A Penalty Method Based Approach for Autonomous Navigation using Nonlinear Model Predictive Control [⋆]

Ben Hermans [*] **Panagiotis Patrinos** [**] **Goele Pipeleers** [*]

[*] *MECO Research Team, Department of Mechanical Engineering, KU Leuven, Leuven, Belgium*
DMMS lab, Flanders Make, Leuven, Belgium
(e-mail: ben.hermans2@kuleuven.be).
[**] *Division ESAT-STADIUS, Department of Electrical Engineering, KU Leuven, Leuven, Belgium*

Abstract: This paper presents a novel model predictive control strategy for controlling autonomous motion systems moving through an environment with obstacles of general shape. In order to solve such a generic non-convex optimization problem and find a feasible trajectory that reaches the destination, the approach employs a quadratic penalty method to enforce the obstacle avoidance constraints, and several heuristics to bypass local minima behind an obstacle. The quadratic penalty method itself aids in avoiding such local minima by gradually finding a path around the obstacle as the penalty factors are successively increased. The inner optimization problems are solved in real time using the proximal averaged Newton-type method for optimal control (PANOC), a first-order method which exhibits low runtime and is suited for embedded applications. The method is validated by extensive numerical simulations and shown to outperform state-of-the-art solvers in runtime and robustness.

Keywords: Obstacle avoidance, Predictive control, Nonlinear programming, Optimization, Optimal control

1. INTRODUCTION

Driverless cars, fruit-picking robots and automated guided vehicles in a warehouse are examples of autonomous motion systems that are up-and-coming in industry. In all these applications, the computation of a collision-free trajectory is essential. Computing motion trajectories that satisfy collision-avoidance constraints has been the topic of substantial research, resulting in a variety of methods, including the potential field method (Ge and Cui, 2002; Montiel et al., 2015) and methods using velocity obstacles (Guy et al., 2009). Another popular approach relies on constructing a graph by discretizing the geometric domain, using vonoroi diagrams (Takahashi and Schilling, 1989) or a simple grid of square elements, and performing a subsequent graph search. These graph searches are typically performed using Dijkstra's algorithm or one of its variants with additional heuristics, such as the popular A* (Hart et al., 1968). Grid based motion planning problems can also be solved by wavefront planners, such as D* (Stentz, 1994) and its variants. The main drawback of graph search methods is that they usually do not consider kinematic constraints of the motion vehicle, which is a problem especially for nonholomic vehicles. Recently, optimization-based strategies for solving motion planning problems, such as Model Predictive Control (MPC), are also becoming more popular.

MPC is a control strategy in which an optimal control problem is solved at every time instant (Rawlings and Mayne, 2009). Of the resulting optimal control sequence, the first one is applied to the plant, and the procedure is repeated. A significant advantage of MPC is its ability to take into account constraints on the inputs and states, such as collision-avoidance constraints.

One of the main challenges of the practical application of MPC is the strict real-time constraint for solving the optimal control problems. High sampling frequencies are typically required for the system to be able respond appropriately to disturbances and changes in the environment. Moreover, solvers often have to run on resource-constrained embedded hardware. The traditional solvers for numerical optimization, Sequential Quadratic Programming (SQP) and Interior Point (IP) methods, are not very suitable for this purpose as they require the costly operation of solving a linear system of equations at every iteration. In contrast, first-order methods do not require this operation and often involve only simple steps. This explains their increasing popularity for solving MPC problems, (Richter et al., 2012; Patrinos and Bemporad, 2014; Jerez et al., 2014).

Furthermore, the optimization algorithm and the problem form, in particular the constraints, are generally linked. For example, most SQP solvers assume that the linear independence constraint qualification (LICQ) is satisfied.

[⋆] This work benefits from KU Leuven-BOF PFV/10/002 Centre of Excellence: Optimization in Engineering (OPTEC), from the project G0C4515N of the Research Foundation-Flanders (FWO-Flanders), from Flanders Make ICON project: Avoidance of collisions and obstacles in narrow lanes, and from the KU Leuven Research project C14/15/067: B-spline based certificates of positivity with applications in engineering, from KU Leuven internal funding: StG/15/043, from Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no 30468160 (SeLMA) and from FWO projects G086318N, G086518N.

Often only simple obstacle shapes, such as circular (Wang and Ding, 2014) and rectangular obstacles are considered. Another approach is based on the separating hyperplane theorem (Boyd and Vandenberghe, 2004), and allows for the separation of a convex motion system and convex obstacles, or between convex motion systems, as illustrated by (Debrouwere et al., 2013) and (Mercy et al., 2017). Recently, Sathya et al. (2018) have proposed a novel constraint formulation to incorporate general obstacle shapes, described as the intersection of a set of nonlinear inequalities, in the optimization problem.

