Engineering Applications of Artificial Intelligence 153 (2025) 110690

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Polygon decomposition for obstacle representation in motion planning with Model Predictive Control☆

Aleksey Logunov [a], Muhammad Alhaddad [b], Konstantin Mironov [a,c,d], Konstantin Yakovlev [e], Aleksandr Panov [a,c,e,*]

[a] Moscow Institute of Physics and Technology, Moscow, Russia
[b] University of Aleppo, Aleppo, Syria
[c] AIRI, Moscow, Russia
[d] Ufa University of Science and Technology, Ufa, Russia
[e] Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, Moscow, Russia

ARTICLE INFO

Keywords:
Model predictive control
Polygon decomposition
Artificial potential function
Trajectory planning
Mobile robots

ABSTRACT

Model Predictive Control (MPC) is a powerful tool for planning the local trajectory of autonomous mobile robots. The paper considers a new algorithm for trajectory planning and obstacle avoidance based on the MPC technique known in Artificial Intelligence (AI) planning and robotics. We have proposed an original method for decomposing obstacles to form a potential field, which in turn is used as an additional component in MPC. Thus, we propose a new intelligent trajectory planning method that takes into account the special shape of obstacles, which in turn significantly improves the metrics of intelligent agent movement on the well-known Moving AI benchmark. The challenging aspect of MPC planning is collision avoidance on large and complicated grid maps. We propose the Polygon Segmentation for obtaining Artificial Potential Field (PolySAP). This local planner approximates the obstacles on the map with a set of polygons. We address the question of how to partition a map with polygons to make it fast and effective for a practical MPC planner. We propose a decomposition algorithm based on Straight Skeleton. Our algorithm returns a set of polygons, which are then convexified. Numerical experiments show that our method outperforms basic algorithms in performance and provides sufficient partition quality for effective planning. We propose an artificial potential function calculated for polygonal obstacles and added to the MPC objective for collision avoidance. We evaluate our approach on city map dataset and on a real robotic platform. Numerical experiments show that PolySAP allows for polygon decomposition that is five times faster than Interior Extensions. Our MPC solver provides a fast solution for the MPC task compared to the state-of-the-art MPC planners. Our planner ensured the safe motion of the real mobile robot through a narrow indoor environment. Our code is available at https://github.com/alhaddad-m/PolySAP.

1. Introduction

Autonomous robots can effectively act and solve various tasks in different environments: offices (BrainCorp, 2023b), homes (Szot et al., 2021), shops (BrainCorp, 2023a), medical facilities (MOXI, 2023; Vogel et al., 2021; Parikh et al., 2023), outdoor landscapes (Kayacan and Chowdhary, 2019), and even other planets (Daftry et al., 2022). The planning methods depend on the surroundings of the robots (Panov, 2019). When dealing with complex paths with multiple obstacles, a two-stage planning process is typically used (Jian et al., 2021; Bojadžić et al., 2021). In the initial stage, known as global planning, a rough path is generated from the starting point to the target destination. Subsequently, in the second stage, termed local planning, this preliminary path is refined into an executable trajectory through smoothing or optimization. This local planning phase can be performed in real time using receding horizon planning, incorporating sensor data to dynamically adapt the trajectory (Tong, 2020; Li, 2020). In this case, local planning is often formulated as a Model Predictive Control (MPC) problem (Bojadžić et al., 2021; Schoels et al., 2020a; Thirugnanam et al., 2022; Zuo et al., 2020; Li et al., 2021). This problem may be solved using direct numerical solvers based on interior point method (Waechter and Biegler, 2005–2022) or sequential quadratic

☆ This work was supported by the Ministry of Science and Higher Education of the Russian Federation under Project 075-15-2024-544.
* Corresponding author at: Moscow Institute of Physics and Technology, Moscow, Russia.
E-mail address: panov.ai@mipt.ru (A. Panov).

https://doi.org/10.1016/j.engappai.2025.110690
Received 10 November 2024; Received in revised form 19 March 2025; Accepted 24 March 2025
Available online 22 April 2025

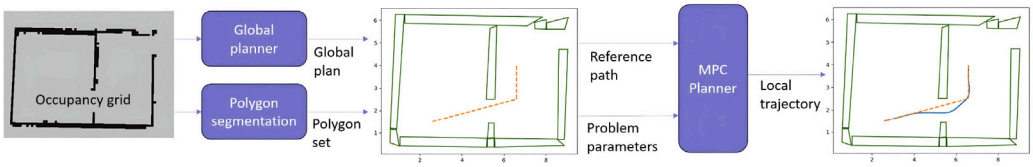


Fig. 1. Common scheme of the PolySAP planning stack. The state-of-the-art rough global planner generates a polyline path based on the occupancy grid. Our decomposition algorithm extracts a set of convex obstacle polygons from the occupancy grid. By projecting the global path onto the polygon map, we can identify the polygons that pose a danger to the robot. The parameters of these polygons are then passed to the obstacle-aware MPC local planner, which generates the robot's trajectory.

programming (Verschueren et al., 2022). Obstacle avoidance may be inserted into MPC formulation as a set of constraints, e.g. Schoels et al. (2020a), or as a set of obstacle-related cost terms, e.g. Thirugnanam et al. (2022). The first approach is sensitive to the quality of the initial guess, which is better to be collision-free for the correct convergence of the numerical solution (Schoels et al., 2020a). Alternatively the stochastic techniques like MPPI (Williams et al., 2016) may be utilized instead of numerical solver; however such techniques may produce unstable or oscillating solutions. Contrary, the second approach allows the numerical optimizer to converge from the initial path, which contains collisions. In this case, a repulsive Artificial Potential Function (APF) may be introduced, which take lower values far from obstacles, very high values inside them, and descend with receding from them (Alhaddad et al., 2024). In our work, we address MPC for local planning, incorporating collision avoidance based on an APF.

Such an APF should be differentiable, as its gradient indicates the direction in which the trajectory should be adjusted to enhance safety. The challenge to be addressed is how to design it for large, arbitrary cell maps, such as Occupancy Grids (OG), which are commonly used in practical applications. Computational heaviness of the numerical MPC may be decreased by integrating neural networks (Song et al., 2023), fuzzy logic (You et al., 2024) or linearizing techniques (Morato et al., 2021). Many existing works consider the obstacles as a known set of simple-shaped figures, e.g Blackmore et al. (2011), Szmuk et al. (2017) and Thirugnanam et al. (2022). Other works aim to find an approximation of the collision danger (Schoels et al., 2020a,b; Kurenkov et al., 2022; Adamkiewicz et al., 2022) or to replace numerical MPC solution with the sampling-based technique that does not require analytical obstacle models (Williams et al., 2016, 2017; Mohamed et al., 2020).

Our work is motivated by the fact that many real environments consist of objects with flat surfaces. These environments can be represented as a maze constructed of polygons. Examples of such cases are corridors and rooms in office buildings, shops, warehouses, or living spaces. It seems useful to develop a planner, which exploits such a structure of the environments. One can decompose mazes into convex polygons and set the simple APF formulation for these polygons. The convex nature of polygons is important for two reasons. First, moving away from the center of such a polygon guarantees a greater distance from the obstacle it represents, based on its geometry. Second, in complex mazes, convexification allows for the generation of polygons with fewer parameters. During the local planning process, small polygons near obstacles are selected, which significantly reduces the number of parameters in the problem. The formulation of APF could be rather simple, without the need to apply a complicated techniques like in Schoels et al. (2020a,b), Kurenkov et al. (2022), Adamkiewicz et al. (2022), Williams et al. (2016, 2017) and Mohamed et al. (2020) This approach is challenging due to two factors. Firstly, polygon decomposition can be computationally intensive. Secondly, the straight lines of the maze may appear non-straight on the occupancy grid due to sensor and quantization errors.

