Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-8 (2019) 368–374

Free-space Polygon Creation based on Occupancy Grid Maps for Trajectory Optimization Methods ⋆

Christian Meerpohl * **Matthias Rick** * **Christof Büskens** *

* *Center for Industrial Mathematics, University of Bremen, Germany (e-mail: {cmeerpohl, mrick, bueskens}@math.uni-bremen.de)*

Abstract: In this paper, we present a strategy for incorporating occupancy grid maps into local trajectory optimization methods to allow for safe autonomous navigation. We construct polygons as surface representations of the locally visible area and merge them with the aid of third-party software to form connected free-spaces. The method can be applied to any two-dimensional environments and different representations of vehicle dimensions. It allows direct optimization methods to be used consistently for different problem levels from control to planning. The strategies developed are applied to a rover system that moves in the plane. We evaluate our algorithms on the basis of simulation results and real-world experiments.

Keywords: optimal control, automatic control, autonomous mobile robots, predictive control, optimal trajectory, robot control, robot navigation, polygons

1. INTRODUCTION

Technological and economic growth in the 21st century will most likely be essentially influenced by artificial intelligence. This progress is accompanied by highly automated systems like robot manipulators, self-driving cars or exploring rover systems. The scientific fields behind these innovations are highly interdisciplinary and require powerful but also flexible technologies. Nonlinear optimization is said to be such a key technology. As a subcategory of this, optimal control techniques can, among other things, be utilized to plan trajectories - as has been especially the case in space and aeronautic applications - or to design potent control strategies like LQR, LMPC or NMPC. The success of an optimization approach, however, depends to a large extent on the right formulation. For one thing, it can be especially challenging to generate solutions if the problem statement involves a variety of stationary points.

One characteristic that can cause this in particular is the presence of obstacles, as pointed out in Ziegler and Stiller (2010). Precisely for this reason and because it is difficult to define meaningful gradients for every state, the complexity of environmental restrictions is often deliberately kept small for both, trajectory optimization and model predictive control. Furthermore, a-priori knowledge is essential for a robust iteration process. If the environment is known in advance, one can easily define various algebraic representations for static obstacles, such as circular discs, ellipses, rectangles or polygons. Alternatively, one can define suitable potential or navigation functions (see Rimon and Koditschek, 1992), but this can be very challenging in terms of universality.

If, however, knowledge about the environment is only probabilistic - as in mobile robotics - this constitutes a severe restriction. Classifiable objects like passing cars can be estimated by the aforementioned primitives (see Weiskircher et al., 2017; Rosolia et al., 2017) but if even this information is not available, a universal modeling technique is desirable. For autonomous driving, Ziegler et al. (2014) presented an approach of how static and dynamic obstacles can be modeled as a series of convex polygons. In particular, the authors ensure that the optimization strategy used can converge to only one global optimum. This, however, takes the knowledge of the routing for granted. If one also wants to remove this restriction, a generalization in the form of free-spaces comes in handy. Moutarlier and Chatila (1991); González-Baños and Latombe (2002) proposed ideas of incrementally modeling the environment by a series of polygons or polylines. The methods rely directly on sensory information, i.e., they represent a partial alternative to the SLAM approach and are not directly comparable with the prerequisites here. Nevertheless, the general idea is promising to take up again in the context of collision detection.

1.1 Contribution

We present a procedure for the construction and integration of polygonal free-spaces into direct optimal control methods. By doing so, we are able to perform both, trajectory optimization and model predictive control of mobile robots in the context of autonomous navigation. In a first step, we use ray-casting techniques to transform occupancy grid maps to polygonal representations of the free-space. By merging them in a second step, we can specify a form that is in many ways usable for collision checks,

⋆ This work was supported by the German Aerospace Center (DLR) with financial means of the German Federal Ministry for Economic Affairs and Energy (BMWi), project "EnEx-CAUSE" (grant No. 50 NA 1505). C. Meerpohl acknowledges support by the German Research Foundation (DFG) - project No. 281474342/GRK2224/1.

10.1016/j.ifacol.2019.08.107

such as a nonlinear constraint within local optimization problems. We evaluate our algorithms by means of two scenarios. First, we demonstrate in a simulation that the concept of global free-space construction allows us to plan and follow trajectories in complex environments. Second, we verify in a real experiment that the proposed strategy also allows usage within a reactive collision avoidance system.

2. OPTIMIZATION & OPTIMAL CONTROL

In the field of motion planning, there are a variety of strategies for leading a robotic system from one state to another while taking into account the dynamics of the system. Due to its applicability to high-dimensional problems, its direct relation to control variables, and, above all, its optimality character, which applies to many different criteria, optimal control is one strong candidate here.
Various tasks can be reduced to the formulation of such an *optimal control problem* (OCP) that is defined in its general form for a time interval $I = [T_a, T_b]$ as follows:

$$\begin{aligned} \min_{x,u,T_b} \quad & J(x,u) = g(x(T_a), x(T_b)) \\ & \qquad + \int_{T_a}^{T_b} f_0(x(t), u(t))\,\mathrm{d}t \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t), t) \;\forall\, t \in I, \\ & \psi\left(x(T_a), x(T_b)\right) = 0, \\ & C\left(x(t), u(t)\right) \leq 0 \;\forall\, t \in I. \end{aligned} \tag{OCP}$$

Here, J is the objective functional, which represents different objective criteria. x is the state vector, which can be changed by (nonlinear) dynamics f via the controls u. ψ describes the start and end conditions and C are further path constraints.