This paper embeds the obstacle constraint formulation presented by Sathya et al. (2018) in a penalty method framework to calculate a trajectory while satisfying collision-avoidance constraints. The penalty parameters allow for a trade-off between the optimality of the trajectory and the extent to which the obstacle constraints may be violated. Virtual enlargements ensure that this trade-off results in a trajectory that avoids all real obstacles. Moreover, the application of the penalty method lowers the likelihood of getting stuck in local optima due to obstacles, as the trajectory is gradually formed around the obstacle. In addition, some heuristics are developed for dealing further with these local optima.

The resulting optimization problems are solved using the proximal averaged Newton-type method for optimal control (PANOC), as proposed in (Stella et al., 2017). As this method combines projected gradient and limited-memory quasi-Newton steps, its implementation is simple and it can achieve a fast rate of convergence. An automatic differentiation toolbox, CasADi (Andersson et al., In Press, 2018), is used to efficiently compute the value of the objective function and its derivative.

The proposed algorithm is validated for a set of obstacle configurations and for different vehicle models. It is shown to be successful in avoiding obstacles of arbitrary shape, as long as they can be described by the intersection of a set of nonlinear inequalities. In addition, our algorithm is benchmarked against state-of-the-art SQP and IP methods, and is found to outperform these methods both in terms of runtime and robustness.

This paper is organized as follows: Section II describes the obstacle constraint formulation introduced by Sathya et al. (2018). In addition, the resulting optimization problem is presented. Section III discusses the methodology for solving this problem, consisting of necessary reformulations, the first-order algorithm PANOC, the quadratic penalty method and several heuristics. Section IV shows and discusses the numerical simulations results. Section V draws the main conclusions of the paper.

2. PROBLEM FORMULATION

An obstacle constraint formulation that can deal with general obstacle shapes was first introduced by Sathya et al. (2018). This constraint formulation is an essential part of the optimization problem considered in this paper. Another element of the problem are the kinematics of the motion system for which trajectories are calculated. Both these elements are analyzed in the subsections below and finally incorporated in a nonlinear model predictive control (NMPC) framework.

2.1 Obstacle constraint formulation

In this work, we consider obstacle shapes that can be defined as the intersection of a set of m nonlinear inequalities:

$$O = \{z \in \mathbb{R}^{n_z} : h_i(z) > 0,\ i = 1,...,m\}. \tag{1}$$

Here, z denotes the position vector, n_z the number of dimensions, and $h_i : \mathbb{R}^{n_z} \to \mathbb{R}$ are continuously differentiable functions with Lipschitz continuous gradients. In the remainder of this paper the considered geometry will be two dimensional, thus $n_z = 2$ and $z = (x, y)$. To take into account moving obstacles, time dependent functions $h_i(z,t)$ can also be used in this formulation. However, this paper will only consider the static case. The obstacle avoidance constraint ($z \notin O$) can then be written as follows:

$$\exists i \in 1,...,m : \qquad h_i(z) \le 0. \tag{2}$$

In other words, at least one of the inequalities defining the obstacle must be violated. This condition can be rewritten as the following equality constraint:

$$\psi(z) := \prod_{i=1}^{m} [h_i(z)]_+ = 0, \tag{3}$$

where the operator $[h_i(z)]_+$ is defined as $\max(h_i(z), 0)$.

The obstacle avoidance constraint (3) is linked to vertical complementary constraints (Scheel and Scholtes, 2000), as it can be rewritten as

$$\min([h_1(z)]_+^2, ..., [h_m(z)]_+^2) = 0. \tag{4}$$

Here, the terms are squared to obtain continuously differentiable functions. In general, the linear independence constraint qualification (LICQ) is not satisfied for such constraints. In our case, for example, the gradient of the rewritten obstacle avoidance constraint is zero at and outside the obstacle boundary.

Numerous obstacles can be described using the above formulation. For example, any polyhedral set can be cast as a set of affine constraints

$$O = \{z \in \mathbb{R}^{n_z} : b_i - a_i^\intercal z > 0,\ i = 1,...,m\}.$$

Also non-convex polytopes, such as a cross shaped obstacle, can often be constructed as the union of a set of intersecting convex polygons. Another type of obstacle shape that can be considered are balls and ellipsoids, given by

$$O = \{z \in \mathbb{R}^{n_z} : 1 - (z-c)^\intercal E(z-c) > 0,\ i = 1,...,m\}.$$

Furthermore, sections of discs can be described as the combination of an outer radius constraint, an inner radius constraint, and a separating hyperplane (affine) constraint. For example, a half-disc obstacle is shown in Figure 2.

In addition, a set of polynomial functions $h_i(z)$ can be used to define a more general semi-algebraic set. Finally, other functions, such as trigonometric functions, are also possible, as long as they are continuously differentiable.

2.2 Vehicle models

The problem under consideration is the real-time computation of the optimal trajectory for a motion system. This motion system can be a robot, a satellite, a car, etc. For the remainder of this paper, this system will be called 'vehicle', as the example models discussed below will be of the vehicle type.

