_2 + q_3)$, and $s_{23} = \sin(q_2 + q_3)$. Let us assume that the trajectory admits a singularity for the following configuration: $q_1 = \bar{\theta}_1$, $q_2 = \bar{\theta}_2$, $q_3 = \bar{\theta}_3$. Evidently, axes 4 and 6 are aligned and the following condition is satisfied

$$\hat{\mathbf{z}}_4 = \begin{bmatrix} \bar{c}_1 \bar{c}_{23} \\ \bar{s}_1 \bar{c}_{23} \\ \bar{s}_{23} \end{bmatrix} = \begin{bmatrix} z_x \\ z_y \\ z_z \end{bmatrix} = \hat{\mathbf{z}}_6, \tag{2.2}$$

where $\bar{c}_1 = \cos(\bar{\theta}_1)$, $\bar{s}_1 = \sin(\bar{\theta}_1)$, $\bar{c}_{23} = \cos(\bar{\theta}_2 + \bar{\theta}_3)$, and $\bar{s}_{23} = \sin(\bar{\theta}_2 + \bar{\theta}_3)$. Is it possible to have another singular configuration $q_1 = \bar{\bar{\theta}}_1$, $q_2 = \bar{\bar{\theta}}_2$, and $q_3 = \bar{\bar{\theta}}_3$ along the same trajectory? If yes, the following equation must apply

$$\begin{bmatrix} \bar{c}_1 \bar{c}_{23} \\ \bar{s}_1 \bar{c}_{23} \\ \bar{s}_{23} \end{bmatrix} = \begin{bmatrix} \bar{\bar{c}}_1 \bar{\bar{c}}_{23} \\ \bar{\bar{s}}_1 \bar{\bar{c}}_{23} \\ \bar{\bar{s}}_{23} \end{bmatrix} = \begin{bmatrix} z_x \\ z_y \\ z_z \end{bmatrix}, \tag{2.3}$$

where $\bar{\bar{c}}_1 = \cos(\bar{\bar{\theta}}_1)$, $\bar{\bar{s}}_1 = \sin(\bar{\bar{\theta}}_1)$, $\bar{\bar{c}}_{23} = \cos(\bar{\bar{\theta}}_2 + \bar{\bar{\theta}}_3)$, and $\bar{\bar{s}}_{23} = \sin(\bar{\bar{\theta}}_2 + \bar{\bar{\theta}}_3)$. If q_1 is variable along the trajectory then, clearly, $\bar{\bar{\theta}}_1 \neq \bar{\theta}_1$, so that (2.3) cannot be satisfied by any combination of $\bar{\bar{\theta}}_2$ and $\bar{\bar{\theta}}_3$: the trajectory admits only one singularity. Conversely, if q_1 is constant, i.e., $\bar{\bar{\theta}}_1 = \bar{\theta}_1$, then condition $\hat{\mathbf{z}}_4 = \hat{\mathbf{z}}_6$ is satisfied if the following equality applies

$$\bar{\theta}_2 + \bar{\theta}_3 = \bar{\bar{\theta}}_2 + \bar{\bar{\theta}}_3. \tag{2.4}$$

Let us study such eventuality. The origin of the fourth frame, i.e., the wrist position, can be obtained from the direct kinematics of the manipulator and written as follows

$$\mathbf{p}_4 = \begin{bmatrix} (L_3 c_{23} - L_4 s_{23} + L_2 c_2 + L_1) c_1 \\ (L_3 c_{23} - L_4 s_{23} + L_2 c_2 + L_1) s_1 \\ -L_4 c_{23} - L_3 s_{23} - L_2 s_2 \end{bmatrix}. \tag{2.5}$$

If $\bar{\bar{\theta}}_1 = \bar{\theta}_1$ and (2.4) apply, then the position of a further singular point $\bar{\mathbf{p}}_4$ can be expressed as follows

$$\bar{\mathbf{p}}_4 = \begin{bmatrix} (L_3\bar{\bar{c}}_{23} - L_4\bar{\bar{s}}_{23} + L_2\bar{\bar{c}}_2 + L_1)\bar{\bar{c}}_1 \\ (L_3\bar{\bar{c}}_{23} - L_4\bar{\bar{s}}_{23} + L_2\bar{\bar{c}}_2 + L_1)\bar{\bar{s}}_1 \\ -L_4\bar{\bar{c}}_{23} - L_3\bar{\bar{s}}_{23} - L_2\bar{\bar{s}}_2 \end{bmatrix} \tag{2.6}$$

$$= \begin{bmatrix} (L_3\bar{c}_{23} - L_4\bar{s}_{23} + L_2\bar{\bar{c}}_2 + L_1)\bar{c}_1 \\ (L_3\bar{c}_{23} - L_4\bar{s}_{23} + L_2\bar{\bar{c}}_2 + L_1)\bar{s}_1 \\ -L_4\bar{c}_{23} - L_3\bar{s}_{23} - L_2\bar{\bar{s}}_2 \end{bmatrix} \tag{2.7}$$

or, equivalently, as follows

$$\bar{\mathbf{p}}_4 = \begin{bmatrix} k_1 + k_2\bar{\bar{c}}_2 \\ k_3 + k_4\bar{\bar{c}}_2 \\ k_5 + k_6\bar{\bar{s}}_2 \end{bmatrix} \tag{2.8}$$

where k_i for $i = 1,\ldots,6$ are constants and defined as follows

$$\begin{aligned} k_1 &:= (L_3\bar{c}_{23} - L_4\bar{s}_{23} + L_1)\bar{c}_1, \\ k_2 &:= L_2\bar{c}_1, \\ k_3 &:= (L_3\bar{c}_{23} - L_4\bar{s}_{23} + L_1)\bar{s}_1, \\ k_4 &:= L_2\bar{s}_1, \\ k_5 &:= -L_4\bar{c}_{23} - L_3\bar{s}_{23}, \\ k_6 &:= -L_2. \end{aligned}$$

With a few algebraic manipulations, it can be shown that (2.8) is the equation of a circumference lying on plane $q_1 = \bar{\theta}_1 = \bar{\bar{\theta}}_1$, centered in $[k_1\ k_3\ k_5]^T$, and whose radius is L_2. It contains all the singular points which satisfy $\bar{\theta}_1 = \bar{\bar{\theta}}_1$ and (2.4). Figure 2.2, which shows a schematic side view of the manipulator, provides a graphical interpretation of the result. If the manipulator is executing a linear trajectory, the singular point circumference can be intersected, at most, into two points. ■

Proposition 1 asserts that linear trajectories executed with a constant $\hat{\mathbf{z}}_6$ normally admit no more than one kinematic singularity, since two may only appear for motions in which q_1 is constant. Consequently, if a singularity is avoided with a method which

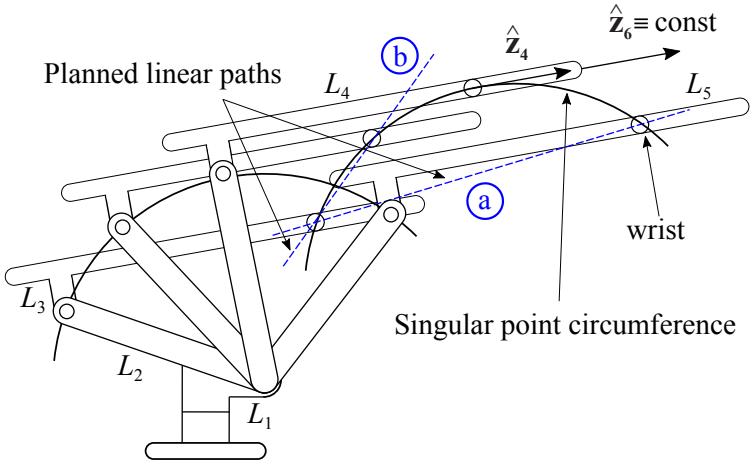


Figure 2.2: Side view of the anthropomorphic manipulator. Any constant vector $\hat{\mathbf{z}}_6$ generates a circumference of singular points lying on plane $q_1 = \hat{\theta}_1$. Any straight trajectory can intersect such circumference, at most, into two points.

brings back the system to the original trajectory, no further problems have to be expected. Evidently, the singularity avoidance transient may be critical since, according to the premises, the tool frame orientation is certainly changed, thus invalidating one of the conditions required by Proposition 1: a novel singular point may potentially appear during the transient. Next proposition poses a necessary condition which must be satisfied in the vicinity of the singularity: if it does not apply the system is driven toward a further singularity.

Proposition 2 *A necessary condition required for the avoidance of kinematic singularities is that* $\mathrm{sgn}(\dot{q}_4)$ *and* $\mathrm{sgn}(\dot{q}_6)$ *do not change during the motion.*

Proof – The tool-frame associated to the modified trajectory will be indicated in the following by $\widetilde{T}$ in order to distinguish it from T, i.e., from the one associated to the nominal trajectory. The Jacobian matrix for $\widetilde{T}$ is defined as follows

$$\mathbf{J}_{\widetilde{T}}(\mathbf{q}) := \left[\frac{\mathbf{J}_{v_{\widetilde{T}}}(\mathbf{q})}{\mathbf{J}_{\omega_{\widetilde{T}}}(\mathbf{q})} \right] .$$

It is consequently possible to express $\boldsymbol{\omega}_{\widetilde{T}}$, i.e., the angular speed of $\widetilde{T}$, through the following equation

$$\boldsymbol{\omega}_{\widetilde{T}} = \mathbf{J}_{\omega_{\widetilde{T}}}(\mathbf{q})\,\dot{\mathbf{q}} := \left[\mathbf{J}_{\omega_{\widetilde{T}_1}}(\mathbf{q}) \,|\, \mathbf{J}_{\omega_{\widetilde{T}_2}}(\mathbf{q})\right]\,\dot{\mathbf{q}}, \tag{2.9}$$

where $\mathbf{J}_{\omega_{\widetilde{T}_1}}(\mathbf{q})$ and $\mathbf{J}_{\omega_{\widetilde{T}_2}}(\mathbf{q})$ are 3x3 matrices obtained by partitioning $\mathbf{J}_{\omega_{\widetilde{T}}}(\mathbf{q})$. $\boldsymbol{\omega}_{\widetilde{T}}$ can also be obtained through the composition rule used for angular velocities. Consequently, it is possible to write

$$\boldsymbol{\omega}_{\widetilde{T}} = \boldsymbol{\omega}_T + {}^0_T\mathbf{R}(\mathbf{q})\,{}^T\boldsymbol{\omega}_{\widetilde{T},T} \tag{2.10}$$

where $\boldsymbol{\omega}_T$ is the nominal angular speed of the tool-frame, while ${}^T\boldsymbol{\omega}_{\widetilde{T},T}$ is the relative angular speed between the nominal tool-frame and the modified one, described w.r.t. T. By combining (2.9) with (2.10) and by performing some algebraic manipulations, the following equation can be obtained

$${}^T_0\mathbf{R}(\mathbf{q})\,\mathbf{J}_{\omega_{\widetilde{T}_2}}(\mathbf{q})\,\dot{\tilde{\mathbf{q}}} = {}^T_0\mathbf{R}(\mathbf{q})\,\left[\boldsymbol{\omega}_T - \mathbf{J}_{\omega_{\widetilde{T}_1}}(\mathbf{q})\,\dot{\bar{\mathbf{q}}}\right] + {}^T\boldsymbol{\omega}_{\widetilde{T},T}, \tag{2.11}$$

where $\dot{\bar{\mathbf{q}}} := [\dot{q}_1\,\dot{q}_2\,\dot{q}_3]^T$ while $\dot{\tilde{\mathbf{q}}} := [\dot{q}_4\,\dot{q}_5\,\dot{q}_6]^T$. Practically, $\dot{\bar{\mathbf{q}}}$ and $\dot{\tilde{\mathbf{q}}}$ are partitions of $\dot{\mathbf{q}} := [\dot{\bar{\mathbf{q}}}^T \,|\, \dot{\tilde{\mathbf{q}}}^T]^T$. By defining

$$\hat{\boldsymbol{\omega}} := {}^T_0\mathbf{R}(\mathbf{q})\,\left[\boldsymbol{\omega}_T - \mathbf{J}_{\omega_{\widetilde{T}_1}}(\mathbf{q})\,\dot{\bar{\mathbf{q}}}\right] + {}^T\boldsymbol{\omega}_{\widetilde{T},T}$$

it is finally possible to write (2.11) as follows

$${}^T_0\mathbf{R}(\mathbf{q})\,\mathbf{J}_{\omega_{\widetilde{T}_2}}(\mathbf{q})\,\dot{\tilde{\mathbf{q}}} = \hat{\boldsymbol{\omega}} \tag{2.12}$$

and, in turn, to obtain

$$\dot{\tilde{\mathbf{q}}} = \mathbf{J}^{-1}_{\omega_{\widetilde{T}_2}}(\mathbf{q})\,{}^0_T\mathbf{R}(\mathbf{q})\,\hat{\boldsymbol{\omega}}. \tag{2.13}$$

The structure of (2.13) can be analyzed by considering the manipulator parameters reported in table 2.1. To this purpose, let us redefine the joint variables as follows $q_i := \hat{\theta}_i + \theta_i$, where $\hat{\theta}_i$ is the value assumed by the joint variables in the singular point, so that it is constant along the trajectory, while θ_i is the displacement w.r.t. such value and it clearly changes during the motion. Evidently, $\hat{\theta}_5 = 0$, while no restrictions are

imposed on the other $\hat{\theta}_i$s since the discussion in the following applies to any wrist singularity of the workspace.

According to the premises $\hat{\mathbf{z}}_T \equiv \hat{\mathbf{z}}_6$ is constant along the nominal trajectory, so that the tool-frame can only rotate around such axis. As a consequence, ${}^0_T\mathbf{R}(\mathbf{q})$ can be obtained from the orientation assumed in the singular point by admitting further rotations around the $\hat{\mathbf{z}}_6$ axis. Practically, the orientation of the tool-frame along the nominal trajectory can be obtained by assuming $\mathbf{q} := \left[\hat{\theta}_1\ \hat{\theta}_2\ \hat{\theta}_3\ (\hat{\theta}_4+\theta_4)\ 0\ (\hat{\theta}_6+\theta_6)\right]^T$. A few algebraic manipulations make it possible to express the three components of (2.13) as follows

$$\dot{q}_4 = \dot{\theta}_4 = \frac{\mathbf{f}_1(\mathbf{q}, \dot{\mathbf{q}}, {}^T\boldsymbol{\omega}_{\widetilde{T},T})}{\sin(\theta_5)} \tag{2.14}$$

$$\dot{q}_5 = \dot{\theta}_5 = \mathbf{f}_2(\mathbf{q}, \dot{\mathbf{q}}, {}^T\boldsymbol{\omega}_{\widetilde{T},T}) \tag{2.15}$$

$$\dot{q}_6 = \dot{\theta}_6 = \frac{\mathbf{f}_3(\mathbf{q}, \dot{\mathbf{q}}, {}^T\boldsymbol{\omega}_{\widetilde{T},T})}{\sin(\theta_5)} \tag{2.16}$$

Functions $\mathbf{f}_i$, which are not reported for space reasons, are highly nonlinear. However, from (2.14) it can be inferred that

$$\sin(\theta_5) = \frac{\mathbf{f}_1(\mathbf{q}, \dot{\mathbf{q}}, {}^T\boldsymbol{\omega}_{\widetilde{T},T})}{\dot{\theta}_4}\ . \tag{2.17}$$

Singularities are certainly avoided if $\theta_5 \neq 0$, i.e., if the sign of θ_5 does not change along a trajectory. Consequently, (2.17) allows one asserting that such condition can be achieved if during the execution of a trajectory the signs of $\mathbf{f}_1$ and of $\dot{\theta}_4$ always switch simultaneously or, conversely, if they do not switch at all. The first condition can be hardly obtained with any real-time method because of the complexity of the functions involved, so that the second method is the only one that can be actually exploited. Similar considerations apply for $\dot{\theta}_6$. ■

The demonstration shows that an alternative solution could be potentially proposed, however functions $\mathbf{f}_1$ and $\mathbf{f}_3$ depend on the Cartesian path through a set of highly nonlinear relationships, so that it was not possible to devise analytical relations for the preservation of the feasibility. The sign maintenance is the easiest one to be guaranteed. Practically, Proposition 2 asserts that, during any transient for the

singularity avoidance, motion directions of joints 4 and 6 must not invert. SAS trajectories fulfill such condition.

2.2 The novelties introduced

Differently from other methods in the literature, the SAS handles separately the inverse kinematics and the singularity problems. More precisely, the first one is solved through a standard algorithm based on efficient closed-form equations, while singularities are handled by the SAS, which only acts on the tool-frame orientation: as shown in figure 2.3, position references are directly sent to the inverse kinematics block, so as to guarantee that assigned Cartesian paths and time-law are preserved with certainty.

In the following, $\mathbf{p}_T(s)$ and ${}^0_T\mathbf{R}(s)$ specify the position and the orientation of the tool-frame expressed as functions of the curvilinear coordinate s. Trajectories are obtained by combining positions and orientations with time-law $s(t)$, so that $\mathbf{p}(t) := \mathbf{p}[s(t)]$ is a Cartesian trajectory, while ${}^0_T\mathbf{R}(t) := {}^0_T\mathbf{R}[s(t)]$ is an orientation trajectory. The same notation is used for velocities and accelerations of the tool-frame.

The main assumption made in this work is that trajectories are planned and then immediately processed by the SAS in real time, so that there is no dead-time between the planning phase and the trajectory execution. Every time a new trajectory is generated, the following information is provided to the SAS: the path equation, given by $\mathbf{p}_T(s)$ and ${}^0_T\mathbf{R}(s)$ where s is the curvilinear coordinate along the path, and subsequently, at each sample time, the instantaneous values of ${}^0_T\mathbf{R}(t)$, $\mathbf{p}_T(t)$, $\boldsymbol{\omega}_T(t)$, $\mathbf{v}_T(t)$, $\boldsymbol{\alpha}_T(t)$, and $\mathbf{a}_T(t)$. The output of the system is represented by a modified trajectory which fulfills, for all the joints, the following velocity and the acceleration constraints

$$\dot{\mathbf{q}}^- \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}^+, \tag{2.18}$$

$$\ddot{\mathbf{q}}^- \leq \ddot{\mathbf{q}} \leq \ddot{\mathbf{q}}^+, \tag{2.19}$$

where $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ are the first and the second time derivatives of joint variables $\mathbf{q} := [q_1, q_2, q_3, q_4, q_5, q_6]^T \in \mathbb{R}^6$, while $\dot{\mathbf{q}}^-, \ddot{\mathbf{q}}^- \in (\mathbb{R}^-)^6$, and $\dot{\mathbf{q}}^+, \ddot{\mathbf{q}}^+ \in (\mathbb{R}^+)^6$ are user de-

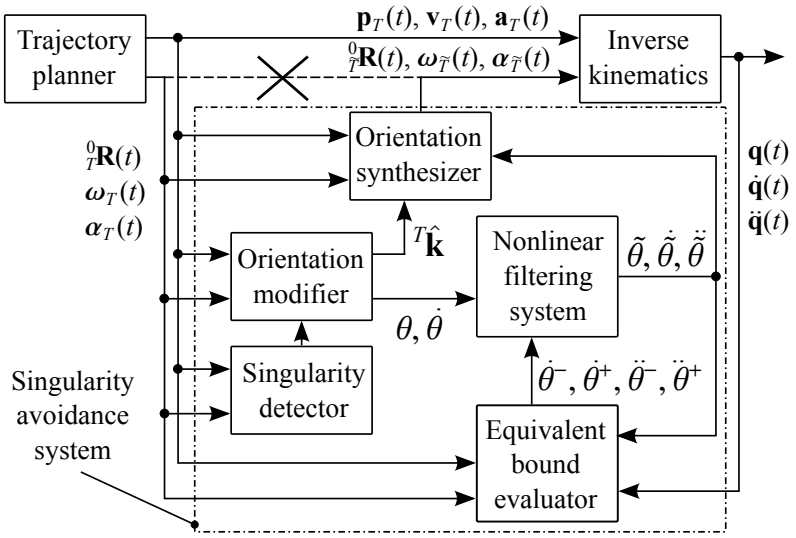


Figure 2.3: Schematic representation of the novel SAS.

fined bounds for joint velocities and accelerations. Such bounds may also be variable, so as to account, for example, for the presence of torque constraints (see also the discussion in [45]). Far from singularities, trajectories must coincide with the original ones, while in critical configurations minor orientation displacements can be admitted in order to fulfill (2.18) and (2.19). The imposition of specific bounds represents an improvement w.r.t. classical methods, which generically limit joint velocities and accelerations, but do not explicitly bound them.

The novel SAS is based on the functional scheme shown in figure 2.3. Its behavior is synthetically recalled in the following. When the singularity detector block ascertains that the tool-frame is moving toward a singularity, the orientation modifier proposes a candidate angular displacement between T and $\widetilde{T}$ – where $\widetilde{T}$ indicates the orientation-modified tool-frame – so as to allow its avoidance. The displacement is specified by defining an appropriate rotation axis, described through unit-vector ${}^T\hat{\mathbf{k}}$, and an angular offset θ. A proper choice of ${}^T\hat{\mathbf{k}}$ and θ allows the fulfillment of the condition proposed by Proposition 2.

${}^T\hat{\mathbf{k}}$ is directly sent to the orientation synthesizer, i.e., to the block which combines

the orientation of the nominal trajectory with the SAS angular displacement, while θ is processed in order to allow smooth and feasible orientation changes. An instantaneous change of θ would indeed cause unfeasible joint speeds and accelerations. The nonlinear filtering system (see [47,48] for details concerning its implementation) is used to solve possible feasibility issues. It generates an output signal given by $\widetilde{\theta}$, $\dot{\widetilde{\theta}}$, and $\ddot{\widetilde{\theta}}$ which represents the best approximation of θ which satisfies the following limits

$$\dot{\theta}^{-} \leq \dot{\widetilde{\theta}} \leq \dot{\theta}^{+}, \tag{2.20}$$

$$\ddot{\theta}^{-} \leq \ddot{\widetilde{\theta}} \leq \ddot{\theta}^{+}. \tag{2.21}$$

It will be shown in section 2.5, bounds $\dot{\theta}^{-}$, $\dot{\theta}^{+}$, $\ddot{\theta}^{-}$, and $\ddot{\theta}^{+}$ directly descend from $\dot{\mathbf{q}}^{-}$, $\ddot{\mathbf{q}}^{-}$, $\dot{\mathbf{q}}^{+}$, and $\ddot{\mathbf{q}}^{+}$, so that (1.1) and (1.2) are fulfilled as long as (2.20) and (2.21) are satisfied.

The functional scheme of figure 2.3 is similar to the one proposed in [45], but all the blocks have been actually redesigned on the basis of different concepts in order to allow crossing the wrist singularities and to achieve a four time smaller computational time. Let us summarize in the following the main changes which have been introduced.

The previous version of the Orientation Modifier block was designed so as to evaluate the angular displacements between T and $\widetilde{T}$ at each sample time. As a consequence, ${}^{T}\hat{\mathbf{k}}$ was frequently changed in the neighborhood of critical configurations, thus worsening the system performance. In the new approach, if the procedure proposed in section 2.3 detects a singularity, ${}^{T}\hat{\mathbf{k}}$ is at first selected by means of the new analytic procedure proposed in section 2.4.1 and, then, it is kept constant until the critical configuration is passed. Furthermore, while in [45] ${}^{T}\hat{\mathbf{k}}$ was chosen within a set of 26 candidate directions, in this paper any generic direction in the 3D space can be selected. This choice makes the SAS effective within a wider volume of the workspace. Finally, in the new SAS the axis-angle minimal notation replaces the XYZ Euler-angles for the representation of the candidate angular displacements. This apparently minor change actually reduces the problem dimensionality and complexity, thus leading to the above mentioned efficiency improvement.

Sections 2.3, 2.4, and 2.5 will describe the novel parts of the SAS.

2.3 Singularity detector

The best SAS performance can be achieved if ${}^{T}\hat{\mathbf{k}}$, i.e., the rotation axis chosen for the singularity avoidance, is kept constant during the execution of a critical trajectory. It is worth recalling that the actual path and orientation of the tool frame are unknown to the SAS. As a consequence, it first finds an estimate of $\mathbf{p}_T(s)$ and ${}^{0}_{T}\mathbf{R}(s)$ – where s is the curvilinear coordinate – then inspects the path so as to predict the occurrence of wrist singularities which are revealed, as known, by values of q_5, i.e., of the fifth joint variable, close to zero. Industrial manipulators usually implement very simple path primitives given by linear segments or arcs, so that path equations can be easily estimated from the past set-points. For example, for linear segments the estimated-path equation can be expressed as follows

$$\mathbf{p}_T(s) = \mathbf{p}_T(t_0) + \frac{\mathbf{v}_T(t_0)}{\|\mathbf{v}_T(t_0)\|}\, s \,,$$

where $\mathbf{p}_T(t_0)$ and $\mathbf{v}_T(t_0)$ are the current reference signals for position and velocity. Analogous path primitives can be obtained for circular arcs and, with similar considerations, it is also possible to synthesize proper expressions for ${}^{0}_{T}\mathbf{R}(s)$. More complex path primitives can be approximated with arcs since small deviations from the most appropriate value of ${}^{T}\hat{\mathbf{k}}$ have only a minimal impact on the system performance. Given these premises, an estimate of $\mathbf{p}_T(s)$ and of ${}^{0}_{T}\mathbf{R}(s)$ is supposed to be available in the remainder of the paper.

A point close to the wrist singularity, which is then used for the evaluation of unit vector ${}^{T}\hat{\mathbf{k}}$, is obtained by means of Algorithms 1 and 2. Algorithm 1 is executed at each sample time until a singularity, if any, is detected. It preliminary inspects $\mathbf{p}_T(s)$ for a length s_f which depends on the current speed $\mathbf{v}_T(t_0)$ and on a lookahead time t^*. In particular, it initially selects $N_s + 1$ points equally distributed along the path, where N_s is kept small for efficiency reasons ($N_s \simeq 10$ always given good results). For each point s it saves the following data in *Arr* (see lines 1–7 of Algorithm 1): $\mathbf{p}_T(s), {}^{0}_{T}\mathbf{R}(s), \mathbf{q}(s)$, and s. In the second part of Algorithm 1 (see lines 8–14), the

Algorithm 1: The path is preliminarily inspected looking for possible critical configurations.

Data: t^*, lookahead time; $\mathbf{v}_T(t_0)$, current Cartesian speed;
Result: *sing*, singularity alarm; *Sol*, data required by Algorithm 2;

$s_f \leftarrow \|\mathbf{v}_T(t_0)\|\, t^*$;
$sing \leftarrow false$;
foreach $i \in 0,\ldots,N_s$ **do**
    $s \leftarrow \frac{i s_f}{N_s}$;
    Evaluate: ${}^0_T\mathbf{R}(s)$, $\mathbf{p}_T(s)$;
    $\mathbf{q}(s) \leftarrow InverseKinematics\left(\mathbf{p}_T(s), {}^0_T\mathbf{R}(s)\right)$;
    $Arr(i) \leftarrow \mathbf{p}_T(s), {}^0_T\mathbf{R}(s), \mathbf{q}(s), s$;
foreach $i \in 1,\ldots,N_s - 1$ **do**
    if $(|Arr(i).q_5| \leq |Arr(i-1).q_5|)$ & $(|Arr(i).q_5| \leq |Arr(i+1).q_5|)\,\&(|q_5| < \bar{a}_2)$ **then**
        $sing \leftarrow true$;
        $Sol(0) \leftarrow Arr(i-1)$;
        $Sol(2) \leftarrow Arr(i)$;
        $Sol(4) \leftarrow Arr(i+1)$;
        break;

acquired data are scrutinized looking for a possible singularity. Practically, the inspection aims at discovering if q_5 admits a minimum in one of the acquired points, and if such minimum is lower than a given threshold. If both conditions are satisfied the manipulator is probably running toward a critical configuration, so that the point admitting the minimum value of q_5 is saved together with its two adjacent points into vector *Sol* in order to be later used by Algorithm 2.

If a singularity is detected, Algorithm 2 initially refines the solution provided by Algorithm 1 by looking for the path-point which is the closest one to the singularity. The point assumed for the evaluation of ${}^T\hat{\mathbf{k}}$ is placed immediately before the mini-

Algorithm 2: Search for the point along the path which is the closest one to the singularity and evaluation of ${}^{T}\hat{\mathbf{k}}$.

Data: Sol, first tentative solution; s^*, threshold;
Result: ${}^{T}\hat{\mathbf{k}}$, rotation axis;

$Step \leftarrow Sol(2).s - Sol(0).s$;
while $Step > s^*$ **do**
  $s \leftarrow (Sol(2).s + Sol(0).s)/2$;
  Evaluate: ${}^{0}_{T}\mathbf{R}(s)$, $\mathbf{p}_T(s)$;
  $\mathbf{q}(s) \leftarrow InverseKinematics\left(\mathbf{p}_T(s), {}^{0}_{T}\mathbf{R}(s)\right)$;
  $Sol(1) \leftarrow \mathbf{p}_T(s), {}^{0}_{T}\mathbf{R}(s), \mathbf{q}(s), s$;
  $s \leftarrow (Sol(4).s + Sol(2).s)/2$;
  Evaluate: ${}^{0}_{T}\mathbf{R}(s)$, $\mathbf{p}_T(s)$;
  $\mathbf{q}(s) \leftarrow InverseKinematics\left(\mathbf{p}_T(s), {}^{0}_{T}\mathbf{R}(s)\right)$;
  $Sol(3) \leftarrow \mathbf{p}_T(s), {}^{0}_{T}\mathbf{R}(s), \mathbf{q}(s), s$;
  foreach $i \in 1,2,3$ **do**
    if $(|Sol(i).q_5| \leq |Sol(i-1).q_5|)$ & $(|Sol(i).q_5| \leq |Sol(i+1).q_5|)$ **then**
      $Sol(0) \leftarrow Sol(i-1)$;
      $Sol(4) \leftarrow Sol(i+1)$;
      $Sol(2) \leftarrow Sol(i)$;
      break;
  $Step \leftarrow Sol(2).s - Sol(0).s$;
Evaluate: ${}^{T}\hat{\mathbf{k}}$ for $Sol(0)$;

mum of q_5, i.e., it is sufficiently close to the critical configuration to allow selecting an appropriate rotation axis according to the procedure proposed in section 2.4.1, but it is certainly non-singular. Such choice is important since the evaluation of ${}^{T}\hat{\mathbf{k}}$ requires a full rank Jacobian matrix. The selected value of ${}^{T}\hat{\mathbf{k}}$ is then kept constant, i.e., Algorithms 1 and 2 are no more executed unless the critical configuration has been

passed or the path has been changed.