There are two kinds of strategies to solve these problems (see von Stryk and Bulirsch, 1992). The indirect approach uses Pontryagin's maximum principle to state the necessary optimality conditions, which leads to a boundary value problem that can be solved by collocation or multiple shooting methods, for example. More popular for applications is the direct approach, in which the problem at first is being discretized to a potentially high-dimensional nonlinear program (NLP) that can be solved by sequential quadratic programming (SQP) or interior point algorithms in a second step.

We use the software packages TransWORHP (Knauer and Büskens, 2012) and WORHP (Büskens and Wassel, 2012; Kuhlmann and Büskens, 2018) that are most suitable for high-dimensional problems with sparse matrix structures. To transcribe the OCP, TransWORHP offers several strategies, of which we use the full discretization method with the trapezoidal integration scheme. The discretized problem is then solved by the WORHP-SQP routine with an interior-point method on the QP level.

3. COLLISION AVOIDANCE

For a robotic system that can autonomously explore its environment, it is crucial to ensure that the system will not collide with an object at any time. To check whether a collision will occur, not only the dimensions of present obstacles have to be accounted for, but also the dimensions of the robotic system. The concept of free-spaces as in LaValle (2006) is an elegant way to link the two together. In order to use this concept, at first we have to introduce the *configuration space* $\mathcal{C}$, which is the space of all configurations q the system can adopt. For a rover system that only moves in two dimensions, this classically is the configuration manifold $\tilde{\mathcal{C}} = \mathbb{R}^2 \times \mathbb{S}^1$. The space within the world $\mathcal{W} = \mathbb{R}^2$, which is being occupied by the system with configuration q is denoted by $\mathcal{R}(q) \subset \mathcal{W}$. We assume the rover to have the dimensions of a ball B_r with radius r, so that we can ignore the orientation below, i.e. we define $\mathcal{C} := \mathbb{R}^2$. If there are any obstacles $\mathcal{O} \subset \mathcal{W}$, one can define the *obstacle region* by

$$\mathcal{C}_{\text{obs}} = \{q \in \mathcal{C} | \mathcal{B}_r(q) \cap \mathcal{O} \neq \emptyset\}, \tag{1}$$

and consequently the *free-space* by $\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \mathcal{C}_{\text{obs}}$.

In the following, we will deduce a characterization of such a free-space. Map-based representations are closely associated with so-called occupancy grids (Elfes, 1989) that are a quasi-standard in mobile robotics. Here, a map of the environment m is being discretized to a certain number of cells m_i, $i = 1, \ldots, N$, so that each cell represents the probability of an obstacle being present $P : m_i \to [0, 1]$. To extract the free-space, one simply has to apply a convolution operation (see Kavraki, 1995). Map-based representations are a good match for many path and motion planning strategies (see section 4) but are hard to reconcile with continuous optimization techniques. Another widespread approach is to use feature-based representations such as polygons or polyhedrons. These allow efficient and precise point-wise calculations but can be difficult to determine when the environment is not known in advance, as pointed out in Ziegler and Stiller (2010). To combine the benefits of both, we develop a strategy that allows us to efficiently generate polygonal free-spaces on the basis of the latest occupancy grid map.

3.1 Free-Space Polygons

The online construction of free-space polygons is no new finding per se. In general, there are two ways to use polygons in the context of collision checking. One way is to characterize the boundaries of the free-space by a closed polyline. This can be achieved by applying ray-casting techniques. The complementary approach is to define occupied areas as a set of (often convex) polygons (see Ziegler et al., 2014). In this section, we take up the first approach, but with a few adjustments. Since we strive for an efficient gradient-based implementation, the free-space polygons should be sparse and "smooth" (i.e. without sharp edges) on the one hand, while on the other hand false classifications in critical areas should be avoided. Therefore, we perform the collision-check itself not with points on a ray but on a half-circle that moves along the ray. This allows us to construct polygons with fewer line segments - but without cutting off valuable information at the same time. The corresponding algorithm is depicted in Alg. 1.

The rasterization of lines and circles can be computed by using Bresenham's line and circle drawing algorithms (Bresenham, 1965, 1977). To prevent the algorithm from missing some cells when the half-circle is being moved, we

Algorithm 1: Free-space polygon construction: For a probability function P_m that assigns to a given map position the probability that the corresponding map cell be occupied, a free-space polygon $\mathcal{P}$ can be calculated. The structure of the polygon additionally depends on the selected origin $(x, y, \theta)_o$.