The vehicle is described by a state vector q denoting its position and orientation. In this paper, the considered geometry is two-dimensional. The state therefore has three components: position components x and y, and a heading angle θ. The vehicle is steered by control inputs u, and its system dynamics are governed by nonlinear ordinary differential equations $\dot{q} = f(q,u)$.

Two vehicle models will be used in the numerical validation of the proposed methodology, cf. Section 4. The first vehicle model is the simple bicycle model (Rajamani, 2011), where slip of the wheels is neglected. A bicycle is controlled by two control inputs, the velocity v and the steering angle of the front wheel(s) δ_f. The corresponding equations of motion are

$$\begin{aligned}\dot{x} &= v \cdot \cos(\theta)\\ \dot{y} &= v \cdot \sin(\theta)\\ \dot{\theta} &= \frac{v}{L}\tan(\delta_f).\end{aligned} \tag{5}$$

Here, L is the distance between the centers of mass of the wheels of the bicycle.

The second nonlinear vehicle model considered in this paper is a simplified trailer model (Sathya et al., 2018). Again, slip of the wheels is neglected. This model's inputs are the velocity reference u_x and u_y of the towing vehicle. This velocity reference is tracked by a low-level velocity controller. The equations of motion of the trailer model are

$$\begin{aligned}\dot{x} &= u_x + L\sin(\theta) \cdot \dot{\theta}\\ \dot{y} &= u_y - L\cos(\theta) \cdot \dot{\theta}\\ \dot{\theta} &= \frac{1}{L}(u_y\cos(\theta) - u_x\sin(\theta)).\end{aligned} \tag{6}$$

Here, L is the distance between the center of mass of the trailer vehicle and the fulcrum connecting to the towing vehicle.

2.3 NMPC formulation

The continuous-time dynamics describing the motion of the system are discretized using a nonlinear integrator, in this case a fourth order explicit Runge-Kutta method, resulting in the discrete-time representation:

$$q_{k+1} = \Phi_k(q_k, u_k) \tag{7}$$

The NMPC problem is the following:

$$\begin{aligned}\textbf{minimize} \quad & \ell_N(q_N) + \sum_{k=0}^{N-1} \ell_k(q_k, u_k), && (8)\\ \textbf{subject to} \quad & q_{k+1} = \Phi_k(q_k, u_k),\ k = 0, ..., N-1, && (9)\\ & \psi_i(z_k) = 0,\ i = 1, ..., N_O,\ k = 1, ..., N, && (10)\\ & u_k \in U,\ k = 0, ..., N-1, && (11)\end{aligned}$$

where N denotes the horizon length and N_O the number of obstacles. The obstacle cost functions ψ_i are defined as in (3), and the position z_k is a subvector of the state vector q_k. The stage costs are quadratic functions expressing the distance of the state and input variables to the reference state and input:

$$\ell_k(q_k, u_k) = (q_k - q_{\text{ref}})^\intercal Q_k (q_k - q_{\text{ref}}) + (u_k - u_{\text{ref}})^\intercal R_k (u_k - u_{\text{ref}}),$$

with the terminal cost

$$\ell_N = (q_N - q_{\text{ref}})^\intercal Q_N (q_N - q_{\text{ref}}).$$

The matrices Q_N, Q_k and R_k are positive (semi-)definite matrices. A set of input constraints U on which it is easy to project, can straightforwardly be accounted for by the PANOC algorithm. Typical constraints of this type are box constraints of the form $U = \{u \in \mathbb{R}^{n_u} : u_{\min} \leq u \leq u_{\max}\}$.

3. METHODOLOGY

This section discusses the method employed for solving problem (14), which consists of four parts: (i) a reformulation of the optimization problem itself; (ii), an optimization algorithm for solving the problem for a fixed value of the penalty parameters; (iii), a penalty method algorithm, which allows for an adequate trade-off between the least-squares objective and the obstacle cost; and (iv), heuristics that facilitate convergence to a trajectory that both reaches the destination and avoids all obstacles.

3.1 NMPC reformulation

Two transformations are applied to the optimization problem before we can introduce a first-order algorithm to solve it. First, the equality constraints representing obstacle avoidance in problem (8) are replaced by appropriate penalty functions in the objective, also known as soft constraints. Given the formulation of the obstacle cost function (3), it is straightforward to construct a quadratic penalty function, $\widetilde{\psi}(z_k) = \frac{1}{2}\mu_k \psi(z_k)^2$, with penalty factors μ_k. This obstacle penalty function has the advantage of being continuously differentiable, in contrast to an exact penalty formulation of these constraints. It is also better conditioned than higher order penalties, with gradient:

$$\nabla \widetilde{\psi}(z_k) = \mu_k \sum_{i=1}^{m} [h_i(z_k)]_+ \prod_{j \neq i} [h_j(z_k)]_+^2 \nabla h_i(z_k). \tag{12}$$

Note that

$$\nabla([w]_+^2) = 2[w]_+ \nabla([w]_+) = 2[w]_+ \nabla w.$$

Second, the state vectors are eliminated from the optimization problem by integrating the nonlinear kinematic equations of the motion system

$$F_{k+1}(u) = \Phi_k(F_k(u), u_k), \tag{13}$$

with $F_0(u) = q_0$. This is the so-called single-shooting formulation, where the control inputs are the only remaining decision variables, and the initial state vector is a parameter.