The computational load of Algorithms 1 and 2 is mainly given by the evaluation of ${}^{T}\hat{\mathbf{k}}$. The procedure proposed in section 2.4.1 to this purpose is very efficient: its average evaluation time, measured with a Intel Core2 Duo PC running at 3.0GHz, i.e., with the same PC used for the experimental tests proposed in section 3.2, is equal to $1.063 \cdot 10^{-5}$ s, i.e., it is much smaller than the process sample time ($2 \cdot 10^{-3}$ s).

2.4 Orientation modifier

The orientation modifier is the second block of the SAS and it is devoted to the evaluation of a proper rotation axis and the relative rotation angle in order to introduce the required modification to the nominal orientation. Section 2.4.1 is devoted to the synthesis of the rotation axis while section 2.4.2 shows a procedure to properly tune the system variables, i.e. the rotation angle and the activation/deactivation distance.

2.4.1 Selection of rotation axis ${}^{T}\hat{k}$

The ellipsoid of manipulability provides information concerning the relationships between the velocities in the operational space and the ones in the configuration space. In particular, its shape allows one to immediately individuate which directions of the operational space have a minimal impact on the joint speeds, and which others should be avoided since they would require high joint speeds. As known, the firsts coincide with the major principal semi-axis of the ellipsoid, the seconds with the minor principal semi-axis [49].

The synthesis of ${}^{T}\hat{\mathbf{k}}$ requires knowing a configuration $\mathbf{q}$ corresponding to a path-point placed very close to the singularity. As shown in section 2.3, it can be obtained through Algorithms 1 and 2.

The relationship between velocities in the configuration space and the ones in the operational space can be expressed as follows

$$\overline{\mathbf{v}}_T = \mathbf{J}_T(\mathbf{q})\,\dot{\mathbf{q}}\,, \tag{2.22}$$

where $\mathbf{J}_T(\mathbf{q})$ is the Jacobian matrix associated to the tool-frame, while

$$\overline{\mathbf{v}}_T := \begin{bmatrix} \mathbf{v}_T \\ \boldsymbol{\omega}_T \end{bmatrix} \tag{2.23}$$

is the tool-frame generalized velocity. Equation (2.22) can be inverted since $\mathbf{q}$ is a non-singular point (see section 2.3), so that it is possible to write

$$\dot{\mathbf{q}} = \mathbf{J}_T^{-1}\,\overline{\mathbf{v}}_T \ . \tag{2.24}$$

By partitioning $\mathbf{J}_T^{-1}(\mathbf{q})$ as follows

$$\mathbf{J}_T^{-1} := \left[\overline{\mathbf{J}}_{v_T}(\mathbf{q})\,|\,\overline{\mathbf{J}}_{\omega_T}(\mathbf{q})\right] \ , \tag{2.25}$$

it is possible to rewrite (2.24) into the following form

$$\dot{\mathbf{q}} = \overline{\mathbf{J}}_{v_T}(\mathbf{q})\,\mathbf{v}_T + \overline{\mathbf{J}}_{\omega_T}(\mathbf{q})\,\boldsymbol{\omega}_T \ .$$

Any change of the tool-frame orientation carried out during the motion causes a change of its angular speed. By recalling that the tool-frame associated to the modified orientation is indicated by $\widetilde{T}$, its angular speed can be expressed as follows

$$\boldsymbol{\omega}_{\widetilde{T}} = \boldsymbol{\omega}_T + \boldsymbol{\omega}_{\widetilde{T},T} \ , \tag{2.26}$$

where $\boldsymbol{\omega}_{\widetilde{T},T}$ is the relative angular speed between frames $\widetilde{T}$ and T, while $\mathbf{v}_T = \mathbf{v}_{\widetilde{T}}$ because the SAS preserves the assigned Cartesian path. Through obvious algebraic manipulations, it can be shown that joint speeds associated to frame $\widetilde{T}$ can be expressed as follows

$$\begin{aligned} \dot{\widetilde{\mathbf{q}}} &= \overline{\mathbf{J}}_{v_T}(\mathbf{q})\,\mathbf{v}_T + \overline{\mathbf{J}}_{\omega_T}(\mathbf{q})\,\boldsymbol{\omega}_{\widetilde{T}} \\ &= \overline{\mathbf{J}}_{v_T}(\mathbf{q})\,\mathbf{v}_T + \overline{\mathbf{J}}_{\omega_T}(\mathbf{q})\,\boldsymbol{\omega}_T + \overline{\mathbf{J}}_{\omega_T}(\mathbf{q})\,\boldsymbol{\omega}_{\widetilde{T},T} \\ &= \mathbf{J}_T^{-1}(\mathbf{q})\,\overline{\mathbf{v}}_T + \overline{\mathbf{J}}_{\omega_T}(\mathbf{q})\,\boldsymbol{\omega}_{\widetilde{T},T} \end{aligned} \tag{2.27}$$

or, equivalently,

$$\dot{\widetilde{\mathbf{q}}} = \dot{\mathbf{q}} + \Delta\dot{\mathbf{q}} \ , \tag{2.28}$$

where

$$\Delta\dot{\mathbf{q}} := \overline{\mathbf{J}}_{\omega_T}(\mathbf{q})\,\boldsymbol{\omega}_{\widetilde{T},T} = \overline{\mathbf{J}}_{\omega_T}(\mathbf{q})\,{}^0_T\mathbf{R}(\mathbf{q})\,{}^T\boldsymbol{\omega}_{\widetilde{T},T}\,, \tag{2.29}$$

while $\dot{\mathbf{q}}$ is defined according to (2.24). ${}^0_T\mathbf{R}$ is the rotation matrix which describes the orientation of the tool-frame w.r.t. the base, so that ${}^T\boldsymbol{\omega}_{\widetilde{T},T} := [\omega_x\ \omega_y\ \omega_z]^T$ is the angular velocity between frames $\widetilde{T}$ and T, described w.r.t. frame T.

The first term in (2.28) cannot be modified being associated to the nominal trajectory. Conversely, it is possible to act on the second one, i.e., on (2.29), in order to avoid the singularity. Equation (2.29) creates a relationship between ${}^T\boldsymbol{\omega}_{\widetilde{T},T}$ and $\Delta\dot{\mathbf{q}}$ that can be used for the selection of an appropriate axis ${}^T\hat{\mathbf{k}}$ around which the tool-frame should rotate in order to avoid the singularity. In particular, ${}^T\hat{\mathbf{k}}$ will be selected by means of reasonings concerning the ellipsoid of manipulability, so as to guarantee that the additional speed introduced by the SAS has a minimal impact on joints 4 and 6, i.e., the most solicited ones.

Let us further reduce the problem dimensionality. The manipulator frames are supposed to be assigned according to the modified Denavit-Hartenberg method [50], so that the $\hat{\mathbf{z}}$ axis of the tool-frame always coincides with the rotation axis of the sixth joint variable, i.e., with the axis associated to q_6. This implies that matrix $\overline{\mathbf{J}}_{\omega_T}(\mathbf{q})\,{}^0_T\mathbf{R}(\mathbf{q})$ always assumes the following structure

$$\overline{\mathbf{J}}_{\omega_T}(\mathbf{q})\,{}^0_T\mathbf{R}(\mathbf{q}) := \begin{bmatrix} j_{11} & j_{12} & 0 \\ j_{21} & j_{22} & 0 \\ j_{31} & j_{32} & 0 \\ j_{41} & j_{42} & 0 \\ j_{51} & j_{52} & 0 \\ j_{61} & j_{62} & 1 \end{bmatrix},$$

i.e., ω_z only influences $\Delta\dot{q}_6$ while it has no impact on the other joint speeds and, in particular, on $\Delta\dot{q}_5$. The immediate consequence of this remark is that ω_z cannot be used to force q_5 away from the singular configuration, so that it can be posed equal to zero. As a consequence, (2.29) can be simplified as follows

$$\Delta\dot{\mathbf{q}} = \overline{\overline{\mathbf{J}}}_{\omega_T}(\mathbf{q})\,{}^T\overline{\boldsymbol{\omega}}_{\widetilde{T},T}\,, \tag{2.30}$$

where

$$\bar{\bar{\mathbf{J}}}_{\omega_T}(\mathbf{q}) := \begin{bmatrix} j_{11} & j_{12} \\ j_{21} & j_{22} \\ j_{31} & j_{32} \\ j_{41} & j_{42} \\ j_{51} & j_{52} \\ j_{61} & j_{62} \end{bmatrix} \tag{2.31}$$

is given by the first two columns of $\bar{\mathbf{J}}(\mathbf{q})$, while

$${}^{T}\overline{\boldsymbol{\omega}}_{\tilde{T},T} := \begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix} \tag{2.32}$$

contains the first two components of ${}^{T}\boldsymbol{\omega}_{\tilde{T},T}$.

The relationship between $\Delta\dot{\mathbf{q}}$ and ${}^{T}\overline{\boldsymbol{\omega}}_{\tilde{T},T}$ can be studied by means of a classic analysis based on the ellipsoid of manipulability

$$\Delta\dot{\mathbf{q}}^{T}\Delta\dot{\mathbf{q}} := {}^{T}\overline{\boldsymbol{\omega}}_{\tilde{T},T}^{T}\bar{\bar{\mathbf{J}}}_{\omega_T}^{T}(\mathbf{q})\bar{\bar{\mathbf{J}}}_{\omega_T}(\mathbf{q}){}^{T}\overline{\boldsymbol{\omega}}_{\tilde{T},T} = 1\ . \tag{2.33}$$

Matrix $\bar{\bar{\mathbf{J}}}_{\omega_T}(\mathbf{q})$ is maximum rank since position $\mathbf{q}$ is not singular, so that $\bar{\bar{\mathbf{J}}}_{\omega_T}^{T}(\mathbf{q})\bar{\bar{\mathbf{J}}}_{\omega_T}(\mathbf{q})$ is positive-defined and, consequently, (2.33) is the equation of an ellipsoid which minor principal semi-axis individuates a spinning direction which could lead to critical joint speeds. For this reason, the most appropriate direction for ${}^{T}\overline{\boldsymbol{\omega}}_{\tilde{T},T}$ to be used for the singularity avoidance coincides with the major principal semi-axis, that can be individuated by evaluating the eigenvector of matrix $\bar{\bar{\mathbf{J}}}_{\omega_T}^{T}(\mathbf{q})\bar{\bar{\mathbf{J}}}_{\omega_T}(\mathbf{q})$ associated to the minimum eigenvalue. The computational burden is limited since matrix $\bar{\bar{\mathbf{J}}}_{\omega_T}^{T}(\mathbf{q})\bar{\bar{\mathbf{J}}}_{\omega_T}(\mathbf{q})$ has dimension 2x2, but it can be further simplified according to the procedure proposed in the following. In the neighborhoods of a wrist singularity, j_{4i} and j_{6i}, $i = 1,2$ always assume modules which are much bigger than the ones of all the other terms and, moreover, $j_{4i} \simeq -j_{6i}$, $i = 1,2$. As a consequence, many terms of $\bar{\bar{\mathbf{J}}}_{\omega_T}^{T}(\mathbf{q})\bar{\bar{\mathbf{J}}}_{\omega_T}(\mathbf{q})$ are negligible w.r.t. the others. After a few manipulations, the following simplified matrix can be obtained

$$\bar{\bar{\mathbf{J}}}_{\omega_T}^{T}(\mathbf{q})\bar{\bar{\mathbf{J}}}_{\omega_T}(\mathbf{q}) \simeq 2\begin{bmatrix} j_{41}^2 & j_{41}j_{42} \\ j_{41}j_{42} & j_{42}^2 \end{bmatrix}\ . \tag{2.34}$$

The original matrix $\overline{\overline{\mathbf{J}}}^{T}_{\omega_T}(\mathbf{q})\overline{\overline{\mathbf{J}}}_{\omega_T}(\mathbf{q})$ was positive defined, with one eigenvalue very close to zero, while the simplified one is positive semi-defined. According to the premises, the eigenvector associated to the zero eigenvalue individuates the major principal semi-axis of the ellipsoid of manipulability. A few manipulations lead to the following eigenvector

$$ {}^{T}\overline{\mathbf{k}} := \frac{[j_{42} \;\; -j_{41}]^T}{\sqrt{j_{41}^2 + j_{42}^2}} , \tag{2.35} $$

which can be consequently selected as the rotation-axis for the singularity avoidance procedure.

The effectiveness of this choice can be also indirectly verified by analyzing the impact on the joint speeds of an additional angular velocity defined as follows

$$ {}^{T}\overline{\boldsymbol{\omega}}_{\widetilde{T},T} := {}^{T}\overline{\mathbf{k}}\,\dot{\theta} , \tag{2.36} $$

where $\dot{\theta} = \left\| {}^{T}\overline{\boldsymbol{\omega}}_{\widetilde{T},T} \right\|$. Velocity changes for joints 4, 5, and 6 can be obtained by applying (2.36) to (2.30). A few algebraic manipulations make it possible to write

$$ \Delta\dot{q}_4 = \frac{j_{41}j_{42} - j_{42}j_{41}}{\sqrt{j_{41}^2 + j_{42}^2}}\,\dot{\theta} = 0 , \tag{2.37} $$

$$ \Delta\dot{q}_5 = \frac{j_{51}j_{42} - j_{52}j_{41}}{\sqrt{j_{41}^2 + j_{42}^2}}\,\dot{\theta} , \tag{2.38} $$

$$ \Delta\dot{q}_6 = \frac{j_{61}j_{42} - j_{62}j_{41}}{\sqrt{j_{41}^2 + j_{42}^2}}\,\dot{\theta} = 0 . \tag{2.39} $$

${}^{T}\overline{\boldsymbol{\omega}}_{\widetilde{T},T}$ has no impact on the speeds of joints 4 and 6 by virtue of (2.37) and (2.39), so that there is no risk to worsen an already critical situations. Conversely, ${}^{T}\overline{\boldsymbol{\omega}}_{\widetilde{T},T}$ influences the velocity of joint 5 and can be used to avoid the singularity by keeping q_5 far from zero. This result can be achieved by imposing the same sign of q_5 to $\Delta\dot{q}_5$, i.e., by redefining ${}^{T}\overline{\mathbf{k}}$ as follows

$$ {}^{T}\overline{\mathbf{k}} := \operatorname{sgn}(q_5)\operatorname{sgn}(j_{51}j_{42} - j_{52}j_{41})\,\frac{[j_{42} \;\; -j_{41}]^T}{\sqrt{j_{41}^2 + j_{42}^2}} . $$

The rotation axis is finally obtained by adding the missing $\mathbf{z}$ component to ${}^T\overline{\mathbf{k}}$, according to the following equation

$$ {}^T\hat{\mathbf{k}} := \begin{bmatrix} {}^T\overline{\mathbf{k}} \\ 0 \end{bmatrix} . \tag{2.40} $$

Figure 2.4a schematically shows what happens when a straight trajectory passes in the surroundings of a kinematic singularity. At the point of minimum distance between trajectory and singularity, the motion of the first joint is subject to a sharp acceleration, with speeds that may be unfeasible. Unit vector ${}^T\hat{\mathbf{k}}$, associated to the major principal axis of the ellipsoid, if evaluated in such point, indicates the motion direction which would produce the lowest joint velocities. ${}^T\hat{\mathbf{k}}$ can be used to generate a small speed $\alpha\,{}^T\hat{\mathbf{k}}$ to be added to $\mathbf{v}_T$, so as to generate a path passing farther from the singularity. Consequently, the speed of the first joint will be lowered.

Trajectories crossing singular points must be handled differently. Such trajectories, as suggested by the alignment between the major principal axis of the ellipsoid and the given path, may be theoretically executed by avoiding the π turn of the first joint (see also figure 2.4b). However, under actual operating conditions, any small numerical rounding in the forward/inverse kinematics forces an undesired sudden turn of joint 1. As a consequence, it is always better to force a controlled rotation of joint 1. To this purpose, a speed $\alpha\,{}^T\hat{\mathbf{k}}^*$ can be added to the nominal $\mathbf{v}_T$, where ${}^T\hat{\mathbf{k}}^*$ is obtained by rotating ${}^T\hat{\mathbf{k}}$ of an appropriate angle. In figure 2.4 such angle is equal to $\pi/2$ but, for the problem at hand, the actual amplitude was chosen through the procedure later proposed in this section.

In the planar example just considered, kinematic singularities are avoided by modifying linear velocities and, in turn, the path. However, the same concept also applies for angular velocities, so that, for a 6 degrees of freedom manipulator, singular configurations can be avoided by changing the tool frame orientation instead of its path. To this purpose the orientation modifier proposes a candidate rotation in the following form

$$ {}^T\mathbf{k} := \theta\ {}^T\hat{\mathbf{k}} . \tag{2.41} $$

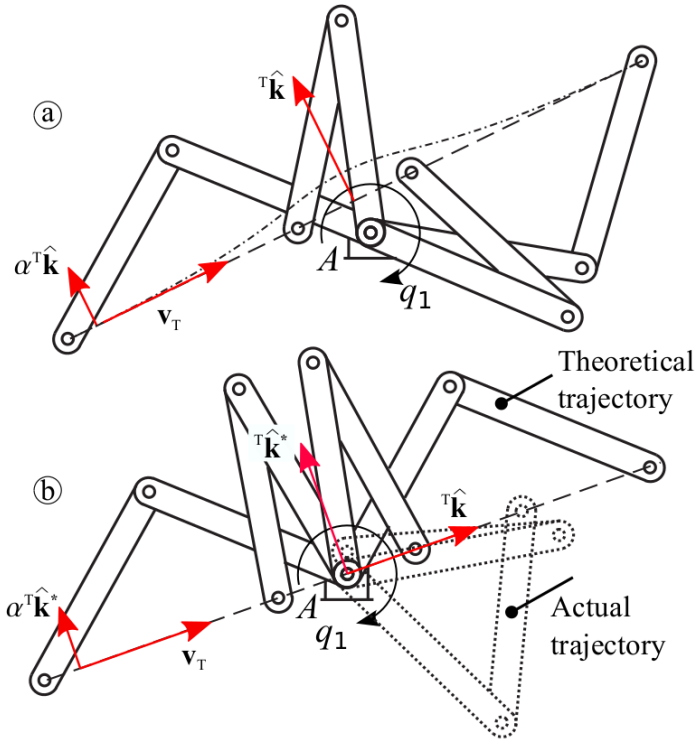


Figure 2.4: Case (a): the trajectory (dashed line) passes close to the singularity. The ellipsoid of manipulability indicates ${}^T\hat{\mathbf{k}}$ as a possible escape direction. Velocity $\alpha\,{}^T\hat{\mathbf{k}}$ is added to $\mathbf{v}_T$ in order to modify the path (dash-dotted line) thus reducing the joint speed. Case (b): the trajectory crosses the singularity. Theoretically, no link flip is required, but minor rounding problems always force it. As a consequence, ${}^T\hat{\mathbf{k}}^*$ must be used instead of ${}^T\hat{\mathbf{k}}$, which is suggested by the ellipsoid of manipulability, in order to escape from the singularity. In this planar example the singular configuration is avoided by modifying the linear velocity and, consequently, the cartesian path. The same concept applies for angular velocities and the tool frame orientation in a 6 DoF manipulator.

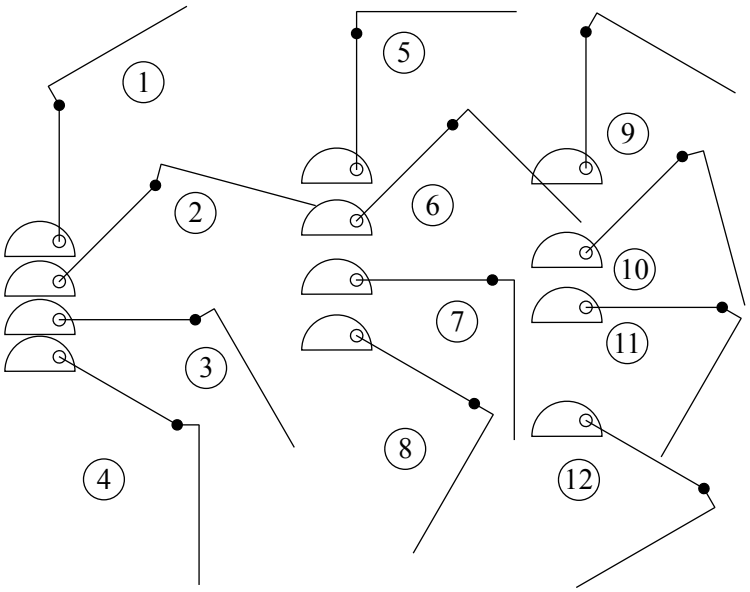


Figure 2.5: Schematic representation of the 12 poses assumed by links 2 and 3 for the experimental validation of the SAS.

2.4.2 Tuning procedure

The association of a proper value of θ to ${}^{T}\hat{\mathbf{k}}$ represents a complex problem, since no evident theoretical considerations are available for the synthesis of adequate analytical equations. For evident reasons, small values of θ are desirable. In the same way, the angular deviation from the nominal trajectory should last for a short period: curvilinear coordinates $s_a := s^* - d_a$ and $s_d := s^* + d_a$, at which the SAS must be respectively activated and deactivated, are very important. s^* indicates a point along the path which is located just before the singular configuration and d_a is the activation distance which must be kept as small as possible.

θ and d_a have been tuned through a procedure totally based on simulative tests. Its outcomes have been later verified on the actual manipulator. A set of wrist singular points, uniformly distributed in the workspace, were first selected by aligning the fourth and the sixth joint axes, i.e., by posing $q_5 = 0$, and, subsequently,

by assigning all possible combinations of q_2 and q_3 taken from the following sets: $q_2 \in \{-\pi/2\ -\pi/4\ \ 0\ \pi/6\}$, $q_3 \in \{-\pi/6\ 0\ \pi/6\}$. In facts, as shown in figure 2.5, the combinations of q_2 and q_3 were chosen so as to cover the whole workspace. Upper and lower bounds on q_2 and q_3 were imposed by the end-strokes of the actual manipulator.

A "star" of straight trajectories passing through the resulting 12 singular points was then generated so as to cover all possible directions in the 3D space. Joint variables q_1, q_4, and q_6 have no influence on the singularity analysis: the results obtained apply independently from their values. The tuning set is potentially composed by 4440 unfeasible trajectories (370 for each singular point) but, since some of them partially fall outside the workspace, the actual one contains 3110 cases. The following bounds have been assumed for velocities and accelerations ($i = 1,2,\ldots,6$): $\dot{q}_i^- = -10$ rad s^{-1}, $\dot{q}_i^+ = 10$ rad s^{-1}, $\ddot{q}_i^- = -25$ rad s^{-2}, and $\ddot{q}_i^+ = 25$ rad s^{-2}.

The tuning procedure starts by first choosing, for each trajectory of the tuning set, the proper rotation axis ${}^T\hat{\mathbf{k}}$ and, then, the trajectory is executed at the maximum speed (0.4 ms^{-1}) by activating the SAS and by assuming a constant value of θ for the whole segment: the procedure is repeated by progressively increasing θ until a feasible trajectory is obtained. At the end of the process, a threshold value $\overline{\theta}$ is associated to each feasible trajectory of the tuning set.

As early asserted, since tuning trajectories exactly cross singular configurations, ${}^T\hat{\mathbf{k}}$ must be perturbed w.r.t. to the one suggested by the ellipsoid of manipulability: the tuning procedure was repeated for different orientations of ${}^T\hat{\mathbf{k}}$ – which must always lie on the *xy*-plane of the tool-frame – trying to minimize the average value of all $\overline{\theta}$s: the best performances were achieved by adopting a rotation for ${}^T\hat{\mathbf{k}}$ in the range [0.8, 0.9] rad.

The acquired data highlighted, for each trajectory, a relationship between $\overline{\theta}$ and the derivative of q_5 w.r.t. to s, i.e., $q_5'(s^*) = |[dq_5(s)]/ds|_{s=s^*}$, where s^* indicates a position along the path located just before the singular configuration. Such relationship is shown in figure 2.6a: for each trajectory, a dot indicates the value of $\overline{\theta}$ associated to the corresponding $q_5'(s^*)$. Such information was used to define the following function

$$\overline{\theta}[q_5'(s^*)] := a + b \min\{\left|q_5'(s^*)\right|, c\} , \tag{2.42}$$

which output, obtained by assuming $a = 0.0147$, $b = 0.1033$, and $c = 1.6$ is shown in figure 2.6a by means of a red line. Coefficient a and b are the intercept and the slope coefficients of the leftmost linear segment, respectively. $\overline{\theta}$ is superiorly saturated by means of c. For the problem at hand, c was chosen so as to guarantee a maximum angular displacement equal to 0.18 rad (10.31 deg). It is worth to mention that smaller angular displacements may be imposed by reducing c, but lower travel speeds must be assumed in order to maintain high success rates. With the values chosen for the three coefficients, 91.0% trajectories lie below the red line and, consequently, are feasible.

The angular displacements obtained through (2.42) are excessive if used for $\|\mathbf{v}_T\|$ lower than 0.4 ms^{-1} and, consequently, $\overline{\theta}$ is subsequently downscaled according to the following equation

$$\theta := \overline{\theta}_0 + \frac{\overline{\theta} - \overline{\theta}_0}{0.4} \|\mathbf{v}_T\| . \tag{2.43}$$

The red lines in figure 2.6 show the output of (2.43) at different speeds for $\overline{\theta}_0 = 0.009$. $\overline{\theta}_0$ is not critical: it represents the minimum angular displacement to be introduced when the SAS is activated. It can be noticed that the percentage of points above the red lines, corresponding to unfeasible trajectories, decreases together with the speed. For $\|\mathbf{v}_T\| = 0.1$ ms^{-1} the success rate increases up to 96.8%. Evidently, $\|\mathbf{v}_T\| = 0.4$ ms^{-1} is a critical speed which strongly solicits some joints and which would potentially require higher values for θ.

The time interval during which the SAS modifies the nominal trajectory depends on d_a and must be kept as small as possible. To this purpose the tuning set was newly executed by evaluating θ according to (2.43) and by progressively reducing d_a until feasibility was lost. Simulations pointed out a relationship between $q_5'(s^*)$ and d_a: the higher $\left|q_5'(s^*)\right|$, the lower d_a. In the same way, a relationship between d_a and $\|\mathbf{v}_T\|$ was observed. Good success results were obtained by selecting d_a through the

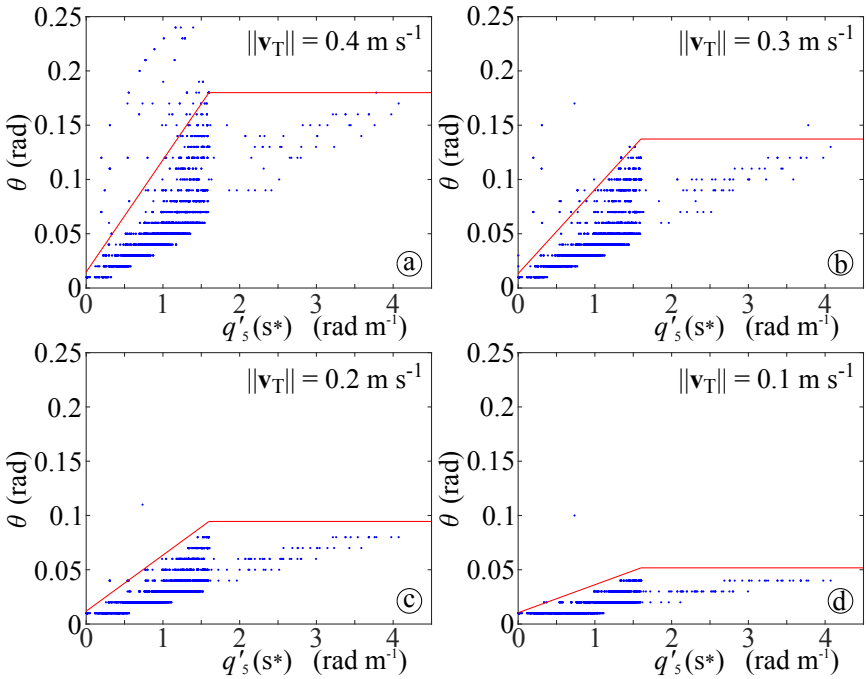


Figure 2.6: For each trajectory of the test-set a blue dot associates $q'_5(s^*)$ to the minimum value $\overline{\theta}$ which guarantees feasibility. Red lines represent the output of (2.43) and are used by the SAS for the evaluation of the tool-frame rotation angle.

following hyperbolic function $(d_a - \overline{d})(|q'_5(s^*)| - \overline{q}) = k$ or, equivalently,

$$d_a := \frac{\overline{d}\,|q'_5(s^*)| + k - \overline{q}\,\overline{d}}{|q'_5(s^*)| - \overline{q}} \tag{2.44}$$

with

$$\overline{d} := d_0 + \frac{d_1}{0.4}\,\|\mathbf{v}_T\|\ , \tag{2.45}$$

and where $k = 0.1$, $d_0 = -0.027$, $d_1 = 0.147$, and $\overline{q} = -0.8$. The output of (2.44) is represented by the red lines shown in figure 2.7. Coefficients k, $\overline{d}$, and $\overline{q}$ were tuned so as to bound as many samples as possible below the red line of figure 2.7a, i.e., the one corresponding to the maximum speed: k acts on the shape of the hyperbole, while $\overline{d}$ and $\overline{q}$ change its vertical and horizontal displacement, respectively. Then the

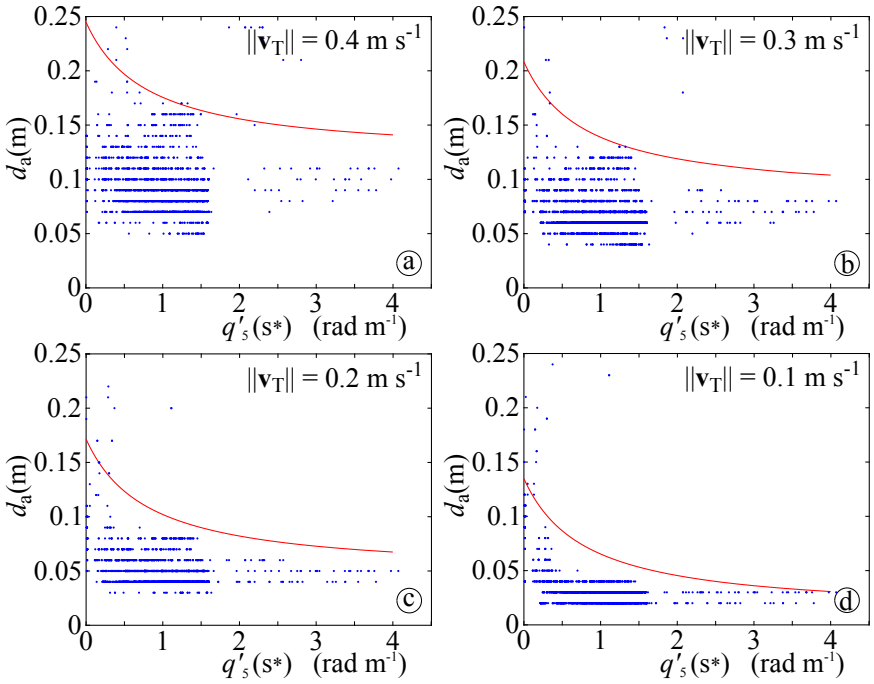


Figure 2.7: For each trajectory of the test-set a blue dot associates $q'_5(s^*)$ to the minimum value d_a which guarantees feasibility. Red lines represent the output of (2.44) and are used for the evaluation of the SAS activation distance.

obtained value of $\overline{d}$ is subsequently scaled down by means of (2.45), so as to account for smaller longitudinal speeds: obviously $d_0 + d_1$ must coincide with the value of $\overline{d}$ previously obtained. The subdivision between d_0 and d_1 is made by maximizing, for all possible speeds, the number of samples lying below the red lines.

