NMPC Strategy for Mobile Robot Navigation in an Unknown Environment

Iuro B. P. Nascimento [*] **Antonio Ferramosca** [**]
Luciano C. A. Pimenta [*,***] **Guilherme V. Raffo** [*,***]

[*] *Graduate Program in Electrical Engineering, Federal University of Minas Gerais, CEP 31270-901, Belo Horizonte, Minas Gerais, Brasil. (iuro@ufmg.br)*
[**] *CONICET - UTN Facultad Regional Reconquista, Santa Fe 3560, Argentina. (ferramosca@santafe-conicet.gov.ar)*
[***] *Department of Electronic Engineering, Federal University of Minas Gerais, CEP 31270-901, Belo Horizonte, Minas Gerais, Brasil (lucpim@cpdee.ufmg.br,raffo@ufmg.br)*

Abstract: This work presents a Nonlinear Model Predictive Control strategy for mobile robot navigation in unknown environments. The control system aims to reach a goal safely, as fast as possible, minimizing the control effort, and the distance between the current trajectory and the goal. A LIDAR (Light Detection and Ranging) sensor is used to obtain obstacles information as these are approached by the robot. The LIDAR output is then processed to obtain inequality constraints describing a collision-free area. This area is partitioned in convex subregions in order to be used within the proposed NMPC approach to perform path tracking, ensuring obstacle avoidance. Numerical results are provided to corroborate the performance of the system.

Resumo: Este trabalho apresenta uma estratégia de controle preditivo não linear para a navegação de um robô móvel em um ambiente desconhecido. O sistema de controle é projetado com o objetivo de levar o robô ao alvo de forma segura, com tempo mínimo, minimizando o esforço de controle e a distância da trajetória ao alvo. A saída de um sensor LIDAR (Light Detection and Ranging) é processada de forma a se obter restrições de desigualdade de uma area livre de colisão a qual o robô pode se movimentar. Esta area livre é dividida em subpartes convexas de forma a ser usada no NMPC proposto para seguimento de trajetórias assegurando o evitamento de obstáculos. Resultados numéricos são apresentados de forma a demonstrar a perfomance do sistema.

Keywords: NMPC; Mobile Robot; Obstacle Avoidance; Unknown Environment
Palavras-chaves: NMPC; Robô Móvel; Evitamento de Obstáculo; Ambiente Desconhecido

1. INTRODUCTION

Academic and Industrial researches of wheeled mobile robots have been active in the past decades. There are many applications, such as domestic tasks, access to dangerous areas, planet exploration, agriculture, and mining. In many cases, an indoor mobile robot is considered, and in most works with a known and mapped environment. In these cases, problem solution can be divided into two phases: trajectory planning and trajectory tracking control. In this approach, the trajectory planning can be obtained offline and can be executed only once. In the literature, many algorithms undertake the planning task. Some algorithms sample collision-free portions of the map to form graphs of paths, like probabilistic roadmaps (PRM) (Kavraki et al., 1996), rapidly-exploring random trees (RRT) (LaValle, 1998), and their asymptotically optimal versions PRM* and RRT* (Karaman and Frazzoli, 2011), among others. Depending on the characteristics of the environment, this planning can have a high cost. Any path tracking control strategy can combine with these planners, altough they will not deal with changes in the environment.

In other applications, the environment is unknown, and therefore, the location and shape of obstacles are not known beforehand. At every change in the environment or every discovery of a new obstacle, the planning algorithm needs to be recomputed, leading to a high computational cost.

Some methods recompute part of the path to adapt to small changes in the environment, as D^*(Stentz, 1994). These methods will lead to a lower computational cost, although they are designed to be efficient with small changes in the environment. An entirely unknown environment will

⋆ This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Codes 001 e 88887.136349/2017-00, by the projects INCT InSAC and Universal under the grants CNPq 465755/2014-3 and 426392/2016-7, and also by the Brazilian agency FAPEMIG.
Luciano C. A. Pimenta and Guilherme V. Raffo are also members of the National Institute of Science and Technology (INCT) for Cooperative Autonomous Systems Applied to Security and Environment.

DOI: 10.17648/sbai-2019-111556

still demand a constant change of the path and therefore be costly.

Another approach is to combine optimal path planning and optimal control into a single optimization problem. This approach has the advantage of obtaining an optimal solution that takes into account the whole problem, considering the robot model, the constraints, and possibly an economic criterion.

Model Predictive Control (MPC) is well suited to solve this unified control problem. In the MPC framework, we obtain an optimal control law by solving an optimal control problem (OCP) with some performance index and subject to path constraints such as obstacles, limits on states and inputs, among others.

