



# Minimum-Time Trajectory Planning for a Differential Drive Mobile Robot Considering Non-slipping Constraints

I. F. Okuyama<sup>1</sup> · Marcos R. O. A. Maximo<sup>1</sup> · Rubens J. M. Afonso<sup>2,3</sup>

Received: 14 February 2020 / Revised: 24 August 2020 / Accepted: 8 October 2020 / Published online: 4 November 2020  
© Brazilian Society for Automatics–SBA 2020

## Abstract

We propose a real-time minimum-time trajectory planning strategy with obstacle avoidance for a differential-drive mobile robot in the context of robot soccer. The method considers constraints important to maximize the system's performance, such as the actuator limits and non-slipping conditions. We also present a novel friction model that regards the imbalance of normal forces on the wheels due to the acceleration of the robot. Theoretical guarantees on how to obtain a minimum-time velocity profile on a predetermined parametrized curve considering the modeled constraints are also presented. Then, we introduce a nonlinear, non-convex, local optimization using a version of the Resilient Propagation algorithm that minimizes the time of the curve while avoiding obstacles and respecting system constraints. Finally, employing a new proposed benchmark, we verified that the presented strategy allows the robot to traverse a cluttered field (with dimensions of 1.5 m × 1.3 m) in 2.8 s in 95% of the cases, while the optimization success rate was 85%. We also demonstrated the possibility of running the optimization in real-time, since it takes less than 13.8 ms in 95% of the cases.

**Keywords** Trajectory planning · Optimization · Differential-drive mobile robot · Robotics

## 1 Introduction

Differential-drive mobile robot (DDMR) is a very pervasive class of robots. They are car-like and have only two wheels driven by electric motors, making them cheap and easy to manufacture. However, this hardware simplicity comes with nonlinear, nonholonomic, and underactuated dynamics (LaValle 2006); hence trajectory planning and control are hard, especially in competitive scenarios, such as military

applications and robot soccer, where high performance is needed to beat an opponent.

The Very Small Size Competition League (VSS) is a robot soccer competition where two teams compete using three autonomous mobile robots each one (IEEE 2008). Each robot is constrained to fit within a cube of side 7.5 cm. The poses of the robots and the ball are determined by a vision system based on an overhead camera. Moreover, computationally intensive algorithms, including computer vision, decision-making, and motion planning, run in a central computer. Given the tight space restrictions, differential-drive mobile robot (DDMR) seems to be the optimal solution since all top teams opt for it.

Classic approaches for robot navigation divide the problem into three sequential steps: path planning, trajectory planning, and tracking control (Rimon and Koditschek 1992). Even if each subproblem is solved optimally, this divide-and-conquer approach usually leads to suboptimal solutions due to performance losses in transitions between steps.

Many practical implementations frame path planning as a graph search problem (LaValle 2006), which requires encoding the space as a graph. A straightforward representation involves using a uniform grid, whereas more advanced solu-

✉ I. F. Okuyama  
franzoni315@gmail.com

Marcos R. O. A. Maximo  
mmaximo@ita.br

Rubens J. M. Afonso  
rubensjm@ita.br

<sup>1</sup> Autonomous Computational Systems Lab (LAB-SCA), Computer Science Division, Aeronautics Institute of Technology, São José dos Campos, São Paulo, Brazil

<sup>2</sup> Department of Aerospace and Geodesy, Institute of Flight System Dynamics, Technical University of Munich, Munich, Bavaria, Germany

<sup>3</sup> Electronic Engineering Division, Aeronautics Institute of Technology, São José dos Campos, São Paulo, Brazil

tions are visibility graphs (Lozano-Pérez and Wesley 1979) and cell decomposition (Latombe 2012).

Visibility graphs create a graph formed by all collision-free lines drawn from the obstacles' vertices. If no approximation is used to represent the obstacles' shapes, the technique guarantees that the optimal graph path coincides with the shortest length path in continuous space, but this solution may pass dangerously close to obstacles. Furthermore, it is not time-optimal in general, since dynamic behavior is not considered.

Cell decomposition, on the other hand, divides the free space into cells, such that it is easy to determine a collision-free path between adjacent cells. The graph formed by connecting all adjacent cells is called the connectivity graph. Then, the path planning problem is reduced to finding the sequence of cells (a "channel"), that should be traversed in order to get to the goal. Nevertheless, this approach does not have any optimality guarantees, since the graph nodes are usually set in the middle of the cell.

After graph construction, a path may be found using A\* (Hart et al. 1968), an informed graph search algorithm that finds a minimum cost path between two nodes in the graph given that the heuristics used obey certain conditions.

Another family of methods for path planning is artificial potential fields (APF) (Khatib and Le Maitre 1978). The basic scheme places attractive and repulsive potential functions on targets and obstacles, respectively. Then, the potential functions are summed to yield a resultant potential field. APF is popular due to the ease of implementation and low computational cost. Nevertheless, the method has its drawbacks. It is suboptimal, tuning the algorithm's parameters is difficult, and the robot tends to get stuck due to local minima in the potential field. A variant of APF called Univector (Kim et al. 2001) is particularly popular in robot soccer competitions, where parameters are often tuned by metaheuristic optimization (Kim et al. 2001; Lim et al. 2008).

None of the methods discussed so far are concerned with model-based minimum-time trajectory planning. Shin and McKay (1985) proposed an algorithm to compute a minimum-time velocity profile given a parametrized path for a robot's dynamics written in the manipulator format. Yamamoto et al. (1998) adapted this work for a DDMR model, while also generalizing it to search for the optimal predefined parametrized path. The authors used a combination of cell decomposition and B-spline (LaValle 2006) to encode the path and the local optimization method Hill Climbing (Russell and Norvig 2009). Sprunk (2008) further developed this approach by analyzing which type of spline is better suited for the problem. Morsali et al. (2019) considered multiple constraints such as friction rollover due to banking of roads, and moving obstacles to optimize a speed profile based on a customized A\*, although the path itself is

not optimized and load transfer between wheels is not considered.

There are also many approaches to the path planning problem based on the Pontryagin's minimum principle of optimal control theory. Dubins (1957) constructed primitives that can be combined to create a minimum-length path for a car with limited curvature, and Hérisse and Pepy (2013) extended this approach by allowing the maximum curvature to depend on the vehicle position, although the constraint considered was restricted to be monotonically increasing over the  $y_w$  axis. Manor et al. (2018) worked on the same problem for the differential-drive mobile robot (DDMR), but also analyzed velocity dynamics and provided six motion primitives for the minimum-time trajectory planning (MTTP) problem under tangential and centripetal acceleration limits. If obstacles are considered (Ben-Asher et al. 2019), two more primitives arise, which consist of obstacle hugging segments. We point out that both Manor et al. (2018) and Ben-Asher et al. (2019) assume direct control of the angular velocity, while in the present work we only control the battery voltage applied to the motors. Furthermore, they only provide primitives without showing how to find the optimal sequence of primitives that minimize the traversal time; the paths are actually constructed by using numerical solutions with direct collocation methods.

Sampling-based methods, such as the Rapidly-Exploring Random Trees (RRT) (LaValle 1998), are another way of solving path planning problems and they are known to solve high-dimensional tasks in reasonable time. Its optimal version, the RRT\* (Karaman and Frazzoli 2010), produces an initial path and iteratively enhance it by pruning suboptimal paths. An extension for handling kinodynamics was introduced by Webb and van den Berg (2013), although optimality for nonlinear dynamics (as in the case of the differential-drive mobile robot (DDMR)) is not guaranteed, since one needs to solve a boundary value problem when connecting two nodes of the tree in the pruning stage. Since RRT usually produces infeasible jerky trajectories and can handle only simplified dynamics, Zhu et al. (2015) locally enhanced the path by framing the smoothing and velocity profile optimization as a convex optimization, although these steps are computationally expensive.