Parameters: $r_{\mathcal{C}}$, $r_{\mathcal{HC}}$, n_{pts}, τ_{obs}
Input : $p_o := (x, y)_o, \theta_o, P_m$
Output : $\mathcal{P}$

forall $k = 1, \ldots, n_{\text{pts}}$ **do**
  $\theta_k \longleftarrow \theta_o + \pi - \frac{2\pi k}{n_{\text{pts}}+1}$;
  $\mathcal{L}_k \longleftarrow$ `line`$(\theta_k, p_o, r_{\mathcal{C}})$;
  $\mathcal{HC}_k \longleftarrow$ `halfcircle`$(\theta_k, p_o, r_{\mathcal{HC}})$;
  forall map positions p on $\mathcal{L}_k$ **do**
    forall map positions q on $\mathcal{HC}_k$ **do**
      if $P_m(q) > \tau_{\text{obs}}$ **then**
        $\mathcal{P}$.append(q); `// collision cell`
        go to 1;
    $\mathcal{HC}_k \longleftarrow$ `halfcircle`$(\cdot, p, \cdot)$; `// shift`
  $\mathcal{P}$.append(p^{last}); `// last point on line`

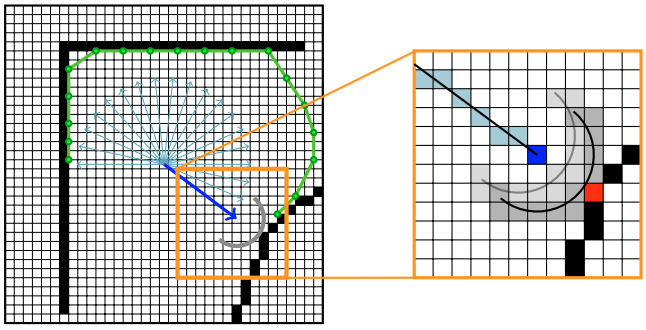

Fig. 1. Free-space polygon creation. *Left:* Occupancy grid map with partially constructed polygon (green). An additional point is to be added at an angle of $\theta = -36°$. *Right:* A rasterized half-circle is being moved (light gray to gray) along the sampled line (blue) until there is a point of overlap (red) with the occupied cells (black).

have to add some elements to the list of half-circle points, so that each point does not only have neighbors on the diagonal (cf. Fig. 1).

3.2 Free-Space Unification

Since the free-space polygon we construct depends only on what can be recognized from one specific point of view, the accessible area is restricted to a rather local description. To explore a larger environment, we have to extend Alg. 1 to a globally applicable procedure. This can be achieved by first creating a set of polygons, each depending on a different ray-casting origin p_i, and merging them in a second step. In contrast to a time-incremental approach as in the works of Moutarlier and Chatila; González-Baños and Latombe, we always take into account only the current position and map information provided by a sensor fusion system: By virtually placing ray-casting origins on the map - for example along a path - we are able to cover the area of interest as a whole. For this, it is necessary that the

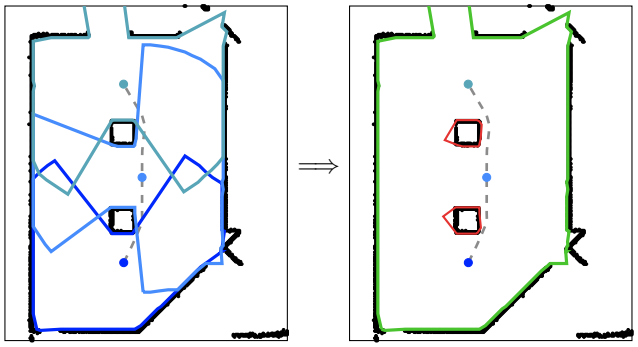

Fig. 2. Polygon unification. Three primal polygons (blue) are merged to form a combined free-space that includes one exterior polygon (green) and two interior polygons (red).

respective origins lie close enough to each other so that the primal polygons overlap, that is

$$\mathcal{P}_i \cap \mathcal{P}_{i+1} \neq \emptyset \ \forall i = 1, \ldots, n_{\mathcal{P}} - 1. \tag{2}$$

Mathematically speaking, we aim for the union of polygonal sets. It is mandatory to carry out this operation, since the closed-distance calculation to polylines (to come below) otherwise does not correspond to the distance to the nearest obstacle.[1] Several software libraries can perform the unification-operation. For instance there is *Clipper* (see Johnson, 2014). As an outcome, one however cannot expect to obtain a single free-space polygon only. Individual areas don't have to be connected and there can be polygons within other polygons. In fact, when using *Clipper*, we obtain a list of exterior and interior polygons $\mathcal{E}$ and $\mathcal{I}$ (for an example, see Fig. 2), for which the following properties apply:

$$\mathcal{E}_i \cap \mathcal{E}_j = \emptyset \ \forall i \neq j, \ i, j \in \mathrm{I}_{\mathcal{E}} = \{1, \ldots, n_{\mathcal{E}}\}, \tag{3}$$
$$\mathcal{I}_k \cap \mathcal{I}_l = \emptyset \ \forall k \neq l, \ k, l \in \mathrm{I}_{\mathcal{I}} = \{1, \ldots, n_{\mathcal{I}}\}, \tag{4}$$
$$\forall k \in \mathrm{I}_{\mathcal{I}} \ \exists! \, j \in \mathrm{I}_{\mathcal{E}} : \mathcal{I}_k \subsetneq \mathcal{E}_j^{\circ}, \tag{5}$$