The resulting optimization problem then becomes

$$\underset{u \in U^N}{\textbf{minimize}}\ \ell(u), \tag{14}$$

where the objective function is given by

$$\ell(u) = \ell_N(F_N(u)) + \sum_{k=0}^{N-1} \ell_k(F_k(u), u_k) + \frac{1}{2}\sum_{k=1}^{N}\sum_{i=1}^{N_O} \mu_{ik}\psi_i^2(F_k(u)). \tag{15}$$

and $U_N = \underbrace{U \times \cdots \times U}_{N \text{ times}}$.

3.2 PANOC algorithm

For solving problem (14) with a fixed value for the penalty parameters μ_{ik}, we employ the recently introduced proximal averaged Newton-type method for optimal control (Stella et al., 2017). This algorithm, presented in Alg. 1, achieves a fast convergence by combining proximal gradient and limited memory quasi-Newton (L-BFGS) steps. In this manner, curvature information of the optimization problem is incorporated without calculating second-order derivatives. A set of input constraints U^N can straightforwardly be taken into account via the projection step, step 2, in the iterative scheme.

Algorithm 1 PANOC algorithm for problem (14)

```
    Input: $L_\ell > 0$, $\gamma \in (0, \frac{1}{L_\ell})$, $\sigma \in (0, \frac{\gamma}{2}(1-\gamma\frac{L_\ell}{2}))$, $u^0 \in \mathbb{R}^n$, $\tau > 0$, L-BFGS memory $m$.
1:  for $k = 0, 1, 2, \ldots$ do
2:      $\bar{u}^k \leftarrow \Pi_{U^N}(u^k - \gamma\nabla\ell(u^k))$
3:      $r^k \leftarrow \frac{u^k - \bar{u}^k}{\gamma}$
4:      if $\|r^k\|_\infty < \tau$ then
5:          stop with solution $\bar{u}^k$.
6:      end if
7:      $d^k = -H_k r^k$ using L-BFGS
8:      $u^{k+1} \leftarrow u^k - (1-\alpha_k)\gamma r^k + \alpha_k d^k$, with $\alpha_k$ the largest in $\{\frac{1}{2^i} : i \in \mathbb{N}\}$ such that
            $\varphi_\gamma(u^{k+1}) \le \varphi_\gamma(u^k) - \sigma\|r^k\|^2$    (16)
9:  end for
```

In this algorithm, L_ℓ denotes the Lipschitz constant of the objective function, ℓ. If this is not known a priori, as is often the case in practice, the PANOC algorithm can also run with a Lipschitz estimate which is then updated in between iterations, by adding another step after step 3

```
3bis: if $\ell(\bar{u}^k) > \ell(u^k) - \gamma\nabla\ell(u^k)^\top r^k + \frac{L_\ell}{2}\|\gamma r^k\|^2$ then
      $\gamma \leftarrow \frac{\gamma}{2}$, $L_\ell \leftarrow 2L_\ell$, $\sigma \leftarrow \frac{\sigma}{2}$, go to step 2.
```

In addition, two new functions were introduced. The first is the fixed-point residual operator

$$R_\gamma(u) = \frac{1}{\gamma}(u - \Pi_{U^N}(u - \gamma\nabla\ell(u))) \tag{17}$$

The second is the forward-backward envelope, first introduced by Patrinos and Bemporad (2013), which can be computed as

$$\varphi_\gamma(u) = \ell(u) - \frac{\gamma}{2}\|\nabla\ell(u)\|^2 + \mathrm{dist}^2_U(u - \gamma\nabla\ell(u))). \tag{18}$$

For a more in-depth discussion on the properties of PANOC, the reader is referred to (Stella et al., 2017).

3.3 *Penalty method algorithm*

Given the definition of the obstacle constraint function (3), the obstacles are completely avoided when for every point z of the trajectory and for all obstacles, $\psi_O(z) = 0$ holds. However, solving the optimization problem with the objective as defined in (15), the solution will likely show a trade-off between low stage costs and low obstacle costs. This trade-off depends on the value of the penalty factors. In order to enforce the obstacle constraints to an acceptable predefined tolerance, we employ a penalty method, as shown in Alg. 2. Here, it is made explicit that the objective function is parametrized in the penalty factors, hence the notation $\ell(u,\mu)$. Outer iterations are denoted by subscripts, inner iterations by superscripts, and u and μ represent the vector of control inputs and penalty factors, respectively.