The main contributions of the present work are:

- A framework for performing the minimum-time trajectory planning (MTTP) for a DDMR. We are partly inspired by VSS, where the team ITAndroids competes, but the methods presented herein certainly generalize to other applications. In the proposed formulation, obstacle avoidance, non-slipping, non-toppling, and actuation constraints are taken into account. This framework considers tangential and centripetal accelerations as inputs,

differing from Ben-Asher et al. (2019) that uses tangential acceleration and angular velocity.

- An elliptical friction model which considers the distribution of the normal forces on the robot's support points is developed, which enables the use of more realistic acceleration constraints. This model is more generic than the commonly used thresholds on linear and centripetal accelerations (or angular velocity) (Ben-Asher et al. 2019; Webb and van den Berg 2013; Yamamoto et al. 1998; Zhu et al. 2015).
- The requirements of minimum-time, obstacle avoidance and feasible trajectory are addressed by a newly designed cost function used in the optimization procedure.
- This is the first work, to the best of our knowledge, to simultaneously consider all these constraints while providing computational performance suitable for real-time.
- A benchmark for MTTP frameworks based on Monte Carlo simulations.

The remaining of the article is organized as follows. Section 2 presents the DDMR dynamic model. Section 3 explains the method used to solve the minimum-time velocity profile problem. Section 4 defines the minimum-time trajectory planning problem and presents our solution. Section 5 proposes a Monte Carlo benchmark to assess the overall performance of the proposed algorithms. Section 6 shows simulation results to validate our approach. Finally, Sect. 7 concludes and shares future research directions.

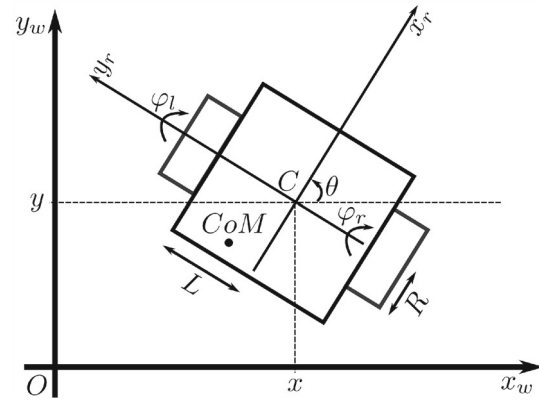
## 2 Differential Drive Mobile Robot Model

We show a summarized presentation due to space restrictions, but a full derivation of the ideas presented here may be seen in Okuyama (2019). Moreover, a general differential-drive mobile robot (DDMR) dynamics model (see Fig. 1) for the VSS robot is derived in Okuyama et al. (2017). This model tries to capture the whole complexity of the system. However, to simplify the equations, we decided to neglect dry friction, delay, the dynamic coupling between wheels, and asymmetry between the motors.

Under these assumptions, the equations of motion of the wheel velocities can be written as:

$$\begin{cases} \dot{\omega}_r = \beta \omega_r + \gamma V_r, \\ \dot{\omega}_l = \beta \omega_l + \gamma V_l, \end{cases} \quad (1)$$

where  $\beta$  and  $\gamma$  are model parameters,  $\omega_r$  and  $\omega_l$  are the angular speeds of the right and left wheels, respectively, and  $V_r$  and  $V_l$  are voltages fed to the right and left motors, respectively. All modeling parameters used in this work are shown in Table 4 in the “Appendix”. Additionally, consider the following relationships between linear and angular velocities



**Fig. 1** Schematic diagram of the robot depicting the world ( $x_w, y_w$ ) and robot ( $x_r, y_r$ ) frames, where  $C$  is the robot's geometric center,  $\theta$  is the robot's orientation w.r.t. the world frame,  $R$  is the wheel's radius,  $L$  is the distance from  $C$  to each wheel,  $CoM$  is the center of mass,  $\phi_r$  and  $\phi_l$  are the wheels' angles and  $(x, y)$  is the position of the robot's center

and accelerations:

$$\begin{cases} v = \frac{v_r + v_l}{2} = (\omega_r + \omega_l) \frac{R}{2}, \\ \omega = \frac{v_r - v_l}{2L} = (\omega_r - \omega_l) \frac{R}{2L}, \\ a = \frac{dv}{dt} = (\dot{\omega}_r + \dot{\omega}_l) \frac{R}{2}, \\ \alpha = \frac{d\omega}{dt} = (\dot{\omega}_r - \dot{\omega}_l) \frac{R}{2L}, \\ \omega = \frac{v}{\rho} = \kappa v, \end{cases} \quad (2)$$

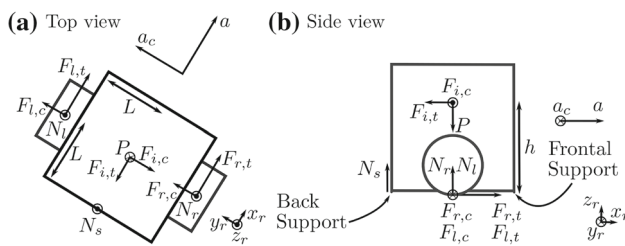
where  $v$  is the linear velocity,  $\omega$  is the angular velocity,  $a$  is the linear acceleration,  $\alpha$  is the angular acceleration,  $\rho$  is the radius of curvature,  $\kappa$  is the curvature,  $R$  is the wheel's radius and  $L$  is the distance between the robot's center and the wheel. In the following sections, we determine what are the valid sets of velocity and acceleration due to motor, friction and toppling constraints. The intersection of these sets will be used in the construction of the minimum-time velocity profile.

### 2.1 Motor Constraints

Consider the robot is driving along a curve parametrized over the path displacement,  $s$ . Each point of the curve has a curvature,  $\kappa(s)$ , and a derivative of curvature with respect to the path displacement that is represented by  $\eta(s) = \frac{d\kappa(s)}{ds}$ . From (1) and (2) and by imposing the voltage must be less than the maximum available voltage  $V_{\max}$ , we can determine parabolic validity regions in the  $(v, a)$  plane for the robot's acceleration and velocity given a curvature,  $\kappa$ , and its derivative,  $\eta$ , along the curve:

$$-u_m \leq a(\kappa L + 1) + \eta L v^2 - \beta(\kappa L + 1)v \leq u_m, \quad (3)$$

$$-u_m \leq a(\kappa L - 1) + \eta L v^2 - \beta(\kappa L - 1)v \leq u_m, \quad (4)$$



**Fig. 2** Forces over the robot, assuming it leans on the back support (non-inertial reference frame)

where  $u_m = \gamma RV_{\max}$ . A more detailed derivation of these inequalities is shown in Okuyama (2019).