where $(\cdot)^{\circ}$ denotes the interior of a set. According to (5), each interior polygon can be assigned to exactly one exterior polygon. This mapping can be realized easily because the polylines do not intersect; so one only has to perform a point-in-polygon test for a single vertex once. We define the indices that are assigned to a specific exterior polygon $\mathcal{E}_j$ by

$$\mathrm{I}_{\mathcal{I}}^{j} := \{k \in \mathrm{I}_{\mathcal{I}} : \mathcal{I}_k \subsetneq \mathcal{E}_j^{\circ}\}.$$

With those assignments, the obstacle region is determined by the list of exterior and interior polygons as follows:

$$\mathcal{O} = \mathcal{W} \setminus \Big(\bigcup_{j \in \mathrm{I}_{\mathcal{E}}} \big(\mathcal{E}_j \setminus \bigcup_{k \in \mathrm{I}_{\mathcal{I}}^{j}} \mathcal{I}_k\big)\Big) \tag{6}$$

If we revisit the circular dimensions of the robot, then, in accordance with the concept of LaValle, we can finally define the free-space as

$$\mathcal{C}_{\text{free}} = \mathcal{W} \setminus \big(\mathcal{O} \ominus \mathcal{B}_r(0)\big), \tag{7}$$

with $\ominus$ being the Minkowski difference of the two sets.

3.3 Distance Function Calculation

With the free-space being described by polygons, it is relatively easy to perform collision checks, as the only two

[1] One can confirm this for instance when the position to test against is located inside the intersecting area of two polygons.

Algorithm 2: Signed distance function calculation

Function $\texttt{dist}(q, \mathcal{O})$ **is**
  $\texttt{dist}_{\mathcal{E}} \longleftarrow -\infty$;
  forall $i \in \mathrm{I}_{\mathcal{E}}$ **do**
    $\texttt{dist}_{\mathcal{E}_i} \longleftarrow \texttt{dist}(q, \mathcal{E}_i)$;
    if $q \in \mathcal{E}_i$ **then**
      $\texttt{dist}_{\mathcal{I}^i} \longleftarrow \infty$;
      forall $k \in \mathrm{I}^i_{\mathcal{I}}$ **do**
        $\texttt{dist}_{\mathcal{I}_k} \longleftarrow \texttt{dist}(q, \mathcal{I}_k)$;
        if $q \in \mathcal{I}_k$ **then**
          return $-\texttt{dist}_{\mathcal{I}_k}$;
        else
          $\texttt{dist}_{\mathcal{I}^i} \longleftarrow \min(\texttt{dist}_{\mathcal{I}^i}, \texttt{dist}_{\mathcal{I}_k})$;
      return $\min(\texttt{dist}_{\mathcal{I}^i}, \texttt{dist}_{\mathcal{E}_i})$;
    $\texttt{dist}_{\mathcal{E}} \longleftarrow \max(\texttt{dist}_{\mathcal{E}}, -\texttt{dist}_{\mathcal{E}_i})$;
  return $\texttt{dist}_{\mathcal{E}}$;

necessary tools are to compute distances to polylines and to perform point-in-polygon tests. By combining these two primitives to a signed distance function as being depicted in Alg. 2, the test is complete: A configuration q is defined to be admissible if the signed distance function is greater than a minimum distance, i.e., $q \in \mathcal{C}_{\text{free}}$, if and only if

$$\texttt{dist}(q, \mathcal{O}) \geq r. \tag{8}$$

Here, the inflate radius is composed of the system size plus an additional safety clearance: $r = r_0 + r_{\text{s}}$.

Furthermore, one might want to consider the collision check not only as a binary state but rather as an optimizable decision-process. For this, we introduce a cost function $g : \mathcal{C} \to \mathbb{R}$ such that

$$g(q) = \max\{0, r_0 + r_p - \texttt{dist}(q, \mathcal{O})\}, \tag{9}$$

where $r_p \geq r_s$ is a penalty distance. The whole term is unequal to zero if the distance function evaluation falls below the dimensions of the system plus the additional penalty distance. Note that the function of Alg. 2 - and therefore also (9) - is only a piecewise $\mathcal{C}^1$-function. If, for instance, the rover drives through a narrow passage, this can interfere with the iteration process of the optimization routine.

4. TRAJECTORY PLANNING

As already mentioned in section 2, there are strong arguments for using optimal control strategies as a tool for trajectory planning. However, it should not be concealed that probabilistic roadmaps (Kavraki et al., 1996) or rapidly-exploring random trees (RRT) (LaValle, 1998) and its derivatives perform particularly well if environmental constraints are complex - such as for maze-like problems. Here, nonlinear optimization methods highly depend on a good initial guess (see Tilove, 1990; Shiller, 2015), which can be very challenging to provide the solver with.