2.5 Orientation synthesizer and equivalent bound evaluator

Once the singularity alarm has been raised, the orientation modifier proposes, by means of (2.41), a candidate rotation ${}^T\mathbf{k}$ between T and $\widetilde{T}$. Such rotation generically

reduces the speed of critical joints, but it does not guarantee that (2.18) and (2.19) are satisfied. For this reason, the orientation modifier is immediately followed by a nonlinear filtering stage, whose output $\widetilde{\theta}$ represents the best approximation of θ which is compatible with constraints (2.20) and (2.21). Details on the filter structure and its behavior are here omitted since they have been largely discussed in [47]. Conversely, the evaluation of equivalent bounds $\dot{\theta}^-$, $\dot{\theta}^+$, $\ddot{\theta}^-$, and $\ddot{\theta}^+$ and the equations required for the synthesis of the modified tool-frame orientation are proposed in the following.

The Orientation Synthesizer generates a modified trajectory by combining the nominal one with ${}^T\hat{\mathbf{k}}$ and with the output of the nonlinear filter, represented by signals $\widetilde{\theta}, \dot{\widetilde{\theta}}$, and $\ddot{\widetilde{\theta}}$. By virtue of the requirements, the linear components of the trajectory are not altered by the SAS, so that its output is only given by reference signals ${}^0_{\widetilde{T}}\mathbf{R}$, $\boldsymbol{\omega}_{\widetilde{T}}$, and $\boldsymbol{\alpha}_{\widetilde{T}}$. The modified tool-frame orientation can be obtained as follows

$$ {}^0_{\widetilde{T}}\mathbf{R} = {}^0_T\mathbf{R}\,{}^T_{\widetilde{T}}\mathbf{R}\,, $$

where ${}^0_T\mathbf{R}$ is associated to the nominal trajectory, while ${}^T_{\widetilde{T}}\mathbf{R}$ is the rotation matrix which expresses the angular displacement between T and $\widetilde{T}$. It is given by an axis-angle matrix in which the rotation axis is given by ${}^T\hat{\mathbf{k}}$, while the angle is given by $\widetilde{\theta}$.

Since ${}^T\hat{\mathbf{k}}$ is kept constant during transients, the relative angular speed between frames T and $\widetilde{T}$, described w.r.t. frame T can be expressed as follows

$$ {}^T\boldsymbol{\omega}_{\widetilde{T},T} = {}^T\hat{\mathbf{k}}\,\dot{\widetilde{\theta}} $$

and, consequently, the angular velocity of $\widetilde{T}$ can be posed in the following form

$$ \boldsymbol{\omega}_{\widetilde{T}} = \boldsymbol{\omega}_T + {}^0_T\mathbf{R}\,{}^T\boldsymbol{\omega}_{\widetilde{T},T} = \boldsymbol{\omega}_T + {}^0_T\mathbf{R}\,{}^T\hat{\mathbf{k}}\,\dot{\widetilde{\theta}}\,. \tag{2.46} $$

Finally, the angular acceleration can be obtained by differentiating (2.46) still bearing in mind that ${}^T\hat{\mathbf{k}}$ is constant. With a few algebraic manipulations it is finally possible to obtain the following equation

$$ \begin{aligned} \boldsymbol{\alpha}_{\widetilde{T}} &= \boldsymbol{\alpha}_T + {}^0_T\dot{\mathbf{R}}\,{}^T\hat{\mathbf{k}}\,\dot{\widetilde{\theta}} + {}^0_T\mathbf{R}\,{}^T\hat{\mathbf{k}}\,\ddot{\widetilde{\theta}} \\ &= \boldsymbol{\alpha}_T + \boldsymbol{\omega}_T \times {}^0_T\mathbf{R}\,{}^T\hat{\mathbf{k}}\,\dot{\widetilde{\theta}} + {}^0_T\mathbf{R}\,{}^T\hat{\mathbf{k}}\,\ddot{\widetilde{\theta}}\,. \end{aligned} \tag{2.47} $$

The remainder of this section is devoted to the synthesis of the equivalent bounds $\dot{\theta}^-, \ddot{\theta}^-, \dot{\theta}^+$, and $\ddot{\theta}^+$ which are required by the nonlinear filter.

The modified trajectory, by construction, avoids singularities, so that the Jacobian matrix can always be inverted. As a consequence, bearing in mind (2.22), joint velocities associated to the modified trajectory can be expressed as follows

$$\dot{\tilde{\mathbf{q}}} = \mathbf{J}_{\widetilde{T}}^{-1}\,\overline{\mathbf{v}}_{\widetilde{T}}\,, \tag{2.48}$$

where $\overline{\mathbf{v}}_{\widetilde{T}} = [\mathbf{v}_T^T\ \boldsymbol{\omega}_{\widetilde{T}}^T]^T$ is the generalized velocity associated to $\widetilde{T}$. According to the premises, its linear component coincides with the one of the nominal trajectory. By considering the same partitioning scheme proposed for (2.25), (2.48) can be rewritten as follows

$$\dot{\tilde{\mathbf{q}}} = \overline{\mathbf{J}}_{v_{\widetilde{T}}}\,\mathbf{v}_T + \overline{\mathbf{J}}_{\omega_{\widetilde{T}}}\,\boldsymbol{\omega}_{\widetilde{T}}\,. \tag{2.49}$$

Equation (2.49) is fundamental for finding the equivalent constraints for $\dot{\tilde{\theta}}$. By virtue of (2.49) and (2.46), bounds (2.18) can be posed into the following form

$$\dot{\mathbf{q}}^- \le \overline{\mathbf{J}}_{v_{\widetilde{T}}}(\tilde{\mathbf{q}})\,\mathbf{v}_T + \overline{\mathbf{J}}_{\omega_{\widetilde{T}}}\left[\boldsymbol{\omega}_T + {}^0_T\mathbf{R}^T\hat{\mathbf{k}}\,\dot{\tilde{\theta}}\right] \le \dot{\mathbf{q}}^+$$

or, analogously, bearing in mind (2.23) and (2.25), as follows

$$\dot{\tilde{\mathbf{q}}}^- \le \mathbf{a}\,\dot{\tilde{\theta}} \le \dot{\tilde{\mathbf{q}}}^+\,, \tag{2.50}$$

where

$$\dot{\tilde{\mathbf{q}}}^- := \dot{\mathbf{q}}^- - \mathbf{J}_{\widetilde{T}}^{-1}\,\overline{\mathbf{v}}_T, \tag{2.51}$$

$$\dot{\tilde{\mathbf{q}}}^+ := \dot{\mathbf{q}}^+ - \mathbf{J}_{\widetilde{T}}^{-1}\,\overline{\mathbf{v}}_T, \tag{2.52}$$

$$\mathbf{a} := \overline{\mathbf{J}}_{\omega_{\widetilde{T}}}\,{}^0_T\mathbf{R}^T\hat{\mathbf{k}}. \tag{2.53}$$

Any $\dot{\theta}$ which fulfills (2.50) generates feasible joint velocities. All the terms in (2.50)–(2.53) can be efficiently evaluated online. It must be noticed that some of them depend on the nominal trajectory, while others depend on the modified one.

Being **a** a vector, (2.50) can also be posed into the following scalar form

$$\dot{\tilde{q}}_i^- \le a_i\,\dot{\tilde{\theta}} \le \dot{\tilde{q}}_i^+\,, \quad i = 1,2,\ldots,6,$$

where $\dot{\tilde{q}}_i^-$, $\dot{\tilde{q}}_i^+$, and a_i indicate the components of $\dot{\tilde{\mathbf{q}}}^-$, $\dot{\tilde{\mathbf{q}}}^+$, and $\mathbf{a}$, respectively. Since all inequalities must be simultaneously satisfied, it can be asserted that any feasible $\dot{\tilde{\theta}}$ must fulfill condition (2.20) where $(i = 1,2,\ldots,6)$

$$\dot{\theta}^- = \begin{cases} \max\limits_i \left\{ \dot{\tilde{q}}_i^- / a_i \right\} & \text{if} \quad a_i > 0 \\ \max\limits_i \left\{ \dot{\tilde{q}}_i^+ / a_i \right\} & \text{if} \quad a_i < 0 \end{cases}, \tag{2.54}$$

$$\dot{\theta}^+ = \begin{cases} \min\limits_i \left\{ \dot{\tilde{q}}_i^+ / a_i \right\} & \text{if} \quad a_i > 0 \\ \min\limits_i \left\{ \dot{\tilde{q}}_i^- / a_i \right\} & \text{if} \quad a_i < 0 \end{cases}. \tag{2.55}$$

The equivalent constraints for $\ddot{\tilde{\theta}}$ can be derived with similar manipulations. More precisely, the generalized acceleration of $\widetilde{T}$, obtained by differentiating (2.22), can be expressed as follows

$$\bar{\mathbf{a}}_{\widetilde{T}} = \dot{\mathbf{J}}_{\widetilde{T}} \dot{\tilde{\mathbf{q}}} + \mathbf{J}_{\widetilde{T}} \ddot{\tilde{\mathbf{q}}},$$

and then it can be posed, by also considering (2.48), into the following form

$$\ddot{\tilde{\mathbf{q}}} = \mathbf{J}_{\widetilde{T}}^{-1} \left[\bar{\mathbf{a}}_{\widetilde{T}} - \dot{\mathbf{J}}_{\widetilde{T}} \dot{\tilde{\mathbf{q}}} \right] = \mathbf{J}_{\widetilde{T}}^{-1} \left[\bar{\mathbf{a}}_{\widetilde{T}} - \dot{\mathbf{J}}_{\widetilde{T}} \mathbf{J}_{\widetilde{T}}^{-1} \bar{\mathbf{v}}_{\widetilde{T}} \right]. \tag{2.56}$$

Bearing in mind the partitioning scheme suggested in (2.25) and by also recalling that $\bar{\mathbf{a}}_{\widetilde{T}} = [\mathbf{a}_T^T \ \boldsymbol{\alpha}_{\widetilde{T}}^T]^T$, it is possible to write

$$\ddot{\tilde{\mathbf{q}}} = \bar{\mathbf{J}}_{v_{\widetilde{T}}} \mathbf{a}_T + \bar{\mathbf{J}}_{\omega_{\widetilde{T}}} \boldsymbol{\alpha}_{\widetilde{T}} - \mathbf{J}_{\widetilde{T}}^{-1} \dot{\mathbf{J}}_{\widetilde{T}} \mathbf{J}_{\widetilde{T}}^{-1} \bar{\mathbf{v}}_{\widetilde{T}},$$

which, in turn, can be written, with the aid of (2.47), as follows

$$\begin{aligned} \ddot{\tilde{\mathbf{q}}} = {} & \bar{\mathbf{J}}_{v_{\widetilde{T}}} \mathbf{a}_T + \bar{\mathbf{J}}_{\omega_{\widetilde{T}}} \left\{ \boldsymbol{\alpha}_T + \boldsymbol{\omega}_T \times {}_T^0\mathbf{R}^T \hat{\mathbf{k}} \dot{\tilde{\theta}} + {}_T^0\mathbf{R}^T \hat{\mathbf{k}} \ddot{\tilde{\theta}} \right\} \\ & - \mathbf{J}_{\widetilde{T}}^{-1} \dot{\mathbf{J}}_{\widetilde{T}} \mathbf{J}_{\widetilde{T}}^{-1} \bar{\mathbf{v}}_{\widetilde{T}}. \end{aligned} \tag{2.57}$$

By substituting (2.57) into (2.19), the following inequalities can be obtained after a few manipulations

$$\ddot{\tilde{\mathbf{q}}}^- \leq \mathbf{a} \ \ddot{\theta} \leq \ddot{\tilde{\mathbf{q}}}^+, \tag{2.58}$$

where

$$\begin{aligned}\dddot{\tilde{\mathbf{q}}}^{-} &:= \ddot{\mathbf{q}}^{-} - \mathbf{J}_{\tilde{T}}^{-1}\left[\bar{\mathbf{a}}_T - \dot{\mathbf{J}}_{\tilde{T}}\mathbf{J}_{\tilde{T}}^{-1}\bar{\mathbf{v}}_{\tilde{T}}\right] \\ &\quad - \bar{\mathbf{J}}_{\omega_{\tilde{T}}}\left\{\boldsymbol{\omega}_T \times {}^{0}_{T}\mathbf{R}^{T}\hat{\mathbf{k}}\dddot{\tilde{\theta}}\right\}, \\ \dddot{\tilde{\mathbf{q}}}^{+} &:= \ddot{\mathbf{q}}^{+} - \mathbf{J}_{\tilde{T}}^{-1}\left[\bar{\mathbf{a}}_T - \dot{\mathbf{J}}_{\tilde{T}}\mathbf{J}_{\tilde{T}}^{-1}\bar{\mathbf{v}}_{\tilde{T}}\right] \\ &\quad - \bar{\mathbf{J}}_{\omega_{\tilde{T}}}\left\{\boldsymbol{\omega}_T \times {}^{0}_{T}\mathbf{R}^{T}\hat{\mathbf{k}}\dddot{\tilde{\theta}}\right\},\end{aligned}$$

while $\mathbf{a}$ is still defined according to (2.53). Vectors $\dddot{\tilde{\mathbf{q}}}^{-}$ and $\dddot{\tilde{\mathbf{q}}}^{+}$ can be evaluated from the knowledge of the nominal trajectory, the modified one, and the acceleration bounds. The equivalent constraints for $\dddot{\tilde{\theta}}$ can be obtained through manipulations similar to the ones considered for $\dot{\tilde{\theta}}$. In particular, the following limits can be derived from the scalar representation of (2.58) ($i = 1, 2, \ldots, 6$)

$$\ddot{\theta}^{-} = \begin{cases} \max\limits_{i}\left\{\dddot{\tilde{q}}_i^{-}/a_i\right\} & \text{if} \quad a_i > 0 \\ \max\limits_{i}\left\{\dddot{\tilde{q}}_i^{+}/a_i\right\} & \text{if} \quad a_i < 0 \end{cases}, \tag{2.59}$$

$$\ddot{\theta}^{+} = \begin{cases} \min\limits_{i}\left\{\dddot{\tilde{q}}_i^{+}/a_i\right\} & \text{if} \quad a_i > 0 \\ \min\limits_{i}\left\{\dddot{\tilde{q}}_i^{-}/a_i\right\} & \text{if} \quad a_i < 0 \end{cases}. \tag{2.60}$$

By comparing the SAS proposed in this chapter with the previous version proposed in [45] it is possible to infer that the total computational burden of the new strategy is certainly much lower since the new algorithm is more compact and the nonlinear filtering stage now manages a single signal instead of three. However, the most relevant improvement in terms of efficiency was achieved for the computation of the equivalent bounds, which do no more require the online solution of a linear programming problem.

Chapter 3

Results

The system proposed was tested, at first, by means of simulated experiments. Then, the SAS was implemented on the control system of a real anthropomorphic manipulator for the execution of an experimental test campaign. Section 3.1 focuses on the performances obtained through simulations, while section 3.2 reports the outcomes of the experiments carried out on the Comau SmartSix-6.14 robot.

3.1 Simulation results

The early discussed tuning procedure only accounts for trajectories passing through singularities. In order to verify if the obtained parameters can be adopted to manage trajectories which do not exactly cross singularities, they were tested by also considering alternative scenarios. In particular, for each one of the 12 test configurations, 4 additional points were placed 10^{-3} m far from the singularity and, in each of them, a "star" of 370 trajectories was generated. The experiment was then repeated by considering 4 more points located $2 \cdot 10^{-3}$ m far from the singularity, and so on. The obtained success rates are listed in table 3.1. The tuning procedure, which is very fast and easily adaptable to alternative manipulators or working conditions, always guarantees success rates higher than 82%. Evidently, the best performances are achieved for trajectories crossing singularities, since they were used for the system

Table 3.1: Success rates obtained through the tuning procedure proposed in section 2.4.2. d indicates the distance between trajectories and singular points.

	$\|\mathbf{v}_T\|$ [m s^{-1}]			
d [m]	0.1	0.2	0.3	0.4
$0 \cdot 10^{-3}$	97.7%	96.8%	95.1%	92.3%
$1 \cdot 10^{-3}$	91.1%	87.8%	85.4%	82.7%
$2 \cdot 10^{-3}$	94.2%	91.0%	88.2%	85.9%
$5 \cdot 10^{-3}$	96.2%	94.7%	92.7%	90.7%
$10 \cdot 10^{-3}$	96.8%	95.3%	94.0%	91.9%

tuning. However for primitives passing close to singular points, the success rate drop is limited.

It must be pointed out that 100% success rate can never be reached for several reasons. Many configurations in figure 2.5 (see for example 4, 7, 8, 11, and 12) admit trajectories passing close or through shoulder singularities, which are not managed by the SAS. Furthermore, some trajectories are almost singular everywhere and, finally, some others are characterized by two singular configurations.

3.2 Experimental results

The SAS was tested by means of a Comau Smart SiX 6-1.4 anthropomorphic manipulator. An external Linux-RTAI PC was used to generate the trajectories and to process them with the SAS at a sample rate equal to $2 \cdot 10^{-3}$ s. The obtained reference signals are sent, with the same sample rate, to the feedback control loops of the robot controller through a real-time Ethernet connection.

Two experiments were performed on the real manipulator, for both of them the following bounds were assumed for the joint velocities and accelerations ($i = 1, 2, \ldots, 6$): $\dot{q}_i^- = -10$ ms^{-1}, $\dot{q}_i^+ = 10$ ms^{-1}, $\ddot{q}_i^- = -25$ ms^{-2}, and $\ddot{q}_i^+ = 25$ ms^{-2}. The tool-frame linear speed was posed equal to $\|\mathbf{v}_T\| = 0.4$ ms^{-1}.

The first experiment concerned the execution of 100 random trajectories lying

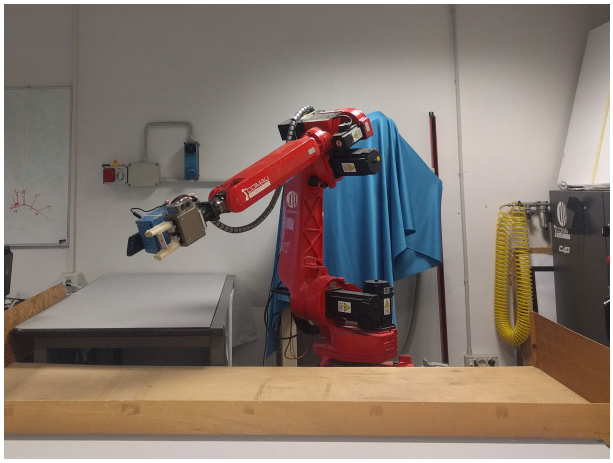


Figure 3.1: The presence of desks limits the workspace of the Comau manipulator used for the experiments.

on the vertical plane $y = 0.83$ m and passing through a singular point placed in $\overline{\mathbf{p}}_T = [0\ 0.83\ 1.077]^T$ with orientation $\mathbf{\Phi}_{T,0} = [\pi/2\ \pi/2\ 0]^T$ expressed through the ZYZ RPY notation, Cartesian coordinates are expressed in meters, while angles, expressed in radiant, were adopted for the tool-frame orientation. According to the requirements, the trajectory generator block is totally disjointed from the SAS one. This latter only receives, at each sample time, reference signals $\mathbf{\Phi}_{T,0}(t)$, $\mathbf{p}_T(t)$, $\boldsymbol{\omega}_T(t)$, $\mathbf{v}_T(t)$, $\boldsymbol{\alpha}_T(t)$, and $\mathbf{a}_T(t)$: any decision is taken in real time by the SAS on the basis of those signals. As specified in Section 2.2, $\mathbf{p}_T(t)$, $\mathbf{v}_T(t)$, and $\mathbf{a}_T(t)$ are sent, unmodified, to the inverse kinematics block – so as to guarantee with certainty that the nominal-path is not changed – while the tool-frame orientation is handled by the SAS. The maximum angular deviation from the nominal trajectories was equal to $\widetilde{\theta}_{max} = 5.87$ deg. All trajectories satisfied the imposed velocity and acceleration constraints.

The second experiment was relative to the execution of a "star" of straight trajectories passing through the singular points used for the tuning phase. As shown in figure 3.1, the manipulator workspace is limited by the presence of desks. As a consequence, only a part of the simulated tests were replicated in the real environment and,

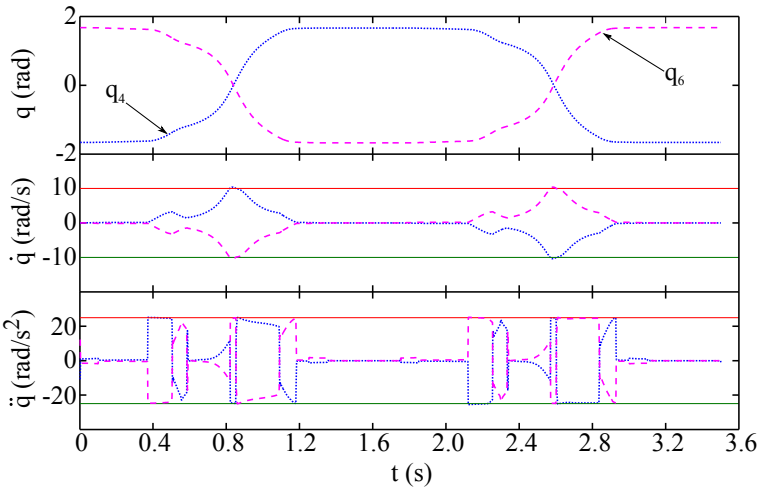


Figure 3.2: The velocity and acceleration signals for joints 4 and 6 associated to two horizontal trajectories passing through $\overline{\mathbf{p}}_T$: the given bounds are fulfilled.

more precisely, the ones corresponding to configurations 1, 2, 5, and 9 of figure 2.5. The experimentally acquired success rates were even better than simulated ones, as proved by table 3.2. The reason of such performances is that unmanageable configurations occur more frequently in areas which are precluded to the real manipulator. All the trajectories that were classified "manageable" by the algorithm were actually executed with the aid of the SAS and feasibility was never lost, i.e., all trajectories were feasible w.r.t. the imposed velocity and acceleration constraints. Some of them are particularly critical since they are close to a workspace border or to an elbow singularity, which is not managed by the SAS.

Figure 3.2 shows a typical transient for joints 4 and 6, i.e., the most solicited ones. The figure proves that, differently from other approaches in the literature, the proposed strategy does not simply reduce joint velocities and accelerations, but it explicitly bounds them between given limits. Similar results were obtained for all the trajectories of the test set. Another detail, still pointed out by figure 3.2, concerns the exchange of position between joints 4 and 6: it occurs for all the trajectories and it is a direct consequence of Proposition 2. In order to prove such assertion, let us

Table 3.2: Success rates obtained with the actual manipulator. d indicates the distance between trajectory and singular point.

	$\|\mathbf{v}_T\|$ [m s^{-1}]			
d [m]	0.1	0.2	0.3	0.4
0	96.6%	94.9%	94.7%	93.2%
$1 \cdot 10^{-3}$	95.1%	93.9%	92.9%	91.1%
$2 \cdot 10^{-3}$	96.3%	95.0%	93.9%	92.5%
$5 \cdot 10^{-3}$	97.3%	96.5%	95.4%	94.2%
$10 \cdot 10^{-3}$	97.9%	97.0%	96.5%	95.2%

consider a simplified representation of (2.14) and (2.16) obtained for very common operating conditions. Many applications do not require the tool-frame rotation, i.e., $\hat{\mathbf{x}}_6$ and $\hat{\mathbf{y}}_6$ can be assumed constant together with $\hat{\mathbf{z}}_6$, so that $\boldsymbol{\omega}_T = \mathbf{0}$. Furthermore, the SAS strategy actuates the angular displacement between nominal and modified tool-frame when the system is sufficiently far from the singularity: close to critical configurations it can be reasonably assumed ${}^T\boldsymbol{\omega}_{\tilde{T},T} = \mathbf{0}$ and, additionally, $\sin(\theta_5) \simeq \theta_5$ and $\cos(\theta_5) \simeq 1$. Bearing in mind such premises, (2.14) and (2.16) simplify as follows

$$\dot{\theta}_4 = \dot{\theta}_1 \mathrm{c}_{23} + \frac{\dot{\theta}_1 \mathrm{c}_4 \mathrm{s}_{23} + (\dot{\theta}_2 + \dot{\theta}_3)\mathrm{s}_4}{\theta_5}, \tag{3.1}$$

$$\dot{\theta}_6 = -\frac{\dot{\theta}_1 \mathrm{c}_4 \mathrm{s}_{23} + (\dot{\theta}_2 + \dot{\theta}_3)\mathrm{s}_4}{\theta_5} \tag{3.2}$$

where

$$\begin{aligned}
&\mathrm{c}_4 = \cos(\hat{\theta}_4 + \theta_4) = \cos(q_4),\\
&\mathrm{s}_4 = \sin(\hat{\theta}_4 + \theta_4) = \sin(q_4),\\
&\mathrm{c}_{23} = \cos(\hat{\theta}_2 + \hat{\theta}_3 + \theta_2 + \theta_3) = \cos(q_2 + q_3),\\
&\mathrm{s}_{23} = \sin(\hat{\theta}_2 + \hat{\theta}_3 + \theta_2 + \theta_3) = \sin(q_2 + q_3).
\end{aligned}$$

Close to the singularity $\theta_5 \simeq 0$, so that from (3.1) and (3.2) it is possible to infer that

$$\dot{\theta}_4 \simeq \frac{\dot{\theta}_1 c_4 s_{23} + (\dot{\theta}_2 + \dot{\theta}_3) s_4}{\theta_5} = -\dot{\theta}_6, \tag{3.3}$$

i.e., the velocities of joints 4 and 6 are each other opposite so that the positions of the corresponding joints swap. It is important to stress that the simplifications were only introduced in order to propose a compact representation of (2.14) and (2.16), but (3.3) is still valid even when they do not apply.

For which concerns the performances, the average computational time, obtained with an Intel Core2 Duo PC running at 3.0GHz, was equal to $4.211 \cdot 10^{-5}$ s. It typically spans in the range $[2.800 \cdot 10^{-5}, 1.690 \cdot 10^{-4}]$ s: evaluation times are, roughly, four times smaller than the ones obtained in [45] and are plenty compatible with the manipulator sample time ($2 \cdot 10^{-3}$ s).

Part II

An online jerk-bounded velocity planner for autonomous vehicles with safety requirements

Introduction to Part II

In autonomous warehouses several independent agents share their workspace. In fact, autonomous guided vehicles directly interact with human-operated forklifts and with other coworkers. The collaboration between humans and machines is fundamental for the effectiveness of industrial plants, but poses several problems which must be consequently managed. First of all, the safeness of human workers and of the overall plant must be ensured. Serious security issues may raise in autonomous warehouses, so that appropriate rules are imposed by the laws for their avoidance. For example, for safety reasons, autonomous Laser Guided Vehicles (LGV) are typically equipped with certified safety laser sensors and only move along pre-specified path segments. Additionally to the safety requirements, industrial plants must also account for possible efficiency concerns since, evidently, productivity must be maximized. The efficiency requirement is antithetic with respect to safeness. In fact, in many applications LGVs speeds are kept limited for safety reason.

A possible method, which allows overcoming such restrictions, is represented by the use of smart velocity planners. More specifically, such planners should own the following three properties:

a) real-time requirements. As anticipated, LGVs do not follow a fixed path but, conversely, paths are composed by combining variable sequences af short segments. For such reason, the velocity profile must be evaluated online during the motion of the vehicle;

b) jerk bounded: LGVs often carry huge loads. By bounding the longitudinal jerk between specified limits, smooth trajectories can be obtained, so as to reduce

both the vehicle and load solicitations;

c) variable velocity bounds: safety is the first requirement that must be met. It will be shown in the thesis that such target can be achieved by handling variable velocity bounds for the longitudinal motion.