## 2.2 Friction Constraints

The robot's frontal and back supports are made of a material with negligible friction such as Teflon and have a small gap w.r.t. the wheels' points of contact with the ground. Therefore, when accelerating, the robot leans over the opposite direction of the acceleration. Considering the equilibrium of fictitious forces in the frame shown in Fig. 2, we have the following relationships:

$$\begin{aligned} N_r &= \frac{m}{2} \left( g - \frac{h}{L} |a| + \frac{h}{L} a_c \right), \\ N_l &= \frac{m}{2} \left( g - \frac{h}{L} |a| - \frac{h}{L} a_c \right), \\ F_{r,t} &= \frac{maL + I_c \alpha}{2L}, \\ F_{l,t} &= \frac{maL - I_c \alpha}{2L}, \end{aligned} \quad (5)$$

where  $a_c$  is the centripetal acceleration,  $m$  is the robot's mass,  $g$  is the gravity acceleration,  $h$  is the center of mass (CoM) height,  $I_c$  is the robot's moment of inertia w.r.t. its geometric center,  $N_r$  and  $N_l$  are normal forces applied to the right and left wheels, respectively, and  $F_{r,t}$  and  $F_{l,t}$  are tangential friction forces acting on the right and left wheels, respectively. The non-slipping condition is:

$$\begin{cases} F_r = \sqrt{F_{r,t}^2 + F_{r,c}^2} \leq \mu N_r, \\ F_l = \sqrt{F_{l,t}^2 + F_{l,c}^2} \leq \mu N_l, \end{cases} \quad (6)$$

where  $\mu$  is the friction coefficient. From (5) and (6), we can determine the complete non-slipping conditions:

$$\begin{cases} m^2 \left( \frac{a^2}{2} + a_c^2 \right) + \frac{I_c^2 \alpha^2}{2L^2} \leq \mu^2 (N_r^2 + N_l^2) + 2\sqrt{AB}, \\ A = (\mu N_r)^2 - F_{r,t}^2 \geq 0, \\ B = (\mu N_l)^2 - F_{l,t}^2 \geq 0. \end{cases} \quad (7)$$

Nevertheless, this friction model is too complex for a real-time formulation. Thence, we perform some simplifications to obtain a more conservative but simpler model that does not depend on the angular acceleration  $\alpha$ . The first one is to eliminate the square root term by observing that  $\sqrt{AB} \geq 0$ . Then, we employ another conservative approximation over the  $A \geq 0$  and  $B \geq 0$  conditions, by using triangle inequality:

$$\begin{aligned} &\begin{cases} (\mu N_r)^2 - F_{r,t}^2 \geq 0, \\ (\mu N_l)^2 - F_{l,t}^2 \geq 0, \end{cases} \Leftrightarrow \begin{cases} |F_{r,t}| \leq |\mu N_r|, \\ |F_{l,t}| \leq |\mu N_l|, \end{cases} \\ &\Leftrightarrow \begin{cases} |maL + I_c \alpha| \leq \mu m |gL - h|a| + ha_c|, \\ |maL - I_c \alpha| \leq \mu m |gL - h|a| - ha_c|, \end{cases} \quad (8) \\ &\Leftrightarrow \begin{cases} |maL| + |I_c \alpha| \leq \mu m |gL - h|a| + ha_c|, \\ |maL| + |I_c \alpha| \leq \mu m |gL - h|a| - ha_c|, \end{cases} \\ &\Leftrightarrow |maL| + |I_c \alpha| \leq \mu m |gL - h(|a| + |a_c|)|. \end{aligned}$$

where  $\Leftarrow$  means that satisfying the condition on the right-hand-side of the symbol is sufficient for the condition on the left-hand-side to hold. Additionally, as suggested in Purwin and D'Andrea (2006) and Sherback et al. (2006), we limit the angular acceleration to a predefined maximum value:

$$|\alpha| \leq \alpha_{\max} \iff |\eta v^2 + a\kappa| \leq \alpha_{\max}, \quad (9)$$

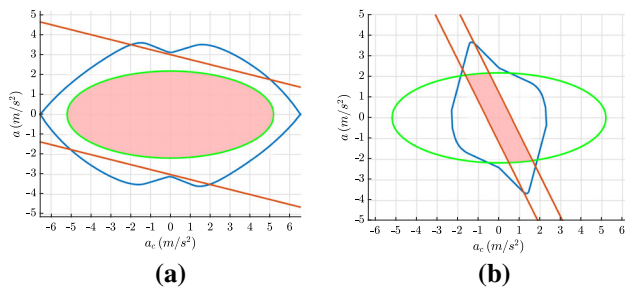
where  $\alpha_{\max}$  is the maximum angular acceleration. Applying the aforementioned approximations into (7):

$$\begin{cases} m^2 \left( \frac{a^2}{2} + a_c^2 \right) + \frac{(I_c \alpha_{\max})^2}{2L^2} \leq \mu^2 (N_r^2 + N_l^2), \\ |maL| + I_c \alpha_{\max} \leq \mu m |gL - h(|a| + |a_c|)|. \end{cases} \quad (10)$$

We remark that (10) together with (9) is more conservative, i.e., satisfaction of (9) and (10) for any values of the accelerations  $a$ ,  $a_c$  and  $\alpha$  implies that (7) is also valid, but satisfaction of (7) does not necessarily imply (9) and (10). In other words, the set of possible accelerations obtained through (9) and (10) is a subset of the accelerations that satisfy (7). We further reduce the model complexity by considering an ellipse that best approximates the region defined by (10). The elliptical model for describing friction limits of a DDMR was introduced by Lepetič et al. (2003), but the model was determined experimentally. The approximate elliptical model is:

$$\frac{a_c^2}{a_{c,\max}^2} + \frac{a^2}{a_{t,\max}^2} \leq 1, \quad (11)$$

where  $a_{t,\max}$  is the maximum tangential acceleration and  $a_{c,\max}$  is the maximum centripetal acceleration, under which there is no slip. The ellipse parameters are determined by a numeric optimization procedure to minimize the difference



**Fig. 3** Comparison of the region defined by all constraints and the proposed conservative approximation, for: **a**  $|\kappa| = 20 \text{ m}^{-1}$ ,  $\eta = 10^{-2}$  and **b**  $|\kappa| = 50 \text{ m}^{-1}$ ,  $\eta = 5000^{-2}$

between the actual constraint's area and the approximating ellipse's area (Okuyama 2019). Considering our robot's parameters, Fig. 3 presents a comparison of the region defined by all constraints in (7) and the proposed conservative approximation of (11). The curve in blue denotes the region defined by all constraints (7), while the green curve is our ellipse approximation (11) and the red lines represent the maximum angular acceleration constraint (9). The pink shaded region represents the combination of the constraints (9) and (11).

### 2.3 Toppling Constraints

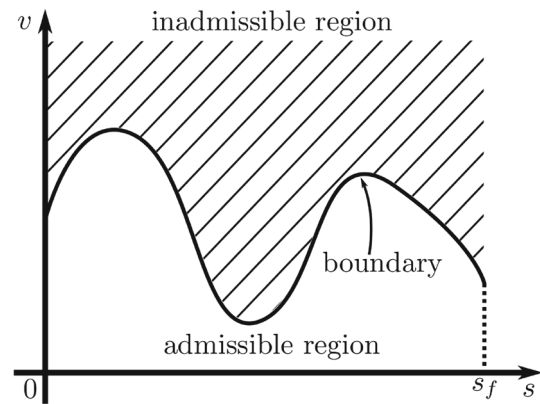
Some robots topple due to their own acceleration. This hardly happens with well-designed robots or cars, since experienced hardware designers know that low CoM effectively solves this problem. To express this in mathematical terms, notice that the robot is in imminence to tumble when its weight is concentrated over one of the frontal or back supports or one of the wheels (see Fig. 2b). In the first case, we can write:

$$\begin{cases} N_s = P = mg, \\ N_r = N_l = 0. \end{cases} \quad (12)$$

Now, considering the equilibrium moment over the back support point in the  $y_r$  direction:

$$F_{i,t}h + (N_r + N_l - P)L = 0 \Rightarrow a_{t,\text{topp}} = \frac{gL}{h}, \quad (13)$$

where  $a_{t,\text{topp}}$  is the maximum tangential acceleration without toppling. Notice that it depends on the ratio  $L/h$ , so it is desirable to have a low CoM and to maximize the robot width, as intuition suggests. The same analysis can be done for the case when the whole weight is over one of the wheels, yielding  $a_{c,\text{topp}} = gL/h$ , where  $a_{c,\text{topp}}$  is the maximum centripetal acceleration without toppling. For our robot, this results in linear and centripetal acceleration limits of  $8.8993 \text{ ms}^{-2}$ , much higher than the limits imposed by friction, so we dis-



**Fig. 4** Example of admissible and inadmissible regions

regard toppling in our framework, although this condition could be considered for a distinct robot if necessary.