Initial Guess Since it is important to us to plan optimal trajectories and do so within a vast environment, we depend on additional support to generate an admissible initial guess. For this, we first use a two-dimensional grid search (Lu et al., 2018) that accounts only for obstacles but not for the dynamics or optimality criteria that are different from path length. After obtaining a path $(x, y)_{i=1,\dots,N}$, we can extract the orientation at some time by applying a difference quotient scheme $\theta_i = \text{atan2}(\frac{y_{i+1}-y_i}{x_{i+1}-x_i})$. Due to the necessity of continuity for all state variables within gradient-based approaches, we have to resolve the discontinuity at $-\pi$ resp. π; hence the domain of θ is extended to $(-\inf, \inf)$. Finally, points along the path are selected to be origins for creating the free-space.

System Model To describe the dynamic behavior of our mobile robot, we use a simple kinematic differential drive model and do not take any external forces into account. The position, orientation, velocity, and yaw rate of the rover are represented by the state vector $\boldsymbol{x} := (x, y, \theta, v, \omega)$. These state variables can be effected by the control vector $\boldsymbol{u} := (a, \sigma)$, which includes the translational and rotational acceleration. The relation between states and controls is given by a first-order differential equation system:

$$\dot{\boldsymbol{x}}(t) = (v\cos(\theta), v\sin(\theta), \omega, a, \sigma)^T \tag{10}$$

In reality, we do not actually apply a and σ as control variables. The interface via ROS (Robot Operating Software) demands us to command values for v and ω, so that an on-board high-frequency PID controller will perform the actual force-related control.

Cost Functions As in many applications, we aim for a mixture of time-optimality $J_0^I := T_b - T_a$ and energy-optimality $J_1^I := \int_{T_a}^{T_b} a(t)^2 + \beta\sigma(t)^2 \mathrm{d}t$. Additionally, we want to minimize the risk of collisions. To achieve that, we pick up (9) as a third cost term:

$$J_2^I := \int_{T_a}^{T_b} g(q)^2 \mathrm{d}t \tag{11}$$

In the context of autonomous exploration, other cost functions might be worthwhile, such as the maximization of knowledge about the environment or the minimization of conflicting information (see Clemens et al., 2016; Miller and Murphey, 2013). As for this paper, we leave these aspects aside or, more specifically, this is being taken care of by another higher-level planning module in form of selecting admissible destinations.

Problem Formulation With the model and the cost functions defined above and some additional box-constraints such as limitations to speed and acceleration,

$$\begin{aligned} \boldsymbol{x}_l \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_u, \\ \boldsymbol{u}_l \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_u, \end{aligned} \tag{12}$$

we can formulate our nominal trajectory planning problem (TPP) on a time interval $I_T = [t_0, t_f]$ with free end time as follows:

$$\begin{aligned} \min_{\boldsymbol{x}, \boldsymbol{u}, t_f} \quad & \alpha_0 J_0^{I_T} + \alpha_1 J_1^{I_T} + \alpha_2 J_2^{I_T} \\ \text{s.t.} \quad & \text{eq.(8)(10)(12) hold } \forall\, t \in I_T, \\ & \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \\ & \boldsymbol{x}(t_f) = \boldsymbol{x}_f. \end{aligned} \tag{TPP}$$

5. NONLINEAR MODEL PREDICTIVE CONTROL

In order to compensate for deviations during trajectory execution, we use a nonlinear model predictive control (NMPC) technique. This is a powerful and flexible approach since it allows to take nonlinear dynamics as well

as additional path constraints into account. On the contrary, the computational effort is comparatively high and statements on stability and runtime are hard to give. In many similar applications, attempts are therefore made to decrease the level of complexity, e.g. by convexification (Wang et al., 2018) or successive linearization techniques (Falcone et al., 2007). NMPC is also used without these restrictions (see Rosolia et al., 2017), although a stable convergence behavior can often only be achieved for trajectory tracking problems as in Faulwasser et al. (2017).

5.1 NMPC Formulation

The general idea of MPC is to repetitively set up and solve optimal control problems while applying few control signals (mostly only the first) to the system at a time. The formulation can be made very similar to (TPP), except for a few adaptations: Given a reference trajectory $\boldsymbol{x}_{\text{ref}}$, we introduce a fourth cost term that represents the tracking behavior:

$$J_3^I := \int_{T_a}^{T_b} \langle \varphi(t), \boldsymbol{x}(t) - \boldsymbol{x}_{\text{ref}}(t) \rangle \, \mathrm{d}t, \tag{13}$$

with $\varphi : I \to \mathbb{R}_+^5$ being a weighting function that is monotonically increasing componentwise. To reach real-time capability, the process time is abbreviated to a common prediction and control horizon T_p. Here, problems are solved at a fixed frequency $f_\text{s} = 1/T_\text{s}$, with T_s being the step size between two successive calculations. We assume piecewise linear control functions and apply interpolated signals at a frequency f_ctrl. With a definition of the respective time interval as $I_k = [t_k, t_k + T_\text{p}]$, the complete NMPC problem can be formulated as