The approach proposed in the thesis satisfies all these requirements. For the first time in the literature, safety constraints are directly considered in the optimization problem which must be solved for the trajectory generation. Even if the approach proposed is based on a heuristic procedure, it always returns feasible solutions whose traveling times are comparable with the ones obtained through a nonlinear solver. Conversely, nonlinear solvers show evaluation times which do not meet the real-time requirements. Alternative real-time methods appeared in the literature do not satisfy simultaneously the three requirements proposed above. In fact, often jerk is only minimized, but it is not specifically bounded, or velocity bounds are assumed constant.

This work has been supported in part by the European Project SAFERUN (Secure And Fast rEal-time planneR for aUtoNomous vehicles) in the context of ECHORD++ (The European Coordination Hub for Open Robotics Development) financed in the framework of the FP7 EU program. The collaboration between universities and major companies was expected as a target of the project, in fact the obtained results has been achieved through the partnership among University of Parma, Elettric80 and PreGel.

The second part of the thesis is organized as follows. Chapter 4 illustrates the state of the art for which concerns the velocity planners proposed in the literature and proposes a brief survey on the existing safety laws. Chapter 5 describes the proposed algorithm. Two versions have been implemented. The first one is specifically conceived for real warehouses, since it provides sub-optimal solutions which limit vehicle stresses. Conversely, the second one aims at obtaining a very efficient velocity profile, in order to make it possible to compare its performances with the ones obtained through the use of a nonlinear solver. Finally, chapter 6 reports the simulated and the experimental results. The proposed algorithm has been implemented and tested in a real warehouses: part of the experimental results directly descends from the statistic of the industrial plant.

Chapter 4

State of the art

LGVs are largely used in autonomous warehouses for the efficient transportation of huge loads. As asserted in [51], vehicles used in industrial contexts must be largely autonomous: they must localize themselves, follow the assigned path, and manage emergency situations. Hence the velocity planners cover a central role in order to manage the concurrent needs of a warehouse, they must maximize the plant productivity while simultaneously assuring the safeness of other vehicles and, even more important, of human coworkers which share the workspace with the automated LGVs. Section 4.1 analyzes the state of the art of the existing velocity planner, while a brief survey relative to the existent safety regulations is presented in section 4.2.

4.1 Related works

Working spaces for humans and machines, which have been historically kept separate for safety reasons, progressively tend to overlap due to the need of maximizing the plant productivity. For example, the separation between the two worlds has almost disappeared in automatic warehouses, where coworkers directly interact with LGVs for supervision and maintenance reasons: differently from other industrial contexts, completely disjointed working spaces cannot be conceived at all. In fact, in large warehouses vehicles maintenance is carried out by avoiding plant stops for productive

reasons: in case of failures, human operators necessarily enter the vehicles workspace for the required restorations. Additionally, load and transport operations which cannot be managed by LGVs are handled through human-operated forklifts, which naturally share the workspace with the autonomous ones.

Working conditions in LGV plants raise serious hazard issues which are handled at different levels. First of all, rigid rules are imposed to the vehicles motion. For example, the plant layout is defined in advance and cannot be modified, so that human coworkers become used of the LGVs habits and naturally tend to leave free any space which is used by the machines. Operators who need to enter zones crossed by trajectories are spontaneously inclined to pay attention. For the same reason, vehicles must never abandon the assigned trajectories, so as to avoid unexpected, and consequently dangerous, situations. This implies that, in the presence of unexpected obstacles along the route, which may be objects or coworkers, vehicles must be safely stopped in order to avoid impacts.

The motion safety problem has also been widely discussed by the scientific literature, [52] and [53]. A classification of the possible hazard situations, and of the techniques that can be used for their management, is proposed in [54]. The survey paper classifies the known approaches into three classes on the basis of the obstacle typology and of the time horizon considered. The very first approaches managed safety as a problem of obstacle avoidance: collisions were prevented by acting on the path and/or on the speed of the autonomous vehicle. Some algorithms handled obstacles on the basis of position information [55–57], while others took decisions on the basis of velocity considerations. Among them, it is worth citing [58] whose proposed method was based on the so-called “Velocity Obstacles” (VO). The approach based on VO identifies dynamically a set of speeds that must not be assumed in order to avoid collisions, such concept was at first introduced in [58]. Subsequently a probabilistic version has been proposed in [59] and the same concept was further extended in [60], so as to consider real-time contexts and to account for multi-agent systems. More recently the methodology has been revised both in [61], so as to account for the presence of kinematic constraints, and in [62], where the concept of Acceleration-Velocity Obstacles was proposed in order to guarantee the continuity of the vehicle

speed.

The above mentioned strategies have a common denominator: the position and the velocity of the moving obstacles are supposed to be analytically known. In real contexts, such hypothesis is often inadmissible since autonomous vehicles have limited sensing capabilities: obstacles are detected only when they are below a given distance from the sensor. An alternative method, which overcomes the limitations of the VO approaches, is based on the so called "Inevitable Collision States" (ICS). T. Fraichard and H. Asama in [63] were the first to introduce such concept and it is used to indicate motion states that must be necessarily skipped in order to avoid collisions. The initial approach to the problem was totally deterministic and based on the knowledge of a model of the dynamic environment. Later, an efficient algorithm was developed [64] in order to make ICSs suitable for practical applications: indeed, the characterization of an ICS is normally difficult due to its computational complexity. The original concept was also revised in [65] by introducing a probabilistic formulation of the ICS: the motion is planned on the basis of a probabilistic estimation of its safety. Originally the ICS concept was conceived for the car-like model, then, in the literature, it has also been adopted for the management of other vehicles. For example, in [66] ICSs have been reformulated in order to consider a motorcycle interacting with some cars in a dynamic environment. The approach used for the safety management in LGV based plants can be classified among the other ICS methods. Each LGV is equipped with laser sensors which inspect a restricted area around the vehicle: a potential obstacle is revealed only when it enters the sensors field.

In addition to the safety regulations, it must be considered that LGVs routes are obtained by composing elementary curves, which are planned when the plant is designed. Such choice is motivated by several reasons, among them it is worth citing the traffic management. Routes can be modified at run time by the plant supervisor, which selects the most proper path sequences depending on the tasks that must be accomplished. A direct consequence of such operating mode is that velocity references must be synthesized in real time on the basis of the operating conditions, detailed information will be give in section 5.1. For such reason, vehicles trajectories are always planned according to the path-velocity decomposition concept early proposed in [55]

and later revised in [56].

LGV plants have different concurrent needs, for example safety is antithetic with respect to productivity. Generated trajectories must guarantee the safety of coworkers but they must also be minimum-time, compatibly with the assigned constraints. Furthermore, it can be easily verified that the safety constraints impose variable velocity bounds, such limits need to be fulfilled and they have a great impact on the planner complexity. Moreover, one may desire to obtained jerk-limited trajectories, in fact LGVs are used for the displacement of huge loads and smooth trajectories can reduce vehicles and loads stresses.

In summary, a velocity planner for LGVs used in warehouses must have the following properties:

a) real-time capabilities;

b) the capability to handle jerk constraints;

c) the capability to handle position and time dependent velocity bounds.

Earlier real-time planners proposed in the literature can handle problems characterized by constant constraints or problems admitting, at most, second order constraints [67]. Several real-time planners have been recently proposed in the literature, but they are all suited to generate trajectories admitting constant bounds on speeds, accelerations, and jerks [8, 9, 68]. Problems concerning variable-speed constraints have already been addressed in the literature especially for applications concerning industrial manipulators [33, 69]. Mobile vehicles have been addressed at the beginning in [67] or, in more recent works, in [70–72], but up to now jerk limits have been neglected mainly because of the computational burden they would introduce.

Table 4.1 highlights the compatibility of different velocity planners with the proposed LGV application. Compared to the number of planners proposed in the past, the list is not exhaustive: it only includes the ones with real-time or almost real-time characteristics. As shown in table 4.1, the sole approach [84] fulfills all the required properties but, being based on numerical integration methods, shows computational times which may be too high depending on the application. Similarly, the technique

Table 4.1: Properties of some velocity planners proposed in the literature: a) real-time capabilities; b) the capability to handle jerk constraints; c) the capability to handle position and time dependent velocity bounds. "*" indicates methods which minimize but do not bound jerk.

	(a)	(b)	(c)
[33, 69, 73]			✓
[74–77]	✓	✓*	
[70–72, 78]	✓		✓
[7–9, 68, 79–83]	✓	✓	
[84]	✓	✓	✓
[85]	✓	✓*	✓

considered in [85] is not completely suited for the proposed application because jerk is minimized but it is not bounded.

The previous discussion makes it possible to evince that all mentioned velocity planners are not suited for LGV plants, bearing also in mind that they do not take into account the safety problem.

4.2 Safety management in LGV plants

Automated plants in which humans and machines share the same working areas may give rise to safety concerns and, consequently, they must comply with apposite rules issued by specific committees like the American ITSDF (Industrial Truck Standards Development Foundation) and the European ISO (International Standards Organization). In particular, the directives which refer to Automated Guided Vehicles (AGV) are respectively specified in the "ANSI/ITSDF B56.5-2012" document [86] for the ITSDF committee and in the two documents "EN ISO 1525:1997" [87] and "EN ISO 1526:1997+A1:2008" [88] for the ISO committee. The ANSI and the ISO guidelines are very similar: the first parts of both directives are dedicated to pose the safety requirements for automatic guided industrial vehicles, while the second ones con-

cern automated functions for manned industrial vehicles. Such international directives pose general rules and only provide a set of guidelines to be followed. In details, both directives agree that plants must be supervised by competent technicians and trained operators who have to move carefully in the areas shared with vehicles, since hazard situations cannot be avoided only by mechanical means. For what concerns the stopping distance the "ANSI/ITSDF B56.5-2012" directive states "The determination of the vehicle's stopping distance (...) depends on many factors, (...). The prime consideration is that the braking system in conjunction with the detection system and the response time of the safety control system shall cause the vehicle to stop prior to impact between the vehicle structure (...) and an obstruction (...)." The same directive also provides an exception which is at the basis of the safety management techniques used for LGVs: "Although the vehicle braking system may be performing correctly and as designed, it cannot be expected to function as designed and specified (...) should an object suddenly appear in the path of the vehicle and within the designed safe stopping distance." In practice, this implies that safety systems must be conceived to promptly react and stop vehicles in case of static obstacles, while in case of moving ones which suddenly enter the LGV workspace, the kinetic energy must be rapidly lowered, so as to minimize possible consequences. Such concept is implemented by the ASTM F3265-17 [89] directives, which propose a methodology for testing the vehicles reactions in case of detection of moving obstacles.

Standards posed by ISO and ANSI have been acknowledged by local legislations. For example, in Europe all machineries must be produced and installed complying with the European Machinery Directive 2006/42/EC. The same directive has been adopted by each single country of the European Community through local laws, e.g., in Italy through Legislative Decree 27.01.2010, no. 17 and through the Legislative Decree 9.04.2008, no. 81. This latter states: "If work equipment is moving in a work area, circulation rules must be established (...); Organizational measures must be taken in order to prevent workers on foot entering the area of operation of the self-propelled work equipment. If the presence of workers on foot is necessary for the proper execution of the work, appropriate measures must be taken to prevent workers from being injured by the equipment."

(a) (b)

Figure 4.1: On board safety laser sensors (yellow circles) and emergency stop devices (cyan ellipsoids) of a LGV (Courtesy of Elettric80).

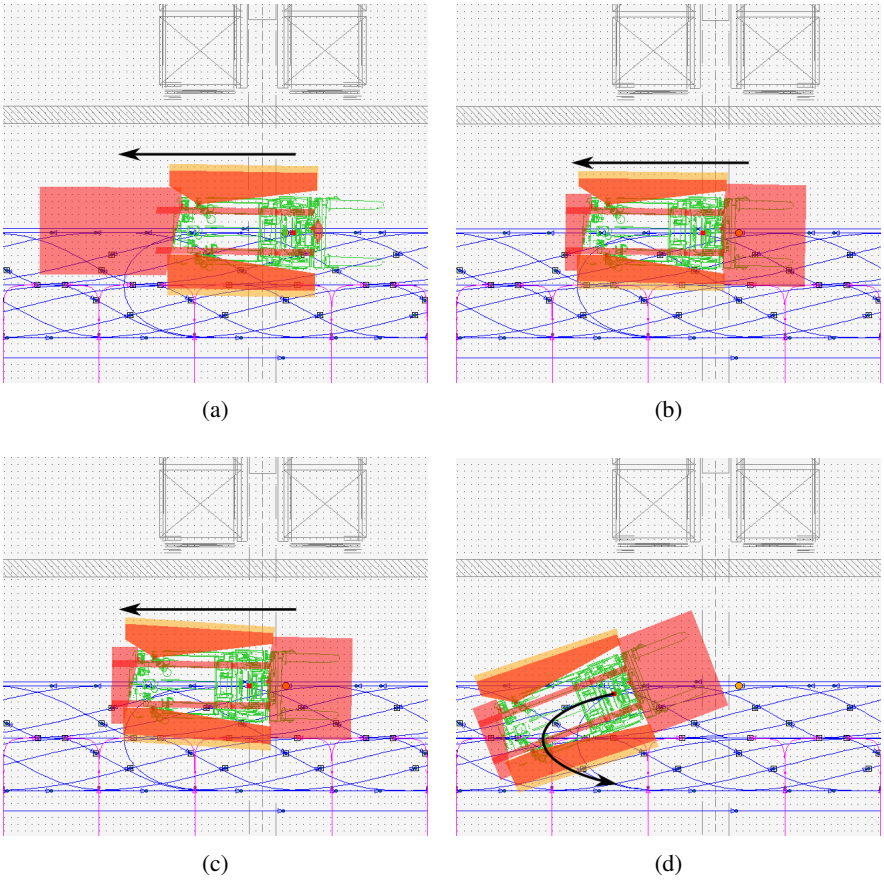
In order to comply with the safety directives, LGVs used in automated warehouses are equipped with certified Safety Laser Scanners (SLS), which operate similarly to standard scanners but provide a different output. Indeed, common lasers return the distances from possible obstacles measured along the directions scanned by the laser beam. Starting from such information, it is potentially possible to create a customized algorithm which reconstructs the vehicle surroundings, thus allowing a very accurate management of the safety concerns. However, any custom algorithm directly implemented by LGV manufacturers should undergo a long and expensive certification process, so that companies always adopt an alternative solution. More precisely, the market offers certified SLSs which are internally equipped with an algorithm for the safety management: users specify the areas that must be kept under control and the sensor verifies if they are free from obstacles. Practically, the sensor output is binary, i.e., the area is clear or not, while obstacle positions are not provided at all. The advantage of this approach is that safety algorithms are directly certified by sensors manufactures, thus relieving LGV producers from any liability. The draw-

back of such solution is that obstacle positions are unknown: obstacles, if detected, are somewhere within the programmed safety area.

LGVs are equipped with several certified SLSs. At least four laser scanners are installed by assuming the classic configuration shown in figure 4.1: sensor 1 scans the front of the vehicle, sensor 2 is placed on its rear, while scanners 3 and 4 inspect its lateral sides. Mentioned devices are installed under the vehicle bodywork. Such position allows detecting obstacles up to an elevation roughly equal to $2 \cdot 10^{-2}$ m. Objects located over such threshold are sensed through further scanners indicated in the figure with 5 and 6, which are located on the top of the vehicle. Additionally, each LGV is provided with several manual emergency devices, like the ones indicated in figure 4.1a by 7 and 8.

A limited number of safety areas can be programmed offline on each SLS of the vehicle. They are designed depending on the plant and on the vehicle characteristics. Shapes of the safety areas cannot be changed at runtime: the control system can only select, among them, the most appropriate one for each segment. The number of programmable areas is currently limited by the technology: available sensors only admit 16 safety areas. Evidently, better performances might be obtained if additional shapes were available. More in details, the limited number of programmable areas sometimes imposes using the same configurations for both tight and large bends. In the first case the vehicle speed is naturally reduced because of the kinematic limits and, consequently, safety areas are redundantly large: since tight curves are used for swift maneuvers in narrow spaces between shelves and columns, unnecessarily large safety areas could cause undesired stops. In the second case, curves could potentially be traveled at higher speeds, but the use of small areas imposes, for safety reasons, reduced velocities. In both cases the plant efficiency is penalized.

Figure 4.2 makes it possible to appreciate how safety areas are managed in real plants. In figure 4.2a the LGV is moving at plain speed along a straight path. In terms of safety, the frontal zone is the most critical one, so that it is protected by a very large safety area. Conversely, the rear part of the vehicle cannot collide with static obstacles, so that the corresponding sensor is switched off. If the vehicle needs to execute a turn, the speed is reduced, the frontal area is scaled down accordingly, and



(a) (b)

(c) (d)

Figure 4.2: LGV safety areas could be modified only when the vehicle switch between two subsequent segments.

the rear area is enlarged as shown in figure 4.2b: in fact, along turns the load sweeps laterally and may hit obstacles or coworkers. Since turns essentially imply a lateral motion, the side areas are modified as shown in figure 4.2c and the turn is finally executed (see figure 4.2d).

In industrial warehouses safety has been historically managed by means of trial-and-error approaches. Designers first define the plant layout, which is obtained through

the composition of path segments, each of them being described through appropriate path primitives. Then, the same technicians assign, on the basis of their own experience, the most suitable set of safety areas to each segment. Areas are selected among the ones programmed in the SLS. Subsequently, for each segment, the most proper constant speed is selected depending on the combination between curve and safety area: it must satisfy the vehicles constraints and it must be safe. A single safety area and a single constant speed is assumed for each segment of the plant. Such choice is motivated by several reasons. One of them was specified in the previous section: a constant velocity reference guarantees a simplified real-time generation of the composite speed profile. Another reason is essentially practical: each plant is composed by thousand of segments, so that the manual choice, for each of them, of multiple safety areas and multiple speed references would enormously complicate the design phase.

The implementation of the system in a real plant requires a final tuning "on the field" since safety areas may be inadequate, speeds may be unfeasible, and safety conditions may be violated: they are all the result of heuristic choices. Since each segment of the plant needs to be tested, the tuning process requires long times. Previous considerations make it possible to conclude that plant performances are strongly affected by the designers experience.

The SAFERUN project aimed at the partial automation of the design process, so as to make it more deterministic. More precisely, while plant designers still need to select curve layouts and safety areas, a smart velocity planner synthesizes, in real time, a speed reference which fulfills all the system constraints, including the safety ones.

Chapter 5

SAFERUN velocity planner

The safety risks prevention and the simultaneous generation of efficient motions motivated the development of the velocity planner explained in this chapter, the proposed planner is indicated in the following as SAFERUN Algorithm (SA). Differently from other real-time solutions, the SA handles variable speed constraints by also managing acceleration and jerk limits. For the first time, safety has been converted into an analytical constraint used for the synthesis of minimum-time velocity references. Hence, the proposed planner is conceived to satisfy both the real-time and the efficiency requirements, by simultaneously managing possible safety issues. Section 5.1 illustrates the problem formulation while section 5.2 shows how the velocity constraint is computed. Finally section 5.3 explains in details the proposed algorithm.

5.1 Problem formulation

In automated warehouses, LGVs move along pre-specified paths defined when the plant is designed. The plant layout is obtained by composing several basic primitives like straight lines, circular arcs, clothoids, splines, etc. The sequence of path-segments that LGVs must cover is variable, being possible bifurcations at the end of each segment, like it happens in the example shown in figure 5.1. Path segments allocated for each LGV are decided on-the-fly by a supervisor, which selects them on the

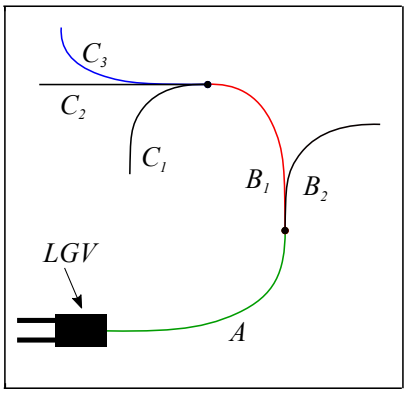


Figure 5.1: The sequence of path-segments is decided on-the-fly by the supervisor.

basis of the plant status: without any loss of generality, it could be assumed that an LGV is always aware of two path-segments, i.e., the one is traveling on and the next one. Shortly before the end of the first path-segment, the vehicle controls if a further one has been assigned. In case of a positive answer the velocity function is updated, otherwise the LGV is stopped at the end of the second segment. A typical planning case is proposed in figure 5.1. The vehicle is moving along path A and knows that B_1 will be the next segment: the initial velocity reference is planned such to generate a stop at the end of B_1. If a new path, for example C_3, is assigned before the LVG reaches the end of A, a new velocity function is evaluated. Evidently, it cannot be the same that would have been used for C_1 or C_2, because of the different path shapes. This implies that speed set-points must be necessarily planned in real time. Because of the limited computational capabilities of the LGV control-boards, velocity references are always obtained through computationally light algorithms. Such limitations prevents computing the velocity reference through nonlinear optimizers, since their computational times are not compatible with the real-time requirement.

Figure 5.2a shows the most commonly used approach. For each segment of the composite path the velocity upper-bound is assumed constant and its value is selected so as to satisfy simultaneously the kinematic and the safety constraints in the most demanding point of the path. When two segments are linked together, the velocity

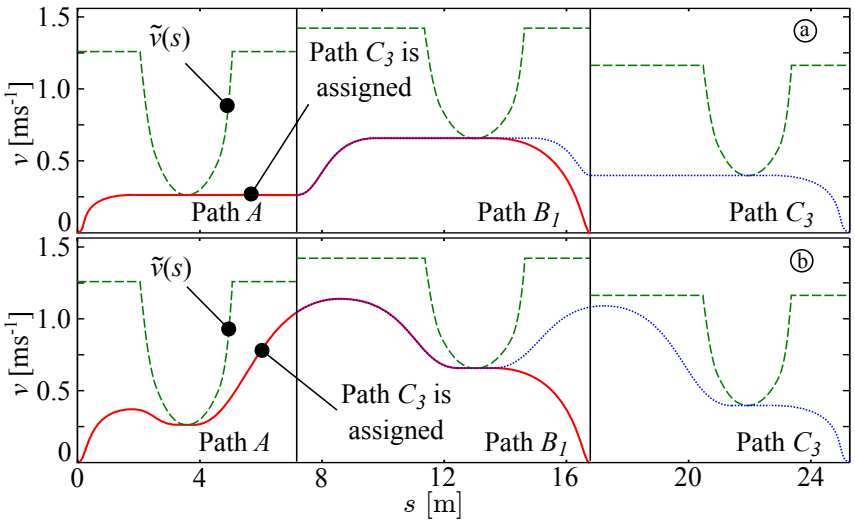


Figure 5.2: Comparison between the two planning approaches: (a) standard planning strategy; (b) novel planning strategy. Vertical solid lines separate paths A from B_1 and B_1 from C_3. Dashed lines indicate $\tilde{v}(s)$. The initial velocity references are shown by means of solid lines, while dotted lines are used for the updated profiles.

reference is obtained by joining the respective upper-bounds by means of Double-S profiles [90] or by means of any other lightweight planning primitive. This approach is very common (see for example [91] for a transportation vehicle), but it has an evident drawback: for each segment, the maximum cruising speed must be assumed equal to the minimum value of $\tilde{v}(s)$. In LGV plants this is evidently reason of productivity losses and motivated the alternative, lightweight planning-strategy proposed in section 5.3.

Figure 5.2b is instrumental for the comprehension of the velocity planning strategy proposed in this work. The vehicle is initially aware of the data concerning paths A and B_1, hence a rest-to-rest velocity reference is planned so as to stop the LGV at the end of B_1. Differently from the standard planning technique, the actual velocity upper-bound, i.e., $\tilde{v}(s)$, is considered. During the motion along A, segment C_3 is assigned: the velocity profile is immediately updated in order to stop the LGV at the

end of C_3. The planning process is repeated if a new path-segment is assigned while the vehicle is moving along B_1. The use of the actual velocity constraints evidently allows higher speeds and, consequently, plant efficiency necessarily improves.

The above considerations motivated the development of the online SA planner. In order to properly formulate the problem, let us indicate with $s(t)$ the LGV position along an assigned Cartesian path. $s(t)$ is assumed twice differentiable, so that speed $v(t)$ and acceleration $a(t)$ are both continuous function. Finally, the corresponding jerk signal, i.e., $j(t)$, is a piecewise-constant function. The path length is known and it is indicated by s_f, while the total traveling time is represented by t_f.

Functions $s(t)$, $v(t)$, $a(t)$, and $j(t)$ can always be algebraically manipulated in order to express velocities, accelerations, and jerks as function of curvilinear coordinate s, i.e., the following functions can be synthesized: $\hat{v}(s)$, $\hat{a}(s)$, and $\hat{j}(s)$, with $s \in [0, s_f]$, defined such that $\hat{v}(s(t)) := v(t)$, $\hat{a}(s(t)) := a(t)$, and $\hat{j}(s(t)) := j(t)$. For such reason, the velocity function will be indifferently mentioned in the following as $v(s)$ or $v(t)$.

The real-time planning problem can be formulated as follows:

Problem 1 *Design an algorithm for the real-time generation of almost minimum-time trajectories $s(t)$ which solves the following nonlinear, semi-infinite optimization problem*

$$\min\{t_f\} \tag{5.1}$$

subject to

$$s(0) = 0, \quad s(t_f) = s_f, \tag{5.2}$$

$$v(0) = v_0, \quad v(t_f) = 0, \tag{5.3}$$

$$a(0) = a_0, \quad a(t_f) = 0, \tag{5.4}$$

$$0 \leq v(t) \leq \tilde{v}[s(t)], \quad \forall t \in [0, t_f], \tag{5.5}$$

$$-\tilde{a} \leq a(t) \leq \tilde{a}, \quad \forall t \in [0, t_f], \tag{5.6}$$

$$-\tilde{j} \leq j(t) \leq \tilde{j}, \quad \forall t \in [0, t_f], \tag{5.7}$$

where $\tilde{v}(s) : [0, s_f] \Longrightarrow \mathbb{R}^+ \backslash \{0\}$ is the velocity upper bound function, while $\tilde{a}, \tilde{j} \in \mathbb{R}^+ \backslash \{0\}$ represent constant limits on acceleration and jerk.

For two practical reasons, the solution proposed is based on a heuristic method. First of all, an exact solution can only be found by means of nonlinear programming strategies, whose computational burden is not compatible with the real-time context here considered. Secondly, a strict minimum-time solutions would continuously impose the execution of transients while, in order to reduce the mechanical stress acting on the LGV and, even more important, on its load, it is wise accepting suboptimal solutions characterized by constant-speed sections. For such reason two different version of SA are proposed in the following: the first one is specifically conceived for LGV plants while the second one aims at demonstrating that through the proposed heuristic procedure it is possible to obtain traveling times comparable with the ones of a nonlinear solver, while respecting the real-time requirements which, instead, cannot be guaranteed by a nonlinear solver.

5.2 Velocity bounding function

The solution of *Problem 1* is based on the knowledge of $\tilde{v}(s)$. This section is devoted to its synthesis. Let us suppose that the vehicle path is defined through a parametric function, namely $\mathbf{p}(s) := [x(s)\ y(s)]^T$, whose curvature $\kappa(s)$ and curvature derivative $\frac{d\kappa}{ds}(s)$ are both known. The velocity bounding function, i.e., $\tilde{v}(s)$, is chosen so as to guarantee that, if the vehicle speed fulfills condition $v(s) \in [0\ \tilde{v}(s)]$, a given set of constraints, specified by the users, is automatically satisfied. The ones reported in the following are typical for the problem at hand, but other constraints can be easily added.