We propose an obstacle avoidance approach based on the fast decomposition of the 2D OG into a set of simple polygons. We call our local planner **PolySAP**, which means **Poly**gon **S**egmentation for obtaining **A**rtificial **P**otential Field. A common scheme of our approach is presented in Fig. 1. A local plan is obtained in two operations. First, polygon decomposition algorithm transforms the occupancy grid into a set of convex polygons. Second, MPC solver optimizes the trajectory regarding the polygons nearest to the global path.

1.1. Contribution

In this paper, we develop both a decomposition algorithm and an MPC local planner. The significance of our contribution lies in the novel decomposition algorithm we propose for transforming an obstacle map into a set of convex polygons. The main purpose of this method is to reduce the initial set of partitions using information from the spatial skeleton. Existing decomposition methods often generate many small polygons and then merge them according to an optimization problem. In a noisy environment, this initial decomposition can result in numerous polygons, which slows down the optimization process. Our method addresses this issue by ignoring small details of polygons that do not impact the robot's ability to navigate the environment. After decomposition, the polygons are convexified to eliminate the possibility of the robot colliding with these minor details. Our algorithm has demonstrated a decomposition process that is five times faster than the baseline Interior Extensions method.

Our contribution also includes a novel APF formulation, which introduces trajectory repulsion from convex obstacle polygons within the MPC optimization loop. We define a sigmoid APF term that depends on the distance between the robot and the nearest line of the obstacle polygon. Experiments showed that our formulation provides significantly faster trajectory optimization than the CIAO (Schoels et al., 2020a) MPC local planner.

1.2. Structure

The rest of the paper is structured as follows. The next section analyzes the existing works on motion planning, collision avoidance, and polygon decomposition. The following two sections introduce our polygon decomposition algorithm and the MPC approach. Then, we describe the implementation and experimental results of our local planner. Last sections discuss the results and provide concluding remarks.

2. Related works

2.1. Motion planning for mobile robots

Robotic planning is a broad research area; see, e.g., the review by González et al. (2016). Global planners are mostly based on random sampling and/or systematic search. Also there are some approaches based on artificial potential field (Khatib, 1985) or bio-inspired discrete optimization (Ding, 2020). Most search-based planners emanate from A* (Hart et al., 1968), a heuristic extension of the graph search algorithm by Dijkstra (1959). Sampling methods are chiefly based either on Rapidly-exploring Random Trees (RRT) by LaValle and Kuffner (2001) or on Probabilistic RoadMaps (PRM) by Kavraki et al. (1996). Note that the concept of APF was initially introduced for global planning (Khatib, 1985; Erdmann and Lozano-Perez, 1986). In APF global planners, moving toward the destination point is considered as a gradient descent of robot coordinates in the artificial potential field. This approach was

further developed in some novel works (Kim and Shin, 2006; Ren et al., 2006; Szczepanski et al., 2022). Its main disadvantage is that it can get stuck in a dead-end, while search-based and sampling-based planners cope with this challenge as they are multi-hypothesis. Optimization seems more effective for local planning where the global plan lies near the global minimum.

There are several works where search-based and sampling-based global planners are modified or extended with specific post-processing to provide an executable local path (e.g., reviews by Heiden et al. (2020) and Gammell and Strub (2021)). Alternative approach is to perform the search not on a grid map, but on a set of motion primitives (Butzke et al., 2014). This approach has high computational capacity due to the large branching factor when searching for primitives. We consider another concept where a rough and fast global planner provides the path as a simple polyline and then turns into an executable trajectory with fast and powerful optimization. Significant examples of such rough planners are Theta* (Nash et al., 2007) and visibility graph (Lozano-Pérez and Wesley, 1979; Yang et al., 2022). The first is an A* extension, which supports any-angle paths on 2D grids. The second search method is done on the graph of obstacle polygon vertices instead of the cell structure of OG. It is fast and provides a short path. However, these paths are dangerous as they touch obstacle borders.

2.2. Trajectory optimization and collision avoidance

Trajectory optimization can be done in two modes. In the first one, the trajectory is considered holistically. There are specific algorithms for this statement, such as CHOMP (Ratliff et al., 2009), STOMP (Kalakrishnan et al., 2011), TrajOpt (Schulman et al., 2014), or GuSTO (Bonalli et al., 2019). This mode does not cover the possibility of meeting previously unknown obstacles while moving along the path. The second mode (MPC) assumes that the optimization is done for a certain part of the future path (prediction horizon), and re-optimization is done after a certain period (control horizon). Some approaches, e.g. CIAO (Schoels et al., 2020a), allow the planning problem to be solved in both modes. MPC has to be strictly real-time with a specified re-planning rate. There are specific tools that provide real-time solutions to correctly stated MPC problems (including nonlinear cases). IPOPT (Waechter and Biegler, 2005–2022) and ForcesPro (Zanelli et al., 2017) utilize the interior point method, while ACADO (Houska et al., 2011a,b) and Acados (Verschueren et al., 2022) apply sequential quadratic programming to obtain the solution. In this work, we use Acados, a fast and novel tool for MPC.

There is a number of works on common collision avoidance (Gilbert et al., 1988; Stoican et al., 2019; Zhang et al., 2022; Zimmermann et al., 2022), when the task is recognize the fact of collision for the given robote pose. Contrary, the MPC statement requires collision avoidance to be expressed analytically as a cost term or a set of constraints. In the first case, the obstacle-related cost term (APF) needs to be differentiable to allow gradient repulsion from the obstacles. If the obstacles are represented with a known set of simple geometric shapes (points, circles, ellipses, or polygons), defining analytical APF is not a very difficult job. This statement is considered, e.g. by Szmuk et al. (2017), Luis et al. (2020) and Wu et al. (2021). The task is more challenging for the arbitrary cell map with unstructured obstacles. Analytical solutions to MPC problems can be replaced with sampling-based Model Predictive Path Integral approach for local planning (Williams et al., 2016, 2017; Mohamed et al., 2020).

Another approach is to train a neural model of collision danger (Adamkiewicz et al., 2022; Kurenkov et al., 2022; Salzmann et al., 2024; Katerishich et al., 2023; Alhaddad et al., 2024). Solver by Alhaddad et al. (2024) work in real time however, it require high computation power, which is obtained from remote server. Other mentioned neural models are non-realtime.

We are considering another approach where an arbitrary obstacle map is approximated with simpler geometric figures to minimize the computations. In CIAO (Schoels et al., 2020a,b), free space around the robot is approximated with a simple convex figure (circle or square). The use of convex free space models guarantees the absence of collisions, provided there is reachable initial guess. However, it significantly limits the ability to modify the trajectory during local planning. Point-wise and circular obstacles are handled by Ji et al. (2016) and Zeng et al. (2021) respectively. In Ziegler et al. (2014), the trajectory of the autonomous cars is constrained with two polylines for the lane-following task. Papaioannou et al. (2023) represent obstacles with cuboids; obstacle detection probability is assigned to each cuboid. Blackmore et al. (2011) introduces an approach for avoiding polytopic obstacles; the question of how to obtain polygons from the map is not considered. Thirugnanam et al. (2022) puts forth a collision model for the case when both the robot and the obstacles are polytopic. This approach requires discrete-time dynamics.