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{u},T_p} \quad & \alpha_0 J_0^{I_k} + \alpha_1 J_1^{I_k} + \alpha_2 J_2^{I_k} + \alpha_3 J_3^{I_k} \\ \text{s.t.} \quad & \text{eq.(8)(10)(12) hold } \forall\, t \in I_k \\ & \boldsymbol{x}(t_k) = \tilde{\boldsymbol{x}}_k. \end{aligned} \tag{NMPC}$$

It is being set up at t_{k-1} and has to be solved before t_k. For this, the initial state $\tilde{\boldsymbol{x}}_k$ has to be estimated, such as by applying a forward-integration scheme. We use the latest pose and velocity estimation available at t_{k-1} to integrate only the pose forward. By analogy, the free-space is computed using a local obstacle grid that could be provided at t_{k-1}. Note that the further the rover moves, the more likely it is for this free-space to deviate from the one calculated at planning level, especially if a loop-closure for the SLAM problem has taken place.

5.2 Performance Enhancements

There are various ways to speed up the iteration process of a NMPC scheme and to make it more predictable (cf. Grüne and Pannek, 2017, chap. 12.4f). We pick up some ideas that can be implemented quickly and lead to significant improvements:

Warm-Start The NLP solver WORHP offers the possibility to perform a warm-start. We can make use of this feature here, since the structure of the NLP does not change from one NMPC step to the next one. For one thing, this increases the performance level, as matrix structures have to be determined only once for the very first step. In addition, already allocated memory can be reused and does not have to be reallocated. To further accelerate the iteration process, a good initial guess is required, which can be determined by interpolating the last calculated trajectory and reusing the Lagrangian multipliers. Besides the speed, the warm-start feature especially increases robustness, making convergence to different local optima less likely.

Algorithm 3: NMPC relaxation strategy: If certain timeouts $\tau_\text{opt} < \tau_\text{feas} < \tau_\text{crit} = T_\text{s}$ are reached, the optimization routine either reduces the optimality tolerance ε_opt, goes into a feasible-only mode, or aborts.

```
Init : ε_opt, ε_comp, ε_feas ⟵ 1E-6
if τ ≥ τ_opt then
  | ε_opt ⟵ 1E-4;            /* relax optimality */
if τ ≥ τ_feas then
  | ε_opt, ε_comp ⟵ 1E20;     /* feasible-only */
if τ ≥ τ_crit then
  | abort;                    /* abortion */
```

Relaxation Even with the enhancements presented above, the iteration process still might struggle in specific situations, for example if an obstacle is being newly detected and has to be avoided. To overcome this, we can gradually relax the problem as being depicted in Alg. 3. This could cause us to lose optimality, but mostly only temporarily. We can, however, ensure that even a non-optimal solution still fits the purpose of navigating the system safely since we do not relax the feasibility conditions at all.

6. RESULTS

In this section, we want to show that the proposed free-space concept can be applied for trajectory planning and control under real-time requirements. As a test vehicle we use the Pioneer 3-AT mobile robot. The methods presented are implemented within a ROS-framework. Parallel to our algorithms, further modules for sensor fusion, decision making, and fault detection are running.

The parameters for the controller are selected as follows: It runs at a frequency of $f_\text{s} = 10\,\text{Hz}$ with a prediction horizon of $T_\text{p} = 5\,\text{s}$. The problems to be solved consist of $n_\text{dis} = 11$ discretization points and controls are applied at $f_\text{ctrl} = 20\,\text{Hz}$. The relaxation timeouts are $\tau_\text{opt} = 0.05\,\text{s}$ and $\tau_\text{feas} = 0.075\,\text{s}$. For the polygon creation we select as parameters $r_\mathcal{C} = 5\,\text{m}$, $r_{\mathcal{HC}} = 0.2\,\text{m}$, $n_\text{pts} = 90$, and $\tau_\text{obs} = 0.2$. The occupancy grid map has a resolution of $0.05\,\text{m}$ per cell and is updated at $10\,\text{Hz}$.

6.1 Simulation: Willow Garage Office

At first, we evaluate our algorithms by means of the simulation environment Gazebo. We run our tests on a system with an Intel i7-4790 CPU and 32 Gbytes of RAM. The used map *willow garage office* has many rooms and several narrow corridors and is therefore best suited for verifying the effectiveness of our algorithms. The underlying dynamical model is very similar to ours, with the exception that Gaussian noise is added.

In this scenario, we navigate through the map by selecting a series of target poses that the robot should reach. The environment has been explored before. Therefore, we can

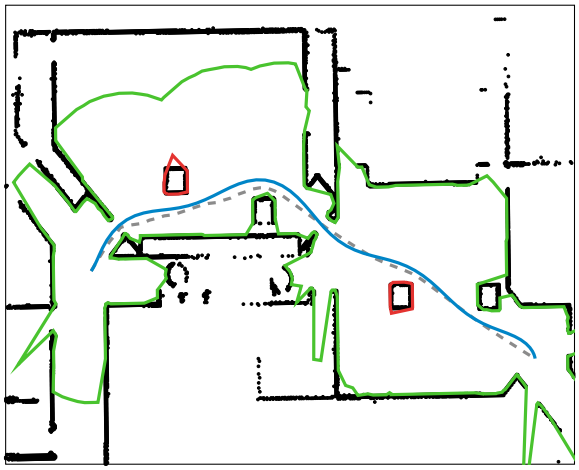

Fig. 3. Scenario 1. The robot has to pass through different rooms with blocking objects and narrow doorways to reach its destination. The initial path (gray) is optimized to a trajectory (blue) that smoothly avoids obstacles.