The norm of the lateral acceleration is normally limited to an assigned value $a_s > 0$, in order to reduce path tracking errors, wheels wears, and load solicitations. This result is achieved by imposing that $v(s) \leq \tilde{v}_a(s)$ where (see also [73, 91])

$$\tilde{v}_a(s) := \sqrt{\frac{a_s}{|\kappa(s)|}}\,. \tag{5.8}$$

The longitudinal velocity is also limited by the steering capability of the vehicle. If θ is the steering angle, its first derivative is certainly superiorly limited, so that feasible movements must fulfill inequality $\left|\dot{\theta}\right| \leq \dot{\theta}_s$, where $\dot{\theta}_s > 0$ is a given upper

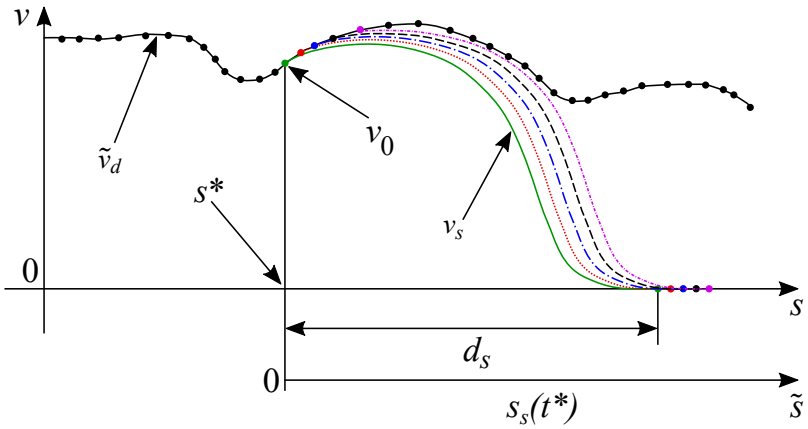


Figure 5.3: Procedure used to build function $\tilde{v}_d(s)$. For each point s of the path a velocity v_0 is obtained by solving problem (5.11)–(5.17). Resulting points are then interpolated through a spline function. Auxiliary functions used in the optimization problem start from s^*

bound. With a few algebraic manipulations such constraint can be converted into the following limit for $\hat{v}(s)$

$$\tilde{v}_b(s) := \frac{(1+(L\kappa(s))^2)\dot{\theta}_s}{L\left|\frac{d\kappa}{ds}(s)\right|} \tag{5.9}$$

where L is the distance between front and rear wheels. Finally, a constant upper bound $\tilde{v}_c > 0$ is added. It may depend on the traction capabilities of the motors or, more commonly, on law restrictions which apply to LGV plants. Depending on the application, further upper bounds might be considered: the final limit is obtained by combining all of them according to the following equation

$$\tilde{v}(s) := \min\{\tilde{v}_a(s), \tilde{v}_b(s), \tilde{v}_c, \ldots\}. \tag{5.10}$$

Then, a novelty of this work is related to the inclusion of the safety constraint in the velocity planning problem. The safety laser sensors are certified devices which only provide a binary information concerning the presence (or not) of obstacles inside an Inspection Area (IA) surrounding the vehicle: no data is available concerning the obstacle position. As a consequence, any algorithm for the safety management must

only be based on such information. Currently, as explained in Section 4.2, safety regulations only consider potential collisions w.r.t. static obstacles, so that as long as the IA is declared "clean", the LGV can move freely inside it. The traveling distance $\hat{d}$ which can be safely crossed by the vehicle before any point of its chassis could touch the border of the IA can be evaluated for each position of the path: safety is guaranteed if the vehicle can be stopped before a distance $d_s := \hat{d} - \tilde{d}$ is covered, where $\tilde{d}$ is an additional safety margin added to account for potential tolerances. Bearing in mind this idea, safe movements can be guaranteed by evaluating a bounding velocity function $\tilde{v}_d(s)$ obtained by interpolating, through a spline function, all points v_0 found by solving, for each position s^* of the path, the following semi-infinite optimization problem

$$\max_{t_s}\{v_0\} \tag{5.11}$$

subject to

$$s_s(0) = 0, \quad s_s(t_s) = d_s, \tag{5.12}$$

$$v_s(0) = v_0, \quad v_s(t_s) = 0, \tag{5.13}$$

$$a_s(0) = \tilde{a}, \quad a_s(t_s) = 0, \tag{5.14}$$

$$0 \leq v_s(t^*) \leq \tilde{v}(s_s(t^*) + s^*), \quad \forall t^* \in [0, t_s], \tag{5.15}$$

$$-\tilde{a} \leq a_s(t^*) \leq \tilde{a}, \quad \forall t^* \in [0, t_s], \tag{5.16}$$

$$-\tilde{j} \leq j_s(t^*) \leq \tilde{j}, \quad \forall t^* \in [0, t_s], \tag{5.17}$$

where t^* is an auxiliary time which is equal to 0 at the beginning of each transient starting from s^*, $s_s(t^*)$, $v_s(t^*)$, $a_s(t^*)$, and $j_s(t^*)$ are auxiliary position, velocity, acceleration, and jerk functions, respectively. It should be noticed that the upper velocity bound in constraint (5.15) is based on the original curvilinear coordinate s. As could be seen in figure 5.3, the axis of the auxiliary position function $s_s(t^*)$ starts from s^*, in order to access the corresponding value of $\tilde{v}(s)$ the vehicle position must be known w.r.t. the beginning of the curve, in fact $s = \tilde{s} + s^* = s_s(t^*) + s^*$. The meaning of v_0 is evident: it coincides with the highest value that the initial velocity can assume in s which allows a vehicle stop within a distance d_s, i.e., within the area that has been

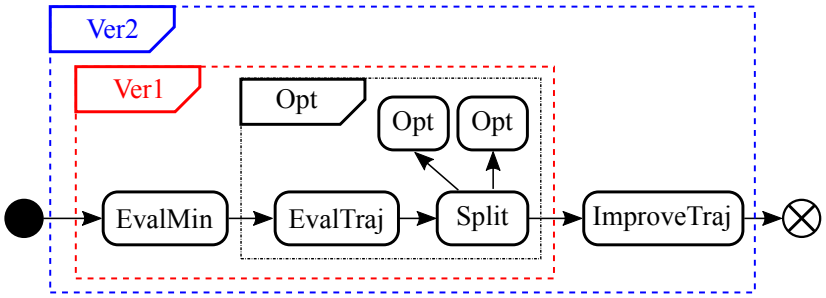


Figure 5.4: Flow diagram of the SA velocity planner.

declared clean from obstacles. For the same position s, any initial speed which is lower than v_0 would generate stopping distances shorter than d_s. Consequently, $\tilde{v}_d(s)$ can be assumed as the speed upper-bound which guarantees safe motions. It must be noticed that the initial acceleration has been posed equal to $\tilde{a}$ since this choice causes longer transients and, consequently, lower values of v_0.

Problem (5.11)–(5.17) is a nonlinear, complex one. Fortunately, computational times do not represent an issue since $\tilde{v}_d(s)$ can be evaluated offline. In fact, the expression of $\tilde{v}_d(s)$ does not depend on real-time data: it is only function of geometric parameters like the path shape, the vehicle shape, and the IA shape.

The final velocity constraint is a combination of all the bounds and, consequently, it is given by the following expression

$$\tilde{v}(s) := \min\{\tilde{v}_a(s), \tilde{v}_b(s), \tilde{v}_c, \tilde{v}_d(s)\}, \ \forall s \in [0, s_f] \, . \tag{5.18}$$

5.3 SAFERUN Algorithm

In this section the general functioning of the SA velocity planner will be explained. Two versions are proposed, the first one (SAv1) presents a velocity planner specifically conceived for the application at hand. In fact, in LGV plant multiple acceleration and deceleration transients should be avoided in order to reduce mechanical stresses on the vehicle and its load, due to this concept a threshold has been inserted in SAv1 in order to stop the optimization algorithm and to obtain a velocity profile

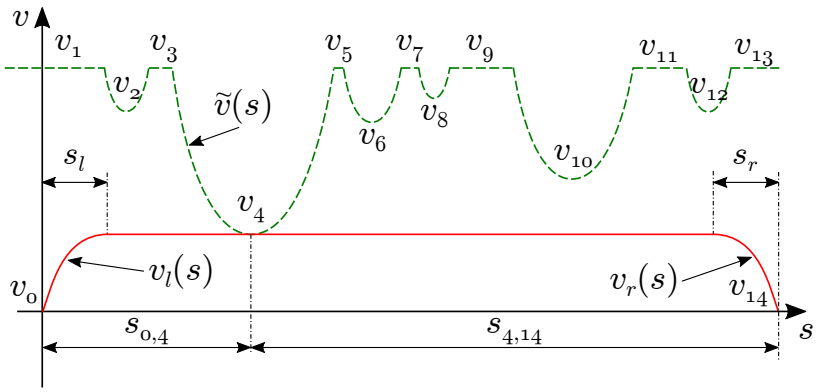


Figure 5.5: First iteration of the velocity planner. The initial feasible solution is represented through a solid line, while a dashed line is used for the upper velocity bound. Upper bound v_4 is active, so that a new iteration is required.

characterized by constant speed segments. Conversely, the second version (SAv2) slightly modifies the previous algorithm in order to obtain the best velocity profile and it further improves the solution through a function conceived "ad hoc". The flow diagram and the differences between the two SA versions are graphically represented in figure 5.4.

The general flow of the first version of SA is explained by means of figure 5.4 and Algorithm 3. Algorithm 3 is able to solve *Problem 1* in real time, let us analyze its behavior. Without any loss of generality the example case here proposed considers a rest-to-rest transient, but the approach can be easily used with starting speeds different from 0.

The Main procedure initially calls $EvalMin(\tilde{v}(s))$ for a preliminary inspection of the velocity constraint, this is also the first step of the flow diagram. In detail, $EvalMin(\tilde{v}(s))$ evaluates the number of local minima and maxima of $\tilde{v}(s)$, their respective values and the distances along the path between adjacent minima. The following notation is adopted: N indicates the total number of maxima and minima, which values are identified by v_m: $m \in 2,4,\ldots,N-1$ for local minima, $m \in 1,3,\ldots,N$ for local maxima. Finally, v_0 and v_{N+1} are the initial and the final velocities, respec-

Algorithm 3: General flow of SA and the optimization function $Opt(i,j,k)$

Data: $\tilde{v}(s)$, velocity constraint function; v_0, initial velocity;

Result: *List*, list with the feasible velocity functions;

Main
- $N, \mathbf{s}, \mathbf{v} \Leftarrow EvalMin(\tilde{v}(s), v_0)$;
- $q \leftarrow \min_{q \in [1,N]} \{v_q\}$;
- $Opt(0, q, N+1)$;

Function $Opt\ (i,j,k)$
- $v_l(t), s_l, v_r(t), s_r \leftarrow EvalTraj(i,j,k)$;
- **if** $s_l + s_r = s_{ik}$ **then**
  - $AddList(i,j,v_l(t))$;
  - $AddList(j,k,v_r(t))$;
- **else**
  - $Check(i,j,k,s_l,s_r)$;
  - **if** $s_l > \mathrm{ST}/100\ s_{ij}$ **then**
    - $AddList(i,j,v_l(t))$;
  - **else**
    - $q \leftarrow \min_{q \in [i+1,j-1]} \{v_q\}$;
    - $Opt(i,q,j)$;
  - **if** $s_r > \mathrm{ST}/100\ s_{jk}$ **then**
    - $AddList(j,k,v_r(t))$;
  - **else**
    - $q \leftarrow \min_{q \in [j+1,k-1]} \{v_q\}$;
    - $Opt(j,q,k)$;

tively. Practically, as shown in figure 5.5, odd indexes refer to maxima, while even indexes refer to minima. The distance along the path between the mth minimum and

Algorithm 4: Function *Check* called by $Opt(i,j,k)$

Data: **v**, vector of minima and maxima; $s_{i,j}$, distances between two minima;
Result: updated **v** and $s_{i,j}$;

```
1  Function Check (i, j, k, s_l, s_r)
2      if s_l > s_ij then
3          p ← j;
4          while s_l > s_{i,(p+2)} do
5              p ← p + 2;
6          v_p ← v_j;
7          s_{p,(p+2)} ← s_{p,(p+2)} − [s_l − s_ip];
8          s_{(p−2),p} ← s_{(p−2),p} + [s_l − s_ip];
9          j ← p;
10     else if s_r > s_jk then
11         p ← j;
12         while s_r > s_{(p−2),k} do
13             p ← p − 2;
14         v_p ← v_j;
15         s_{(p−2),p} ← s_{(p−2),p} − [s_r − s_pk];
16         s_{p,(p+2)} ← s_{p,(p+2)} + [s_r − s_pk];
17         j ← p;
```

the subsequent one is indicated as $s_{m,(m+2)}$, where $m \in 0,2,\ldots,N-3$. In the same way, the distance between two generic minima v_i and v_j, where $i \in 0,2,\ldots,N-1$ and $j \in i+2,\ldots,N+1$, is referred as $s_{i,j}$.

Vector $\mathbf{v} := [v_0\ v_1\ \cdots\ v_{N+1}]^T$ is scrutinized to find the minimum v_q, $q = 1,2,\ldots,N$ (line 3 of Algorithm 3). The minimum is used for the generation of the first feasible profile. To this purpose, the algorithm calls $Opt(i,j,k)$, which is a recursive function that generates an almost minimum-time velocity profile through a sequence of consecutive refinements. Speed functions are actually synthesized by $EvalTraj(i,j,k)$,

which returns a minimum-time profile from v_i to v_k by assuming a maximum speed equal to v_j.

In order to obtain the initial velocity profile, the following minimum-time problem is solved:

$$\min\{t_k - t_i\} \tag{5.19}$$

subject to

$$s(t_i) = s_i, \quad s(t_k) = s_k, \tag{5.20}$$

$$v(t_i) = v_i, \quad v(t_k) = v_k, \tag{5.21}$$

$$a(t_i) = 0, \quad a(t_k) = 0, \tag{5.22}$$

$$0 \leq v(s) \leq v_j, \quad \forall s \in [s_i, s_k], \tag{5.23}$$

$$\underline{a} \leq a(s) \leq \bar{a}, \quad \forall s \in [s_i, s_k], \tag{5.24}$$

$$\underline{j} \leq j(s) \leq \bar{j}, \quad \forall s \in [s_i, s_k], \tag{5.25}$$

$$\tag{5.26}$$

where v_j is the minimum element of vector $\mathbf{v}$, hence for the first iteration $v_j = v_q = v_4$. Then $t_i = 0$ and $t_k = t_f$, so that the cost index coincides with the total traveling time, $s_i = 0$, $s_k = s_f$, and $v_i = v_k = 0$ (see figure 5.5).

The optimal solution of problem (5.19)–(5.26) can be efficiently evaluated through closed form expressions and it is given by a piecewise constant-jerk, composite function made of $n \leq 7$ segments, each of them of time length t_l, whose corresponding jerk, acceleration, velocity, and position primitives can be expressed as follows ($l = 1, 2, \ldots, n$, $t \in [0, t_l]$)

$$j_l(t) = j_l \tag{5.27}$$

$$a_l(t) = a_{l-1}(t_{l-1}) + j_l t, \tag{5.28}$$

$$v_l(t) = v_{l-1}(t_{l-1}) + a_{l-1}(t_{l-1})\, t + \frac{1}{2} j_l t^2, \tag{5.29}$$

$$s_l(t) = s_{l-1}(t_{l-1}) + v_{l-1}(t_{l-1})\, t + \frac{1}{2} a_{l-1}(t_{l-1})\, t^2 + \frac{1}{6} j_l t^3, \tag{5.30}$$

where j_l is the constant jerk value assumed for the lth segment. Equations (5.27)–(5.30) analytically express $s(t)$, $v(t)$, $a(t)$, and $j(t)$ as functions of the time.

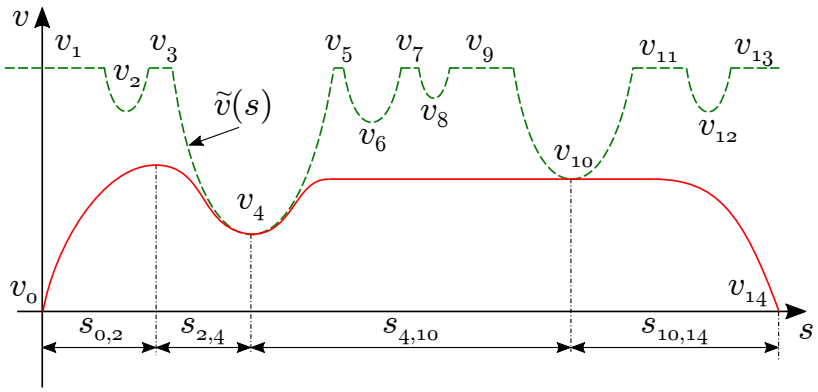


Figure 5.6: Second iteration of the velocity planner. The feasible solution is represented through a solid line, while a dashed line is used for the upper velocity bound. Upper bound v_2 is not active, so that the leftmost part of the profile is no more updated, while a further iteration is required for the right section, since v_{10} is active.

Hence the initial solution found by $EvalTraj(i,j,k)$, shown in figure 5.5 through a solid red line, is characterized by a bang-zero-bang jerk profile, that is $j_l \in \{\underline{j}, 0, \overline{j}\}$, in according with the Pontryagin principle. The profile generated can be divided into two minimum-time transient curves, namely $v_l(t)$ and $v_r(t)$, joined by a constant speed segment.

Depending on the shape of the first profile found by $EvalTraj(i,j,k)$, two situations may arise:

1. the velocity function does not reach upper bound v_j. In such a case, the constant velocity section is missing, so that $s_l + s_r = s_{ik}$;

2. the velocity function reaches v_j.

In the first case, the algorithm has already converged to the optimal solution, which is represented by functions $v_l(t)$ and $v_r(t)$ that are consequently added to the *List* of the final profiles (lines 7–9 of Algorithm 3). Each element in the *List* contains the information required for the generation of a single transient:

- i, id number of the starting point;

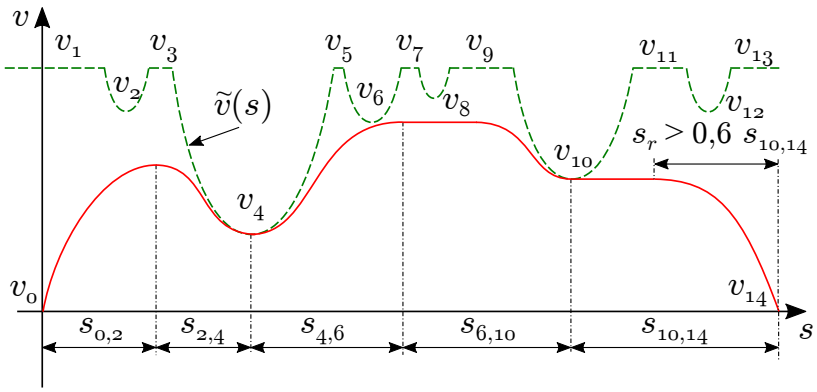


Figure 5.7: Third iteration of the velocity planner. The final feasible solution is represented through a solid line, while a dashed line is used for the upper velocity bound.

- j, id number of the end point;
- s^*, longitudinal coordinate of the starting point;
- $v(t)$, velocity function associated to the ij transient.

In the second case, i.e., if upper bound v_j is active, the initial solution may probably be improved. This is what happens for example in figure 5.5, so that a new iteration is started (lines 14–16 and 19–21 of Algorithm 3). The original problem is decomposed into two parts: a new trajectory is generated between v_0 and v_4 by assuming a maximum speed equal to v_2, and another one is generated between v_4 e v_{14}, by assuming a maximum speed equal to v_{10}. Figure 5.6 shows that the maximum speed is not reached in segment $s_{0,4}$, so that the corresponding velocity functions, according to the premises, are considered optimal and saved into the *List*. Path lengths $s_{0,2}$ and $s_{2,4}$ are modified, so as to coincide with the lengths of the raising and falling transients, respectively. For which concerns segment $s_{4,14}$, the maximum speed is reached, so that a new iteration, whose results are shown in figure 5.7, is started. It is worth noticing that, differently from the expectations, the profile in $s_{10,14}$ is left unchanged. This is a consequence of a design choice that was made for which, if s_l or s_r are longer than 60% of the total path-segment length, the velocity function is not

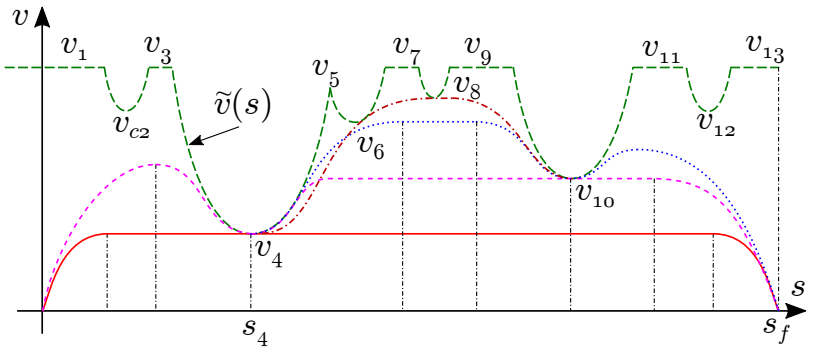


Figure 5.8: The SAv2 iteratively raises the velocity profile, so as to reduce traveling time.

updated (lines 12,13 and 17,18 of Algorithm 3). The underlying idea is that, when a similar configuration occurs, time improvements deriving from a further optimization are limited, so that it could be wise to keep a constant speed in order to limit the mechanical stresses acting on the LGV and on its load. The Stop Threshold (ST) value can be changed depending on the user requirements: higher percentages return solutions closer to the optimal one, but with higher evaluation times. Coming back to the example, the 60% rule does not apply to segment $s_{4,10}$ that is consequently updated. As shown in figure 5.7, the resulting transient from v_4 to v_6 is longer than the distance between the two adjacent minima. This particular case is managed by the *Check* function (line 11 of Algorithm 3 calls the function explained in Algorithm 4): the transient associated to segment $s_{4,6}$ is considered definitive, while a further optimization might be potentially possible in $s_{6,10}$. However, the falling transient is longer than 60% of $s_{6,10}$, so that the planning process of SAv1 ends.

The second version of the SAFERUN Algorithm, SAv2, contains the whole process shown in figure 5.4. The first step, which is $EvalMin(\tilde{v}(s))$, and the choice of the minimum among the elements of vector **v** performs exactly the same as the previous version. The first difference is the removal of ST, hence $Opt(i,j,k)$ computes the best obtainable profile through consecutive iterations of $EvalTraj(i,j,k)$, the detailed algorithm is explained in subsection 5.3.1.

Figure 5.8 shows the iterations performed by the second version of SA and it is useful to understand the introduced improvements. The first two iterations, indicated respectively with a solid red line and a dashed magenta one, performs as the previous algorithm then, starting from the third iteration, the introduced improvements modifies the flow and the velocity profile computed by the algorithm. Looking at the last transient, between v_{10} and the final velocity, the SAv2 further improves the velocity profile (see the dotted-blue curve in Figure 5.8), conversely in Figure 5.7 the algorithm was stopped because of the stop threshold. A different strategy is adopted also in the central part of the profile, between v_4 and v_{10} the SAv1 stops and calls the *Check* function, conversely in SAv2 if the maximum admissible speed v_j is reached but $\tilde{v}(s)$ and $v(s)$ do not touch each other, the value of v_j is increased by selecting from $\mathbf{v}$ a higher minimum in the considered interval and by repeating the optimization procedure. For example, the dotted line in figure 5.8 does not touch v_6, so that the velocity function is re-planned by assuming $v_j = v_8$. The dash-dotted brown solution replaces the dotted blue one if the latter shows a higher traveling time. The function refinement continues until no further improvements are possible because all the elements in $\mathbf{v}$ have been scrutinized.

Function $Opt(i, j, k)$ returns a very efficient solution, which however can be further improved. To this purpose, in SAv2 the profile returned by $Opt(i, j, k)$ is subsequently processed by $ImproveTraj$, so as to generate a final solution characterized by traveling times comparable with the ones obtained through a nonlinear solver. Details on $ImproveTraj$ are given in subsection 5.3.2, while subsection 5.3.1 explains how $EvalTraj(i, j, k)$ works.

The main properties of the planning approach proposed can be summarized as follows. First of all, at each iteration the novel velocity profile reduces with certainty the total traveling time. Moreover, the final solution is found through a sequence of feasible steps, so that the optimization procedure can be stopped at any stage in order to limit the evaluation time. Furthermore, the number of constant jerk segments of the final solution depends on the complexity of $\tilde{v}(s)$, i.e., on the number of its local minima and maxima: the complexity of the solution is modulated depending on the needs.

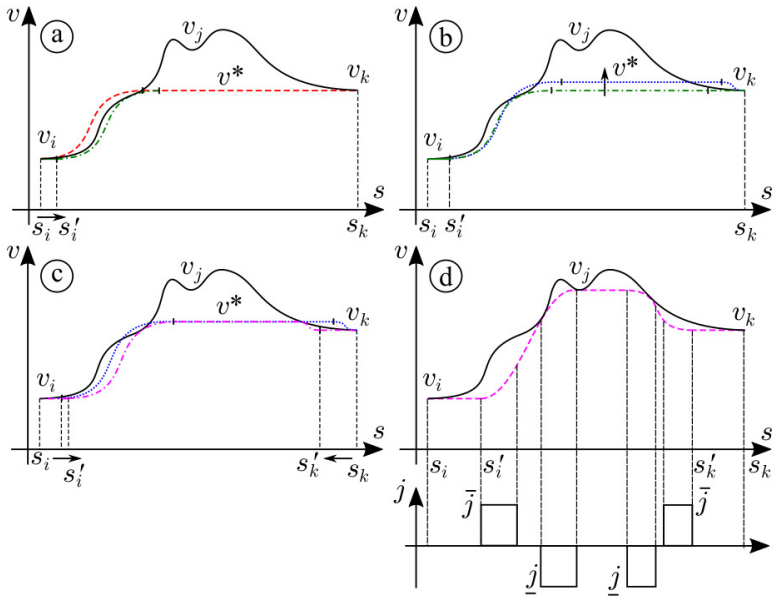


Figure 5.9: Algorithm 5 generates a velocity function according to the following strategy: (a) starting from the unfeasible dashed red profile, a feasible solution is obtained (dash-dotted green line) by right shifting s_i'; (b) v^* is increased and feasibility is lost (dotted blue line); (c) a new feasible solution is obtained by shifting s_i' and s_k' (dash-dotted magenta line). If improvements are achieved, the procedure is reiterated from (b); (d) a possible final solution for which $v^* = v_j$. The final solution is made of 9 constant jerk intervals.

5.3.1 The EvalTraj function

This section is devoted to the explanation of $EvalTraj(i,j,k)$, its basic principles can be explained with the aid of figure 5.9 and Algorithm 5. $EvalTraj(i,j,k)$ finds an almost optimal solution for (5.19)–(5.26) based on an iterative method and by assuming a maximum of 9 constant-jerk intervals.

After the initialization of some internal variables (lines 2–5 of Algorithm 5), the

Algorithm 5: The *EvalTraj* algorithm used to find an initial velocity function.

Data: $\tilde{v}(s)$, velocity constraint; $\mathbf{h} := [s_i, s_k, s_i', s_k', v_i, v_k, \underline{a}, \overline{a}, \underline{j}, \overline{j}]$, interpolating conditions; v_j, maximum speed for the segment; N, maximum number of iterations.

Result: $v_o(s)$ velocity function for $s \in [s_i, s_k]$; $\mathbf{h}_o$, final interpolating conditions;

```
1  Function EvalTraj
2      v* = max{v_i, v_k};
3      f_cs ← 0;
4      t_best ← ∞;
5      Fail ← 0;
6      while (f_cs = 0) & (v* < v_j) & (Fail < N) do
7          v̂(s), t_new, f_cs ← DS(h, v*);
8          if !(v̂(s) ≤ ṽ(s)), s ∈ [s_i, s_k] then
9              h ←DSReduce(ṽ(s), v̂(s), h);
10         else
11             if t_new ≥ t_best then
12                 Fail ← Fail + 1
13             else
14                 t_best ← t_new;
15                 v_o(s) ← v̂(s);
16                 h_o ← h;
17                 Fail ← 0;
18             v* ← Incr(v*) ;
```

algorithm enters the while cycle. The interpolating conditions for the problem, together with the acceleration and the jerk bounds are stored into vector:

$$\mathbf{h} := [s_i, s_k, s_i', s_k', v_i, v_k, \underline{a}, \overline{a}, \underline{j}, \overline{j}].$$

s'_i and s'_k are auxiliary variables initially assigned as follows: $s'_i = s_i$ and $s'_k = s_k$. At the first iteration, the maximum admissible speed, i.e., v^*, is posed equal to the highest minimum between v_i and v_k. By initially neglecting the velocity constraint, function DS$(\mathbf{h}, v^*)$ finds a minimum-time solution through closed form expressions based on a double-S profile [90] (line 7). The result is the dashed red curve shown in figure 5.9a, which is composed by 4 constant-jerk intervals. The outputs of DS$(\mathbf{h}, v^*)$ are represented by velocity function $\hat{v}(s)$, by traveling time t_{new}, and by flag f_{cs} which indicates if v^* has been reached. The resulting profile is often unfeasible, thus function DSReduce$(\tilde{v}(s), \hat{v}(s), \mathbf{h})$ (line 9) right shifts the position of s'_i (or left shifts s'_k) until feasibility is reestablished. To this purpose a constant speed segment is added at the beginning (or at the end) of the profile, between s_i and s'_i. DSReduce$(\tilde{v}(s), \hat{v}(s), \mathbf{h})$ returns an updated version of $\mathbf{h}$ and a new, certainly feasible profile – see the dash-dotted green line in figure 5.9a – whose parameterization is subsequently evaluated by DS$(\mathbf{h}, v^*)$ (line 7). If t_{new} is smaller than t_{best} (line 11), the novel profile becomes the best one (lines 14–16) and the failure counter is reset (line 17). In any case, v^* is increased in order to repeat the procedure by assuming an higher maximum speed (line 18). Another call to DS$(\mathbf{h}, v^*)$ (line 7) returns the blue dotted profile in figure 5.9b. Function DSReduce$(\tilde{v}(s), \hat{v}(s), \mathbf{h})$ further shifts s'_i and s'_k in order to make the profile feasible, so that the dash-dotted magenta curve shown in figure 5.9c is obtained. The process continues until v_j is reached ($v^* = v_j$), or v^* cannot be further increased by DS (in that case f_{cs} is posed equal to 1), or for N consecutive iterations the solution does not improve. Figure 5.9d shows a possible final solution, which always admits a maximum of 9 constant-jerk intervals.

5.3.2 The ImproveTraj function

The solution provided by Algorithm 5 shows time performances compatible with the real-time requirement. However, further improvements can be achieved by reshaping such solution so as to make it closer to $\tilde{v}(s)$, thus increasing average speeds and, consequently, reducing traveling times. This is the purpose of Algorithm 6 described in the following.