Algorithm 2 Penalty method for problem (14)

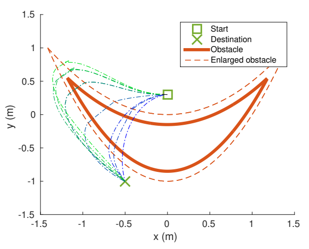
```
    Input: $u_0^0 \in \mathbb{R}^n, \mu_0 \in \mathbb{R}^{N_O N}, \eta_* > 0, \tau_* > 0, \{\tau_k\} \to \tau_*, \omega > 1$
1:  for $k = 0, 1, 2, \ldots$ do
2:      Minimize $\ell(u, \mu_k)$ with starting point $u_k^0$, using PANOC with termination criterion $\|R_\gamma(u)\|_\infty < \tau_k$ to find $u_k^*$.
3:      if $\|R_\gamma(u_k^*)\|_\infty \le \tau_* \wedge \|\psi(u_k^*)\| \le \eta_*$ then
4:          stop with solution $u_k^*$.
5:      end if
6:      $\mu_{k+1} \leftarrow \omega\mu_k$
7:      $u_{k+1}^0 \leftarrow u_k^*$
8:  end for
```

In the penalty method, the penalty factors are raised until the optimization solver has converged to a solution for which the norm of the obstacle cost function is lower than a certain tolerance η_*. The quadratic penalty method is well known to only be exact ($\eta_* = 0$) if the penalty factors are equal to infinity (Nocedal and Wright, 2006). Therefore, a strictly positive tolerance is chosen. In our algorithm, this is often on the order of 10^{-2}. Virtual enlargements of the obstacle complement this formulation, so that the real obstacles can in fact be completely avoided, even though the constraint tolerance is strictly positive. Such enlargements also allow the formulation to be used for a vehicle with a finite width.

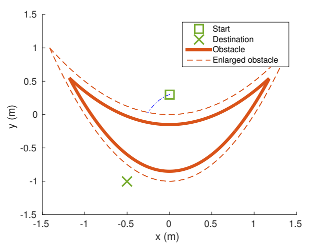
The penalty update factor ω is used to increase the penalty factors at each outer iteration. In practice, appropriate values of this factor lie between $2 - 10$ and there is always a trade-off in choosing this value: low values make the different optimization problems more similar and thus easier to warm-start, but more problems will have to be solved in order to converge to a feasible solution. High values in contrast, render the consecutive optimization problems more difficult, but fewer of them are needed. It is observed that for our motion planning problem, the optimization problems do not suffer from a high penalty update factor, thus ω is here chosen to be 10. In addition, after every update of the penalty factor, the solver is warm-started with the solution from the previous iteration, step 7. After every MPC step, it is common practice to warm-start the next optimal control problem, shifting the vector of control inputs over one time instant and adding an initial guess, often the zero vector, for the last time instant. Similarly, the penalty factors are shifted, and a vector of ones is added for the last time instant.

An illustration of the penalty method applied to a problem with a crescent obstacle is shown in Figure 1a. The trajectories ranging from blue to green correspond to increasing penalty factors, which determine the balance between a feasible trajectory and one that is optimal for the least squares objective. The enlarged obstacle can never be completely avoided, but for high enough penalty factors, the original obstacle is avoided, as illustrated by the final two green trajectories. The combination of a virtual enlargement of the obstacle and finite values for the penalty factors is therefore indeed successful. Figure 1a also demonstrates that successive trajectories are usually similar in shape even though the penalty factors are updated somewhat aggressively in this paper. Therefore, warm-starting aids tremendously in the convergence of the penalty method.

The application of the penalty method may have an additional benefit for obstacle avoidance problems, also illustrated by Figure 1a. Assuming some obstacle is blocking the shortest path



(a) Penalty method.



(b) Fixed (high) penalty factor.

Fig. 1. Illustration of the penalty method. The enlarged obstacle is defined by $O = \{(x,y) : y > x^2, y < 1 + x^2/2\}$.

from start to destination, the initial trajectory calculated with low penalty factors is very likely to arrive at the destination while violating obstacle avoidance constraints. Subsequent iterations with higher and higher penalty factors tend to push the trajectory to the edge of the obstacle, while remaining connected to the destination. In contrast, solving the problem only once with a high value for the penalty factors can impede convergence to a trajectory that reaches the destination, as the vehicle is more likely to get stuck behind an obstacle, as shown in Figure 1b. Using the penalty method for the problems considered in this paper therefore aids in avoiding the local minimum.

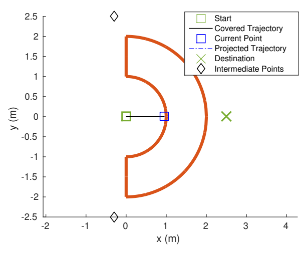
3.4 Heuristics

Optimization problem (14) is a general nonlinear, non-convex problem. The obstacles render the solution space non-convex, and thus often local minima exist near obstacles. An example hereof is illustrated in Figure 2a. To aid in the convergence to a feasible trajectory that reaches the destination, three additional heuristics are applied to the algorithm: (i) the penalty factors are capped at an appropriate value; (ii) the vehicle is stopped if the obstacle costs do not satisfy the specified tolerance η_*; and (iii) whenever the vehicle remains in place for more than one time instant, it is guided to an intermediate destination before continuing on towards the final destination. Below, the rationale for each of these heuristics is explained.