Table 1. NMPC optimization performance results of the simulation test run.

Optimization Status	Share		Iterations		Computation time [s]	
	N	%	mean	σ	mean	σ
optimal	404	99.0	5.51	1.84	0.007	0.003
suboptimal	4	1.0	65.50	11.12	0.055	0.006
feasible-only	0	0.0	0.0	0.0	0.0	0.0
infeasible	0	0.0	0.0	0.0	0.0	0.0
all	408	100.0	6.10	6.27	0.008	0.005

assure that the pose and map estimation provided by the sensor fusion module is relatively stable - i.e. there are no major discontinuities with respect to the time - and that the paths planned are always feasible. An exemplary trajectory that leads through different rooms is visualized in Fig. 3. In this test, the creation and merging of free-spaces only takes a few milliseconds (with non-optimized code). The performance results of the NMPC optimization process when following the exemplary trajectory are depicted in Tab. 1. In almost every case, the optimization routine terminates with an optimal solution output. The computation time is far below the timeout τ_{crit}, then. On the other hand, if the solver struggles to find solutions - which is most likely to happen, when the rover encounters doorways or obstacles in its way - the computational effort raises quickly. Nevertheless, the iteration process never has to be interrupted.

6.2 Real-World Demonstration: Evasive Maneuver

The second test is performed as a real-world experiment. We want to demonstrate that our formulation of non-collisions with obstacles as a nonlinear constraint is flexible enough, so that the system can react to newly detected obstacles, especially to those making it impossible to keep to the prior plan. For that, we compute a trajectory trough a previously explored area and then put an obstacle in its way. The mounted laser scanner, which is supposed to detect this obstacle, has a field of view of 180 degrees.

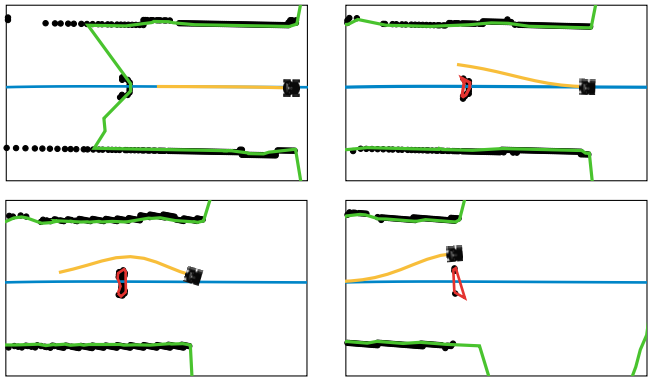

Fig. 4. Scenario 2. The initially planned trajectory (blue) is blocked by an obstacle. The controller generates trajectories one after the other (yellow) so that the rover is guided around it.

As can be seen in Fig. 4, the controller is capable of avoiding the obstacle and it leads the vehicle back to the nominal trajectory. Moreover, the computational effort raises significantly only once. When the initial guess is infeasible with respect to the collision check for the first time - that is the case when the end of the trajectory touches the interior polygon of the free-space - the computation time exceeds the optimality timeout and raises to $\tau_{\text{comp}} = 53\,\text{ms}$. After and before that, the computation times are comparable to those shown in Tab. 1.

7. CONCLUSION & FUTURE WORK

In this paper, we addressed the problem of how two-dimensional occupancy grid maps can be integrated into optimal control methods as path constraints. We took up the concept of free-spaces and presented a strategy to construct polygonal representations of densely structured environments in real-time, making collision-free navigation possible. Our methodology results in two qualities:
On the one hand, by merging free-spaces we demonstrated that trajectory planning through nonlinear optimization techniques can be applied to more maze-like problems. This opens the opportunity to enrich the field of mobile robotics with more sophisticated optimization tools, such as multi-objective optimal control to generate plans with respect to different criteria or Fisher information trajectory synthesis in order to identify system parameters accurately (see Wilson et al., 2014).
On the other hand, we showed that our free-space formulation fits very well with a nonlinear model predictive control concept. The computation time is low enough, so that obstacles can be avoided safely, even if the perception is sudden and contradictory to the previous planning.
In combination, one can take advantage of the strong link between trajectory optimization and model predictive control, for instance to develop control concepts that can adapt objective weights to requests at planning level.
Furthermore, the polygonal free-space representation is easily extensible to more complex vehicle dimensions. For instance, collision checks can also be carried out for oriented bounding boxes or hierarchical testing procedures.

REFERENCES

Bresenham, J.E. (1965). Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1), 25–30.