2.3. Polygon decomposition

Area decomposition methods include:

1. Cell decomposition: An area is represented as a polygon. The polygon is divided into a set of smaller polygons. For most cell decomposition methods, two stages are distinguished: the initial decomposition and the stage of combining areas.
2. Area segmentation: An area is represented as a polygon or an occupancy grid. It is used for areas that are rooms inside buildings. Despite this, some algorithms may be used in street spaces.

Area segmentation includes methods such as watershed algorithm, morphological segmentation, and distance-based segmentation (Bormann et al., 2016). They use the morphological erosion and dilation operators to solve the segmentation problems. This approach is unsuitable for solving our task, as it does not consider the context and cannot guarantee the convexity of the partition. Trapezoidal Decomposition (Latombe, 1991) and Boustophedon Decomposition (Choset and Pignon, 1998) are classical algorithms for the space decomposition problem, but they do not take into account any kinematic constraints of the robot. Their generalization (Morse Decomposition by Acar et al. (2002)) is too hard to implement.

Many papers present methods (e.g. Huang (2001), Nielsen et al. (2019), Li et al. (2020, 2011) and Tang et al. (2021)) that include the following steps:

1. Build a polygon's vertices that satisfy the chosen heuristics (all vertex or only concave vertex).
2. Draw split lines in some directions (e.g., an extension of edges) that form the primary decomposition.
3. Solve certain optimization problems and merge cells from primary decomposition.

This approach has the following drawbacks: primary decomposition can include a lot of cells, and the selection of the optimal combination of cells can be very long. Li et al. (2011) tried to fix the first challenge by drawing dividing lines only from the concave vertices. Tang et al. (2021) proposes using a depth-first search to solve the second problem. Notably, in Li et al. (2020), the authors decide to move away from solving the optimization problem and draw split lines between the polygon vertices according to certain heuristics. However, this approach does not draw split lines between the polygon edge's vertices, which can lead to drawing too long split lines.

An alternative way to reduce the area search space is offered by Tang et al. (2021). The authors present the R-DFS method based on a depth-first search. Voronoi graph-based (Thrun, 1998; Preparata and Shamos, 1989) segmentation uses space skeleton to solve segmentation

problem. This approach extracts information from the space skeleton and utilizes unique heuristics for each task. The use of heuristics along with the space skeleton makes it possible to construct various segmentation and decomposition methods that solve a large class of applied problems. Such methods include CDM (Construction Decomposition Method) (Brown, 2017), which uses a Straight skeleton as a space skeleton (Aichholzer and Aurenhammer, 1995).

2.4. Discussion

Analysis of the related works lead to the following outcomes:

- The state of the art approach to trajectory planning for mobile robots include global path planning for generating the rough geometric path and local motion planning for turning this path into smooth trajectory, which avoid obstacles and satisfy kinodynamic constraints. Global planning is effectively solved via search-based algorithms such as Theta* (Nash et al., 2007), which output the geometric path as a polyline. Local planning may be solved via model predictive control (MPC).
- Collision avoidance within MPC planning is challenging, as it require analytic representation of obstacles. This challenge is overcome by use of gradient-free randomized MPPI planners (Williams et al., 2016, 2017), by use of learnable collision model (Adamkiewicz et al., 2022; Kurenkov et al., 2022), or by approximating obstacle map with simple geometric shapes (Schoels et al., 2020a; Thirugnanam et al., 2022; Schoels et al., 2020b). The last approach provide stable and fast results. MPPI and neural fields provide a solution for arbitrary maps by using computationally redundant techniques (random sampling and approximation with the complex neural model).
- Approaches, which rely on geometric approximation, often consider this approximation to be given (e.g. polytopic obstacles in Blackmore et al. (2011), Thirugnanam et al. (2022)). The task of obtaining this representation from obstacle map is considered in Schoels et al. (2020a,b) however, resulting representation mark a lot of free space as obstacles. We cannot specify existing approach, which provide realtime approximation of collision danger, and mark obstacles and free space accurately. In this work we aim to provide MPC local planner, which include fast and accurate geometric obstacle representation.
- Obstacle approximation with a set of convex polygons seems promising as it, first, may be effectively handled within MPC (Blackmore et al., 2011; Thirugnanam et al., 2022) and, second, allow for high accuracy. Space decomposition with polygons is a well-know task in computational geometry. There is a number of works on this task (Bormann et al., 2016; Nielsen et al., 2019; Li et al., 2020, 2011; Tang et al., 2021). However, there is no method, specifically desgned for MPC obstacle avoidance. The task of local planning provide a specific requirements for polygon decomposition. Its performance is critical, while the accuracy has to be tuned according to the resolution of the common obstacle map and linear size of the robot.

In general, this work aim to develop a novel MPC local planner, which utilize polygon decomposition of the obstacle map. The issues, that we try to overcome, is computational complexity of polygon decomposition and low accuracy of obstacle modeling with geometric approximation (which lead to lower quality of the planned trajectories). In Section 5 we show that our algorithm provide fast obstacle decomposition and high-quality real-time planning in the cluttered environments.

3. Polygon decomposition

3.1. Background

Consider a robot navigating an environment populated with obstacles (of non-trivial shapes). The surrounding map is available (e.g. through the simultaneous localization and mapping system) in the form of the occupancy grid. The latter can be considered a binary image, i.e. the image that contains only two types of pixels: black (corresponding to obstacles) and white (corresponding to free space). Denote the set of all blocked pixels on the image as X. We aim to decompose it into the N regions $\{S_i\}_{i=1}^{N}$ so that the following conditions are met:

1. $X = \bigcup_{i=1}^{N} S_i$,
2. $S_i \cap S_j = \emptyset,\ i \neq j$,
3. $\texttt{IsConvex}(S_i \cup S_j) = False\ i \neq j$.

Here `IsConvex` is a predicate that returns true if the region is convex. Informally, we want to split the arbitrary-shaped obstacles that surround the robot into the convex sub-obstacles.

The following two criteria will be employed to measure the effectiveness of the decomposition. The first is the number of sub-regions, N (the lower, the better). The second is Δ (also, the lower, the better), which is computed as:

$$\Delta = \sum_{i=1}^{N} A(C(S_i)) / \sum_{i=1}^{N} A(S_i) - 1. \tag{1}$$

Here $A(\cdot)$ denotes the region's area, and $C(\cdot)$ denotes the convex hull of the region. The introduced criterion is minimal, i.e., it equals 0, when all the regions S_i are convex (as in this case, the areas of their convex hulls equal the areas of the regions themselves). If some S_i is not convex, Δ is positive. Following this formulation, the two existing methods can be mentioned.