In the literature, many algorithms are using the MPC framework for obstacle avoidance. Drozdova et al. (2016) uses a laser scanner to detect obstacles and models them as ellipses. The ellipses equations are used as inequality constraints in the OCP. Liu et al. (2018) combines VPH+ (enhanced vector polar histogram) and MPC for unmanned ground vehicles (UGVs). VPH+ calculates directions to avoid obstacles, and the MPC is used as a local planner and to find optimal control values along with points in the desired direction.

In some cases, the system model and the constraints are nonlinear and non-convex, which leads to non-convex optimal control problems. OCPs are often discretized and solved as a nonlinear programming problem (NLP). In many cases, non-convex optimization problems have multiple minima solutions. The solution found will depend on the trajectory's initial guess. As an example, in a case with one obstacle in front of the robot, there are at least two possible paths, making a left turn and a right turn to avoid the obstacle. Both of these paths will have a local minima solution. There might be more local minima solutions depending on the states, inputs, and constraints used in the problem.

Liu et al. (2017) develops a Nonlinear MPC (NMPC) strategy for an autonomous ground vehicle (AGV) truck in an unknown environment with obstacles modeled as polygons to obtain linear constraints. The OCP is nonlinear and non-convex. In this paper, it is explored several local minima by considering more possible paths. It also uses a cost function tailored to the truck model and state variables.

The present work proposes an NMPC strategy of a wheeled mobile robot in an unknown environment. Differently from Liu et al. (2017), we use a generic quadratic functional cost that could be used with other systems. We also use computational geometry algorithms to model obstacles, which is detected by a laser scanner sensor, as polygons. In order to find a better local minima solution to the OCP, we also explore more possible paths.

This work is organized as follows. Section 2 describes the wheeled mobile robot model. Section 3 presents the proposed control strategy. Section 4 shows the optimal control problem (OCP) formulation, describing the generation of obstacles constraints and the OCP formulation. Section 5 presents numerical simulation results, and section 6 concludes the work.

2. DIFFERENTIAL WHEELED MOBILE ROBOT MODEL

The differential wheeled mobile robot schematic is shown in Fig. 1. A kinematic model is obtained by tracking a point at a distance d in front of the robot's geometric center. The differential equations are given by

$$\begin{aligned}\dot{x}(t) &= \frac{V_R(t)+V_L(t)}{2}\left(\cos\psi(t) - d\sin\psi\right)\\ \dot{y}(t) &= \frac{V_R(t)+V_L(t)}{2}\left(\sin\psi(t) + d\cos\psi\right)\\ \dot{\psi}(t) &= \frac{V_R(t)-V_L(t)}{R}\end{aligned} \tag{1}$$

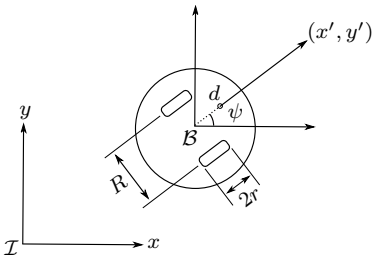

Figure 1. Schematic of differential drive mobile robot.

where V_R and V_L are the right and left wheels linear velocities, respectively. $\boldsymbol{x} = [x\ y\ \psi]^T$ is the state-space vector of the system, where x and y are the position of the robot body frame $\mathcal{B}$ w.r.t the inertial frame $\mathcal{I}$. ψ is the rotation of $\mathcal{B}$ w.r.t. $\mathcal{I}$ expressed in $\mathcal{I}$. R is the distance between the robot wheels. $\boldsymbol{u} = [V_R\ V_L]^T$ is the vector of inputs.

3. CONTROL STRATEGY

3.1 Problem Statement

Considering a wheeled mobile robot in an unknown environment, the problem consists of moving the mobile robot from an initial position to a target position as fast as possible. The environment may contain static obstacles with position, size, and shape not known a priori.

3.2 Control Scheme

The control strategy consists of two main tasks:

- generate the obstacle constraints;
- formulate and solve optimal control problems (OCPs).

The control system schematic is illustrated in Fig. 2.

The constraint generation (CG) task provides the constraints by processing a planar LIDAR (Light Detection and Ranging) output in order to maintain the robot in the obstacle-free region, which is called the safe region.

The NMPC task uses these constraints and others, such as the mathematical model, states and inputs bounds to formulate and solve the OCP. The obtained optimal trajectory and inputs satisfying these constraints will be feasible. These tasks will be described in detail in the next sections.

DOI: 10.17648/sbai-2019-111556

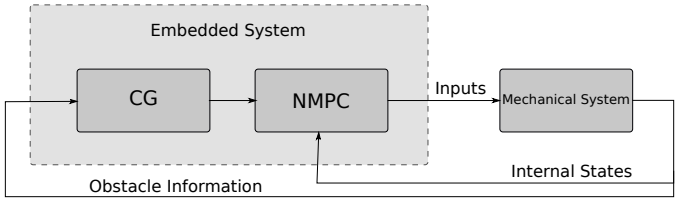

Figure 2. System Schematic.