### 2.4 Collision Avoidance Constraints

Finally, for obstacle avoidance, we consider the teammate and opponent robots, the field's walls and possibly the ball (depending on the task). We use a common approach where the robot is regarded as a point and its dimensions are transferred to the obstacles. Moreover, a safe distance is also added to account for inevitable modeling, estimation and tracking errors. The collision avoidance parameters are the obstacle radius  $r_{\text{obs}}$ , the robot radius  $r_{\text{rob}}$ , and the ball radius  $r_{\text{ball}}$ .

### 3 Minimum-Time Velocity Profile

Given the model constraints and a parametrized path, we approach the problem of determining the minimum-time velocity profile. We use the algorithm proposed by Shin and McKay (1985), which is divided into two steps: determination of the admissible region of velocities; and determination of the minimum-time velocity profile.

The admissible region (Fig. 4) is the region of allowed velocities for each point,  $s$ , of the parametrized path, and it can be directly computed from the constraint equations. Its main idea is that if the system goes outside the admissible region, then at least one of the constraints is necessarily violated. Therefore, any valid velocity profile must lie inside the admissible region.

Depending on the constraints, the admissible region may present islands of inadmissibility, which can be seen as holes in the phase plane. However, even with the nonlinear constraint of friction, our system does not present any islands of inadmissibility independently of the parametrized path. In fact, the symmetry of the friction around the origin guarantees that its intersection with the motor and maximum angular constraints does not present any islands. Thus, the admissible



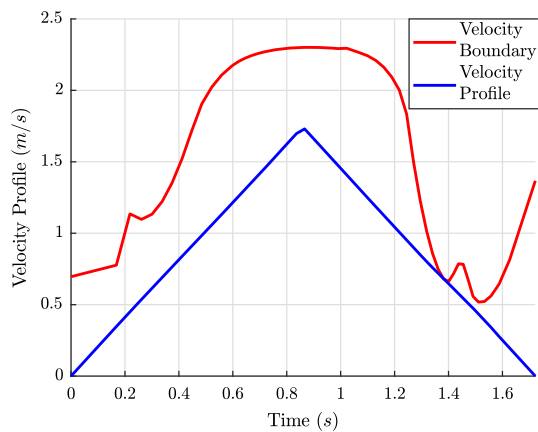


Fig. 5 Example of velocity profile

region can be described by the maximum velocity at every point of the path, which can be determined by the intersection of the constraints:

$$0 \leq v \leq v_{\max}(s). \quad (14)$$

The Minimum-Time Velocity Profile is generated by maximizing the velocity over each point of the path, without ever surpassing the velocity boundary. With the constraint equations (3), (4), (9), (11), it is possible to determine the acceleration that maximizes the velocity without crossing to the inadmissible region. The problem resides, then, in finding the points where the acceleration is changed from maximum to minimum (or vice-versa). These switching points are determined by the ACOT (for systems that do not contain islands of inadmissibility), which is detailed in Shin and McKay (1985). Figure 5 shows an example of optimal velocity profile composed of only one switching point at 0.85s.

## 4 Minimum-Time Trajectory Planning

### 4.1 Problem Definition

The MTTP can be defined in the following way. Given:

- Initial conditions:  $x(0) = x_0$ ,  $y(0) = y_0$ ,  $v(0) = v_0$ ,  $\kappa(0) = \kappa_0$ ,  $\theta(0) = \theta_0$ ;
- End conditions:  $x(s_f) = x_f$ ,  $y(s_f) = y_f$ ,  $v(s_f) = v_f$ ,  $\kappa(s_f) = \kappa_f$ ,  $\theta(s_f) = \theta_f$ ;
- Robot's constraints from (3), (4), (9), and (11):

$$\begin{aligned} |a - \beta v + (\alpha - \beta\omega)L| &\leq \gamma R V_{\max}, \\ |a - \beta v - (\alpha - \beta\omega)L| &\leq \gamma R V_{\max}, \\ \frac{a^2}{a_{t,\max}^2} + \frac{k^2 v^4}{a_{c,\max}^2} &\leq 1, \\ |\eta v^2 + a\kappa| &\leq \alpha_{\max}, \end{aligned} \quad (15)$$

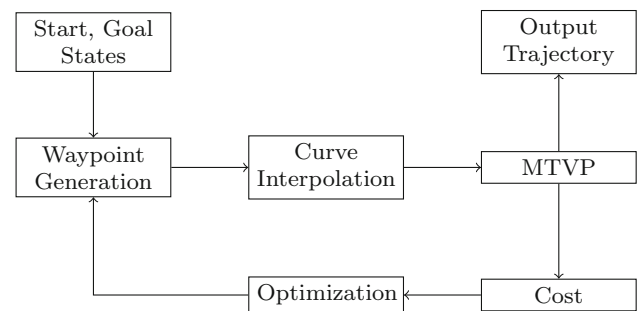


Fig. 6 Proposed MTTP framework

find a parametrized path,  $q(s) = [x(s) \ y(s)]^T$ , with  $0 = s(0) \leq s(t) \leq s(t_f) = s_f$ , that produces a valid velocity profile, minimizes  $T = \int_0^{t_f} dt$  and is collision-free.

### 4.2 MTTP Framework

We propose a general MTTP solution framework (Fig. 6), which is similar to Yamamoto et al. (1999), Haddad et al. (2007), and Sprunk (2008), the novelties being a new interpolation scheme, the consideration of friction and the normal forces over the wheels and the design of the cost function. First, we generate waypoints that are used in the creation of a parametrized path, through an interpolation scheme. Next, a MTVP is computed from the initial curve. Then, we calculate a cost function based on the profile and on some collision checking procedure. This cost is passed to an optimizer that changes the waypoint positions to minimize the time and avoid collisions. Therefore, the MTTP problem is transformed into a nonlinear, non-convex and local parametric optimization problem.

#### 4.2.1 Euclidean Shortest Path with Visibility Graph

The positions of the initial waypoints are crucial for avoiding local minima that are far from the global optimum since we employ a local optimization method. In order to avoid the influence of an inadequate parameter initialization, Yamamoto et al. (1999) picks multiple initial waypoints from a cell-decomposition method. However, this is computationally costly, since we would have to run one optimization for each initial solution and compare all of them.

Considering that the obstacles have predefined shape and there is much free space, we opt for an Euclidean shortest path algorithm based on visibility graphs (Ho and Liu 2009). Our implementation simultaneously builds the graph and searches for the shortest path to avoid constructing parts of the graph which do not contribute to the shortest path. The search results in a variable number ( $n$ ) of waypoints depending on the obstacles.

An important observation is that using the visibility graph path as the parametrized curve does not produce minimum-time trajectories in general, since the robot would have to stop at each waypoint, turn around itself to correct its orientation and seek for the next waypoint. Thus, this initial path needs to be further optimized.