The first method is **Interior Extension of Edges** (Nielsen et al., 2019). Interestingly, it represents an approach with the initial partitioning of space and subsequent solution of the optimization problem to unite the regions. This method works with a polygon representation of the workspace. The polygon vertices are selected whose forming faces make an angle greater than 180 degrees. Further, separating lines are drawn from these faces, extending them already outside the polygon. The intersections of the separating lines divide the workspace into a set of convex polygons S. The second stage of the algorithm is the union of convex polygons. A set of possibilities for merging two polygons Ω is created, and its elements are denoted as j. The following integer programming problem is solved for minimizing the total width of the combined polygons:

$$\begin{aligned} &\min \sum_{j\in\Omega} w_j \lambda_j, \\ &\text{s.t.} \\ &\sum_{j\in\Omega} a_{ij}\lambda_j = 1, \forall i \in S, \\ &\lambda_j \in \{0,1\}, \forall j \in \Omega, \end{aligned} \tag{2}$$

where w_j is polygon width; $a_i j$ is the possibility of polygon i merging with its neighbors, defined in the set Ω, and equals 0 or 1; λ_j is the decision variable, which determines the choice of the possibility of polygon merging. An example of how the algorithm works is shown in Fig. 2. We can think of an obstacle as a polygon with a hole. Thus, on Fig. 2, the initial partition of a polygon consists of ten polygons, and the final partition has five.

Interior Extension of Edges solves an optimization problem for merging polygons. In the case of heavily noisy maps, the initial partitioning may be excessive, which can negatively affect the speed of solving the optimization problem 3. An example of a noisy map is shown in Fig. 3. By noise, we mean various artifacts caused by the

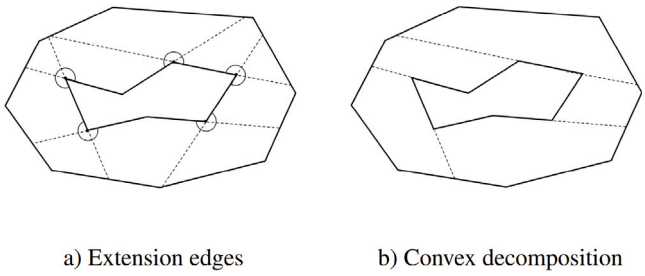


Fig. 2. An example of how the interior extension of edges works (Nielsen et al., 2019). (a) Initial partition after extension of edges. Partition consists ten convex polygons. (b) The final partition after solving the optimization problem. Partition consists five polygons.

Fig. 3. Excessive initial partitioning example. Inaccuracy of the sensors lead to artifacts, which enlarge the number of small polygons.

inaccuracy of the sensors. Indeed, there exists a (limited) number of the algorithms that can be used to remove this type of noise, like the Douglas-Pecker algorithm (Douglas and Peucker, 1973). However, such methods require extensive parameter tuning for each problem instance, which significantly limits their applicability. To this end, we suggest our own method of decomposition based on the ideas of another decomposition method that is relevant to us, i.e. **CDM** (Construction Decomposition Method) (Brown, 2017).

Noteworthy feature of CDM is that it represents the space skeleton approach. The latter extracts information from the skeleton to solve a decomposition problem. The main task of the method is to divide the office space map into rooms. This is achieved by searching for places where space begins to narrow.

CDM works with the polygon representation of the map. Based on the vertices of P polygons, a Straight Skeleton $S(P)$ is constructed. The Straight Skeleton (Aichholzer and Aurenhammer, 1995) is a structure generated through continuous uniform compression of the original polygon. The skeleton is a set of N vertices. Each of these vertices has a "creation time" t_i; event type $\tau \in \{\tau^s, \tau^e\}$, where s and e means which event formed the vertex: area break or edge removal, respectively, as well as a set of neighboring vertices connected by skeleton faces. An example straight skeleton is shown in Fig. 4.

CDM performs a complete traversal of the $S(P)$ skeleton vertices in search of split points. Split points are the skeleton vertex that formed the event when area break τ^s. When the dividing point s is found, a new face e_{ij} is added, connecting the vertices $v_i, v_j \in V$ (V, a set of polygons vertices), called the dividing line. The vertices v_i, v_j are located on different faces and closest to the skeleton s vertex. Creating a dividing line is denoted by the function $\nu(e_{k,l}, v_i) \to v^*$. Vertex v_* is added as follows: the face $e_{k,l}$, between whose vertices v_k, v_l the vertex v_* is located, is divided into two faces: $e_{k,*}$ and $e_{*,l}$. The separating points are those vertices of the s_i skeleton that meet the condition $time(i) < time(i+1) \cap time(i) < time(i+1)$. An example of how the CDM algorithm works is shown in Fig. 4(b). In this image, the skeleton is marked in blue, the split lines are in red (thus, the initial polygon is split into three sub-areas). In this work, we will not use CDM for the decomposition of obstacles straightforwardly but rather use its by-product, i.e. the straight skeleton, to create our own decomposition algorithm.

3.2. Our algorithm

Recall that our main goal is to decompose the obstacles on the input map into a set of convex polygons. The latter representation is beneficial for local planning as a typical local planner can only handle convex obstacles with primitive shapes (e.g. rectangles, ovals, etc.). The input of the decomposing algorithm is the binary occupancy grid obtained through the robot's sensors, which are subject to noise. First, we vectorize the map, i.e. convert each obstacle on a map from a set of occupied cells to a (possibly non-convex) polygon. This can be done by one of the several edge extraction algorithms like (Douglas and Peucker, 1973). Next, we iterate over the obstacles and split each obstacle into a set of primitive shapes (this work uses rectangles).

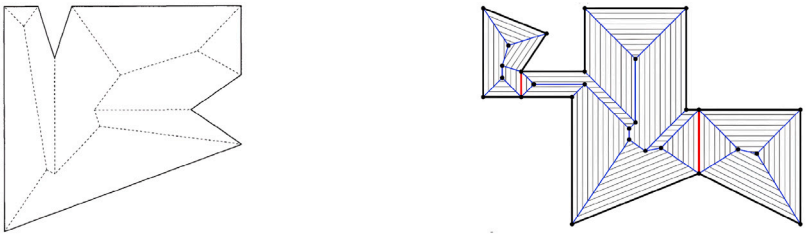
Indeed, one can use the previously described Interior Extension of Edges algorithm for splitting a polygonal obstacle of a complex shape into a set of sub-polygons, but in practice, it does not perform well. The reason is the following. In the real world, faces of numerous obstacles like sofas, boxes, etc. are straight but, due to the inaccurate mapping, they are transferred to the occupancy grid and later to the polygonal representation of this grid map as broken lines. Consider Fig. 5 as an example. On the left, one can see the original obstacle of a rectangular shape. On the right, we show how this obstacle is actually mapped. Clearly, numerous "noisy" vertices of the polygon are introduced. It is these vertices that cause trouble when you straightforwardly apply Interior Extension of Edges method for decomposition, as each vertex that forms a concave angle is the source of the split line. Consequently, there are many split lines, and the initial split of the polygon is composed of numerous sub-polygons. Merging them becomes a bottleneck, especially for the prolonged obstacles.

To this end, we suggest filtering out certain vertices (i.e. not consider them as the sources of the split lines), and we use the previously mentioned Straight Skeletons (or simply skeletons) for that.