Function *ImproveTraj* is called a single time for each one of the 9-interval pro-

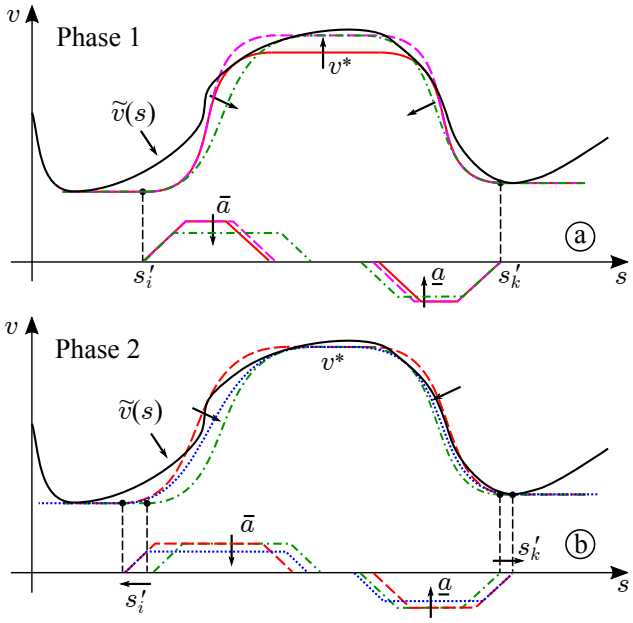


Figure 5.10: Velocity function improvement. a) During Phase 1 the initial solution (solid red line) is modified by increasing v^*. The obtained unfeasible profile (dashed magenta line) is modified by reducing $\bar{a}$ and $|\underline{a}|$ until constraint $\tilde{v}(s)$ is newly satisfied (dash-dotted green line). b) Phase 2 modifies such solution by enlarging s_i' and s_k' (dashed red line) and by subsequently further reducing $\bar{a}$ and $|\underline{a}|$ (dotted blue line) until constraint $\tilde{v}(s)$ is satisfied. Solutions are accepted if traveling time reduces.

files $v_o(s)$, which compose $v(s)$, returned by Algorithm 5. Consequently, its inputs are given by $\tilde{v}(s)$ and by $v_o(s)$ together with its corresponding planning data: vector $\mathbf{h}_o := [s_i, s_k, s_i', s_k', v_i, v_k, \underline{a}, \bar{a}, \underline{j}, \bar{j}]$, traveling time t_{best} and maximum speed v_j. The function output is represented by an updated profile $v_o(s)$ and a novel vector $\mathbf{h}_o$.

The improving strategy considers three phases: the first 2 of them are executed alternatively several times, while the third one represents a final refinement. The following flags are used: f_{ph} individuates the current phase, $f_{im} = 1$ indicates that the velocity function has been improved, $f_{cs} = 1$ indicates that maximum speed cannot be further increased, while *Fail* is a failure counter.

Algorithm 6: The *ImproveTraj* algorithm used to improve the solution found by *EvalTraj*.

Data: $v_o(s), t_{best}$, best solution from Alg. 5; $\tilde{v}(s)$, velocity constraint; N, maximum number of iterations; $\mathbf{h}_o := [s_i, s_k, s_i', s_k', v_i, v_k, \underline{a}, \overline{a}, \underline{j}, \overline{j}]$, interpolating conditions; v_j, maximum speed.

Result: $v_o(s)$ velocity function for $s \in [s_i, s_k]$; $\mathbf{h}_o$, final interpolating conditions.

Function *ImproveTraj*
- $f_{ph} \leftarrow 1; f_{im} \leftarrow 0; f_{cs} \leftarrow 0;$
- $Fail \leftarrow 0; v^* \leftarrow \max_{s \in [s_i, s_k]} v_o(s); \mathbf{h} \leftarrow \mathbf{h}_o;$
- **while** $f_{ph} \neq 3$ **do**
  - **if** $f_{ph} = 1$ **then**
    - $v^* \leftarrow \text{Incr}(v^*)$;
  - **else**
    - $\mathbf{h} \leftarrow \text{UpdateSiSk}(\mathbf{h})$;
  - $\hat{v}(s), t_{new}, f_{cs}, \mathbf{h} \leftarrow \text{UpdateDS}(\mathbf{h}, v^*)$;
  - **if** $t_{new} \geq t_{best}$ **then**
    - $Fail \leftarrow Fail + 1;$
  - **else**
    - $t_{best} \leftarrow t_{new};$
    - $v_o(s) \leftarrow \hat{v}(s);$
    - $\mathbf{h}_o \leftarrow \mathbf{h};$
    - $f_{im} \leftarrow 1$;
  - **if** $[(f_{cs} = 1) || (Fail = N)]$ **then**
    - $Fail \leftarrow 0; v^* \leftarrow max(v_o(s)); \mathbf{h} \leftarrow \mathbf{h}_o;$
    - **if** $f_{ph} = 1$ **then**
      - $f_{ph} \leftarrow 2;$
    - **else if** $f_{im} = 1$ **then**
      - $f_{ph} \leftarrow 1;$
      - $f_{im} \leftarrow 0;$
    - **else**
      - $f_{ph} \leftarrow 3;$

```
26
27    s_l, s_r ← EvalSP(h_o, v_o(s), ṽ(s));
28    if s_l > 0 then
29        v_o(s), h_o ← Left_JzJ(v_o(s), s_l, h_o);
30    if s_r > 0 then
31        v_o(s), h_o ← Right_JzJ(v_o(s), s_r, h_o);
```

As shown in figure 5.10a, initial solution $v_o(s)$ (solid red line) is always in contact with $\tilde{v}(s)$. After some initializations (lines 2 and 3), Phase 1 starts and v^* is increased (line 6), so that an eventual call of DS($\mathbf{h}, v^*$) would return an unfeasible profile, like the dashed magenta one shown in figure 5.10a. In order to obtain a feasible solution, function UpdateDS($\mathbf{h}, v^*$) (line 9) modifies $\bar{a}$ and $\underline{a}$ through an iterative bisection method: roughly speaking, $\bar{a}$ and $|\underline{a}|$ are reduced until feasibility is newly achieved (see the dash-dotted green line in figure 5.10a). The output of UpdateDS($\mathbf{h}, v^*$) is given by a novel feasible profile $\hat{v}(s)$ and its corresponding $\mathbf{h}$. The new solution replaces the best one if the latter is less efficient (lines 13–15). Simultaneously, the improvement flag is raised (line 16). Conversely, if traveling time does not reduce, the failure counter is increased (line 11).

If v^* cannot be further increased ($f_{cs} = 1$) or if the maximum number of failures has been reached (line 17), the failure counter is reset and the algorithm switches to Phase 2 (line 20). The improving procedure changes. As shown in figure 5.10b function UpdateSiSk($\mathbf{h}, v^*$) enlarges interval $[s_i', s_k']$ by preserving v^* (see the dashed red line in figure 5.10b). The obtained solution becomes unfeasible, so that UpdateDS($\mathbf{h}, v^*$) is called in order to newly achieve feasibility by further reducing the accelerations module (see the dotted blue line in figure 5.10b). Iterations continue until the solution improves or a given number of failures has been detected. If the exit condition is met and at least one improvement is detected during one of the two phases, the system switches to Phase 1 and starts a new iteration. If no improvements are detected, the algorithm enters Phase 3.

The activation of Phase 3 requires that the contact between $v_o(s)$ and $\tilde{v}(s)$ occurs

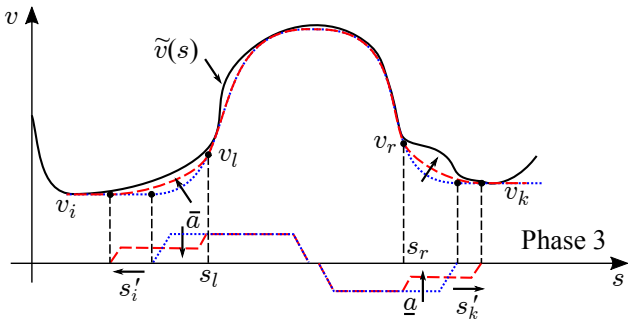


Figure 5.11: Velocity function improvement. Phase 3 activates if $v_o(s)$ intercepts $\tilde{v}(s)$ in the interval in which $a(s) = \bar{a}$ or $a(s) = \underline{a}$. Two split points s_l and s_r are chosen just below the contact point and the accelerations of the first and of the last transients are reduced, so as to increase the speed in the corresponding intervals.

for points in which the acceleration is equal to $\bar{a}$ or $\underline{a}$. By adding additional segments to the velocity function – which becomes composed by a maximum of 11 time intervals, instead of 9 – the traveling time can be further reduced by exploiting the acquired degrees of freedom and, in particular, by reducing the maximum acceleration modulus during the first and the last transient. The procedure can be explained with the aid of figure 5.11. Function EvalSP($\mathbf{h}_o, v_o(s), \tilde{v}(s)$) evaluates two split points – s_l and s_r, respectively – located just below the contact points between $v_o(s)$ and $\tilde{v}(s)$ (line 27). If $v_o(s)$ and $\tilde{v}(s)$ do not intersect for $a(s) = \bar{a}$, then improvements are not possible and $s_l \leftarrow -1$. Similarly, $s_r \leftarrow -1$ if the two functions do not touch each other for $a(s) = \underline{a}$.

If $s_l > 0$, then function Left_JzJ($v_o(s), s_l, \mathbf{h}_o$) (line 29) iteratively reduces the acceleration used for the first transient until feasibility is lost: s_i' moves left and the area enclosed by $v_o(s)$ increases, thus reducing the transient time. A similar strategy is applied to the final transient through function Right_JzJ($v_o(s), s_r, \mathbf{h}_o$) (line 31).

Chapter 6

Results

The SAv1 proposed in the previous chapter has been tested at the beginning by means of simulation tests and the obtained results are reported in section 6.1, then experiments were performed, in part, in a demonstration plant and, in part, in a real warehouse by considering actual operating conditions. In particular, the second set of tests covered a period of several months. The achieved results are discussed in sections 6.2 and 6.3. For what concerns SAv2, only simulation results have been performed since it is conceived to demonstrate that, through the heuristic procedure proposed, it is possible to obtain results comparable with the one of a nonlinear solver. However it is not used in a real warehouse since constant speed segments are almost removed and continuous acceleration and deceleration transients are present. Obtained results are presented in section 6.4.

6.1 Simulation results of SAv1

As early mentioned, the algorithm proposed in section 5.3 returns solutions for Problem 1 which are obtained through a heuristic strategy. The optimality of the results will be proven in the following by comparing them with the ones achieved by using a nonlinear programming algorithm, which evaluation times are 4 orders of magnitude higher.

Table 6.1: Comparison of the traveling times for the first three path-segments and corresponding evaluation times.

	Traveling times [s]			Evaluation times [s]	
	Segment 1	Segment 2	Segment 3	Profile 1&2	Profile 2&3
SP	20.15	12.50	15.49	$3.49 \cdot 10^{-3}$	$0.49 \cdot 10^{-3}$
SAv1 20%	14.55	10.08	9.74	$10.65 \cdot 10^{-3}$	$3.57 \cdot 10^{-3}$
SAv1 60%	13.97	9.93	9.38	$10.88 \cdot 10^{-3}$	$4.26 \cdot 10^{-3}$
IPOPT	13.51	9.52	9.31	$3.45 \cdot 10^{2}$	$2.41 \cdot 10^{2}$

SAv1 has been tested over a set of 1000 randomly generated path-segments. For each of them, the corresponding function $\tilde{v}(s)$ was evaluated through the approach proposed in section 5.2 and the velocity profiles were calculated by means of four methods:

1. an algorithm inspired to the one proposed in [91], which practically coincides with the Standard Planner (SP) described in section 5.1 and shown figure 5.2(a);
2. SAv1 with ST=20%;
3. SAv1 with ST=60%;
4. the IPOPT nonlinear programming algorithm [92].

For all the tests, the acceleration and the jerk bounds were posed equal to $\tilde{a} = 0.4$ ms^{-2} and to $\tilde{j} = 1.5$ ms^{-3}, respectively. All tests were carried out on a PC equipped with an Intel i7-6700HQ processor running at 2.60GHz.

Figure 6.1 visually compares the results achieved with the four strategies. For space reasons, the figure only shows the velocity profiles for the first three path-segments, but similar behaviors have been verified for all the curves of the test-set. As confirmed by table 6.1, the solutions found with the SP algorithm are evidently suboptimal if compared with the ones of the other strategies. Conversely, IPOPT and

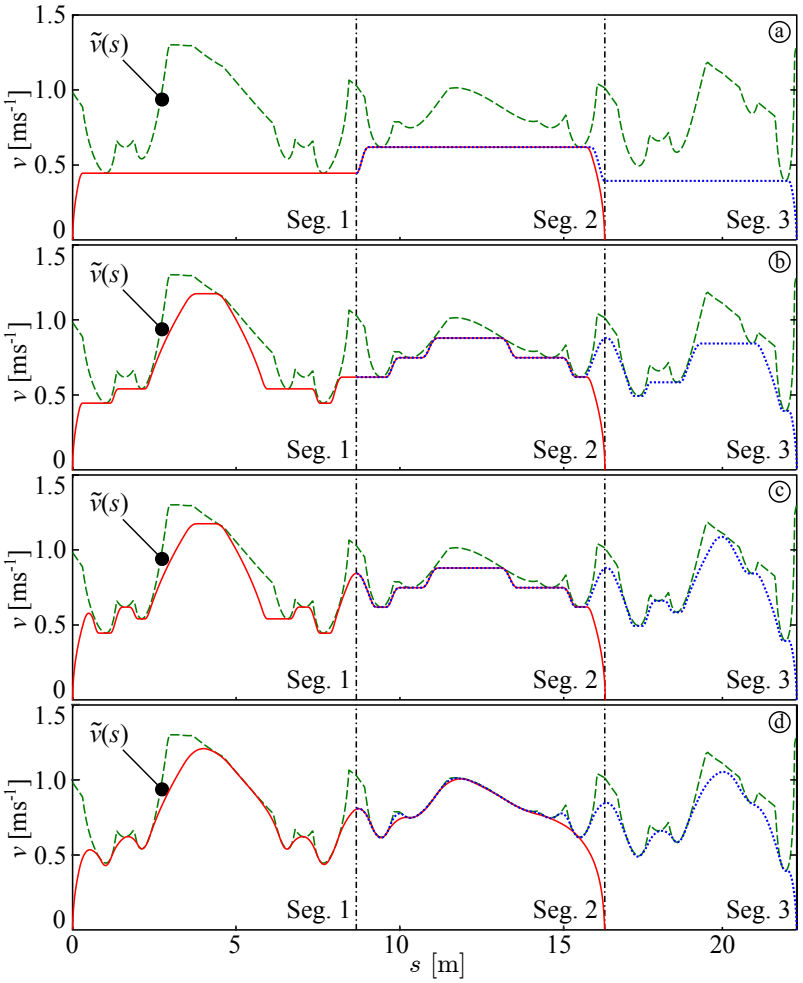


Figure 6.1: Velocity profiles for the first three path-segments of the 1000 elements test-set, obtained by means of the following strategies: (a) SP; (b) SAv1, ST=20%; (c) SAv1, ST=60%; (d) IPOPT. Vertical dash-dotted lines separate the path-segments. The dashed green line indicates $\tilde{v}(s)$. The initial velocity references are shown by means of solid red lines, while dotted blue lines are used for the updated profiles.

Table 6.2: Comparison of the traveling times and of the evaluation times for a composite path made of 1000 randomly chosen path-segments.

	Total Traveling Time [s]	Average Computing Time[s]	Max Computing Time [s]
SP	10897	$3.72 \cdot 10^{-5}$	$3.13 \cdot 10^{-3}$
SAv1 20%	8362	$2.27 \cdot 10^{-3}$	$34.95 \cdot 10^{-3}$
SAv1 60%	8121	$3.08 \cdot 10^{-3}$	$35.61 \cdot 10^{-3}$
IPOPT	8023	$62,16$	$1.12 \cdot 10^{3}$

SAv1 with ST=60% show very similar traveling times: this proves that SAv1 returns almost optimal solutions with evaluation times which are compatible with the real-time requirement.

It is interesting to notice that the solutions provided by SAv1 with ST=20% have traveling times that are only slightly suboptimal w.r.t. the ones returned by IPOPT but, as shown by figure 6.1, they are characterized by long constant-velocity segments, so that they should be generally preferred in order to reduce the vehicle and the load solicitations.

The cumulative results for the whole test-set are summarized in table 6.2: they practically confirm the same trend observed for the first three curves.

6.2 Experiments in the demo plant

SAv1 has been implemented on the LGV controller in order to execute experiments in a real plant. The first experiments were executed with a commercial LGV, produced by Elettric80 and named CB16, running in the demonstration plant shown in figure 6.2. The plant was implemented in the test shed owned by the Elettric80 company. It was conceived so as to include all the curve shapes typically used in industrial warehouses and, in particular, the ones considered critical in terms of safety or efficiency. In figure 6.2, circled numbers identify the start and the end points of each

curve, while the curves IDentification numbers (ID) are highlighted through different colors. Four categories of curves were considered:

- asymmetric 'U' curves (red segments - IDs: 1, 2 and 3);
- symmetric 'U' curves (brown segments - IDs: 10, 11 and 12);
- 90 degrees curves (magenta segments - IDs: 4, 5, and 6);
- 'S' curves (blue segments - IDs: 7, 8, and 9).

The straight segments of the plant are essentially used for single-curve tests: they are required to enter the curve at plain speed and to decelerate and stop the LGV when the curve has been executed. Plant curves were obtained through Bézier path primitives and were further classified on the basis of their maximum curvature. Three representative curvatures were selected for each category of the test set. More in details, each family includes one narrow curve, whose maximum curvature is higher than 1.7 m^{-1}, a medium one, whose curvature is in the range [0.8, 1.7] m^{-1}, and a large one, whose maximum curvature is lower than 0.8 m^{-1}. It is worth pointing out that narrow curves are seldom used in real plants because they would impose unacceptably low speeds: the experimental results proved that, conversely, the SAv1 allows reasonable traveling times also for them.

Three types of experiments were executed. Experiment 1 was conceived so as to test the safeness of the SAv1 velocity profiles and to show that the original SP can reach the same performances only by violating the safety conditions. Experiments 2 and 3 aimed at comparing the SAv1 performances with the ones achieved through the SP. More precisely, Experiment 2 concerned single-curve experiments, while Experiment 3 involved composite paths.

Experiment 1 is described in subsection 6.2.1. The comparative analysis for Experiment 2 is reported in subection 6.2.2, while subsection 6.2.3 proposes the outcomes of Experiment 3.

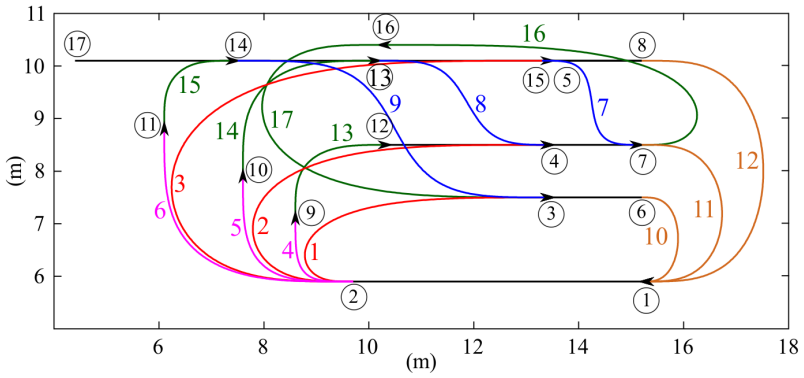


Figure 6.2: The demo plant used for testing the SAFERUN algorithm.

6.2.1 Experiment 1

As previously asserted, Experiment 1 tested the safeness of the SAv1 trajectories. The same experiment also showed that collisions occur with certainty if the SP is tuned so as to achieve the same performances of the SAv1.

Safety tests have been carried out by placing an obstacle along the curves according to the technique described in the following. An LGV was then driven along the plant by means of the SAv1 in order to verify that collisions were actually avoided and that the stop distance from the obstacle were the one theoretically predicted.

The proper choice of the obstacle positions required some preliminary work, since the worst case situations had to be identified. They depend on many factors like, for example, the velocity constraint $\tilde{v}(s)$ that the vehicle must fulfill. The general discussion relative to the synthesis of $\tilde{v}(s)$ was presented in section 5.2, however some additional details regarding its practical implementation are reported in the following. For each position s along the path, the control system can extrapolate a distance $\hat{d}(s)$ that can be safely traveled by avoiding collisions. Such distance depends on the path, on the vehicle shape, and on the shape of the safety area: safeness is guaranteed if the vehicle can be stopped before such distance is covered. In order to preserve a reasonable safety margin, which is added to account for potential tolerances, in real

plants the safety distance is always downrated as follows

$$\tilde{d}(s) := \hat{d}(s) - \Delta d(s),$$

where

$$\Delta d(s) := \min\{d_M, a\hat{d}(s)\}, \quad a \in [0,1]. \tag{6.1}$$

Practically, safety margin $\Delta d(s)$ is proportional to $\hat{d}(s)$ and it is superiorly saturated by d_M. The rationale of this choice is that, if $\hat{d}(s)$ is small, the vehicle speed is intrinsically kept small by the velocity planner, so that the safety margin can be reduced accordingly. For the tests, it was assumed $d_M = 0.2$ m and $a = 0.1$, which implies that the maximum safety margin was equal to 0.2 m. Both parameters can be modified depending on the plant and on the vehicle characteristics. $\tilde{v}(s)$ is then selected so as to guarantee that any vehicle moving at a speed $v(s) \leq \tilde{v}(s)$ could be stopped within a distance $\hat{d}(s) - \Delta d(s)$.

Some preliminary tests highlighted a behavior that was not originally foreseen and which required the implementation of an alternative method for the evaluation of $\tilde{d}(s)$. Alarms generated by SLSs are always "filtered" so as to avoid false positives caused by dust or reflected lights. As a consequence, an obstacle must be detected by several subsequent scans before the safety area is declared unclear. Such detection method evidently introduces a latency time Δt between the moment in which the obstacle is detected and the one in which the vehicle reacts to the emergency. The velocity upper bound used for the synthesis of the speed profile is clearly affected by such latency and, in particular, higher latencies impose lower velocity limits. During the latency time, the acceleration can be assumed constant, so that the vehicle approximately blindly covers the following distance

$$\Delta l = \dot{s}\,\Delta t + \frac{1}{2}\ddot{s}\Delta t^2 = v(s)\Delta t + \frac{1}{2}a(s)\Delta t^2.$$

The knowledge of Δl is clearly important because it influences the shape of $\tilde{v}(s)$. Unfortunately, $v(s)$ and $a(s)$ are computed at run time, so that they are still unknown when $\tilde{v}(s)$ is evaluated. A rough estimate for Δl can be obtained by assuming that actual velocity profiles $v(s)$ would follow very closely $\tilde{v}(s)$. Such consideration suggested the following procedure for the synthesis of the velocity constraint:

1. initially assume $\tilde{d}(s) = \hat{d}(s)$ and tentatively evaluate $\tilde{v}(s)$ with the procedure proposed in section 5.2;

2. evaluate:
$$\Delta l(s) = \tilde{v}(s)\Delta t + \frac{1}{2}\bar{a}\Delta t^2; \tag{6.2}$$

3. evaluate $\Delta d(s)$ according to (6.1) and find $\tilde{v}(s)$ by assuming $\tilde{d}(s) := \hat{d}(s) - \Delta l(s) - \Delta d(s)$.

It is worth remarking that the first term in (6.2) typically overestimates $\Delta l(s)$ since $\tilde{v}(s) \geq v(s)$. In the second term, $\bar{a}$ could be conservatively selected equal to the acceleration upper bound. However, the experience has shown that the overestimation introduced by the first term and the subsequent use of $\Delta d(s)$, allows one imposing $\bar{a} = 0$ in any practical case, with no impact on the safety.

The experimental tests have shown that the actual distance required for the complete stop of the vehicle in emergency situations is generally shorter than $\hat{d}(s) - \Delta d(s)$ because the optimal velocity profile evaluated through the SA is normally smaller than $\tilde{v}(s)$. This evidently leads to higher safety standards.

A single curve of the test set – more precisely the second one, i.e., an asymmetric 'U' curve with a medium curvature – will be extensively discussed in the following, while a concise analysis is provided for the remaining curves of the test set since similar considerations apply.

The velocity constraint is obtained by considering several contributions, for each point s of the curve, the more stringent between the safety and the kinematic speed limits is assumed. For such reason, safety tests were only performed in path positions characterized by a safety constraint more limiting than the kinematic one: in the other points safeness is certainly verified. Figure 6.3 shows the kinematic and the safety limits for the second curve expressed as function of s: for the selected curve all the points are potentially risky. The same figure reveals that seven test points were chosen. Some of them were placed in the most critical locations of the path, i.e., the ones in which the SAv1 velocity reference (solid blue line) touches the safety constraint (dashed red line): β, χ, δ, and ϕ. The remaining points (α, ε, and γ) were

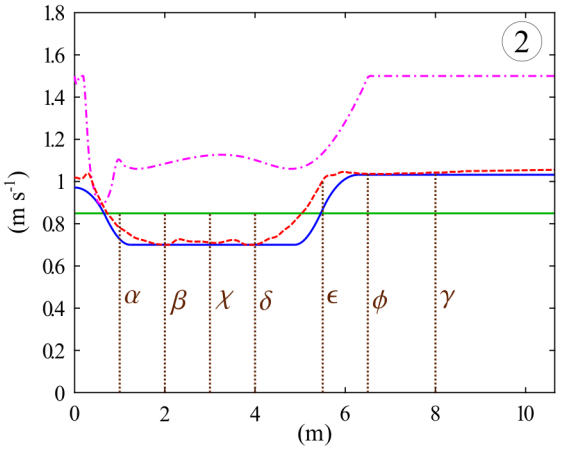


Figure 6.3: Velocity profile for the second curve of the demo plant. The following convention is assumed: the dash-dotted magenta line indicates the kinematic constraint; the dashed red line indicates safety constraint $\tilde{v}(s)$; the solid blue line indicates the SAv1 optimized profile; the solid green line indicates the SP profile used to emulate the SAv1 performances; the dotted vertical brown lines indicates the obstacles positions.

casually placed along the path for validation reasons. The physical collocation of the obstacles was then obtained by means of simulations. For example, figure 6.4a shows the procedure followed for point α. When the vehicle crosses point α (see the blue shape), the SLSs verify if the safety area (see the red shape) is clean from obstacles and, in that case, the LGV can move up to α' (see the green shape) without collision risks: the distance between α and α' coincides with $\hat{d}(s)$. In such situation, the first point in which a collision may occur is indicated by a cross: it is the place in which the obstacle must be posed for the test concerning the α point. The same procedure was followed for all the seven test locations. Figure 6.4b shows that, sometimes, more than one point needs to be tested, while figure 6.5 shows that the critical impact point may change depending on the vehicle position along the path.

The actual obstacle used for the experiments was a small carton box. The tests revealed that the SLSs reactivity is influenced by the box orientation, so that each sin-

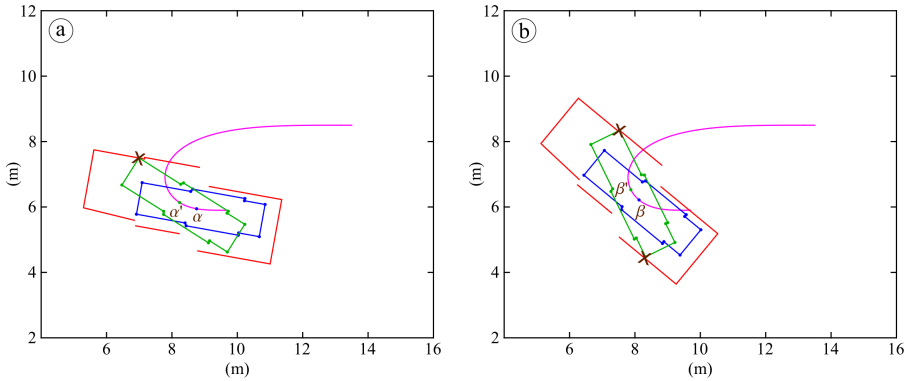


Figure 6.4: Most critical collision points for the α and β experiments on curve ID 2.

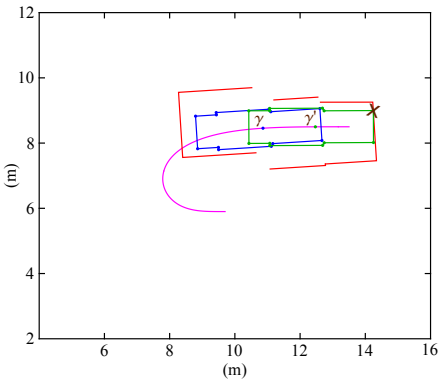


Figure 6.5: Most critical collision point for the γ experiment on curve ID 2.

gle experiment was repeated different times by turning the box in several directions: data reported in table 6.3 refer to the worst case situations detected for each one of the seven positions.

Tests were executed by first using the SAv1 and, then, the SP. The velocity of the SP was augmented in order to obtain the same time performances of SAv1. In the following, the SP with augmented speed will be synthetically referred as ASP. The first two columns of table 6.3 list the distances between LGV and obstacle that were measured after an emergency stop. The last column reports the theoretical values of Δd provided by (6.1) for the seven test points.

Table 6.3: Curve ID 2: comparisons between experimental Δd obtained with SAv1 and with the ASP. OBS indicates the obstacle identification letter, as defined in figure 6.3. A negative value indicates a collision. Δd is the theoretical stop distance obtained with (6.1).

OBS	**Experimental results**		Δd [m^{-2}]
	ASP [m^{-2}]	**SAv1** [m^{-2}]	
α	5.5	22.0	19.1
β	-1.0	14.0	13.7
χ	-3.0	13.5	12.7
δ	-1.0	15.0	13.7
ε	46.0	30.0	20.0
ϕ	49.0	20.0	20.0
γ	49.0	20.0	20.0

A rapid inspection of the table reveals that the ASP caused collisions in configurations β, χ, and δ. Point α is critical as well: the collision was avoided only because of the safety margin Δd. Conversely, with the SAv1 the final distance was always higher or equal to the theoretical value of Δd.

Table 6.3 can be deeply analyzed with the aid of figure 6.3:

- Configuration α: the SAv1 profile (solid blue line) is decreasing and it is smaller than the safety constraint, so that the measured stop distance was slightly higher than Δd. On the contrary, the constant velocity assumed for the ASP is slightly higher than the safe speed and, consequently, the detected stop distance was smaller than Δd. The situation is clearly unsafe: the collision was avoided thanks to the added safety margin Δd.
- Configurations β, χ, δ: the situation is similar in the three cases. The ASP profile evidently violates the safety constraint and, consequently, a collision occurred for all the three cases. Conversely, the SAv1 speed touches the safety constraint, so that the vehicle was stopped at a distance from the obstacle that

practically coincided with Δd.