#### 4.2.2 Curve Interpolation with Quintic Splines

The type of spline to be used for interpolation is not a consensus in the literature. For example, in Yamamoto et al. (1999) a third-order composite B-spline is proposed. However, this curve does not have the desired properties for the problem in Sprunk (2008): stitching,  $C^2$  continuity, localism, and the possibility to set first and second derivatives at endpoints. Indeed, it is concluded in that work that a composite Quintic Bézier could satisfy all desired properties. On the other hand, one drawback is the necessity of heuristics for choosing the curvature and the tangent (the first derivative with respect to the spline parameter) at every waypoint. Additionally, it also includes the tangent elongation factor (the length of the tangent vector) for all waypoints as optimization parameters, which makes the optimization more time-consuming.

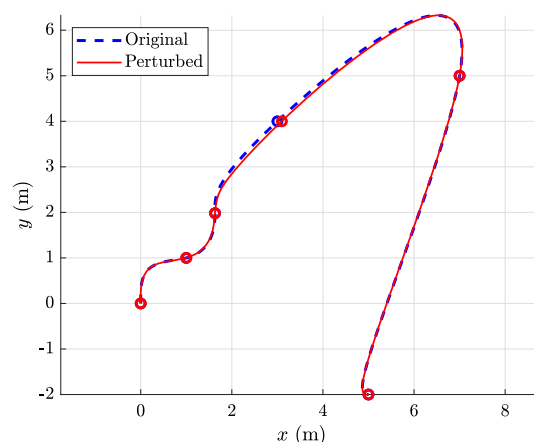
To avoid such overhead, a composite Quintic Bézier curve is used, although we give up the localism property, and require the curve to be continuous up to the fourth derivative so that  $\kappa$  and  $\eta$  at all internal waypoints are already determined and no heuristics are necessary. Even though it may be argued that the localism property is desirable for the optimization procedure, we experimentally observed that for small translations of a waypoint, the modifications over curve segments far from it are very small compared to the translation (see Fig. 7 for one example). Therefore, for gradient-based optimization, we claim that this is not a mandatory property and could be relaxed.

We point out that the proposed Quintic Spline still provides four free parameters that are used in the optimization, so as to control the curve shape: the magnitude of the first and second derivative vector on both start and final waypoints. For more details about the curve interpolation scheme, see Okuyama (2019).

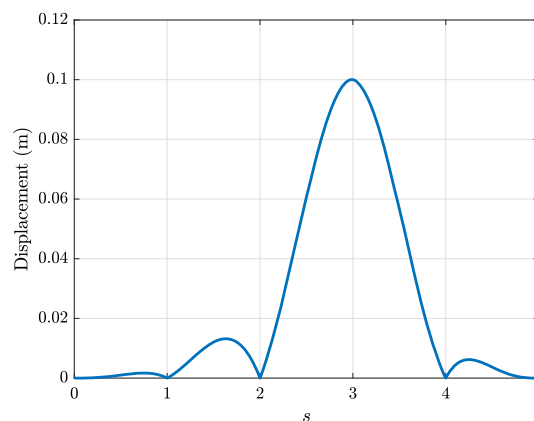
#### 4.2.3 Cost Function

Considering that our objective is to find a minimum-time collision-free trajectory, we propose the following cost function, whose terms are detailed in the next paragraphs:

$$c = t_f + \gamma_1 \sum_{i=1}^{N_d} c_{\text{col},i} + \gamma_2 \mathbb{I} \frac{1}{\frac{d_0^{\text{inv}}}{s_f} + \epsilon} + \gamma_3 \sum_{i=0}^1 c_{\text{neg},i} + \gamma_4 \left( \frac{v_0 - v_0^b}{v_0^b} \mathbb{I}_{v_0^b} + \frac{v_f - v_f^b}{v_f^b} \mathbb{I}_{v_f^b} \right). \quad (16)$$



(a) Waypoint perturbation.



(b) Displacement along the curve.

**Fig. 7** Curve displacement generated by a small perturbation of 0.1 m along the  $x$  axis of the waypoint associated with  $s = 3$ . The waypoints are depicted by circles in (a)

The first term,  $t_f$ , directly penalizes high traversal times. The second term penalizes the collision with the obstacles and is computed as follows: the curve is discretized in  $N_d$  points, and a penetration distance is computed for each point. The collision cost for the curve is the sum of each point penetration and a weight  $\gamma_1$  is assigned to this term.

In order to penalize invalid profiles and search for feasible trajectories, we added the third term of the cost function. We noticed that a common cause for invalid profiles occurs when the robot is driving towards an obstacle with high speed and the space for deceleration is small. Then, the robot would not have enough time to stop or make a curve. Our solution is to add a cost that is inversely proportional to the distance of the beginning of the path to the switching point that makes the velocity profile invalid. In other words, we want to maximize the path length used for deceleration. In (16),  $d_0^{\text{inv}}$  is the distance from the start of the path to the next switching point,  $s_f$  is the total path length,  $\gamma_2$  is the weight attributed to this cost, and  $\mathbb{I}$  is 1 if the profile is invalid, and 0, otherwise. The variable  $\epsilon$  prevents excessively high-cost values if  $d_0^{\text{inv}}$  is too small.

Another issue to be considered is the direction of the end-point velocity vectors (some of the free parameters of the quintic spline), whose signed magnitude is denoted by  $v_0^u$  and  $v_{n-1}^u$ . If they become negative, then the initial direction of the robot would flip  $180^\circ$ , and the orientation constraint would be violated. To avoid this situation we add a cost of  $\gamma_3$  for each negative magnitude, indicated by  $c_{\text{neg},i}$ , which equals 1 if an end-point speed is negative and 0, otherwise, whereas  $i = 0$  denotes the initial velocity, and  $i = 1$ , the final velocity.

The final issue addressed is that the end-point velocities might be over the velocity boundary, invalidating the whole profile. Hence, when this happens, we penalize the difference between the velocity boundary and the initial or final desired velocities. In (16),  $v_0^b$  and  $v_f^b$  are the initial and final value of the velocity boundary and  $\gamma_4$  is the weight given to cost term. The expression  $\mathbb{I}_x^y$  evaluates to 1 if  $x > y$  and 0, otherwise.

#### 4.2.4 Optimization

For the optimization algorithm, we chose the Resilient Propagation (RPROP) (Riedmiller and Braun 1993), since it was also successfully used by Sprunk (2008) in a context similar to this work. It is claimed that RPROP led to faster convergence rates than the usual backpropagation algorithm, because the main idea behind RPROP is that it does not depend on the magnitude of the partial derivatives with respect to the optimization parameters, but only on their sign. Important parameters are:  $\zeta +$  (factor to increase the learning rate),  $\zeta -$  (factor to decrease the learning rate),  $\Delta_{\max}$  (maximum parameter update),  $\Delta_{\min}$  (minimum parameter update) and  $\Delta_s$  (initial parameter update). Nonetheless, RPROP is a local optimizer and, thus, it greatly depends on the quality of the initial guess. To clarify, the optimization parameters are:

$$w = [v_0^u \ v_{n-1}^u \ a_0^u \ a_{n-1}^u \ W_1 \ W_2 \ \cdots \ W_{n-1}]. \quad (17)$$

Notice we have  $N_w = 2n + 2$  parameters. To start the optimization, we need a first set of initial parameters. The initial waypoint positions  $W_j$ ,  $j = 1, 2, \dots, n-1$ , are taken from the visibility graph and for the other parameters, we choose:

$$\begin{aligned} v_0^u &= (v_0 + 0.5) \|W_0 - W_1\|_2, \\ v_{n-1}^u &= (v_f + 0.5) \|W_{n-1} - W_{n-2}\|_2, \\ a_0^u &= a_{n-1}^u = 0, \end{aligned} \quad (18)$$

where  $\|\bullet\|_2$  represents the Euclidean norm, and  $a_0^u, a_{n-1}^u$  represent the magnitude of the second derivative of the spline at the initial and final positions.