We assume that most obstacles can be approximated by a rectangle. Other robots, cars, walls, furniture, and other objects on top can be represented as a rectangle. If the obstacle has a non-convex shape, it can be represented as a set of rectangles. Let us now consider the straight skeleton of a rectangle. The straight skeleton of rectangles has a face parallel to and equidistant from the two largest edges of the rectangle. However, if the obstacle's contour contains a lot of noise, the skeleton's face will appear as a broken line (Fig. 6(b)). The essence of the our method is to select a sequence of vertices on the skeleton in the vicinity of a certain line. We assume that the vertices of the skeleton connected into a broken line approximate a skeleton face of the original polygon. By the definition of the edge of the skeleton, this line will lie between the two parallel edges of the rectangle that make up the obstacle. This approach, as will be shown below, will work well if the polygon consists of several rectangles rather than one.

We will represent the Straight Skeleton as an undirected graph. The vertices of the graph contain information about the metric coordinates of the skeleton point, time t, and all vertices connected with it. We take a random vertex of the skeleton and add connected vertices to it until they no longer approach a certain straight line with a given error. Since we collect the vertices of the skeleton around a certain line, there can be several such sets of vertices. Each of the resulting sets of vertices can correspond to individual rectangles, which can form complex polygons.

We have formed sets of vertices that are located along some lines. Now there is no need to work with every corner of the original polygon, but only with those that are within a certain radius around the extreme points (the two points farthest from each other). We can say that these two extreme points approximate this line. The radius information can be taken from the Straight Skeleton, since its vertices contain a parameter called "creation time" (hereinafter referred to as "time"), which can be used to measure the distance between this point and the nearest vertices of the polygon. This reveals one of the significant practical advantages of using a direct skeleton: the information about the radius of the desired neighborhood is calculated along with the

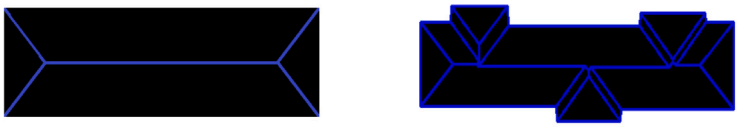


a) Straight skeleton Aichholzer and Aurenhammer (1995) b) CDM decomposition Brown (2017)

Fig. 4. An example of how the CDM works. The skeleton is marked in blue, the split lines are in red (thus, the initial polygon is split into three sub-areas).

a) Origin polygon b) Noisy polygon

Fig. 5. An example of origin polygon and noisy polygon.



a) Straight Skeleton of origin polygon b) Straight Skeleton of noisy polygon

Fig. 6. An example of straight skeletons of origin polygon and noisy polygon.

creation of the skeleton. Next, we extend the faces that form the concave corner into the polygon until the extended edges intersect another face of the polygon 2. Of the two resulting "split lines", we select the shorter one.

This approach makes it possible to significantly reduce the number of dividing lines and, consequently, the number of cells during the initial partitioning; see Fig. 7(h). Also, grouping the vertices of the skeleton around a certain straight line can reduce the contribution of noise to the final partition, since the straight line in this case represents an estimate of the original face of the skeleton, which would be if there was no noise. Finally, we combine these segments according to the optimization problem (2).

So our method works as follows. First, we form the Straight Skeleton, then group its vertices around some lines. In each set of vertices, we select the two extreme ones. Next, we expand only those faces that form concave corners and are located near the extreme points. This way we get the initial partition. And in the end, we combine the resulting segments.

Let us now describe it in detail. Our method includes next stages:

1. Build Straight Skeleton (Fig. 7(a)).
2. Skeleton vertices $S = \{s_i\}_{i=1}^{N}$ are combined into clusters $\{K_j\}_{i=1}^{m} \in S, m \leq N$:

$$|f(s_{k_x}, s_{k_y})| < \max_k(t_{s_k}), \forall s_k \in K_j, k = 1, \dots, n \tag{3}$$

$f(x, y) = ax + by + c$ is the normal equation of the line. The introduction of this heuristic is motivated by the fact that we can select the vertices of the skeleton that lie on the same line. It is assumed that the obstacles consist of a set of rectangles that must be selected. The width of such rectangles is assumed to be $\max_k(t_{s_k})$. The clustering goes like this:

   (a) Give a random skeleton vertex v_0 and their neighbor v_1. If there are several neighbors, choose the closest one.

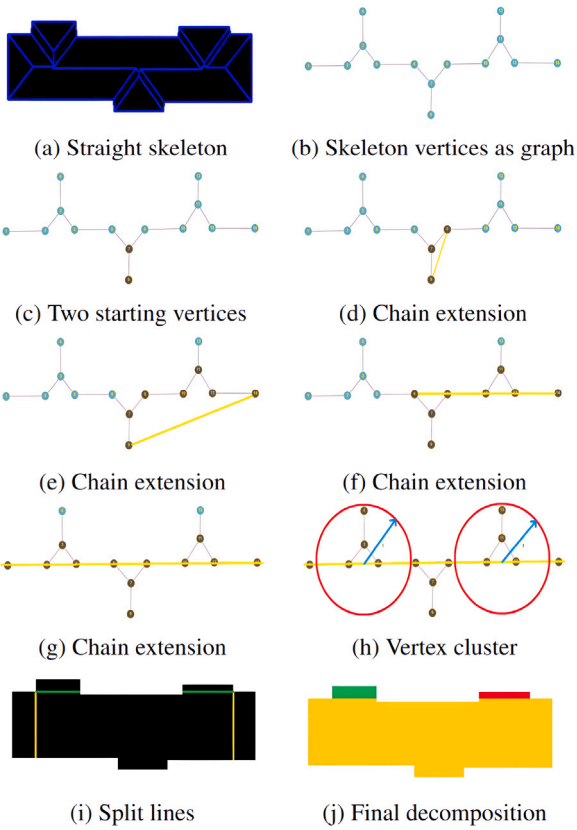


Fig. 7. An example of various stages of the decomposition method.

$C = \{v_0, v_1\}$ and is taken as the beginning s_b and the end s_e of the chain. Hereinafter, the two most distant vertices from each other will be called the start and end vertices.

(b) In a loop. In turn, the predominantly starting vertex of the chain or the end vertex of the chain are connected to the neighboring vertex, and a check is made whether in this case the entire chain forms a straight line. If so, a new vertex is added to the chain; if not, it is discarded. If the initial or final vertex does not have such neighbors, the search for points associated with internal vertices is performed. The cycle ends when the chain stops expanding.

(c) Condition for checking that the vertices lie approximately on the same line: using two extreme points of the chain, one of which is a candidate for joining, the general formula of the line $ax + by + c = 0$ is formed (Fig. 7(b)). Further, intermediate points of the chain are substituted into this formula, and deviations are obtained at the output: the distances from these points to the straight line. Everything is in order if it is less than the specified parameter. They lie almost on the same line.

(d) We save the resulting chain, and if there are still unused vertices, we return to step (a); otherwise, we stop.