3.3 Constraint Generation

The CG task generates constraints to maintain the mobile robot in a safe region, which is accomplished by processing the LIDAR output.

In order to obtain these constraints, the CG models obstacles as polygons by performing three subtasks: line simplification; introduction of a safety margin; and a convex decomposition of the safe region polygon. Fig. 3 shows a representation of the LIDAR output, where the LIDAR distance range R_{LIDAR} is finite, and the angles have a range from -90^o to 90^o. Circular arcs where no obstacles are found by the LIDAR are called openings.

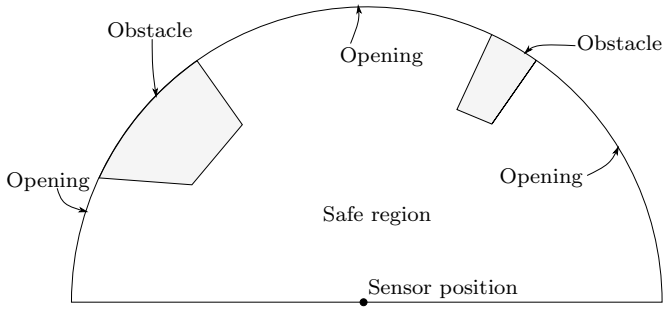

Figure 3. Typical polygon formed by LIDAR output.

Line Simplification: The Ramer-Douglas-Peucker (Ramer, 1972) algorithm is used to perform line simplification. The algorithm is an iterative method where, given a maximum error (ϵ) and a curve represented by a vector of points, it finds simplified lines represented by a subset of the original points.

The method initializes by marking the starting, p_s, and final, p_f, points to be kept. It finds the point p_m that is farthest to the line formed by p_s and p_f. If p_m is farther out of the line than ϵ, p_m is marked as to be kept, and the algorithm is executed recursively with the lines formed by p_s and p_m and the one formed by p_m and p_f.

Fig. 4 shows an example of line simplification. The orange line connects the original points, and the blue line is the simplified line.

Addition of a Safety Margin A safety margin is added to the safe region polygon to take into consideration the vehicle size and possible error in detecting and processing obstacles. Some computational geometry algorithms inflate and deflate polygons. In this work, it is used the Vatti's clipping algorithm (Vatti, 1992).

We only need to expand obstacles, not the openings. Fig. 5. shows an example of the expansion of obstacles.

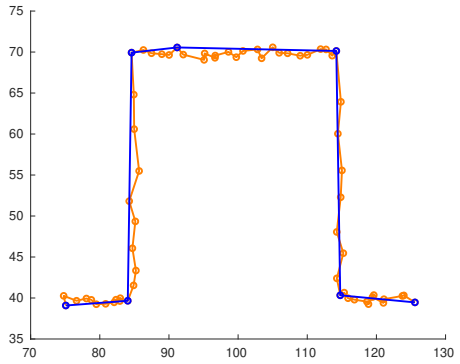

Figure 4. Example of Ramer-Douglas-Peucker algorithm.

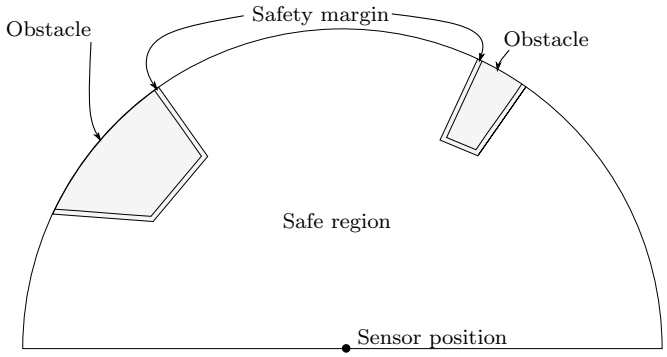

Figure 5. Obstacle Polygon Expansion.

Convex Decomposition The safe region is usually non-convex in the presence of obstacles. It is hard if not impossible to describe a non-convex region as one function. The safe region may also have other problems:

- The NLP solver will not explore other possible solutions. It will find a solution near the initial guess.
- The safe region tends to have edges and/or corners that are not twice differentiable. In this case, the NLP solver will use numerical derivatives that have a higher error. This error might be large enough to cause the solver to have slow convergence or not converge.

In order to deal with these problems, we decompose the safe region into a set of convex polygon subregions. These polygons will form sets of constraints, one set per subregion. For each subregion polygon, each line equation will form one inequality constraint, which is linear.