Figure 8 shows an example of the optimization procedure. The boundary parameters are:  $v_0 = v_f = 0 \text{ ms}^{-1}$ ,  $\kappa_0 = \kappa_f = 0 \text{ m}^{-1}$ ,  $\theta_0 = \theta_f = 0^\circ$ . For the trajectories' colors,

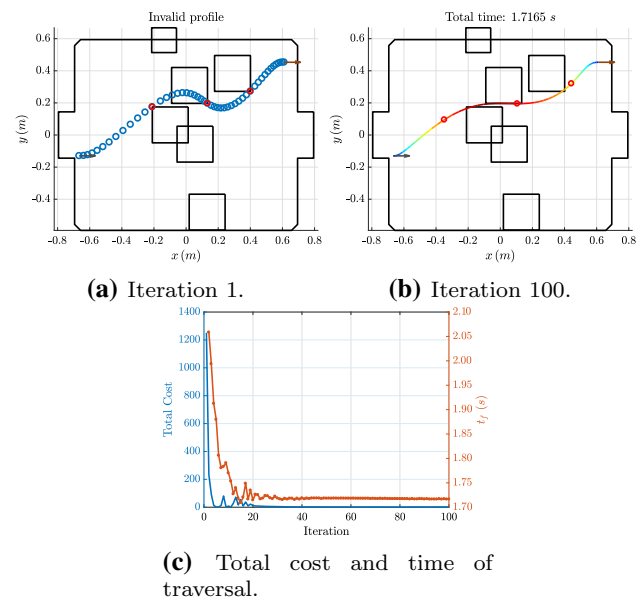


Fig. 8 Example of RPROP iteration steps

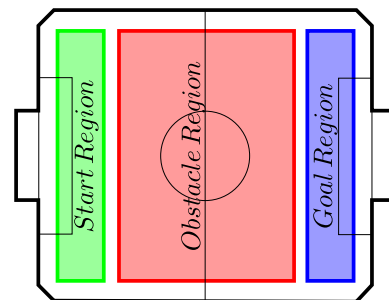


Fig. 9 Positioning of robot, obstacles and goal for Monte Carlo experiments

redder means higher velocity, whereas bluer means slower velocity. The red dots are the waypoints. Observe that the initial spline collides with an obstacle and also has an invalid velocity profile, thus leading to a high initial cost. Note also that the first valid trajectory is executed in more than 2 s and its relatively high time comes from the high curvature in the region of a collision, which forces the robot to slow down. Nonetheless, as we optimize the trajectory and the waypoints are translated, this point of high curvature is smoothed out and the trajectory becomes collision-free. In fact, after 100 iterations, the algorithm could find a collision-free trajectory that can be executed in 1.7165 s, i.e., an improvement of approximately 14%.

## 5 Monte Carlo Benchmark

We consider validation scenarios based on VSS, where the game state is continuously evolving. This means that the trajectory planning constraints are also constantly changing. In



fact, there are 8 boundary values (the initial/final positions, velocities, angles and curvatures) and 6 obstacles (2 teammates, 3 opponents and possibly the ball). This gives 22 input variables, implying that the problem workspace is a subspace of  $\mathbb{R}^{22}$ .

This high dimensionality indicates that it is hard to analyze the performance of our solution for the MTTP problem for a soccer game. Therefore, we choose the Monte Carlo (Metropolis and Ulam 1949) approach to obtain metrics about how our algorithm performs on a soccer game. It relies on sampling of the space and subsequent runs of our algorithm for each sample. The results are collected to create a distribution of some informative metrics.

For our problem, we chose four metrics:

1. The distribution of the trajectory's runtime, represented by the minimum, maximum and 25th, 50th, 75th, and 95th percentiles;
2. The percentage of valid velocity profiles;
3. The percentage of successful collision avoidance, by which we mean that, considering the safety radius to be 1 cm, all collision penetrations are within the safety margin;
4. The percentage of successful trajectories, which accounts for the trajectories that had valid velocity profiles and successful collision avoidance.

We point out that depending on the chosen sample, the problem might be intrinsically infeasible, and these samples constitute the *region of inevitable collision* (LaValle and Kuffner 2001). Therefore, these types of sample are inevitably included among the others. Identifying all these samples is a hard infeasibility detection problem (Byrd et al. 2010) and, for now, we focus only on eliminating the easier cases: we check if the start and final conditions are slip-free.

The end-point velocities, curvatures and orientation are generated according to:

$$\begin{aligned} v_0, v_f &\sim \mathcal{U}\left(0, \frac{v_{\max}}{2}\right), \\ \kappa_0, \kappa_f &\sim \mathcal{U}(-\kappa_{\max}, \kappa_{\max}), \\ \theta_0, \theta_f &\sim \mathcal{U}(-\pi, \pi), \end{aligned} \quad (19)$$

where  $\mathcal{U}(a, b)$  is a uniform random variable from  $a$  to  $b$  and  $v_{\max}$  is the maximum achievable velocity of the robot, which in our case is  $v_{\max} = \left|\frac{\gamma V_{\max} R}{\beta}\right| \approx 2.00 \text{ ms}^{-1}$ . We chose  $\kappa_{\max} = 5/L \approx 151.06 \text{ m}^{-1}$ , which is able to represent small and large radii of curvature compared to the robot length.

The robot, obstacle and goal positions are uniformly sampled from the regions pictured in Fig. 9. They represent challenging tasks that require passing through the whole field while avoiding obstacles. Numerically, the regions are defined by:

- Start Region:  $[-0.65, -0.5] \text{ m} \times [-0.55, 0.55] \text{ m}$ .
- Obstacle Region:  $[-0.35, 0.35] \text{ m} \times [-0.55, 0.55] \text{ m}$ .
- Goal Region:  $[0.5, 0.65] \text{ m} \times [-0.55, 0.55] \text{ m}$ .

Furthermore, an important aspect of algorithms that run on real-time systems, such as the Very Small Size Competition League (VSS), is that they often need to run within a limited time interval. The Monte Carlo benchmark is also useful for assessing the computational time taken by the algorithm, since distinct start and end conditions may demand distinct computational effort.

## 6 Results and Discussions

### 6.1 Monte Carlo Experiments

The conducted experiments were organized in the following way: first, we chose a set of default values for the parameters:  $N_{\text{iter}} = 100$ ,  $\zeta + = 1.4$ ,  $\zeta - = 0.7$ ,  $\Delta_{\max} = 0.5$ ,  $\Delta_{\min} = 0.0001$ ,  $\Delta_s = 0.01$ ,  $\gamma_1 = 5000$ ,  $\gamma_2 = 500$ ,  $\gamma_3 = 1000$ , and  $\gamma_4 = 100$ . Then, one of the parameters was modified, while the others remained fixed. For each set of parameters, we computed the metric for 5000 random game configurations.