3. We use the vertices of the end and the beginning of the chains (s_e and s_b, respectively) and search for the nearest vertices of the obstacle boundary in circles of radius $t_{(s_e)}$ and $t_{(s_b)}$. If these vertices are concave, the faces that form them expand to the first intersection with the face of the original polygon.
4. Of the obtained pairs of separating lines, we leave the lines that are the smallest in length (Fig. 7(c)).
5. If not all vertices are connected, we return to step 2. Otherwise, go for item 5.
6. Adjacent segments that form a convex figure are merged. Solving the optimization problem (2)

The work of the method is shown in Fig. 7. Image (a) shows the Straight Skeleton of the obstacle, and image (b) shows the graph representation of the skeleton. We start at vertex 7 and select 9 as the most distant vertex to the selected (c). Next, we choose 8 as the closest vertex to 7 (d). Vertices 9 and 8 determine the line based on which the condition (3) is checked. Continuing the chain in figure (e), we cannot expand further to the right. Then vertex 6 is chosen. In this case, the start and end vertices are 6 and 14, as the farthest from each other (f). Continuing the chain to the left (g), we are looking for other vertices connected to the inner vertices of the chain. Since the vertices 4 and 13 are at a distance less than $\max_i t_i$ from the line 1-14, they are also members of the cluster (h). Next, the nearest concave vertices of the polygon to the skeleton vertices 1 and 14 are searched for. Separating lines are drawn from them, and the shortest (i) is selected from a pair of lines.

Method pseudo-code is provided in algorithm 1. V are skeleton vertices (which are not contours); E is a set of polygon faces; C is a current chain of skeletal vertices located around the straight line L defined by its extreme points; $f(C)$ is a function that returns the normal formula of the straight line, constructed from two extreme points of the chain C - v_b, and v_e; $\psi(v_i, L)$ is the membership function of the vertex v_i in the neighborhood of the line L containing elements from C. $\omega(s, E)$ are the function searches for concave polygon vertices in some neighborhood around the skeleton vertex v, returns true if such a vertex exists, and false if it does not. $\nu(v, E)$ is a function that takes a skeleton vertex v and a set of polygon faces E as input. It searches for the nearest concave vertex and expands the faces that form that vertex, returns two segments corresponding to the expanded faces.

Following this algorithm, we calculate the convex hulls for the resulting polygons. Our goal is to include the minimum free space in the convex hulls. As shown in the experiments section, this goal is achieved, and the free space included in the convex hull does not affect the final local trajectory.

Algorithm 1 Decomposition method based on Straight Skeleton.

Require: Skeleton vertex N; initialization E: $E' = E$; initialization V
Ensure: Set of polygon faces with split lines E'.

while $|V| \neq 0$ **do**
  $C = \{v_0, v_1\}$
  $V = V \setminus C$
  $L = f(C)$
  for all $v_i \in V$ **do**
    if $\psi(v_i)$ **then**
      $C = C \cup v_i$
      $V = V \setminus v_i$
      $L = f(C)$
    end if
  end for
  if $\omega(v_b, E)$ **then**
    $e_1^b, e_2^b = \nu(v_b, E)$
    $E' = E' \cup \min(|e_1^b|, |e_2^b|)$
  end if
  if $\omega(v_e, E)$ **then**
    $e_1^e, e_2^e = \nu(v_e, E)$
    $E' = E' \cup \min(|e_1^e|, |e_2^e|)$
  end if
end while

4. Trajectory optimization regarding obstacle polygons

4.1. Common MPC formulation for local trajectory planning

Model Predictive Control (MPC) is a feedback control approach that uses a model to predict the future output of a process and determines the next immediate control action by solving an optimal control problem over a receding horizon. We apply the following common nonlinear MPC formulation with continuous-time state dynamics and discrete time control:

$$\{\mathbf{x}_{opt}[i], \mathbf{u}_{opt}[i]\}_{i=k}^{k+m} = arg \min_{x,u,p} \sum_{i=k}^{k+m} J(\mathbf{q}[i], \mathbf{u}[i], \mathbf{p}[i]), \tag{4a}$$

s.t.

$$\dot{\mathbf{x}}[i] = \begin{bmatrix} f_1(\mathbf{x}[i], \mathbf{u}[i], \mathbf{p}[i]) \\ f_2(\mathbf{x}[i], \mathbf{u}[i], \mathbf{p}[i]) \\ \cdots \\ f_\mu(\mathbf{x}[i], \mathbf{u}[i], \mathbf{p}[i]) \end{bmatrix} \tag{4b}$$

$$\begin{aligned} &h_1(\mathbf{x}[i], \mathbf{u}[i], \mathbf{p}[i]) \leq 0, \\ &h_2(\mathbf{x}[i], \mathbf{u}[i], \mathbf{p}[i]) \leq 0, \\ &\cdots \\ &h_\chi(\mathbf{x}[i], \mathbf{u}[i], \mathbf{p}[i]) \leq 0. \end{aligned} \tag{4c}$$

Here m denotes the prediction horizon, $\mathbf{q}[i]$ is a μ-size state vector, $\mathbf{u}[i]$ is a ν-size vector of control inputs (considered as constant within the timestep i), and $\mathbf{p}[i]$ is a κ-size parameter vector. (4a) defines the cost J. (4b) defines the continuous dynamics of the process. (4c) is a set of inequality constraints that must be satisfied within the whole process. The reference of $\{\mathbf{q}_{opt}[i], \mathbf{u}_{opt}[i]\}_{i=k}^{k+m}$ provides minimum value of J. This reference is a result of the optimization procedure.

For the kinematic model of the differential-drive mobile robot, let the state vector be $\mathbf{x} = [x, y, v, \theta]^T$, where x, y is a coordinate of the robot in the global frame, θ is a heading angle, and v is a linear velocity. Model (4b) looks as follows:

$$\dot{\mathbf{x}} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ a \\ \omega \end{bmatrix} \tag{5}$$

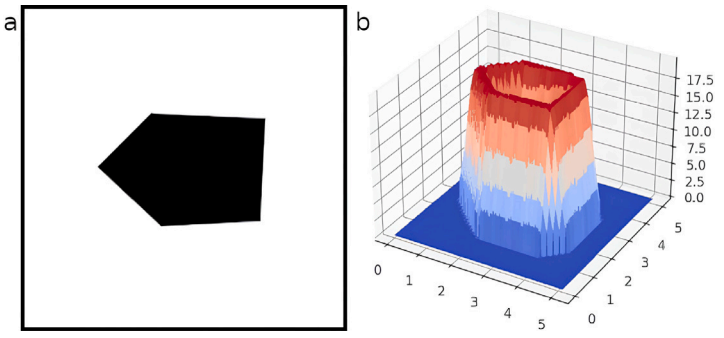


Fig. 8. Polygon (a) and its potential field (b).

where a, ω are the linear acceleration and angular velocity.

The cost function J is a sum of two terms: path-related J_1 and obstacle-related J_2. Obstacle-related term is similar to repulsive potential term in the classic APF algorithm (Khatib, 1985). It is defined in the next subsection, while the path-related term is given as follows:

$$J_1(\mathbf{x}, \mathbf{u}) = \|\mathbf{x}[m] - \mathbf{x}_r[m]\|_{w_e} + \sum_{i=0}^{m-1} l(\mathbf{x}[i], \mathbf{u}[i]), \tag{6}$$

where $l(\mathbf{x}(k), \mathbf{u}(k)) = \|\mathbf{x}(k) - \mathbf{x}_r(k)\|_{w_x} + \|\mathbf{u}(k)\|_{w_u}$, $w_x \in \mathbb{R}^n$, $w_e \in \mathbb{R}^n$, $w_u \in \mathbb{R}^m$ are weights for states and control variables. While the position coordinates (x_r, y_r, θ_r) of the reference path $\mathbf{x}_r$ can be obtained with the global planner, the reference value v_r is suggested based on the target speed.