A problem arises when the robot needs to pass over a narrow subregion. In such a case, it may become hard for the solver to find a feasible solution to the OCP. In order to mitigate this problem, two algorithms are used to decompose the safe region: a polar decomposition and an optimal convex decomposition. If the solver fails to reach a feasible solution using the constraints obtained with the optimal convex partition, the polar decomposition is used. Fig. 6 shows an example of both decompositions. The polar partitioning is not optimal, it will create more subregions, and some of them may become narrow. Since the optimal convex partition algorithm produces the minimum number of partitions, it has a smaller chance of occurrence of a narrow subregion.

DOI: 10.17648/sbai-2019-111556

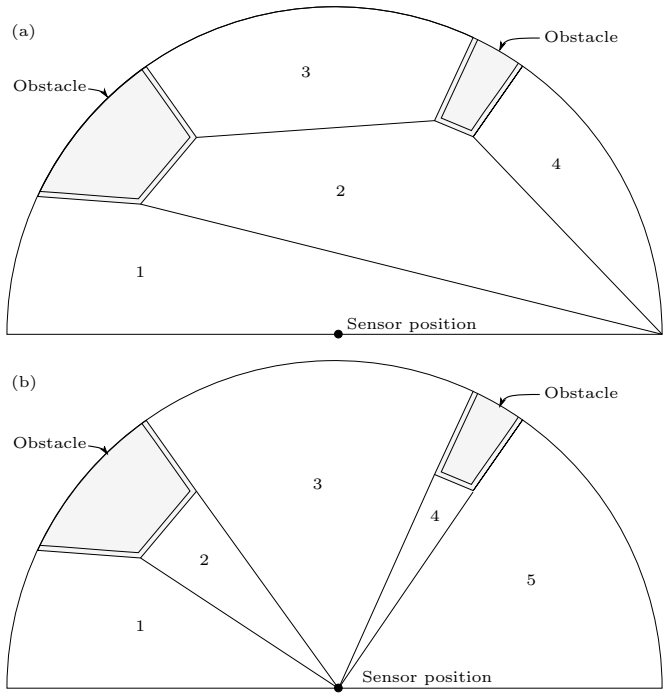

Figure 6. Polygon decomposition: (a) Polar Partition; (b) Optimal Convex Partition.

On the one hand, the polar partitioning consists of creating a new line from each obstacle vertex to the position of the robot. On the other hand, the Optimal Convex Decomposition algorithm is a dynamic programming algorithm (Greene, 1983).

In order to explore more paths, all openings are considered as a possible solution. A path from the robot position through the subregions to each opening is obtained. In order to choose which subregions to pass to reach the opening, we create a graph where each node is a subregion, and each edge represents the adjacency between two subregions. Adjacent subregions share at least a line segment. The path to each opening is obtained by using the Dijkstra shortest path algorithm (Dijkstra, 1959).

As an example, Fig. 6 (b) shows a safe region polygon decomposed into convex polygon subregions.

The start subregion is number 1, which also has an opening. Subregions 3 and 4 also have openings. Fig. 7 shows the adjacency graph.

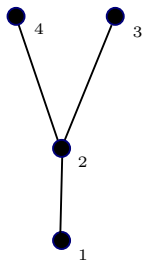

Figure 7. Adjacency Graph.

In this case, using a weight of 1 for each edge in the graph, the best possible paths to each opening will be:

- 1
- $1 \rightarrow 2 \rightarrow 4$
- $1 \rightarrow 2 \rightarrow 3$

Considering the weights of the edges as 1, we aim to find the best path with the smallest number of nodes or subregions to go through. The phases number is minimized, which reduces constraints number. Therefore the computational cost is reduced.

After the decomposition, each subregion inequality constraints will be written as

$$\boldsymbol{A}_i \boldsymbol{\xi}_j - \boldsymbol{b}_i \leq \boldsymbol{0}, \tag{2}$$

where $\boldsymbol{\xi}_j = [x_j \ y_j]^T$, $j = 1, 2...K$ with K points in the discretized trajectory. $\boldsymbol{A}_i$ is a $N \times 2$ matrix, $\boldsymbol{b}_i$ and $\boldsymbol{0}$ are $N \times 1$ column vectors, N is the number of constraints, and $i = 1, 2, ...M$ with M subregions.

3.4 Nonlinear MPC Strategy

Using the subregion information, a constrained multi-phase OCP is formulated. In a multi-phase OCP, the trajectory is divided into segments where each segment may have different dynamical equations and constraints. Some problems demand changes in the system and/or constraints along the planned trajectory. This is the case of the stages of a spaceship launch. Each segment is one phase of the multi-phase OCP.

For each opening, one subregion path is chosen through a graph search, as explained in section 3.3 from the first subregion to each subregion that has an opening. For each of these paths, their constraints are used to formulate an OCP. Since each path has an independent OCP, all paths are formulated and solved in parallel. The solution with the smallest cost is chosen.