- Configuration ε: the two planning techniques generate speeds that are much smaller than the safety constraint, hence the stopping distance from the obstacle was, in both cases, higher than Δd. A lower margin was obtained for SAv1 since the LGV was accelerating at the time the obstacle was detected.
- Configurations ϕ, γ: the two configurations are similar. The SAv1 speed is close to the safety constraint, so that the stop distance was similar to Δd. The ASP speed is much lower than the constraint, so that the LGV was stopped very far from the obstacle.

In conclusion, for the second curve of the test set the SAv1 always guaranteed stop distances which were higher than or equal to Δd, while the ASP caused collisions due to the high speeds required to obtain the same performances.

The same experiment was replicated for the other curves of the test set. However, a single test point was placed in the most critical location of each curve. Such location coincided with a configuration in which the SAv1 speed is very close to, or touches, the safety constraint which, in turn, is violated by the ASP (see, for example, positions β, χ, and δ of figure 6.3). The experimental results are listed in table 6.4, which highlights that, in many cases, the LGV was not able to execute the ASP profile because the required velocity was too high: the path tracking was lost and the vehicle was stopped by the emergency controls.

Summarizing, the results obtained for Experiment 1 confirm the expectations. For each curve of the test set, the SAv1 guarantees stop distances that are higher than $\Delta d(s)$, even considering multiple repetitions of the same experiment. Conversely, if the cruising speed of the SP is increased in order to achieve the same traveling times, three situations may occur: the path tracking is lost, or safety margin $\Delta d(s)$ is not satisfied or, in the worst cases, a collision occurs.

6.2.2 Experiment 2

Experiment 2 is a single curve performance test involving all the curves of the plant. The experiment was first simulated in the Matlab environment and, then, it was repli-

Table 6.4: Comparison between experimental Δd obtained with the SAv1 and with the ASP. ID indicates the curve ID. A negative value indicates a collision. Δd is the theoretical stop distance obtained through (6.1). "–" indicates that the speed was too high to correctly execute the curve.

ID	**Experimental results**		Δd [m^{-2}]
	ASP [m^{-2}]	**SAv1** [m^{-2}]	
1	–	15.0	13.6
3	-10.0	7.5	6.5
4	–	14.5	13.8
5	2.0	9.0	7.3
6	-2.0	15.0	12.7
7	–	17.0	15.2
8	–	11.5	10.3
9	2.0	17.0	16.5
10	–	13.5	11.2
11	–	13.0	12.2
12	3.0	16.0	14.9
13	–	16.0	14.8
14	9.0	20.5	20.0
15	–	11.5	10.0
16	–	12.5	11.6
17	9.5	23.0	20.0

cated in the demo plant, so as to verify if the expected figures were met. It is worth noticing, indeed, that velocity references were planned by neglecting the dynamic constraints, so that discrepancies between simulated and experimental results should have been possible. Table 6.5 summarizes the performances achieved for each one of the 17 curves. The second and the third columns list the simulated traveling times obtained with the SP and with the SAv1, respectively. Velocities for the SP coincide

with the ones that would actually be used during normal operations, i.e., they are all feasible with respect to the constraints. Traveling times do not include the launch and the stop segments, i.e., they only refer to the execution of the test curves. The expected percentage time-gains are shown in the fourth column and are defined as follows

$$gain := \frac{t_{SP} - t_{SAv1}}{t_{SP}} \cdot 100,$$

where t_{SP} and t_{SAv1} are the travelling times achieved with the SP and the SAv1, respectively.

Columns from 5 to 7 report the analogous figures acquired in the demo plant with the test vehicle. They are directly obtained from the LGV log files.

Simulated and actual performances are quite similar. The average time-gain, obtained by considering the whole set of curves, was equal to 31.17%, i.e., it was very close to the figure obtained through simulations, which was equal to 31.68%. It is important to point out that such result was achieved despite the reference speed of the actual vehicle may be influenced by components of the control system which were not simulated.

Figure 6.6 allows to visually compare the results obtained for the second curve. Simulated and actual speeds are indistinguishable for the SP, while for the SAv1 some small differences can be noticed. They are due to some LGV control algorithms that were not simulated and which slightly modify the velocity reference in order to accomplish some secondary tasks. As previously stated, they minimally affect traveling times, as proved by table 6.5.

Figure 6.6 also shows that velocity constraint $\widetilde{v}(s)$ is always strictly satisfied so that the safety requirement, as well as the kinematic constraints, are both fulfilled. Similar figures were obtained for all the curves of the test set.

6.2.3 Experiment 3

The last set of experiments was relative to the execution of 5 composite paths and to the acquisition of the corresponding traveling times. The plant layout is still the same of figure 6.2. The sequences of via-points adopted for each one of the composite

Table 6.5: Comparison between simulated and experimental results. ID is the curve identification number (see also figure 6.2).

ID	Simulation results			Experimental results		
	SP [s]	SAv1 [s]	Gain [%]	SP [s]	SAv1 [s]	Gain [%]
1	33.26	13.23	60.23	33.46	13.39	59.98
2	15.20	12.53	17.56	15.33	12.68	17.27
3	17.84	14.09	21.01	18.07	14.27	21.00
4	9.55	6.78	28.99	9.60	6.92	27.93
5	6.48	5.55	14.38	6.52	5.61	13.98
6	9.25	7.23	21.82	9.35	7.38	21.04
7	20.25	12.19	39.83	20.62	12.54	39.17
8	13.50	9.22	31.68	13.67	9.49	30.55
9	11.15	9.56	14.28	11.33	9.78	13.70
10	59.67	11.94	79.98	60.66	11.87	80.43
11	15.02	10.32	31.30	15.09	10.4	30.98
12	12.83	10.99	14.32	12.90	11.11	13.89
13	11.11	6.28	43.47	11.27	6.47	42.56
14	7.65	6.55	14.42	7.71	6.67	13.49
15	10.81	6.80	37.09	10.89	6.97	36.03
16	31.06	14.87	52.11	31.48	15.11	52.01
17	15.11	12.69	16.04	15.32	12.88	15.85

paths are listed in table 6.6.

The results achieved for Path 1 are summarized in table 6.7, whose columns are organized similarly to the ones of table 6.5. A good agreement between simulated and experimental results has been verified. As expected, the highest time-gains are detected for the segments with the highest curvatures, but the SAv1 also allowed consistent time savings during the execution of straight segments. In fact, vehicles driven with the SA enter and leave curvilinear paths at higher speeds and, conse-

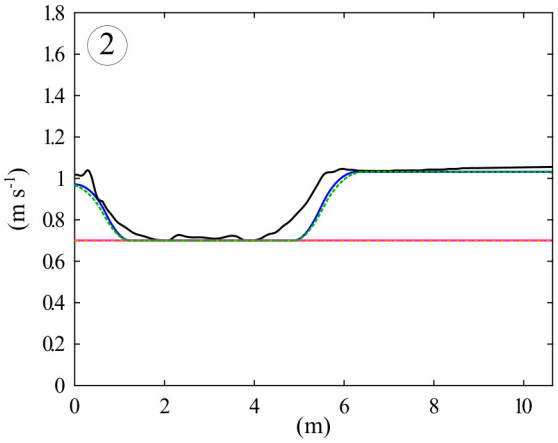


Figure 6.6: Asymmetric 'U' curve: comparisons between the simulated and the actual velocity functions obtained with the SP and with the SAv1 algorithms. A solid black line is used for the kinematic and safety constraint function $\widetilde{v}(s)$; a dash-dotted orange line is used for the experimental (LGV log files) SP velocity function; a dashed green line is used for the experimental (LGV log files) SAv1 velocity function; a solid magenta line is used for the simulated SP velocity function; a solid blue line is used for the simulated SAv1 velocity function.

quently, straight segments following or preceding such curves are executed in less time.

System performances can also be visually compared by means of figure 6.7. The SAv1 velocities are everywhere higher than the SP ones, apart from the positions in which the SAv1 profile coincides with the constraint minima. More evident improvements were achieved for curvilinear segments (the 2nd, the 3rd, the 5th, and the 7th), although time-gains were also detected for straight paths (the 1st, the 4th, and the 6th).

Simulated and experimental profiles are almost identical and differences, like in Experiment 2, are essentially due to controller behaviors that were not modeled. For example, in the highlighted detail, relative to the last curve of the composite path, the two references differ: every time the vehicle is stopped, the velocity function is

Table 6.6: The sequence of via-points for the 5 composite paths used in Experiment 3.

Path 1	$1 \rightarrow 2 \rightarrow 9 \rightarrow 12 \rightarrow 7 \rightarrow 1 \rightarrow 2 \rightarrow 4$
Path 2	$17 \rightarrow 8 \rightarrow 1 \rightarrow 2 \rightarrow 10 \rightarrow 13 \rightarrow 15 \rightarrow 7 \rightarrow 1$
Path 3	$17 \rightarrow 14 \rightarrow 3 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 11 \rightarrow 14$
Path 4	$1 \rightarrow 2 \rightarrow 5 \rightarrow 8 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6$
Path 5	$17 \rightarrow 13 \rightarrow 4 \rightarrow 7 \rightarrow 16 \rightarrow 3 \rightarrow 6$

Table 6.7: Comparison between simulated and experimental results for each segment of composite Path 1. ID identifies the via points of the composite path (see also figure 6.2).

ID	**Simulation results**			**Experimental results**		
	SP [s]	**SAv1 [s]**	**Gain [%]**	**SP [s]**	**SAv1 [s]**	**Gain [%]**
1 $\rightarrow$ **2**	7.37	7.16	2.92	8.23	8.00	2.77
2 $\rightarrow$ **9**	9.57	7.07	26.12	9.58	7.12	25.67
9 $\rightarrow$ **12**	11.11	6.32	43.11	11.26	6.47	42.58
12 $\rightarrow$ **7**	5.90	4.92	16.64	6.20	5.18	16.52
7 $\rightarrow$ **1**	15.02	10.32	31.30	15.10	10.41	31.05
1 $\rightarrow$ **2**	5.67	5.29	6.82	5.94	5.53	6.81
2 $\rightarrow$ **4**	16.47	14.22	13.68	19.05	17.16	9.89

automatically downscaled by the control system in order to approach the end-point at a very low speed. Such behavior guarantees a more accurate final positioning, but it prolongs traveling times.

The visual analysis for the other four composite paths is omitted, but similar considerations could be drawn. Conversely, the numerical analysis for the 5 composite paths is proposed in table 6.8. Average time-gains are quite different depending on the path composition. Higher gains are typical for routes obtained by composing many narrow curves with a minor number of straight segments.

The cumulative time-gain measured for the 5 composite paths was equal to 31.13%.

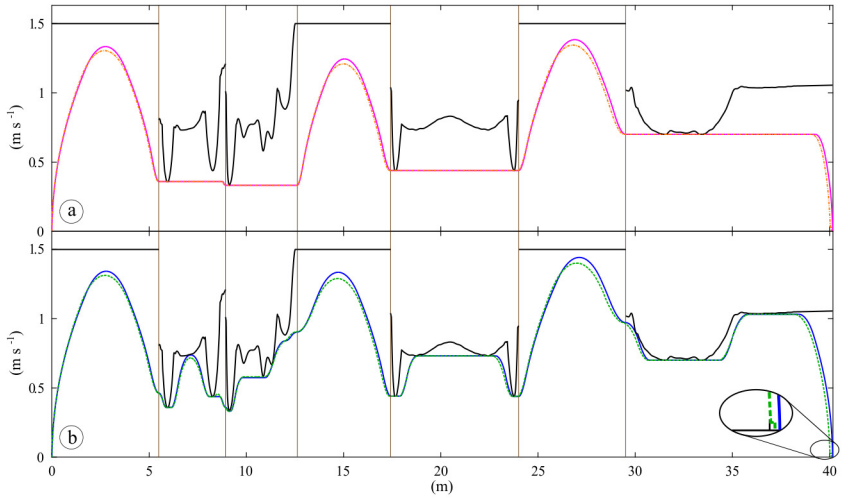


Figure 6.7: Comparison between simulated and experimental results for Path 1, obtained for (a) the SP and (b) the SAv1. The vertical brown lines separate the different segments of the composite path, while black solid lines are used for the velocity constraint. In (a) a solid magenta line is used for the simulated results, while a dash-dotted orange line is used for the experimental data. In (b) a solid blue line is used for the simulated results, while a dashed green line is used for the experimental data.

Table 6.8: Comparison between simulated and experimental results. ID is the path identification number.

ID	Simulation results			Experimental results		
	SP [s]	SAv1 [s]	Gain [%]	SP [s]	SAv1 [s]	Gain [%]
1	71.11	55.29	22.26	75.36	59.88	20.55
2	83.31	64.99	22.00	86.45	67.47	21.96
3	106.93	50.73	52.53	110.77	53.76	51.47
4	83.05	55.29	33.43	87.00	59.55	31.55
5	74.48	50.66	31.98	79.02	55.20	30.15

(a) *Model CB12 vehicle* (b) *Model LT3 vehicle*

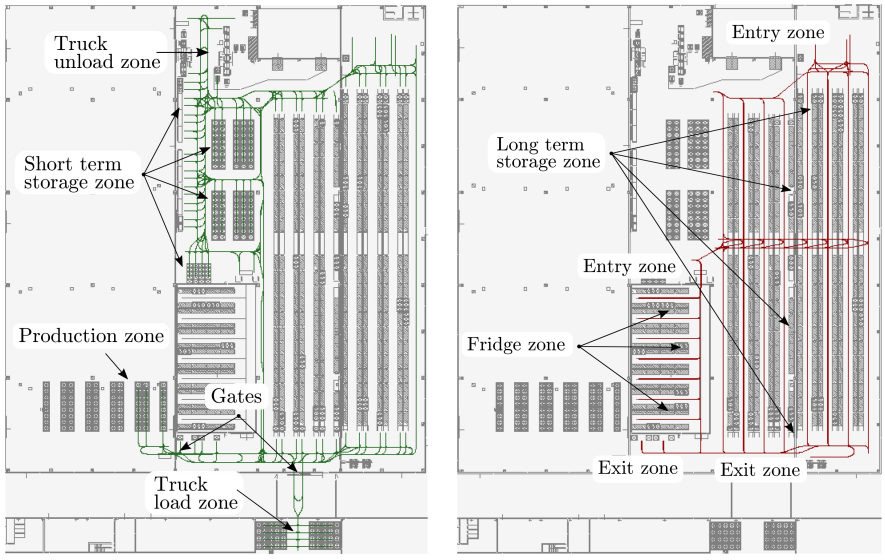
Figure 6.8: The two LGV models used in the industrial plant considered. (Courtesy of Elettric80).

It was very close to the simulated figure that was equal to 32.44%.

6.3 Experiments in an actual warehouse under real operative conditions

The tests proposed in section 6.2 were relative to an ideal situation concerning a single LGV traveling along the routes of an empty plant. Under actual operating conditions, the plant is shared among several vehicles and also with human operators, so that traffic problems may arise and lower time gains have to be expected.

For such reason, an extensive test campaign was executed in a real warehouse by considering actual operating conditions. The selected plant currently uses five LGVs



(a) (b)

Figure 6.9: Plant layouts for the CB12 vehicles (a) and for the LT3 vehicles (b).

subdivided into two categories: two model CB12 vehicles and three model LT3 vehicles. Both LGVs are shown in figure 6.8. The CB12 vehicles were equipped with the SAv1, while the LT3 machines were driven with the SP. The reason of such choice can be understood with the aid of figure 6.9, which shows the plant layouts for the two classes of vehicles. Working areas are clearly different, although some zones are shared. The CB12 vehicles are mainly used to transfer goods between different locations of the plant. Conversely, the LT3 vehicles are principally used to store pallets, so that they mainly operate within shelves: their paths are essential straight segments. As early pointed out, the SAv1 allows marginal time improvements for straight segments, so that upgrade costs were not justified for LT3 vehicles.

The SAv1 was preliminary tested in the warehouse by executing some selected composite paths, in order to verify its reliability. After such initial tuning phase, the SAv1 was used to drive the two upgraded vehicles during the warehouse daily operations. The performance comparisons proposed in the next subsections are based

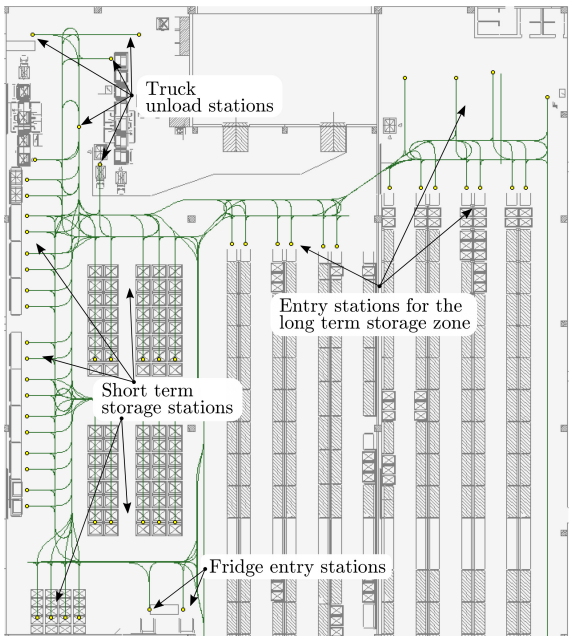
on the log data collected in the period from May to November 2016, i.e., when all LGVs were still driven by the SP, and the ones acquired in the period from September to November 2017, i.e., after the implementation of the SAv1. The results of section 6.3.1 refer to the performances of the sole CB12 vehicles, while the ones in section 6.3.2 extend the comparisons to all the plant vehicles.

Owing to the high number of curves, safety tests only involved the most critical segments, i.e., the ones with high curvatures, plus a set of randomly selected paths. The obtained results were perfectly congruent with the ones reported in section 6.2.1 and, for this reason, they are not discussed.

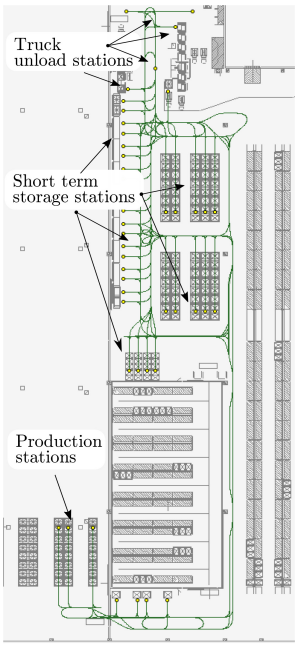
6.3.1 Statistics concerning the CB12 vehicles

In real plants, like the warehouse here considered, LGVs missions change continuously, so that the performance analysis cannot consider repetitive travels. For such reasons, missions were preliminarily classified depending on the travel typology. To this purpose, the vehicles docking stations were first grouped depending on their function. The warehouse used for the experiment has an entry area where raw materials are unloaded from trucks and an exit area used to load the final product on the trucks. Perishable materials are stored into a cooled region, while remaining materials are stored into standard shelf units. Pallets are also moved from the storing sections to the production area and viceversa. Such working organization suggested the following grouping scheme for the docking stations (the number of stations associated to each functionality is indicated within brackets):

- Truck unload stations (6)
- Production stations (3)
- Fridge entrance stations (2)
- Fridge exit stations (4)
- Short term storage stations (32 – actually used: 10)
- Entry points for long term storage stations (15)
- Exit points for long term storage stations (14)
- Truck load stations (8)



(a) Paths from the truck unload stations to fridge entry, to the long term entry, and to short term storage stations.

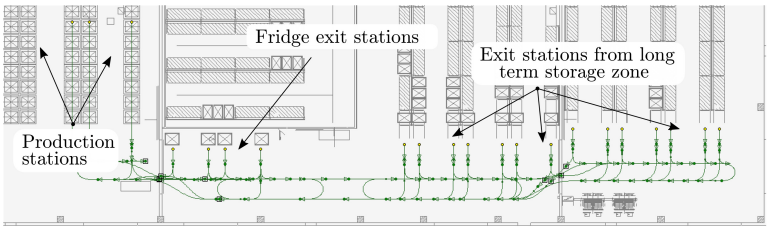


(b) Paths from the truck unload stations and from the short term storage stations to the production stations.

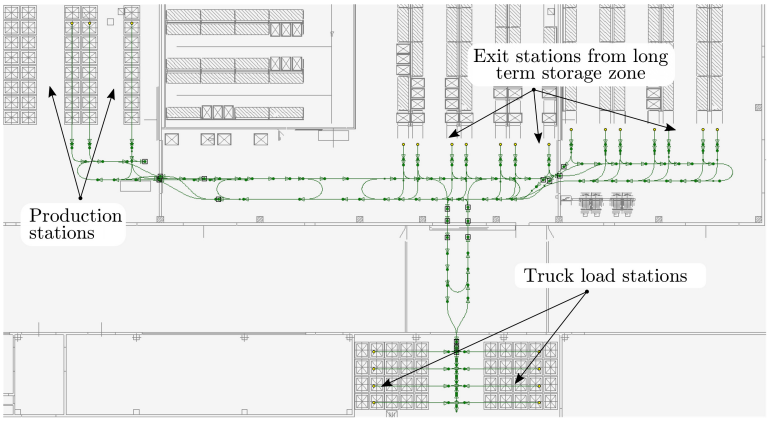
Figure 6.10: Travel categories 1 (a) and 2 (b).

The statistical study has been carried out by considering travels involving homogeneous working conditions. In particular, travels were grouped into the following four categories:

1. from the “Truck unload stations” to the “Fridge entry stations”, or to the “Short term storage stations”, or to the “Entry stations for long term storage zone” (see figure 6.10a);
2. from the “Truck unload stations” or from the “Short term storage stations” to the “Production stations” (see figure 6.10b);
3. from the “Fridge exit stations” or from the “Exit stations from long term stor-



(a) Paths from fridge exit stations and from the long term exit stations to the production stations.



(b) Paths from the production stations and from the long term exit stations to the truck load stations.

Figure 6.11: Travel categories 3 (a) and 4 (b).

age zone" to the "Production stations" (see figure 6.11a);

4. from the "Production stations" or from the "Exit stations from long term storage zone" to the "Truck load stations" (see figure 6.11b).

Statistics were acquired for each travel category. Table 6.9 reports the comparative results. In particular it contains, for each class of travel, the following data:

- number of station-to-station missions;
- the average mission times;
- the resulting time-gains.

Table 6.9: Performance comparisons involving the CB12 vehicles.

Category	Number of missions		Average times		Gain [%]
	2016	2017	2016 [s]	2017 [s]	
1	4862	1827	113.63	101.41	10.71
2	798	254	164.06	161.87	1.33
3	1976	471	118.72	109.64	7.65
4	893	284	131.56	130.12	1.09

Time-gains in table 6.9 evidently depend on the travel category. For example, missions of categories 2, 3 and 4 pass through (at least) one normally closed gate. The time required to open the gate imposes a stop to the LGV and, consequently, the SA does not operate for long periods: any stop – especially long ones – affects negatively the statistics, since average mission-times increase and percentage time-gains necessarily decrease. The most critical situations occur for the paths of category 4, since they often require crossing two gates.

Mission times are also affected by the LGV backward movements required to approach a load/unload station. Backward movements worsen the average performances, since they impose an additional stop in order to revert the motion direction and, furthermore, they require a point-to-point movement executed at a very low speed (0.3 ms^{-1} according to the "EN ISO 1525:1997" [87]), during which the SAv1 does not operate.

The plant traffic can be more or less invasive depending on the mission category and, consequently, its effect on the statistics may be different. Traffic stops may be caused by human operators or by the interaction with other LGVs. For example, category 2 missions intersect the LT3 paths in many points, so that CB12 vehicles are stopped very frequently. Finally, the SA performances depend on the number of straight segments of each mission: improvements are essentially expected for curvilinear segments.

The average traveling time, evaluated over the entire set of missions, was equal

Table 6.10: Performance comparisons involving both the CB12 and the LT3 vehicles.

LGV type	Number of missions		Average times		Gain [%]
	2016	2017	2016 [s]	2017 [s]	
LT3	8095	2639	157.91	156.28	1.03
CB12	8529	2836	121.41	111.11	8.48

to 121.41 s for the SP, while it reduced to 111.11 s when the SAv1 was adopted. The average time-gain for the two CB12 vehicles, obtained by considering a weighting factor which depends on the number of travels, was equal to 8.48%.

6.3.2 Statistics concerning all the vehicles

The SAv1 installed on the CB12 vehicles has a positive influence also on the performances of the LT3 vehicles which still use the SP. A short analysis is proposed in the following in order to quantify the cumulative effects on the plant efficiency. The same statistical analysis proposed in subsection 6.3.1 has been repeated by also considering the performances of the LT3 machines, i.e., the ones that were not upgraded. Table 6.10 reports the results acquired for the two classes of LGVs. It is worth noticing that a small time-gain was detected also for the LT3 vehicles: they found more frequently an empty route, so that traveling times reduced accordingly.

The statistical analysis has been finally repeated by simultaneously considering all the five LGVs. The average traveling time, measured when all vehicles were driven through the SP, was equal to 139.18 s. Conversely, in the period from September to November 2017, the average traveling time decreased to 132.88 s. As a consequence, for the considered plant and time periods, the SAv1 allowed a weighted, omni-comprehensive time-gain equal to 4.53%. This result was achieved despite only two vehicles were upgraded over the five used in the plant. Since the working conditions in the two test periods were exactly the same and no further changes were introduced in the plant, it can be asserted that efficiency improvements were only due to the SAFERUN Algorithm.

6.4 Simulation results of SAv2

The SAv2 has been extensively tested so as to check its performances in terms of optimality of the solution and convergence time. Once again the results provided by the SAv2 have been compared with the ones obtained through IPOPT [92]. This time, in order to obtain fair comparisons, a different strategy has been used: similar degrees of freedom were conferred to both algorithms, which use the same piecewise constant-jerk planning primitive. According to (5.28)–(5.30), acceleration, velocity, and position are continuous functions over time interval $[0, t_f]$, where $t_f := \sum_{l=1}^{m} t_l$ is the total traveling time, and t_l is the time associated to each constant-jerk segment. m is the total number of time intervals.

By defining a bounding box $\mathscr{B} \subset (\mathbb{R}^+)^m$ which contains the origin and vector $\mathbf{t} := [t_1, t_2, \ldots, t_m]^T$, the IPOPT problem was formulated as follows

$$\min_{\mathbf{t} \in \mathscr{B}} \{t_f\} \tag{6.3}$$

subject to ($i = 1, 2, \ldots, m$)

$$\underline{j} \leq j_i \leq \overline{j} \tag{6.4}$$

$$\underline{a} \leq a_i(t) \leq \overline{a}, \quad \forall t \in [0, t_i] \tag{6.5}$$

$$0 \leq v_i(t) \leq \tilde{v}[s(t^*)], \quad \forall t \in [0, t_i], \quad t^* = \sum_{l=1}^{i-1} t_l + t \tag{6.6}$$

$$a_1(0) = 0, v_1(0) = 0, s_1(0) = 0 \tag{6.7}$$

$$a_m(t_m) = 0, v_m(t_m) = 0, s_m(t_m) = s_f, \tag{6.8}$$

where s_f is the total path length. Constraints (6.4)–(6.6) refer to the given kinematic bounds, while (6.7) and (6.8) represent interpolating conditions. Evidently, (6.3)–(6.8) is a semi-infinite optimization problem, which is solved by discretizing (6.5) and (6.6).

It must be pointed out that, while the nonlinear solver assumes an assigned value for m, both versions of the SA selects the number of subintervals depending on the complexity of the velocity constraint. In order to have a fair comparison between the

two methods, each planning problem was first solved with the SAv2 and the obtained value of m was then used for IPOPT, thus guaranteeing "similar" degrees of freedom. Notice the use of term "similar": while IPOPT can assume any jerk value in $\left[\underline{j}, \overline{j}\right]$, the SA can only assume $j_i = \underline{j}$, $j_i = 0$ or $j_i = \overline{j}$, so that more degrees of freedom are actually granted to the nonlinear solver.

The first test experiments were executed by considering the same LGV plant proposed in section 6.2 and shown in figure 6.2. Comparisons are made by only considering the traveling times along the 17 curves, i.e., by neglecting the acceleration and deceleration transients. The second column of table 6.11 shows the performances of the SAv2, while the third one proposes the analogous times which were obtained in section 6.2 with the SAv1: in all the cases, traveling times reduce. The percentage gains reported in the fourth column are obtained by means of the following equation

$$gain = \frac{t_{SAv2} - t_{SAv1}}{t_{SAv2}} \cdot 100 \,, \tag{6.9}$$

where t_{SAv2} and t_{SAv1} are the traveling times for the second and the first versions of SA, respectively. Since gains in the fourth column of table 6.11 are all negative, the SAv2 always performs better than the SAv1. The average time-gain is equal to -8.73% with a standard deviation equal to 5.78%.

The same velocity planning problem was then solved with IPOPT. In the first test campaign, the SP solution was used as the initial solution for the solver (see the solid red profile shown in figure 5.5). The traveling times obtained for the 17 paths are listed in the 5th column of table 6.11 and are referred as IPOPT1. The percentage gains in the 6th column are obtained through the following equation

$$gain = \frac{t_{SAv2} - t_{IPOPT}}{t_{SAv2}} \cdot 100 \,, \tag{6.10}$$

hence positive values indicate that IPOPT returns faster trajectories. The analysis of the 6th column highlights that in many cases the SAv2 provides better minimizers and, indeed, the average time-gain is equal to -4.68%, with a standard deviation equal to 9.29%. The reason of such result is that (6.3)–(6.8) is a highly multimodal problem, so that the nonlinear solver is easily entrapped in local minima.

Table 6.11: Traveling times and gains of SAv2 compared with the previous SAv1 and IPOPT.