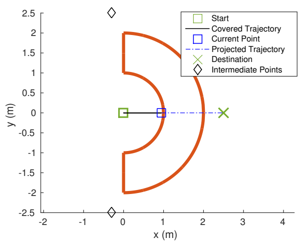
Large penalty factors render the problem ill-conditioned, which impedes fast convergence of the method. In particular, PANOC uses an estimate of the Lipschitz constant of the objective function to determine the stepsize, and the Lipschitz constants of the obstacle cost penalties scale linearly with the penalty factors. Therefore, these factors are capped at a reasonably low value, for example 10^4. With this cap, however, the algorithm is not guaranteed to find a solution for which the obstacle costs are sufficiently low.

In order to solve this problem, the following heuristic is applied: If the obstacle cost is not smaller than the prescribed tolerance within the next three time steps, the vehicle performs an emergency stop. This strategy prevents collisions with obstacles, but causes the vehicle to be more likely to get stuck. Figure 2b illustrates this strategy in case of a half-disc shaped obstacle. Starting from the green square, the vehicle moves to the blue square in seven MPC steps. There, the calculated trajectory threatens to violate the obstacle constraint within the next three time steps, because the corresponding penalty factors are too low. Instead of following this trajectory, the vehicle is stopped.

Finally, to assist the vehicle in circumventing all obstacles, another heuristic is implemented. If the vehicle is stuck behind an obstacle, then the reference state is temporarily replaced by an intermediate destination. The intention behind this is to guide the vehicle around the obstacle. A good intermediate destination is easy to reach from both the point where the vehicle was previously stuck and the final destination. It will usually be close to a corner or edge point of the obstacle. This principle is also illustrated in Figure 2. Appropriate intermediate destinations lie near the black diamonds.



(a) High penalty factors cap.



(b) Low penalty factors cap.

Fig. 2. Illustration of the local minimum behind the obstacle, and the hold-in-place heuristic and choice of intermediate points. The half-disc shaped obstacle is defined by $O = \{(x,y) : x^2 + y^2 > 1, x^2 + y^2 < 4, x > 0\}$.

To avoid getting stuck in a local optimum near an obstacle, a suitable set of intermediate destinations must be available and an appropriate choice from this set of points is necessary. The user may provide such a set, based on knowledge of the obstacle definitions (and therefore the locations of their corners). This approach, however, is hard to justify in an automated setup. In order to generate suitable intermediate points automatically, variants of Dijkstra's algorithm can be used to perform a simple graph search. This paper utilizes the A* search algorithm (Hart et al., 1968), because of its simplicity and efficiency. The worst-case complexity of this algorithm in case a consistent heuristic cost is used, is $O(N)$ (Martelli, 1977), with N the number of nodes in the graph. The heurstic cost used here is the Euclidean distance between a node and the goal node, which is indeed consistent.

Figure 3 shows that the graph search returns a feasible trajectory from the current point to the destination. Intermediate points can be extracted from this trajectory by moving through it and recording points at which the left-right or up-down direction switches. In an unobstructed space, a graph search would find a straight path. Hence, direction changes stem from the presence of an obstacle, and the points at which they occur are close to the corners of the obstacle. These points are therefore suitable candidates for intermediate destinations. In the simulations below, this approach is used and each point of the set of intermediate destinations is successively visited. When this set is exhausted, the reference state is reset to original destination.

4. NUMERICAL SIMULATIONS

The proposed methodology is illustrated and analyzed by means of numerical simulations for a wide variety of obstacle configurations, and two different vehicle models. All simulations were performed on a notebook with Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz x 2 processor and 16 GB of memory.

Figure 4 displays the first set of obstacle configurations, overcome by a vehicle with a trailer. The kinematics of the trailer model, given by (6), are discretized using an explicit fourth order Runge Kutta method. The distance between the center of mass of the trailer and the fulcrum of the towing vehicle

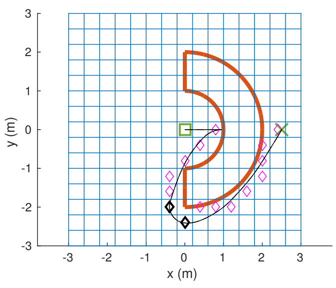
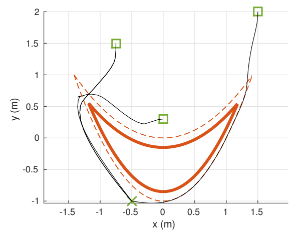


Fig. 3. Illustration of the graph search, represented by the magenta triangles. At both of the black triangles the direction changes (first left-right, then up-down), so these comprise the set of intermediate destinations. The black line denotes the final trajectory.