4. OPTIMAL CONTROL PROBLEM FORMULATION

The multi-phase constrained OCP is formulated as follows

$$\underset{\boldsymbol{x},\boldsymbol{u},T_1,\ldots,T_N}{\text{minimize}} \quad J = \mathscr{T}\left(\boldsymbol{x}^{(N)}(T_P), \boldsymbol{x}_{goal}(T_P), T_P\right) + \sum_{i=1}^{N} \left[\int_{T_{i-1}}^{T_i} \mathscr{I}\left(\boldsymbol{x}^{(i)}(t), \boldsymbol{u}^{(i)}(t),\right) dt\right] \tag{3}$$

$$\underset{\forall i=1,\ldots,N}{\text{subject to}} \quad \dot{\boldsymbol{x}}^{(i)}(t) = \mathscr{V}\left(\boldsymbol{x}^{(i)}(t), \boldsymbol{u}^{(i)}(t)\right) \tag{4}$$

$$\boldsymbol{x}^{(i)}_{min}(t) \leq \boldsymbol{x}^{(i)}(t) \leq \boldsymbol{x}^{(i)}_{max}(t) \tag{5}$$

$$\boldsymbol{u}^{(i)}_{min}(t) \leq \boldsymbol{u}^{(i)}(t) \leq \boldsymbol{u}^{(i)}_{max}(t) \tag{6}$$

$$\mathscr{F}\left(\boldsymbol{x}^{(i)}(t)\right) \leq 0 \tag{7}$$

$$\mathscr{G}\left(\dot{\boldsymbol{u}}^{(i)}(t)\right) \leq 0 \tag{8}$$

$$\boldsymbol{x}^{(i)}(T_{i-1}) = \boldsymbol{x}^{(i-1)}(T_{i-1}) \tag{9}$$

$$t \in [T_{i-1}, T_i], \ T_{i-1} < T_i \tag{10}$$

$$\text{subject to} \quad \mathscr{F}\left[\boldsymbol{x}^{(N)}(T_N), \boldsymbol{x}^{(0)}(T_0)\right] \leq 0 \tag{11}$$

$$T_0 = 0, \ T_N = T_p \tag{12}$$

DOI: 10.17648/sbai-2019-111556

where (3) is the cost functional, with one cost per each of the N phases, and a terminal cost. Equations (4) to (10) are constraints of each phase. Equations (11) and (12) are constraints of all phases. $\boldsymbol{x}_{goal}$ is the target state. (4) are the nonlinear dynamical equations (1) of the mobile robot, with one set of equations (1) per phase. Equations (5) and (6) are the states and inputs bounds. Equation (7) is the position constraint. Equation (8) limits the derivative of the inputs to account for unmodelled actuator dynamics. Equations (9) are imposed to ensure a continuous transition from each phase to the next. Equation (10) enforces the phase start and final times to be monotonically increasing. Equation (11) imposes the initial state to be the current pose of the robot and a terminal region on the final state. Finally, equation (12) sets the initial time to 0 and the final time of the last phase T_N to be equal to the time horizon T_p. The cost functions and constraints will be described in more detail in the next sections.

4.1 Cost Functional

The multi-phase cost function of equation (3) is given by

$$J = w_t T_p + \left\| \boldsymbol{x}^{(N)}(T_p) - \boldsymbol{x}_{goal} \right\|_{\boldsymbol{P}}^2 + \sum_{i=1}^{N} \left[\int_{T_{i-1}}^{T_i} \left(\left\| \boldsymbol{x}^{(i)}(t) - \boldsymbol{x}_{goal} \right\|_{\boldsymbol{Q}}^2 + \left\| \boldsymbol{u}^{(i)}(t) - \boldsymbol{u}_{\boldsymbol{eq}} \right\|_{\boldsymbol{R}}^2 \right) dt \right]. \quad (13)$$

It includes one cost per each of the N phases, where the energy of states far from the target state ($\boldsymbol{x}_{goal}$) and the energy of inputs far from the equilibrium values ($\boldsymbol{u}_{\boldsymbol{eq}}$) are penalized. Considering that the problem is non-convex, the cost on the energy of inputs helps to have a smaller chance of multiple solutions, and with a suitable choice of the matrix $\boldsymbol{R}$, avoid a solution with aggressive inputs. The weighted cost on the time horizon T_p is to achieve a minimum time solution. $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are weighing matrices.

4.2 State and Input Bounds

Equations (5) and (6) are the bounds on states and inputs to take into account limits of the physical system. The limits on each time point of state and input are $\boldsymbol{x}_{min}$, $\boldsymbol{x}_{max}$, $\boldsymbol{u}_{min}$, and $\boldsymbol{u}_{max}$. When the LIDAR detects an obstacle; the input maxima are lowered to $U_{max_{constrained}}$. The value of $U_{max_{constrained}}$ is selected according to the robot to maintain safety in the presence of obstacles.