Table 1 shows the Monte Carlo results for distinct values of the collision cost weight,  $\gamma_1$ . As we expect, the higher the collision cost weight, the higher the collision avoidance success rate. Nonetheless, notice that the collision cost weight has an inverse relationship with the percentage of valid profiles. Indeed, if no collision avoidance is regarded, i.e.,  $\gamma_1 = 0$ , the valid profile rate has its peak at 97.62%, while it decreases to 95.88%, when  $\gamma_1 = 5000$ . We point out that it is necessary to have a high valid profile rate and a high collision avoidance rate since the successful trajectory rate is always less than or equal to the minimum of the two rates.

Observe that for  $\gamma_1 = 0$ , nearly a fifth of the trajectories were obstacle-free. This is due to the fact that some trajectories are very close to straight lines, i.e., it is not necessary to dodge any obstacles. Indeed, 15.74% of the trajectories have only 2 waypoints, the initial and final positions, and there is a chance that no obstacle avoidance is necessary, depending on the initial and final orientations.

Now, considering the results of the variation of the invalid profile weight  $\gamma_2$  shown in Table 2, it is interesting to notice that the maximum trajectory time is considerably lower, when the invalid profile cost is considered: it decreased from 5.13 s for  $\gamma_2 = 0$  to 4.35 s for  $\gamma_2 = 500$ . Then, we can argue that the invalid profile cost helps the optimization to have better results in the most difficult scenarios.

In Table 3, the results for the optimization performance for an increasing number of iterations are displayed. With more iterations, the optimizer was able to find better solu-

**Table 1** Collision cost experiment

Collision cost weight	Trajectory time (s)						Valid profiles (%)	Collision avoidance (%)	Successful trajectories (%)
	Min	p25	p50	p75	p95	Max			
0	1.10	1.58	1.76	2.09	2.73	4.73	97.62	17.36	17.00
10	1.10	1.59	1.79	2.08	2.74	4.73	97.00	71.12	70.54
100	1.10	1.60	1.78	2.09	2.77	4.21	96.90	78.46	77.72
<b>5000</b>	<b>1.10</b>	<b>1.59</b>	<b>1.80</b>	<b>2.10</b>	<b>2.80</b>	<b>4.35</b>	<b>95.88</b>	<b>85.86</b>	<b>84.92</b>

The parameter  $\gamma_1$  is changed, while the others are held constant. In bold, we show the default value for  $\gamma_1$  and its performance

**Table 2** Invalid profile experiment

Invalid profile weight	Trajectory time (s)						Valid profiles (%)	Collision avoidance (%)	Successful trajectories (%)
	Min	p25	p50	p75	p95	Max			
0	1.10	1.59	1.79	2.13	2.77	5.13	94.20	85.60	84.12
10	1.10	1.60	1.81	2.10	2.80	4.35	95.82	86.82	85.98
100	1.10	1.59	1.81	2.11	2.78	4.35	95.60	86.60	85.66
<b>500</b>	<b>1.10</b>	<b>1.59</b>	<b>1.80</b>	<b>2.10</b>	<b>2.80</b>	<b>4.35</b>	<b>95.88</b>	<b>85.86</b>	<b>84.92</b>

The parameter  $\gamma_2$  is changed, while the others are held constant. In bold, we show the default value for  $\gamma_2$  and its performance

**Table 3** Optimization iterations experiment

Number of iterations	Trajectory time (s)						Valid profiles (%)	Collision avoidance (%)	Successful trajectories (%)
	Min	p25	p50	p75	p95	Max			
10	1.11	1.71	2.03	2.43	3.17	4.73	89.72	65.56	63.38
25	1.10	1.63	1.91	2.23	2.95	4.73	94.68	80.28	79.16
50	1.10	1.60	1.85	2.14	2.88	4.35	95.48	84.78	83.78
<b>100</b>	<b>1.10</b>	<b>1.59</b>	<b>1.80</b>	<b>2.10</b>	<b>2.80</b>	<b>4.35</b>	<b>95.88</b>	<b>85.86</b>	<b>84.92</b>

The parameter  $N_{\text{iter}}$  is changed, while the others are held constant. In bold, we show the default value for  $N_{\text{iter}}$  and its performance

tions, regarding all metrics. In fact, the 95th percentile of trajectory time decreased about 11.67%, when the number of iterations went from 10 to 100. The successful trajectory rate also significantly increased from 63.38 to 84.92% for the same parameter variation, mainly due to collision avoidance improvement. This shows that the initial solution provided by the combination of the visibility graph and the spline's initial parameters is in general not able to avoid collisions. An explanation for this phenomenon is that, even though the visibility graph is collision-free, the spline created from its waypoints has a hard to predict behavior, which greatly depends on the initial and final orientations and curvatures.

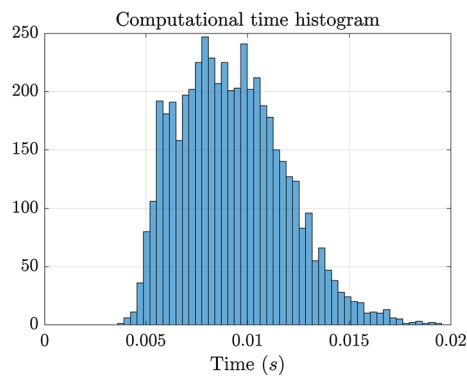
Another noticeable fact is that the increment of performance is marginal when the number of iterations goes from 50 to 100, indicating that the candidate solution is already close to the minimum. Then, in practice, it might not be worthwhile to try more than 100 iterations, when one has computational time restrictions.

## 6.2 Computational Time

We show the computational time distribution of a Monte Carlo experiment (with the default parameters, except the number of iterations was set to 50) in Fig. 10. Then, we can conclude that the algorithm is suitable for robot soccer applications since the 95 percentile was 0.0138 s, which is smaller than the camera sampling period of 0.0166 s (60Hz), commonly used as the allotted time for the strategy algorithm. Additionally, a maximum computational time reserved for the MTTP could be set up and the best solution found so far could be used, although no guarantee can be given that a valid solution is found within this reserved time.

## 7 Conclusions

This work presented a novel method for MTTP of a DDMR, where motor and non-slipping constraints are considered.



**Fig. 10** Computational time histogram of the MTTP algorithm. The algorithm was set to run for 50 iterations. The Monte Carlo experiment used 5000 samples

The framework considers linear and angular acceleration as inputs (or battery voltages) and deals with boundary conditions, such as initial and final velocities, orientations, and curvatures. We warm-start the optimization using a visibility graph path combined with a spline interpolation scheme. Then, after defining a cost function, which encodes the minimum-time objective, obstacle avoidance and the dynamic feasibility of the trajectory, the trajectory is improved iteratively by finding the minimum-time velocity profile and changing the waypoints with RPROP. Another contribution involves a benchmark for the planning framework based on Monte Carlo simulations, which enabled us to assess the performance of the framework in multiple scenarios and conclude that it can be run in real-time.

Future work includes: (1) validation of this method on real robots; (2) analysis of the loss in optimality due to the approximations employed over the robot and friction models in a similar fashion to Sherback et al. (2006); (3) meta-optimization to tune the parameters of the MTTP framework; and (4) including the movement of the obstacles during planning step (Morsali et al. 2019).

**Acknowledgements** Igor Okuyama acknowledges the support of CAPES (fellowship Proc. #88882.161990/2017-01). Rubens Afonso acknowledges the support of CAPES (fellowship Proc. #88881.145490/2017-01) and the Federal Ministry for Education and Research of Germany through the Alexander von Humboldt Foundation. Igor Okuyama and Marcos Maximo would like to thank the ITAndroids' sponsors: Altium, Cenic, Intel, ITAEx, Mathworks, Metinjo, Micropress, Polimold, Rapid, Solidworks, ST Microelectronics, WildLife, and Virtual Pyxis.