Bresenham, J.E. (1977). A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM*, 20(2), 100–106.

Büskens, C. and Wassel, D. (2012). The ESA NLP Solver Worhp. In *Modeling and Optimization in Space Engineering*, 85–110. Springer.

Clemens, J., Reineking, T., and Kluth, T. (2016). An evidential approach to SLAM, path planning, and active exploration. *International Journal of Approximate Reasoning*, 73, 1–26. doi:10.1016/j.ijar.2016.02.003.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6), 46–57. doi:10.1109/2.30720.

Falcone, P., Borrelli, F., Asgari, J., Tseng, H.E., and Hrovat, D. (2007). Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3), 566–580. doi:10.1109/TCST.2007.894653.

Faulwasser, T., Weber, T., Zometa, P., and Findeisen, R. (2017). Implementation of nonlinear model predictive path-following control for an industrial robot. *IEEE Transactions on Control Systems Technology*, 25(4), 1505–1511. doi:10.1109/TCST.2016.2601624.

González-Baños, H.H. and Latombe, J. (2002). Navigation strategies for exploring indoor environments. *I. J. Robotics Res.*, 21(10-11), 829–848. doi:10.1177/027836490202101083 4.

Grüne, L. and Pannek, J. (2017). *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer. doi:10.1007/978-3-319-46024-6.

Johnson, A. (2014). Clipper–an open source freeware library for clipping and offsetting lines and polygons. 2010-2014.

Kavraki, L.E. (1995). Computation of configuration-space obstacles using the fast fourier transform. *IEEE Transactions on Robotics and Automation*, 11(3), 408–413. doi:10.1109/70.388783.

Kavraki, L.E., Svestka, P., Latombe, J.., 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. doi:10.1109/70.508439.

Knauer, M. and Büskens, C. (2012). From WORHP to TransWORHP. In *Proceedings of the 5th International Conference on Astrodynamics Tools and Techniques*.

Kuhlmann, R. and Büskens, C. (2018). A primal–dual augmented lagrangian penalty-interior-point filter line search algorithm. *Mathematical Methods of Operations Research*, 87(3), 451–483. doi:10.1007/s00186-017-0625-x.

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

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press, New York, NY, USA.

Lu, D.V., Ferguson, M., and Hoy, A. (2018). ROS global_planner. `http://wiki.ros.org/global_planner`. Accessed: 2018-11-23.

Miller, L.M. and Murphey, T.D. (2013). Trajectory optimization for continuous ergodic exploration. In *2013 American Control Conference*, 4196–4201. doi:10.1109/ACC.2013.6580484.

Moutarlier, P. and Chatila, R. (1991). Incremental free-space modelling from uncertain data by an autonomous mobile robot. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, 1052–1058 vol.2. doi:10.1109/IROS.1991.174631.

Quigley, M., P. Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Y. Ng, A. (2009). ROS: An open-source robot operating system. *ICRA Workshop on Open Source Software*, 3, 1–6.

Rimon, E. and Koditschek, D.E. (1992). Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5), 501–518. doi:10.1109/70.163777.

Rosolia, U., Bruyne, S.D., and Alleyne, A.G. (2017). Autonomous vehicle control: A nonconvex approach for obstacle avoidance. *IEEE Transactions on Control Systems Technology*, 25(2), 469–484. doi:10.1109/TCST.2016.2569468.

Shiller, Z. (2015). Off-line and on-line trajectory planning. In *Motion and Operation Planning of Robotic Systems*, 29–62. Springer.

Tilove, R.B. (1990). Local obstacle avoidance for mobile robots based on the method of artificial potentials. In *Proceedings., IEEE International Conference on Robotics and Automation*, 566–571 vol.1. doi:10.1109/ROBOT.1990.126041.

von Stryk, O. and Bulirsch, R. (1992). Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1), 357–373. doi:10.1007/BF02071065.

Wang, Z., Li, G., Jiang, H., Chen, Q., and Zhang, H. (2018). Collision-free navigation of autonomous vehicles using convex quadratic programming-based model predictive control. *IEEE/ASME Transactions on Mechatronics*, 23(3), 1103–1113. doi:10.1109/TMECH.2018.2816963.

Weiskircher, T., Wang, Q., and Ayalew, B. (2017). Predictive guidance and control framework for (semi-)autonomous vehicles in public traffic. *IEEE Transactions on Control Systems Technology*, 25(6), 2034–2046. doi:10.1109/TCST.2016.2642164.

Wilson, A.D., Schultz, J.A., and Murphey, T.D. (2014). Trajectory synthesis for fisher information maximization. *IEEE Transactions on Robotics*, 30(6), 1358–1370. doi:10.1109/TRO.2014.2345918.

Ziegler, J., Bender, P., Dang, T., and Stiller, C. (2014). Trajectory planning for Bertha – A local, continuous method. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 450–457. doi:10.1109/IVS.2014.6856581.

Ziegler, J. and Stiller, C. (2010). Fast collision checking for intelligent vehicle motion planning. In *2010 IEEE Intelligent Vehicles Symposium*, 518–522. doi:10.1109/IVS.2010.5547976.