4.3 Path Constraints

Equation (7) is the path constraints. These constraints are the subregion constraints of equation (2). Without obstacles, no constraint is added to the OCP. From equation (2) each subregion constraint is given by

$$\tilde{\boldsymbol{A}}_i = [\boldsymbol{A}_i \ \boldsymbol{0}_{N\times 1}]\, \boldsymbol{x}_j - \boldsymbol{b}_i \leq \boldsymbol{0}. \quad (14)$$

where N is the number of constraints in the subregion, $i = 1, 2, ..., M$ with M subregions, $j = 1, 2, ..., K$ with K points in the discretized trajectory, $\boldsymbol{A}_i$ is an $N \times 2$ matrix of equation (2).

4.4 Input Derivative Constraints

Equation (8) accounts to unmodelled dynamics of actuators. Since real actuators do not change instantaneously from minimum to maximum value, the constraint

$$\left| \frac{d\boldsymbol{u}(t)}{dt} \right| \leq \boldsymbol{dU}_{max} \quad (15)$$

limits the input variation. $U_{max} = [dV_{r_{max}}, dV_{l_{max}}]$ is the vector of maximum derivative values of wheels velocities.

4.5 State Continuity

In order to ensure a continuous transition between phases, the dynamical equation (4) of phase $i-1$ needs to have its final state equal to the initial state of phase i, as stated in (9).

4.6 Terminal Constraints

Three different terminal constraints will be used depending on whether the target states are inside the sensor field of vision or not.

If the target state is not inside the sensor field of vision, we ensure that the final state of the final phase is near the border of the field of vision. The following constraint is added

$$R_{LIDAR} - \delta \leq \left\| \boldsymbol{\xi}^{(N)}(T_N) - \boldsymbol{\xi}^{(1)}(T_0) \right\| \leq R_{LIDAR}, \quad (16)$$

where $\boldsymbol{\xi}^{(i)}(t) = \left[x^{(i)}(t)\ y^{(i)}(t)\right]^T$, R_{LIDAR} is the LIDAR range, and δ is a small value to define the terminal region near to the end of the sensor field of vision.

If the target state is inside the sensor field of vision, the following inequalities are used as terminal constraints

$$\begin{aligned} x_{goal} - \sigma \leq x^{(N)}(T_N) \leq x_{goal} + \sigma, \\ y_{goal} - \sigma \leq y^{(N)}(T_N) \leq y_{goal} + \sigma, \end{aligned} \quad (17)$$

where the terminal region, in this case, is a square of side size of 2σ with its center on the target position $\boldsymbol{\xi}_{goal}$. Equation (18) is added to plan a trajectory without wheels velocities at the target position.

$$\boldsymbol{u}^{(N)}(T_N) = \boldsymbol{0}. \quad (18)$$

5. RESULTS AND DISCUSSION

5.1 OCP Solution

At each sample time, all optimal control problems (OCPs) with paths to all openings are solved in parallel, where the solution with the smallest cost is chosen. The first control signal of the chosen solution is applied to the system. Each OCP of equations (3)-(12) is transcribed to a Nonlinear Programming (NLP) Problem using the direct method HP-adaptive pseudospectral (Darby et al., 2011) and solved by the interior point algorithm with a filter line

DOI: 10.17648/sbai-2019-111556

search implemented in IPOPT (Interior Point Optimizer) (Wächter and Biegler, 2006).

The HP-adaptive pseudospectral is a direct collocation method with an adaptive mesh refinement scheme to transcribe a continuous-time OCP into an NLP. The method approximates states and controls with a variable number of segments of a variable number of polynomial degrees. The dynamical equations of the system are transformed into differential-algebraic constraints evaluated at the Legendre-Gauss-Radau (LGR) points. In each segment, the states and controls are sampled at the LGR points using the Lagrange polynomial as an interpolant. Each segment is evaluated, and if its maximum error is higher than a tolerance ϵ, the algorithm may decide either by dividing into more segments or to augment the polynomial approximation degree in order to reduce the maximum error. Once all segments have their maximum errors below the tolerance, the algorithm finishes.

The transcribed NLP is given by

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{g}_L \leq \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{g}_U \\ & \boldsymbol{x}_L \leq \quad \boldsymbol{x} \quad \leq \boldsymbol{x}_U \end{aligned} \tag{19}$$

where $\boldsymbol{x}$ is the decision variable vector, $\boldsymbol{g}(\boldsymbol{x})$ are the constraints, $\boldsymbol{g}_U$ and $\boldsymbol{g}_L$ are the upper and lower bounds on the constraints, and $\boldsymbol{x}_U$ and $\boldsymbol{x}_L$ are the upper and lower bounds of decision variables. Equality constraints are enforced by making upper and lower bounds equal. The IPOPT is designed to solve large scale NLPs of the form of (19). IPOPT solves a barrier problem for a fixed value of Lagrange multipliers μ and iteratively decreases this value. When μ converges to zero, the Karush-Kuhn-Tucker (KKT) conditions of the original problem are satisfied.