Constraint-based methods such as Schoels et al. (2020a) and Schoels et al. (2020b) use (4c) to express collision avoidance conditions. We do not follow this approach and use (4c) only for the box constraints of the variables.

4.2. Obtaining the obstacle-related cost (APF)

In the proposed problem, we assume that the obstacles are represented as a set of convex polygons, obtained as a result of the decomposition algorithm from the previous section. In this subsection, we define J_2 based on this set. It should be noted that the distance between neighboring points of the trajectory in practice is several times smaller than the considered size of the robot. Since the robot is convex, a slight increase in its considered dimensions allows for accounting for potential collisions between sampling instants.

Assume that a set of polygons includes N_s sides totally. A local coordinate system is attached to the starting point of each side s passed on to the MPC problem and counterclockwise with respect to each polygon. To optimize a collision-free trajectory $\mathbf{x_t}[k] = \{x_k, y_k\}$, $i = 0, \ldots, m$ over prediction horizon m, the position of every point of this trajectory is re-obtained in the local-coordinate system attached to the side s. Suppose (x_0, y_0) is the position of the trajectory point $\mathbf{x_t}[i]$ in the local system attached to the side s. Then, the cost function describing the potential of collision between the polygon's side s and the robot at the position $\mathbf{x_t}[k]$ is given as follows:

$$J_2(\mathbf{x_{t[k]}}, s) = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{y_0^2}{2\sigma^2}) & \text{if } \begin{array}{l} y_0 + d_3 > 0 \\ d_4 < x_0 < l_s + d_4 \end{array} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $\sigma > 0$, l_s is the length of segment s, and d_3, d_4 determine the boundaries of the area in a local coordinate system in which the motion of the robot is penalized. Fig. 8 shows an example of a map image with one polygon and its potential field obtained with (7). Finally, the cost function for the proposed optimization trajectory problem is the following:

$$J = J_1(\mathbf{x}, \mathbf{u}) + \sum_{i=1}^{m} \sum_{j=1}^{N_s} J_2(\mathbf{x_{t[k]}}, s(i)). \tag{8}$$

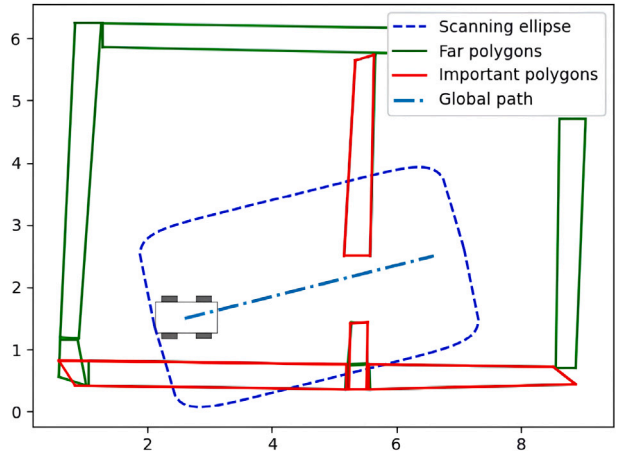


Fig. 9. Choosing important polygons. A six-order ellipse is generated around the global path. Polygons whose parts lie inside this ellipse are passed on to the MPC solver.

4.3. Choice of the surrounding polygons

While the robot can move in the area with an undefined number of obstacles, the number of polygon sides that the proposed algorithm can handle is defined by N_s. Since the reference path in the optimizing trajectory problem is known ($\mathbf{x}_r$), we propose an approach to choose polygons surrounding the path and affecting the trajectory is suggested. Further, the cost function is obtained with the knowledge of these obstacles and the initial path of the robot.

The proposed obstacle classifying algorithm depends on the initial path of the robot in the moving area. Suppose an initial path (Fig. 9) whose length $l \in \mathbb{R}^+$ consists of $S_p \in \mathbb{N}$ segments with $P_s = [x_s, y_s, \theta_s]$ defines the position of the center (x_s, y_s), and orientation θ_s for every segment of the path in a global coordinate system. Assuming that the moving area shown in Fig. 9 has $N_o \in \mathbb{N}$ obstacles, every one of which has $v \in \mathbb{N}$ vertices with $P_o = [x_o, y_o]$ defines the positions of every vertex in the same coordinate system. Let us introduce the variable d_o defined with the following equation:

$$d_o = \frac{((x_o - x_s)cos(\theta_s) + (y_o - y_s)sin(\theta_s))^6}{d_1^6} + \frac{(-(x_o - x_s)sin(\theta_s) + (y_o - y)cos(\theta_s))^6}{d_2^6}, \tag{9}$$

Here d_1, d_2 define the boundaries of a scanning ellipse, where the polygons are classified as obstacles affecting the trajectory of the robot. The value of d_o is obtained for every segment vertex pair. If $d_o \leq 1$, then the polygon corresponding to the current vertex is located inside the scanning area and classified as an important obstacle and included in the optimization problem. Otherwise, the polygon is classified as an obstacle far from the trajectory and is not included in the optimization

problem. As a result of this procedure, $N_{nearest}$ significant obstacles with N_s sides are passed on to the MPC problem.

5. Results

This section discuss an experimental evaluation of our approach. The first subsection provide the technical details and parameters of the experimental implementation. We have made a set of diverse experiments, which evaluate, first, our polygon decomposition algorithm (Section 5.2), second, our MPC solver (Section 5.3), and, third, both algorithms working together (Section 5.4). Numerical evaluation is made both on virtual and real data (Section 5.3), furthermore, we provide a real robot experiments (Section 5.4). In these experiments Husky UGV mobile robot was moving under our approach in the office environment (twisty corridors of the campus building) and in the artificial cluttered environment with a narrow passage.

5.1. Implementation and parameters

Our general approach executes local planning as a sequence of the following operations:

- Apply the decomposition algorithm to obtain the set of non-convex polygons from the occupancy grid;
- Get the convex hulls for obtained polygons;
- Choose the convex polygons nearest to the global path;
- Solve the MPC problem defined by the process model (5) and cost (8) with respect to the nearest polygons.

To demonstrate the reliability of the proposed trajectory optimization problem, we implement its using Acados (Verschueren et al., 2022). The studied optimization problem is described using a Python interface; then a generated self-contained C code for this problem is deployed on an embedded platform. It utilizes the Sequential Quadratic programming to solve the specified problem. For the QP problem in Acados framework, we have used partial condensing high-performance interior-point (HPIPM) solver since Acados relies on HPIPM for reformulating QP problems via (partial) condensing and expansion routines. Prediction horizon for MPC was set to 30 as higher number lead to meaningful enlarging of the SQP optimization time. Weighting coefficients were tuned to $\mathbf{w}_x = [5, 5, 5 * 10^{-5}, 10^{-3}]^T$ and $\mathbf{w}_u = [10^{-2}, 10^{-6}]^T$. This is our common setting for MPC-based local planning. The maximum number of polygon edges, considered by MPC, was set to 100. The reason for this limitation is that Acados solver process up to 400 problem parameters without delays, while one polygon edge correspond to 4 parameters.