## A Model Parameters

Table 4 presents the modeling parameters used in this work.

**Table 4** Modeling parameters

Symbol	Value
$\beta$	$-6.1775 \text{ s}^{-1}$
$\gamma$	$61.640 \text{ rad V}^{-1} \text{ s}^{-2}$
$\mu$	0.675
$a_{t,\max}$	$2.1716 \text{ ms}^{-2}$
$a_{c,\max}$	$5.1948 \text{ ms}^{-2}$
$a_{t,\text{topp}}$	$8.8993 \text{ ms}^{-2}$
$a_{c,\text{topp}}$	$8.8993 \text{ ms}^{-2}$
$g$	$9.8 \text{ ms}^{-2}$
$\alpha_{\max}$	$60 \text{ rad s}^{-2}$
$I_c$	$200 \times 10^{-6} \text{ kg m}^2$
$h$	$36.45 \times 10^{-3} \text{ m}$
$L$	$33.10 \times 10^{-3} \text{ m}$
$R$	$30.00 \times 10^{-2} \text{ m}$
$r_{\text{obs}}$	$56.56 \times 10^{-3} \text{ m}$
$r_{\text{rob}}$	$56.56 \times 10^{-3} \text{ m}$
$r_{\text{ball}}$	$21.00 \times 10^{-3} \text{ m}$

## References

- Ben-Asher, J. Z., Wetzler, M., Rimón, E. D., & Diepolder, J. (2019). Optimal trajectories for a mobile robot with bounded accelerations in the presence of a wall or a bounded obstacle. In *2019 27th mediterranean conference on control and automation (MED)* (pp. 481–488). IEEE.
- Byrd, R. H., Curtis, F. E., & Nocedal, J. (2010). Infeasibility detection and SQP methods for nonlinear optimization. *SIAM Journal on Optimization*, 20(5), 2281–2299.
- Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3), 497–516.
- Haddad, M., Chettibi, T., Hanchi, S., & Lehtihet, H. (2007). A random-profile approach for trajectory planning of wheeled mobile robots. *European Journal of Mechanics-A/Solids*, 26(3), 519–540.
- Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107.
- Hérissé, B., & Pepy, R. (2013). Shortest paths for the Dubins' vehicle in heterogeneous environments. In *52nd IEEE conference on decision and control* (pp. 4504–4509). IEEE.
- Ho, Y. J., & Liu, J. S. (2009). Collision-free curvature-bounded smooth path planning using composite bezier curve based on voronoi diagram. In *2009 IEEE international symposium on computational intelligence in robotics and automation (CIRA)*, Daejeon, South Korea (pp. 463–468). IEEE.
- IEEE. (2008). Rules for the IEEE very small competition. [http://www.cbrobotica.org/wp-content/uploads/2014/03/VerySmall2008\\_en.pdf](http://www.cbrobotica.org/wp-content/uploads/2014/03/VerySmall2008_en.pdf). Accessed April 20th, 2019.
- Karaman, S., & Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104(2).
- Khatib, O., & Le Maitre, J. (1978). Dynamic control of manipulators operating in a complex environment. In *3rd CISM-IFTOMM symposium on theory and practice of robots and manipulators*, PWN, Udine, Italy, vol. 267.

- Kim, Y. J., Kim, J. H., & Kwon, D. S. (2001). Evolutionary programming-based univector field navigation method for past mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(3), 450–458.
- Latombe, J. C. (2012). *Robot motion planning* (Vol. 124). Boston: Springer.
- LaValle, S. M. (1998). *Rapidly-exploring random trees: A new tool for path planning*. Technical report, Ames, IA, USA.
- LaValle, S. M. (2006). *Planning algorithms*. Cambridge: Cambridge University Press.
- LaValle, S. M., & Kuffner, J. J., Jr. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5), 378–400.
- Lepetič, M., Klančar, G., Škrjanc, I., Matko, D., & Potočnik, B. (2003). Time optimal path planning considering acceleration limits. *Robotics and Autonomous Systems*, 45(3–4), 199–210.
- Lim, Y., Choi, S. H., Kim, J. H., & Kim, D. H. (2008). Evolutionary univector field-based navigation with collision avoidance for mobile robot. *IFAC Proceedings Volumes*, 41(2), 12787–12792.
- Lozano-Pérez, T., & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10), 560–570.
- Manor, G., Ben-Asher, J. Z., & Rimón, E. (2018). Time optimal trajectories for a mobile robot under explicit acceleration constraints. *IEEE Transactions on Aerospace and Electronic Systems*, 54(5), 2220–2232.
- Metropolis, N., & Ulam, S. (1949). The monte carlo method. *Journal of the American Statistical Association*, 44(247), 335–341.
- Morsali, M., Frisk, E., & Åslund, J. (2019). Deterministic trajectory planning for non-holonomic vehicles including road conditions. *Safety and Comfort Factors*, 52(5), 97–102.
- Okuyama, I. F. (2019). Minimum-time obstacle avoidant trajectory planning for a differential drive robot considering motor and no-slipping constraints. Master's thesis, Instituto Tecnológico de Aeronáutica.
- Okuyama, I. F., Maximo, M. R. O. A., Cavalcanti, A. L. O., & Afonso, R. J. M. (2017). Nonlinear grey-box identification of a differential drive mobile robot. In *XIII Simpósio Brasileiro de Automação Inteligente*, SBAI, Porto Alegre, RS, BR.
- Purwin, O., & D'Andrea, R. (2006). Trajectory generation and control for four wheeled omnidirectional vehicles. *Robotics and Autonomous Systems*, 54(1), 13–22.
- Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE international conference on neural networks*, vol. 1 (pp. 586–591). IEEE, San Francisco, CA, USA.
- Rimon, E., Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. Departmental Papers (ESE) p. 323.
- Russell, S., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* (3rd ed.). Upper Saddle River, NJ: Prentice Hall Press.
- Sherback, M., Purwin, O., & D'Andrea, R. (2006). Real-time motion planning and control in the 2005 Cornell RoboCup system. *Robot Motion and Control* (pp. 245–263). London: Springer.
- Shin, K., & McKay, N. (1985). Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6), 531–541.
- Sprunk, C. (2008). *Planning motion trajectories for mobile robots using splines*. Technical Report, Freiburg, Germany.
- Webb, D. J., & van den Berg, J. (2013). Kinodynamic RRT\*: Asymptotically optimal motion planning for robots with linear dynamics. In *2013 IEEE international conference on robotics and automation* (pp. 5054–5061). Karlsruhe, Germany. IEEE.
- Yamamoto, M., Iwamura, M., & Mohri, A. (1998). Time-optimal motion planning of skid-steer mobile robots in the presence of obstacles. In *Proceedings of 1998 IEEE/RSJ international conference on intelligent robots and systems. innovations in theory, practice and applications* vol. 1 (pp. 32–37). IEEE, Victoria, BC, Canada.
- Yamamoto, M., Iwamura, M., & Mohri, A. (1999). Quasi-time-optimal motion planning of mobile platforms in the presence of obstacles. In *Proceedings 1999 IEEE international conference on robotics and automation* vol. 4 (pp. 2958–2963). IEEE, Detroit, MI, USA.
- Zhu, Z., Schmerling, E., & Pavone, M. (2015). A convex optimization approach to smooth trajectories for motion planning with car-like robots. In *2015 54th IEEE conference on decision and control (CDC)* (pp. 835–842). IEEE.

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.