5.2 Results

Numerical results are obtained using Matlab R2017b and Simulink software. The nonlinear dynamical model (1) is simulated in Simulink, and the controller is implemented in Matlab using the Casadi (Andersson et al., 2018) toolbox for the formulation of the OCP and for providing exact derivatives and Hessians by the automatic differentiation tool. The Matlab Robotics System Toolbox provides a class to handle 2D maps with software to emulate a 2D LIDAR and other planar map-related functions.

The optimal convex decomposition algorithm used the CGAL library (Hert, 2018) implementation of the dynamic programming decomposition by Greene (1983). The Clipper library (Johnson, 2014) was used to provides polygon offsetting methods. The implemented Rammer-Douglas-Peucker algorithm was from Ahmadzadeh (2017).

The parameters used in the simulation are described in Table 1.

Fig. 8, 9, and 10 show the results. Fig. 9 shows the position, orientation, and their derivatives. The initial robot orientation in the direction of the target is 90 *degrees*, and to avoid obstacles, the robot makes turns with the orientation varying from 60 *degrees* to 115 *degrees*, as shown in Fig. 9. Due to the minimum time cost, a good part of the controls signals is close to the maximum constrained value, as shown in Fig 10. Initially, there is no obstacle in the LIDAR field of vision, and the control signals are close to 29 m/s. From 0.5 s to 38.5 s obstacles appear and the maximum wheel velocity is 9 m/s. After 38.5 s obstacles are behind the vehicle and will not be detected by the LIDAR, therefore the maximum velocity increases to 29 m/s. Around 40 s, the target position is inside the LIDAR field of vision, and the constraint of equation (18) is applied to stop at the target position. Constraints (8) keeps the controls signals in a smooth curve towards zero. This constraint prevents on/off control signals switching between 29 and -5 m/s, which are likely to occur due to the minimum cost. After reaching the terminal region at approximately 46.2 s, robot stops and the inputs are zero.

Table 1. Simulation Parameters.

Parameter	Value
R	1 m
σ	0.1 m
δ	5 m
$dV_{R_{max}}$, $dV_{L_{max}}$	3 m/s^2
ϵ	1
R_{LIDAR}	100 m
w_t	0.1
$\boldsymbol{Q}$	$\text{diag}(1/200^2, 1/200^2, 1/(4\pi^2))$
$\boldsymbol{R}$	$\text{diag}(1/(29+5)^2, 1/(29+5)^2)$
$\boldsymbol{P}$	$\text{diag}([1,1,1])$
U_{max}	29 m/s
U_{min}	-5 m/s
$U_{max_{constrained}}$	9 m/s

The 2D trajectory and environment presented in Fig. 8 shows a scene where obstacles are sufficiently close to needing the reduction of velocity in order to avoid a collision. The mobile robot moves throughout obstacles with a safe distance due to the addition of a safety margin, as described in section 3.3.2.

6. CONCLUSION

This work proposed a multi-phase NMPC control strategy for a differential wheeled mobile robot in an unknown environment. The task was completed as fast as possible and taking into account the energy of states and inputs. A LIDAR sensor output was processed in order to obtain a safe area, which is modeled as a polygon. In order to avoid non-convex constraints and to explore more possible paths, the safe area was divided into convex polygons, which compose the phase constraints of the NMPC. Simulation results showed that it is possible to apply the proposed NMPC strategy to the mobile robot. Future works will focus on developing a hardware-in-loop simulation and implementation in a mobile robot.

REFERENCES

Ahmadzadeh, R. (2017). Ramer-douglas-peucker.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2018). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*.

Darby, C.L., Hager, W.W., and Rao, A.V. (2011). An hp-adaptive pseudospectral method for solving optimal control problems. *Optimal Control Applications and Methods*, 32(4), 476–502.