	SAv2	SAv1		IPOPT1		IPOPT2	
ID	time [s]	time [s]	gain [%]	time [s]	gain [%]	time [s]	gain [%]
1	12.68	13.23	-4.34	12.74	-0.45	12.68	0.00
2	12.21	12.53	-2.62	12.20	0.10	12.17	0.31
3	13.57	14.09	-3.83	13.42	1.11	13.40	1.26
4	5.77	6.78	-17.50	5.80	-0.59	5.77	0.00
5	5.34	5.55	-3.93	5.41	-1.28	5.34	0.00
6	6.65	7.23	-8.72	6.58	0.99	6.64	0.14
7	10.69	12.19	-13.93	13.17	-23.19	10.65	0.42
8	8.33	9.22	-10.68	10.88	-30.65	8.27	0.73
9	9.05	9.56	-5.64	9.01	0.45	9.00	0.53
10	9.80	11.94	-21.84	9.61	1.95	9.80	0.00
11	9.47	10.32	-8.98	10.59	-11.79	9.43	0.45
12	10.54	10.99	-4.27	10.55	-0.12	10.54	0.00
13	5.60	6.28	-12.14	5.67	-1.18	5.58	0.47
14	6.29	6.55	-4.13	6.26	0.50	6.27	0.30
15	5.79	6.80	-17.44	6.64	-14.59	5.74	0.87
16	14.02	14.87	-6.06	14.02	0.01	14.00	0.12
17	12.39	12.69	-2.42	12.50	-0.91	12.39	0.00

In order to improve the performances of IPOPT, the same tests were executed by initializing the algorithm with the SAv2 solutions. Consequently, the IPOPT performances can only be equal or better than the SAv2 ones. The achieved results are reported in the last two columns of table 6.11, where they are referred as IPOPT2. The average time-gain was equal to 0.34% with a standard deviation equal to 0.38%. Figure 6.12 shows the velocity profiles obtained for the 11th curve: the SAv2 and the IPOPT2 solutions are clearly equivalent, while IPOPT1 was stuck at a local minima.

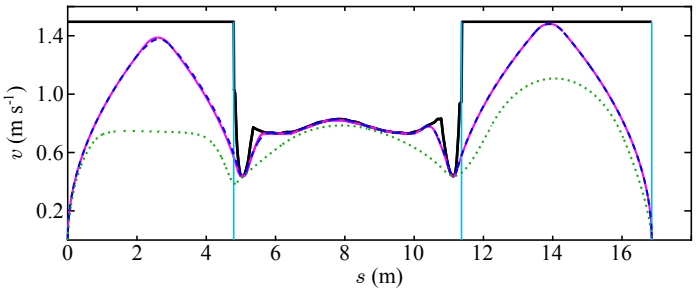


Figure 6.12: The velocity profile generated for curve 11 in figure 6.2. The segment is preceded and followed by straight segments used to accelerate and stop the LGV. The magenta solid line is the solution provided by the SAv2, while the blue dashed line and the green dotted one are the solutions found by IPOPT when starting from the SAv2 solution or from the SP one, respectively.

Some preliminary considerations can be drawn. Differently from IPOPT, the SAv2 does not suffer of convergence problems, so that it always returns optimized feasible solutions. Only marginal improvements are obtained by starting IPOPT from the SAv2 solutions: despite the SAv2 is based on a heuristic technique it provides almost optimal solutions. Additionally, the computational times of the two approaches differ for several orders of magnitude: the SAv2 has shown an average computational time equal to $5.92 \cdot 10^{-2}$ s, while for IPOPT1 and for IPOPT2 it was equal to $2 \cdot 10^{4}$ s and to $7 \cdot 10^{3}$ s, respectively. Such times were obtained with an Intel i7-2670QM processor running at 2.20Ghz and in a MATLAB environment: significantly better performances are expected from C implementations, but the IPOPT strategy would remain unsuited for real-time applications.

The performance test was then repeated by considering an extended set of 1000 curves and a working environment similar to the one of a real warehouse. The velocity functions were planned by initially generating a profile for the first two adjacent segments and considering a rest-to-rest motion (see the solid blue line in figure 6.13). Then, a new segment was added to the chain and the velocity profile was updated by generating a move-to-rest transient involving the last two segments (see the dashed green line in figure 6.13). The process was repeated until all the 1000 curves were

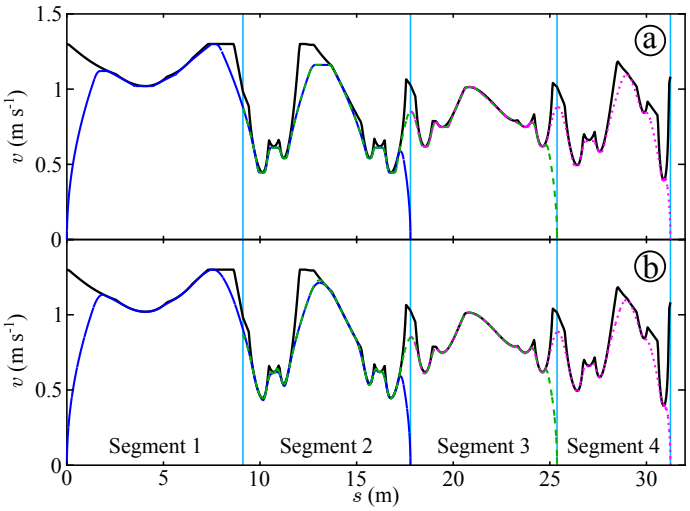


Figure 6.13: Velocity profiles obtained with SAv2 (a) and with IPOPT (b). Both algorithms first plan $v(s)$ for segments 1 and 2 (solid blue line), then for segments 2 and 3 (dashed green line), and finally for segments 3 and 4 (dotted magenta line). $\tilde{v}(s)$ is represented through a solid black line.

evaluated. Such operating mode mimics the typical working conditions of real warehouses, in which the routes of autonomous vehicles are decided in real time by adding new curves to already planned paths.

The experiment was first executed with SAv2 and, then, it was repeated with IPOPT. The initial solutions for IPOPT were still given by the SP profiles. The SAv2 returned feasible solutions for all the 1000 test cases, while very low success rates were achieved with IPOPT: the nonlinear programming algorithm improved the initial guesses in only 615 cases because of the "entrapping" problem early pointed out. In order to improve the performances of IPOPT, the program was executed several times by perturbing the initial guesses. Because of the very high computational times – the management of each curve requires more than 1 hour – repeated attempts only involved critical configurations and a maximum of 5 runs for each of them. Despite the use of such strategy, in 28 cases IPOPT was still not able to improve the ini-

tial solution and in 283 cases returned traveling times longer than the ones provided by SAv2. By considering performance index (6.10) and by limiting the analysis to the 972 curves in which IPOPT improved the initial guess, an average gain equal to 0.25% was measured with a standard deviation equal to 3.74%. Hence, the two planning strategies are potentially equivalent: slightly better average performances have been obtained with IPOPT but, in any case, differences between the two methods are limited to $\pm 4\%$. The actual difference between the two algorithms is represented by the average evaluation time: for the SAv2 it was equal to $5.39 \cdot 10^{-2}$ s, while for IPOPT it reached $4 \cdot 10^{3}$ s. In facts, the SA heuristic strategy represents a valid alternative to nonlinear solvers which, in turn, cannot be used in real-time contexts.

Conclusions

This thesis has dealt with the optimal trajectory planning problem for robotic systems subject to kinematic and safety constraints. Proposed methods allow the real-time management of the planning problem, by simultaneously considering and avoiding possible critical configurations which could appear during the motion.

The technique proposed in the first part of the work aims at the automatic handling of wrist singularities occurring in non-redundant anthropomorphic manipulators. It is explicitly suited to trajectories planned on-the-fly, being totally based on a real-time strategy. Differently from alternative methods in the literature, it preserves the user-defined Cartesian path and time-law. The orientation of the tool-frame is slightly modified in order to avoid critical configurations, however the maximum displacement from the nominal trajectory is bounded and predefined. In addition, joint velocities and accelerations are not generically reduced, but they are explicitly limited within assigned bounds. In particular, a procedural method has been proposed for the tuning of the algorithm, so as to make it more deterministic and to increase success rates. Results were experimentally validated on a real manipulator by means of extensive tests. A very good agreement has been verified between simulated and experimental results.

At the moment, some preliminary tests have been performed by considering curvilinear paths and by assuming the algorithm “as it is”. Clearly, for curvilinear paths Proposition 1 does not apply, so that multiple singular points may appear along a single curve. The success rate will necessarily decrease depending on the path curvature and on the orientation of the osculating-circle associated to the path, but pre-

liminary statistics show very promising performances.

The real-time velocity planner for autonomous vehicles, proposed in the second part of the thesis, generates reference signals which fulfill a set of given bounds on velocity, acceleration, and jerk. Differently from other known planners, the velocity constraint is variable and, more precisely, it depends on the position of the vehicle along the path. It is synthesized such to guarantee safe motions and the fulfillment of several kinematic constraints. In particular, one of the planner characteristic is that it can analytically handle safety concerns which were early managed through a trial-and-error approach. The variability of the speed bound prevents one adopting other known real-time planners. The problem could have been potentially solved by means of nonlinear programming strategies, but their computational burden is not compatible with the real-time requirement which characterizes LGV applications. The planning strategy proposed solves all that issues.

A set of comparative tests have proved that feasible solutions can be generated with short evaluation times: a novel velocity reference can be obtained, in the worst case, in less than $5 \cdot 10^{-2}$ s and, consequently, trajectories can be updated on-the-fly. Simultaneously, solutions provided are characterized by traveling times which are comparable to the ones obtained through a nonlinear programming algorithm. A second version of the velocity planner has been studied in order to improve the obtained solutions and to achieve the same performances of a nonlinear solver.

A set of real experiments were carried out, at first, by using a demonstration plant. The experimental results highlighted performances which are very close to the expected ones. This is very important since it implies that good predictions of the plant behavior can be achieved through the execution of preliminary simulations. Subsequently, the velocity planner has been implemented in an industrial plant and it has gone through a significant validation period under real operative conditions. During such period, which lasted three months, the planner proved its reliability; indeed it remains operative in that industrial plant. Elettric80 will plan to upgrade the existent industrial plants and to propose the novel planning methodology to further customers.

Summarizing, the SA represents an important step toward the complete automa-

tion of the design phase for LGV based plants. Next steps will concern the automatic selection of curves and safety areas, which will be chosen so as to increase the plant efficiency while preserving safe operating conditions.

Bibliography

[1] M. Raineri, C. Guarino Lo Bianco, M. Locatelli, and S. Perri, "A real-time strategy for the management of kinematic singularities: new progresses," in *Int. Conf. Meth. and Models in Autom. and Robot., (MMAR16)*, Międzyzdroje, Poland, Aug. - Sept. 2016.

[2] C. Guarino Lo Bianco and M. Raineri, "An automatic system for the avoidance of wrist singularities in anthropomorphic manipulators," in *IEEE Int. Conf. Autom. Sci. and Eng., (CASE17)*, Aug 2017, pp. 1302–1309.

[3] C. Guarino Lo Bianco and M. Raineri, "An Experimentally Validated Technique for the Real-Time Management of Wrist Singularities in Nonredundant Anthropomorphic Manipulators," *IEEE Trans. Control Syst. Technol.*, pp. 1–10, 2019.

[4] G. Schreiber, M. Otter, and G. Hirzinger, "Solving the singularity problem of non-redundant manipulators by constraint optimization," in *IEEE/RSJ Int. Conf. Intell. Robots and Syst., (IROS99)*, vol. 3, 1999, pp. 1482–1488.

[5] W. Decre, H. Bruyninckx, and J. De Schutter, "Extending the iTaSC constraint-based robot task specification framework to time-independent trajectories and user-configurable task horizons," in *IEEE Int. Conf. Robot. and Autom., (ICRA13)*, May 2013, pp. 1941–1948.

[6] Y. Huang, Y. Yong, Y. Chiba, T. Arai, T. Ueyama, and J. Ota, "Kinematic control with singularity avoidance for teaching-playback robot manipulator system," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 729–742, Apr. 2016.

[7] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: design for real-time applications," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 42–52, 2003.

[8] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of timeoptimal, jerk-limited trajectories," in *IEEE/RSJ Int. Conf. Intell. Robots and Syst., (IROS08)*, 2008, pp. 3248–3253.

[9] T. Kröger and F. M. Wahl, "On-line trajectory generation: basic concepts for instantaneous reactions to unforeseen events," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 94–111, Feb. 2010.

[10] M. Raineri, S. Perri, and C. Guarino Lo Bianco, "Online velocity planner for Laser Guided Vehicles subject to safety constraints," in *IEEE/RSJ Int. Conf. Intell. Robots and Syst., (IROS17)*, Sept 2017, pp. 6178–6184.

[11] ——, "Safety and efficiency management in LGV operated warehouses," *Robot. and Comput.-Integr. Manuf.*, vol. 57, pp. 73–85, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584518302631

[12] M. Raineri, F. Ronchini, S. Perri, and C. Guarino Lo Bianco, "Optimality criteria for the path planning of autonomous industrial vehicles," in *Advances in Robotics Research: From Lab to Market: ECHORD++: Robotic Science Supporting Innovation*, A. Grau, Y. Morel, A. Puig-Pey, and F. Cecchi, Eds. Cham: Springer International Publishing, Jan. 2020, pp. 125–140. [Online]. Available: https://doi.org/10.1007/978-3-030-22327-4_7

[13] M. Raineri and C. Guarino Lo Bianco, "Jerk limited planner for real-time applications requiring variable velocity bounds," in *IEEE Int. Conf. Autom. Sci. and Eng., (CASE19)*, Aug. 2019, pp. 1611–1617.

[14] A. Liégeois, "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms," *IEEE Trans. Syst., Man, Cybern.*, vol. 7, no. 12, pp. 868–871, Dec 1977.

[15] J. Baillieul, J. Hollerbach, and R. Brockett, "Programming and control of kinematically redundant manipulators," in *IEEE Conf. Decision Control, (CDC84)*, Las Vegas, NV, Dec. 1984, pp. 768–774.

[16] A. Maciejewski and C. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 109–117, 1985.

[17] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-Priority Based Redundancy Control of Robot Manipulators," *Int. J. Robot. Res.*, vol. 6, no. 2, pp. 3–15, 1987.

[18] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators ," *IEEE Trans. Robot. Autom.*, vol. 13, no. 3, pp. 398–410, 1997.

[19] Y. Nakamura and H. Hanafusa, "Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control," *J. Dynamic Syst. Meas. Control*, vol. 108, no. 3, pp. 163–171, Sept. 1986.

[20] C. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 1, pp. 93–101, Jan./Feb 1986.

[21] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE J. Robot. Autom.*, vol. 4, no. 4, pp. 403–410, Aug 1988.

[22] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space

augmentation and task priority strategy," *Int. J. Robot. Res.*, vol. 10, no. 4, pp. 410–425, 1991.

[23] L. Everett, J. Colson, and B. Mooring, "Automatic singularity avoidance using joint variations in robot task modification," *IEEE Robot. Autom. Mag.*, vol. 1, no. 3, pp. 13–19, Sept 1994.

[24] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Trans. Control Syst. Technol.*, vol. 2, no. 2, pp. 123–134, Jun 1994.

[25] C. Qiu, Q. Cao, and S. Miao, "An on-line task modification method for singularity avoidance of robot manipulators," *Robotica*, vol. 27, pp. 539–546, 2009.

[26] W. Xu, J. Zhang, B. Liang, and B. Li, "Singularity analysis and avoidance for robot manipulators with nonspherical wrists," *IEEE Trans. Ind. Electron.*, vol. 63, no. 1, pp. 277–290, Jan 2016.

[27] P. From, J. Gunnar, and J. Gravdahl, "Optimal Paint Gun Orientation in Spray Paint Applications - Experimental Results," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 2, pp. 438–442, Apr. 2011.

[28] A. Bemporad, T.-J. Tarn, and N. Xi, "Predictive path parameterization for constrained robot control ," *IEEE Trans. Control Syst. Technol.*, vol. 7, no. 6, pp. 648–656, Nov. 1999.

[29] D. Lam, C. Manzie, and M. C. Good, "Model Predictive Contouring Control for Biaxial Systems," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 2, pp. 552–559, March 2013.

[30] A. Hladio, C. Nielsen, and D. Wang, "Path Following for a Class of Mechanical Systems," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 6, pp. 2380–2390, Nov 2013.

[31] M. Böck and A. Kugi, "Real-time Nonlinear Model Predictive Path-Following Control of a Laboratory Tower Crane," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 4, pp. 1461–1473, July 2014.

[32] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1505–1511, July 2017.

[33] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotics manipulators along specified paths," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 3–17, 1985.

[34] O. Dahl and L. Nielsen, "Torque-limited path following by online trajectory time scaling," *IEEE Trans. Robot. Autom.*, vol. 6, no. 5, pp. 554–561, 1990.

[35] D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *J. Robot. Syst.*, vol. 17, no. 5, pp. 233–249, 2000.

[36] J. Moreno-Valenzuela and E. Oronzco-Manríquez, "A new approach to motion control of torque-constrained manipulators by using time-scaling of reference trajectories," *J. Mech. Sci. Technol.*, vol. 23, no. 12, pp. 3221–3231, Dec. 2009.

[37] C. Guarino Lo Bianco and F. Ghilardelli, "A discrete-time filter for the generation of signals with asymmetric and variable bounds on velocity, acceleration, and jerk," *IEEE Trans. Ind. Electron.*, vol. 61, no. 8, pp. 4115–4125, Aug 2014.

[38] ——, "Real-Time Planner in the Operational Space for the Automatic Handling of Kinematic Constraints," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 3, pp. 730–739, 2014.

[39] F. Lange and M. Suppa, "Predictive path-accurate scaling of a sensor-based defined trajectory," in *IEEE Int. Conf. Robot. and Autom, (ICRA14)*, May 2014, pp. 754–759.

[40] V. Potkonjaka, G. S. Dordević, D. Kostić, and M. Rašić, "Dynamics of anthropomorphic painting robot: Quality analysis and cost reduction," *Robot. and Autonomous Syst.*, vol. 32, no. 1, pp. 17–38, 2000.

[41] D. Conner, A. Greenfield, P. Atkar, A. Rizzi, and H. Choset, "Paint deposition modeling for trajectory planning on automotive surfaces," *IEEE Trans. Autom. Sci. Eng.*, vol. 2, no. 4, pp. 381–392, Oct 2005.

[42] J. Peng, Q. Chen, J. Lu, J. Jin, and C. van Luttervelt, "Real time optimization of robotic arc welding based on machine vision and neural networks," in *IEEE Int. Conf. Ind. Electron., Control, and Instr., (IECON98)*, 1998, pp. 1279–1283.

[43] W. Tillmann, E. Vogli, and B. Krebs, "Influence of the spray angle on the characteristics of atmospheric plasma sprayed hard material based coatings," *J. of Thermal Spray Technol.*, vol. 17, no. 5-6, pp. 948–955, 2008.

[44] P. From and J. Gravdahl, "A Real-Time Algorithm for Determining the Optimal Paint Gun Orientation in Spray Paint Applications," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 4, pp. 803–816, 2010.

[45] F. Ghilardelli, C. Guarino Lo Bianco, and M. Locatelli, "Smart changes of the end-effector orientation for the automatic handling of singular configurations," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 4, pp. 2154–2164, Aug. 2016.

[46] J. J. Craig, *Introduction to robotics mechanics and control*, 3rd ed. Upper Saddle River, New Jersey: Pearson Prentice Hall, 2005.

[47] C. Guarino Lo Bianco and F. Wahl, "A novel second order filter for the real-time trajectory scaling," in *IEEE Int. Conf. Robot. and Autom., (ICRA11)*, Shanghai, China, May 2011, pp. 5813–5818.

[48] O. Gerelli and C. Guarino Lo Bianco, "A discrete-time filter for the on-line generation of trajectories with bounded velocity, acceleration, and jerk," in *IEEE Int. Conf. Robot. and Autom., (ICRA10)*, Anchorage, AK, May 2010, pp. 3989–3994.

[49] L. Sciavicco, B. Siciliano, L. Villani, and G. Oriolo, *Robotics: Modelling, planning and Control*, ser. Advanced Textbooks in Control and Signal Processing. Berlin, Germany: Springer-Verlag, 2011.

[50] W. Khalil and J.-F. Kleinfinger, "A new geometric notation for open and closed-loop robots," in *IEEE Int. Conf. Robot. and Autom., (ICRA86)*, San Francisco, CA, 1986, pp. 1174–1180.

[51] H. Martínez-Barberá and D. Herrero-Pérez, "Autonomous navigation of an automated guided vehicle in industrial environments," *Robot. and Comput.-Integr. Manuf.*, vol. 26, no. 4, pp. 296–311, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584509000994

[52] J. Corrales, F. Candelas, and F. Torres, "Safe human–robot interaction based on dynamic sphere-swept line bounding volumes," *Robot. and Comput.-Integr. Manuf.*, vol. 27, no. 1, pp. 177–185, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584510000840

[53] R. Meziane, M. J. Otis, and H. Ezzaidi, "Human-robot collaboration while sharing production activities in dynamic environment: SPADER system," *Robot. and Comput.-Integr. Manuf.*, vol. 48, pp. 243–253, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584515301447

[54] T. Fraichard, "A Short Paper about Motion Safety," in *IEEE Int. Conf. Robot. and Autom., (ICRA07)*, Apr 2007, pp. 1140–1145.

[55] K. Kant and S. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[56] T. Fraichard and C. Laugier, "Path-velocity decomposition revisited and applied to dynamic trajectory planning," in *IEEE Int. Conf. Robot. and Autom., (ICRA93)*, vol. 2, Los Alamitos, CA, May 1993, pp. 40–45.

[57] T. Fraichard, "Trajectory planning in a dynamic workspace: a 'state-time space' approach," *Advanced Robotics*, vol. 13, no. 1, 1999.

[58] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[59] B. Kluge and E. Prassler, "Reflective navigation: individual behaviors and group behaviors," in *IEEE Int. Conf. Robot. and Autom., (ICRA04)*, vol. 4, April 2004, pp. 4172–4177.

[60] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal Velocity Obstacles for real-time multi-agent navigation," in *IEEE Int. Conf. Robot. and Autom., (ICRA08)*, May 2008, pp. 1928–1935.

[61] D. Wilkie, J. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *IEEE/RSJ Int. Conf. Intell. Robots and Syst., (IROS09)*, Oct 2009, pp. 5573–5578.

[62] J. van den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *IEEE Int. Conf. Robot. and Autom., (ICRA11)*, May 2011, pp. 3475–3482.

[63] T. Fraichard and H. Asama, "Inevitable collision states. A step towards safer robots?" in *IEEE/RSJ Int. Conf. Intell. Robots and Syst., (IROS03)*, vol. 1, Oct 2003, pp. 388–393 vol.1.

[64] L. Martinez-Gomez and T. Fraichard, "An efficient and generic 2D Inevitable Collision State-checker," in *IEEE/RSJ Int. Conf. Intell. Robots and Syst., (IROS08)*, Sept 2008, pp. 234–241.

[65] A. Bautin, L. Martinez-Gomez, and T. Fraichard, "Inevitable Collision States: A probabilistic perspective," in *IEEE Int. Conf. Robot. and Autom., (ICRA10)*, May 2010, pp. 4022–4027.

[66] G. Savino, F. Giovannini, M. Fitzharris, and M. Pierini, "Inevitable Collision States for Motorcycle-to-Car Collision Scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 9, pp. 2563–2573, Sept 2016.

[67] Z. Shiller and Y. R. Gwo, "Dynamic motion planning of autonomous vehicles," *IEEE Trans. Robot. Autom.*, vol. 7, no. 2, pp. 241–249, Apr 1991.

[68] X. Broquère, D. Sidobre, and I. Herrera-Aguilar, "Soft motion trajectory planner for service manipulator robot," in *IEEE/RSJ Int. Conf. Intell. Robots and Syst., (IROS08)*, 2008, pp. 2808–2813.

[69] K. G. Shin and N. D. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Trans. Autom. Control*, vol. 30, no. 6, pp. 531–541, Jun. 1985.

[70] R. Solea and U. Nunes, "Trajectory Planning with Velocity Planner for Fully-Automated Passenger Vehicles," in *IEEE Conf. Intell. Transp. Syst., (ITSC06)*, Sept 2006, pp. 474–480.

[71] X. Li, Z. Sun, A. Kurt, and Q. Zhu, "A sampling-based local trajectory planner for autonomous driving along a reference path," in *IEEE Intell. Veh. Symp., (IV14)*, June 2014, pp. 376–381.

[72] G. Manor and E. Rimon, "The speed graph method: Time optimal navigation among obstacles subject to safe braking constraint," in *IEEE Int. Conf. Robot. and Autom., (ICRA14)*, May 2014, pp. 1155–1160.

[73] V. Muñoz, A. Ollero, M. Prado, and A. Simón, "Mobile robot trajectory planning with dynamic and kinematic constraints," in *IEEE Int. Conf. Robot. and Autom., (ICRA94)*, vol. 4, San Diego, CA, May 1994, pp. 2802–2807.

[74] F. Canali, C. Guarino Lo Bianco, and M. Locatelli, "Minimum-jerk online planning by a mathematical programming approach," *Eng. Optimization*, vol. 46, no. 6, pp. 763–783, 2014.

[75] C. Guarino Lo Bianco, "Minimum-Jerk Velocity Planning for Mobile Robot Applications," *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1317–1326, Oct 2013.

[76] M. Morsali, E. Frisk, and J. Åslund, "Real-time velocity planning for heavy duty truck with obstacle avoidance," in *IEEE Intell. Veh. Symp., (IV17)*, June 2017, pp. 109–114.

[77] J. Huang, P. Hu, K. Wu, and M. Zeng, "Optimal time-jerk trajectory planning for industrial robots," *Mechanism and Mach. Theory*, vol. 121, pp. 530–544, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0094114X17302914

[78] M. Brezak and I. Petrovic, "Time-optimal trajectory planning along predefined path for mobile robots with velocity and acceleration constraints," in *IEEE/ASME Int. Conf. on Adv. Intell. Mechatronics, (AIM11)*, July 2011, pp. 942–947.

[79] V. Zanotto, A. Gasparetto, A. Lanzutti, P. Boscariol, and R. Vidoni, "Experimental Validation of Minimum Time-jerk Algorithms for Industrial Robots," *J. of Intell. Robot. Syst.*, vol. 64, no. 2, pp. 197–219, Nov 2011. [Online]. Available: https://doi.org/10.1007/s10846-010-9533-5

[80] L. Biagiotti and C. Melchiorri, "FIR filters for online trajectory planning with time- and frequency-domain specifications," *Control Eng. Practice*, vol. 20, no. 12, pp. 1385–1399, 2012.

[81] R. Katzschmann, T. Kröger, T. Asfour, and O. Khatib, "Towards online trajectory generation considering robot dynamics and torque limits," in *IEEE/RSJ Int. Conf. Intell. Robots and Syst., (IROS13)*, Nov 2013, pp. 5644–5651.

[82] B. Ezair, T. Tassa, and Z. Shiller, "Planning high order trajectories with general initial and final conditions and asymmetric bounds," *Int. J. Robot. Res.*, vol. 33, no. 6, pp. 898–916, 2014.

[83] F. Cabassi, L. Consolini, and M. Locatelli, "Time-optimal velocity planning by a bound-tightening technique," *Comput. Optimization and Appl.*, vol. 70, no. 1, pp. 61–90, May 2018. [Online]. Available: https://doi.org/10.1007/s10589-017-9978-6

[84] H. Pham and Q. Pham, "On the structure of the time-optimal path parameterization problem with third-order constraints," in *IEEE Int. Conf. Robot. and Autom., (ICRA17)*, May 2017, pp. 679–686.

[85] S. Perri, C. Guarino Lo Bianco, and M. Locatelli, "Jerk bounded velocity planner for the online management of autonomous vehicles," in *IEEE Int. Conf. Autom. Sci. and Eng., (CASE15)*, Aug 2015, pp. 618–625.

[86] "Safety Standard for Driverless, Automatic Guided Vehicles and Automated Functions of Manned Industrial Vehicles," Industrial Truck Standards Development Foundation, Standard ANSI/ITSDF B56.5-2012, 2012.

[87] "Safety of Industrial Trucks - Driverless Trucks And Their Systems," International Standards Organization, Standard ISO 1525:1997, Sept 1997.

[88] "Safety of Industrial Trucks - Additional Requirements for Automated Functions on Trucks," International Standards Organization, Standard ISO 1526:1997+A1:2008 (E), July 2008.

[89] "Standard Test Method for Grid-Video Obstacle Measurement," ASTM International, West Conshohocken, PA, Standard ASTM F3265-17, 2017.

[90] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots*. Heidelberg, Germany: Springer, Berlin, 2008.

[91] J. Villagra, V. Milanés, J. Pérez, and J. Godoy, "Smooth path and speed planning for an automated public transport vehicle," *Robot. and Autonomous Syst.*, vol. 60, no. 2, pp. 252–265, 2012.

[92] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006. [Online]. Available: https://doi.org/10.1007/s10107-004-0559-y

Acknowledgements

Three years since I began my PhD and here I am, time for acknowledgments.

First and foremost, I would like to thank my supervisor Prof. Corrado Guarino Lo Bianco for its guidance and support during the development of this work and for the inspiring conversations we had on scientific and less-scientific topics.

Besides my supervisor, I would like to thank the Elettric80 R&D staff for its profitable support, in particular Eng. Francesco De Mola and Eng. Domenico Di Terlizzi for the excellent collaboration during the SAFERUN project. I wish to express my sincere thanks to Simone Perri, my industrial counterpart during the European project, I found a friend as well as a hard-working colleague.

I would also like to thank Prof. Marco Locatelli and Prof. Luca Consolini, for their useful advice interspersed with laughs. Moreover a special thank goes to the other PhD students who share the journey with me, in particular Andrea Minari, classmate since the bachelor degree, and Laura Sani, my "Sync" office mate. A sincere thank also to Mattia Laurini, even if he loves to scare me while entering in my office, and, for the last months of this adventure, to the other volleyball player of the group Andrea Tagliavini.

Last but not least, I would like to thank my company of Friends, Alessandra, Alex, Giulia, Matteo S., Matteo T., Mirco and Valeria, for the wonderful moments

spent together. A great thank goes to Marco, who believes in me and never leaves me alone.

Finally, a huge thank goes to my parents, who are always by my side and support my choices. Without them, none of that would have been possible.