(a) Enlarged obstacle defined as $\{(x,y) : y > x^2, y < 1 + x^2/2\}$.

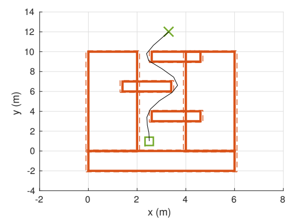
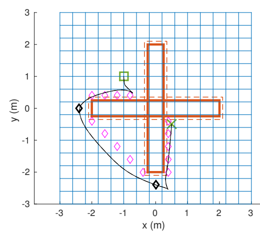

(b) Slightly complex obstacle configuration using rectangular obstacles.

Fig. 4. Two obstacle configurations using the trailer model. The conventions for start, destination and obstacles are the same as those in Figure 1a, and the black lines denote the trajectories that were found in each case.


(a) Cross-shaped obstacle as the combination of two rectangles.

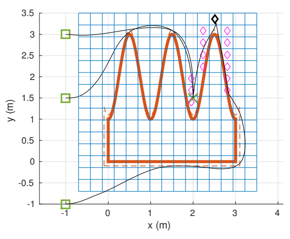
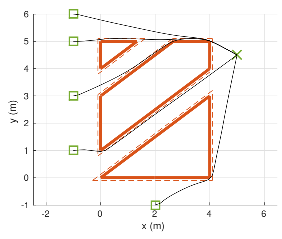

(b) Rack-shaped obstacle, defined by $O = \{(x,y) : y < \sin(2\pi x - \pi/2) + 2, y > 0, 0 < x < 3\}$.

Fig. 5. Illustration of the graph search heuristic for two obstacle configurations using the trailer model.

is $L = 0.5$m. The optimal control problems are solved with sampling time $t_s = 30$ms for Fig. 4a and $t_s = 200$ms for Fig. 4b, and horizon length $N = 50$. The inputs are constrained by box constraints $-4\text{m/s} \leq u_x, u_y \leq 4\text{m/s}$ at every time instant. The penalty factors are capped off at 10^4. It is usually difficult for a first order method, such as PANOC, to find a solution for a strict tolerance, say 10^{-6}. Therefore, the tolerance τ_* is set to 10^{-3}. We observed in simulations that the closed-loop performance is not impacted by this choice, and that a stricter tolerance would be unnecessary. Figure 5 displays two more obstacle configurations for which the graph search heuristic proved necessary to find a trajectory that reached the destination.

Similarly, Figure 6 displays two obstacle configurations overcome by a vehicle modeled as a bicycle. The relevant kinematics are given by (5). The optimal control problem for these simulations is constructed with the same parameters as before, with the following exceptions: the sampling time $t_s = 50$ms, and the input constraints are $-0.1\text{m/s} \leq v \leq 4\text{m/s}$ and $-\frac{\pi}{3} \leq \delta_f \leq \frac{\pi}{3}$ at every time instant. The figures in this section show the versatility of the proposed approach for constructing and solving collision-avoidance problems.


(a) Three polyhedral obstacles that form corridors.

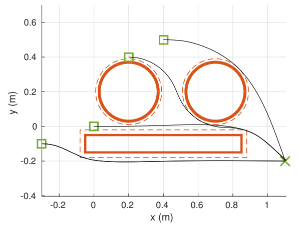

(b) Configuration of one rectangular and two circular obstacles.

Fig. 6. Two obstacle configurations using the bicycle model.

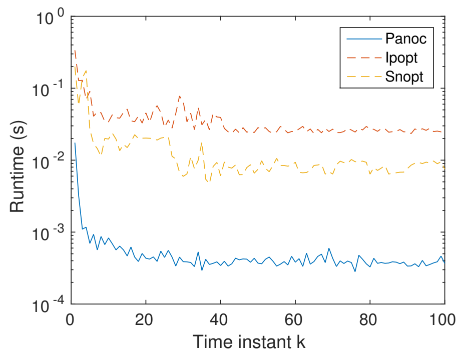


Fig. 7. Runtime comparison of three solvers: PANOC, SNOPT and IPOPT. This comparison is for the problem of Fig. 6b, with initial point $x_0 = (0,0,0)$. All solvers were warm-started with their previous solution after each MPC step. The tolerance for all solvers was set to 10^{-3}.

Figure 7 compares the runtime of the proposed methodology with that of state-of-the-art SQP (Gill et al., 2005, SNOPT) and IP (Wächter and Biegler, 2006, IPOPT) solvers. In these solvers, the obstacle avoidance constraint is incorporated as an inequality constraint, $\psi^2(z) \leq \eta_*^2$. The penalty method algorithm using PANOC clearly outperforms the other solvers, by approximately two orders of magnitude.

Table 1. Comparison of closed-loop costs.