DOI: 10.17648/sbai-2019-111556

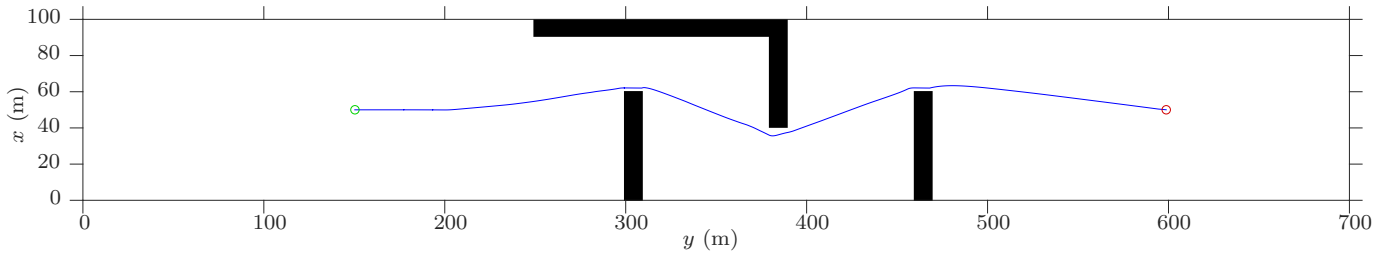

Figure 8. The state space trajectory. The initial position is in the green circle and the goal position is in the red circle.

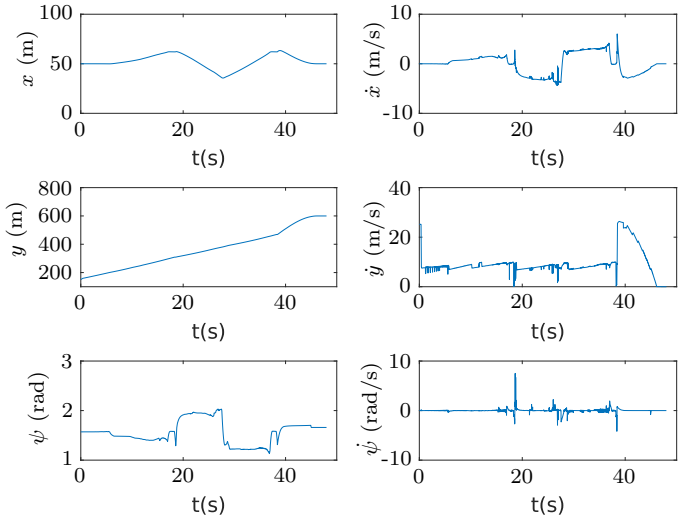

Figure 9. Position, orientation and derivatives.

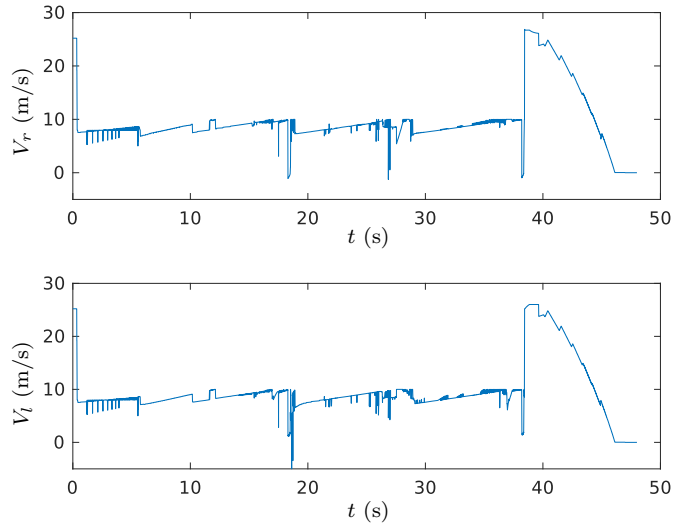

Figure 10. Controls Signals

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271.

Drozdova, E., Hopfgarten, S., Lazutkin, E., and Li, P. (2016). Autonomous driving of a mobile robot using a combined multiple-shooting and collocation method. *The 9th IFAC Symposium on Intelligent Autonomous Vehicles (IAV 2016)*, 49(15), 193–198.

Greene, D.H. (1983). The decomposition of polygons into convex parts. *Computational Geometry*, 1, 235–259.

Hert, S. (2018). 2D polygon partitioning. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.13 edition.

Johnson, A. (2014). 2D polygon partition. In *Clipper Library*. Angus Johson, 6.1.3 edition.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7), 846–894.

Kavraki, L.E., Svestka, P., Latombe, J.C., and Overmars, M.H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4), 566–580.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. *Citeseer*.

Liu, J., Jayakumar, P., Stein, J.L., and Ersal, T. (2017). Combined speed and steering control in high-speed autonomous ground vehicles for obstacle avoidance using model predictive control. *IEEE Transactions on Vehicular Technology*, 66(10), 8746–8763.

Liu, K., Gong, J., and Chen, H. (2018). Vph+ and mpc combined collision avoidance for unmanned ground vehicle in unknown environment. *arXiv preprint arXiv:1805.08089*.

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3), 244–256.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 3310–3317 vol.4. doi:10.1109/ROBOT.1994.351061.

Vatti, B.R. (1992). A generic solution to polygon clipping. *Communications of the ACM*, 35(7), 56–63.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57. doi:10.1007/s10107-004-0559-y.

DOI: 10.17648/sbai-2019-111556