Solver / Example	PANOC	IPOPT	SNOPT
Figure 6b, $x_0 = (0,0,0)$	3.56	4.02	3.54
Figure 4a, $x_0 = (0,0.3,\pi)$	20.78	49.43	47.50
Figure 6a, $x_0 = (-1,1,0)$	17.65	19.62	26.17
Figure 6a, $x_0 = (-1,3,0)$	13.21	12.76	33.48

The proposed approach is not only faster, but also more adept at finding high quality solutions than the other state-of-the-art solvers. This is illustrated in Table 1, which lists the closed-loop costs for different problem scenarios. Sometimes, the solvers all converged to the same trajectory, such as for Fig. 6b. This was, however, not always the case. For example, IPOPT and SNOPT were both temporarily stuck in the local minimum behind the crescent-shaped obstacle, Fig. 4a, whereas our approach found the optimal trajectory in the first optimal control problem. In other cases, such as the corridors example, Fig. 6a, different paths were taken. Table 1 shows that for these cases, the proposed methodology found trajectories with closed-loop costs as good as or better than the other solvers.

5. CONCLUSION

This paper presents a penalty method framework for solving optimal control problems with collision-avoidance constraints that typically arise in motion planning problems. The application of the penalty method, coupled with virtual enlargements, allows for the avoidance of obstacles of complex geometry. It also benefits the convergence to a trajectory that reaches the destination, by gradually finding a path around the obstacles as the penalty factors are successively increased. In addition, several heuristics are employed, which have been observed to improve convergence to a feasible trajectory that reaches the destination. The resulting optimization problems are solved with PANOC, a first-order method which exhibits low runtime. Numerical simulations with nonlinear vehicle dynamics show the versatility of the proposed approach in solving motion planning problems with general obstacle avoidance constraints. In the limited number of cases considered, the proposed algorithm outperforms state-of-the-art SQP and IP solvers in both runtime and robustness.

REFERENCES

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

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Debrouwere, F., Van Loock, W., Pipeleers, G., Diehl, M., De Schutter, J., and Swevers, J. (2013). Time-optimal path following for robots with object collision avoidance using lagrangian duality. In *Robot Motion and Control (RoMoCo), 2013 9th Workshop on*, 186–191. IEEE.

Ge, S.S. and Cui, Y.J. (2002). Dynamic motion planning for mobile robots using potential field method. *Autonomous robots*, 13(3), 207–222.

Gill, P.E., Murray, W., and Saunders, M.A. (2005). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM review*, 47(1), 99–131.

Guy, S.J., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., and Dubey, P. (2009). Clearpath: highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 177–187. ACM.

Hart, P.E., Nilsson, N.J., and 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.

Jerez, J.L., Goulart, P.J., Richter, S., Constantinides, G.A., Kerrigan, E.C., and Morari, M. (2014). Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12), 3238–3251.

Martelli, A. (1977). On the complexity of admissible search algorithms. *Artificial Intelligence*, 8(1), 1–13.

Mercy, T., Van Parys, R., and Pipeleers, G. (2017). Spline-based motion planning for autonomous guided vehicles in a dynamic environment. *IEEE Transactions on Control Systems Technology*.

Montiel, O., Orozco-Rosas, U., and Sepúlveda, R. (2015). Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. *Expert Systems with Applications*, 42(12), 5177–5191.

Nocedal, J. and Wright, S.J. (2006). Numerical optimization 2nd.

Patrinos, P. and Bemporad, A. (2013). Proximal newton methods for convex composite optimization. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, 2358–2363. IEEE.

Patrinos, P. and Bemporad, A. (2014). An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Transactions on Automatic Control*, 59(1), 18–33.

Rajamani, R. (2011). *Vehicle dynamics and control*. Springer Science & Business Media.

Rawlings, J. and Mayne, D. (2009). *Model Predictive Control: Theory and Design*. Nob Hill Pub.

Richter, S., Jones, C.N., and Morari, M. (2012). Computational complexity certification for real-time mpc with input constraints based on the fast gradient method. *IEEE Transactions on Automatic Control*, 57(6), 1391–1403.

Sathya, A.S., Sopasakis, P., Van Parys, R., Themelis, A., Pipeleers, G., and Patrinos, P. (2018). Embedded nonlinear model predictive control for obstacle avoidance using panoc. In *Proceedings of the 2018 European Control Conference*.

Scheel, H. and Scholtes, S. (2000). Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Mathematics of Operations Research*, 25(1), 1–22.

Stella, L., Themelis, A., Sopasakis, P., and Patrinos, P. (2017). A simple and efficient algorithm for nonlinear model predictive control. In *56th IEEE Conference on Decision and Control*, 1939–1944.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *ICRA*, volume 94, 3310–3317.

Takahashi, O. and Schilling, R.J. (1989). Motion planning in a plane using generalized voronoi diagrams. *IEEE Transactions on robotics and automation*, 5(2), 143–150.

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.

Wang, P. and Ding, B. (2014). A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance. *International Journal of Control*, 87(1), 52–63.