Fig. 10 shows an example of the obtained trajectory with two obstacles represented with polygons. The global path of the motion consists of two segments between the point $(2.6, 0.6)$ and $(5.5, 6.5)$ where the initial orientation of the model is zero. The dashed line represents the initial path while the continuous is the optimized trajectory. Fig. 11 shows obtained trajectory with the proposed optimization problem for a part of the map from MovingAI 2D Pathfinding Benchmark (Sturtevant, 2012). These results show that the proposed optimization problem can provide a collision-free trajectory.

The decomposition method has been implemented using the OpenCV and CGAL libraries. OpenCV[1] is applied to find and approximate contours, while CGAL[2] is applied to create and work with Straight Skeleton.

The following issues are also worth highlighting. To decompose the polygons, you need to select contours on the grid. We have used the representation of the grid as an image and selected the contours

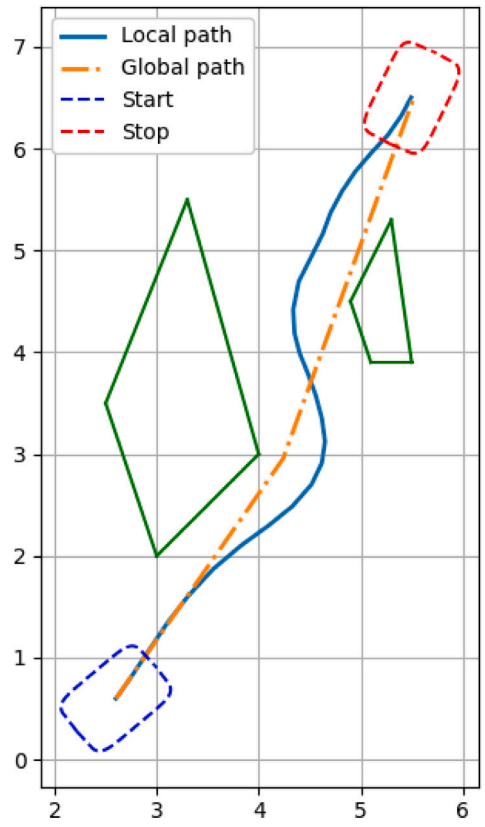


Fig. 10. Example trajectory obtained from the MPC local planner considering two simple polygonal obstacles and a polyline global plan.

using methods from the OpenCV library. To eliminate the excessive number of polygon vertices, the Douglas-Pecker algorithm (Douglas and Peucker, 1973) has been used. Employing it requires manual selection of the approximation parameter ϵ. Also, sometimes the resulting polygons can be self-intersecting, which make it impossible to create a Straight skeleton. Besides, due to noise, the selected polygons have had to be filtered, and in some cases, those have not selected at all.

To solve these problems, it has been decided to allocate an area around the robot 10×7 meters. This decision has significantly accelerated the decomposition algorithm so that the described problems have not affected the work of the local planner. This size of the submap allows for safe re-planning of the trajectory while moving through the environment.

5.2. Comparative numerical experiment on polygon decomposition

PolySAP initial decomposition is compared with the initial decomposition of the Interior Extension of Edges method (Nielsen et al., 2019). The reason of choice is that Interior Extension of Edges is relatively modern algorithm, which rely on numerical optimization for merging the initial set of polygons. The comparison was made on a MovingAI dataset (Sturtevant, 2012) for the planning tasks. Moving AI consist of various maze-like environments, including the environments based on real city maps.

Consider the simplest case from Fig. 7 for illustration. In this example, a rectangular area is shown with some protrusions that act as noise on a lidar map. The task is to divide the given polygon into the smallest number of polygons. The ideal option is that the figure should not break at all. Using the Interior Extension of Edges, this obstacle is broken by 12 dividing lines, and the primary decomposition includes a set of 18 polygons, which, following the merger step, form 7 polygons. At the same time, using our method, only 2 dividing lines are drawn,

[1] https://opencv.org/.
[2] https://www.cgal.org/.

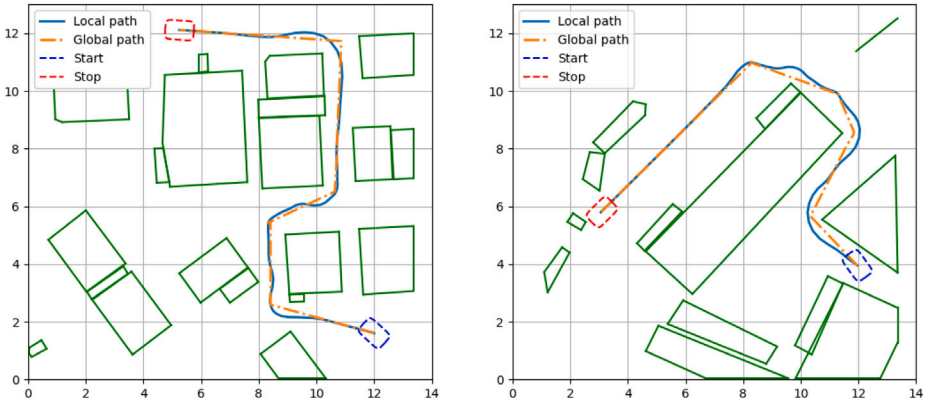


Fig. 11. Results of the MPC local planner for the two maps from the MovingAI 2D Pathfinding Benchmark.

Fig. 12. Comparison of decomposition methods. (a) Decomposition by the interior extension of edges. (b) Decomposition method based on the straight skeleton.

Table 1
Comparison of the convex decomposition using Interior Extension of Edges and PolySAP.

	Interior extension of edges	PolySAP
Δ	0%	11%
Runtime	5673 ± 471 ms	1062 ± 341 ms

forming 3 cells, which are not combined, as a result. Since the Interior Extension of Edges splits the workspace into convex polygons, their convex hulls will not increase the total area of the obstacles. Under the proposed method, convex hulls will increase the total area for this example by only 6%. An example of how these methods work is shown in Fig. 12. Examples of decomposition before and after the selection of convex hulls for the city maps are shown in Fig. 13. The last two figures are enlarged fragments of the same map.

In a set of experiments we have compared our method, PolySAP, to Interior Extensions of Edges according to the criterion Δ (see (1)) and we also compared the runtime (experiments have been executed on an Intel Core i3-7020U (2.30 GHz × 4) running Ubuntu). We provide the average Δ and the average running time for the entire MovingAI dataset in Table 1. Clearly PolySAP is 5 times faster (on average) than Interior Extensions of Edges. Its overhead in Δ is 11%. Indeed it higher than 0% (that corresponds to absolutely accurate decomposition), however this does not globally affect the path planning pipeline as the further experiments demonstrate.

Fig. 14 shows one example of how the decomposition provided by method differs from the one obtained by Interior Extensions of Edges. As can bee seen, our method splits the scene into much lower number of polygons. E.g. the top-row map in Fig. 14 contain 67 polygons for our method and 168 polygons for the competitor (the similar ratio was observed on the other maps).

5.3. Comparative numerical experiment on collision avoidance

We primarily compare our approach with two approximation-based MPC planners: CIAO by (Schoels et al., 2020a,b) and a simple computation of the distance to the nearest boundary point. These baselines are chosen, as they are intended for the MPC trajectory optimization with an arbitrary obstacle map. A newer approach by Thirugnanam et al. (2022) is less relevant: although it considers obstacle polygons, it requires a discrete-time process model, whereas our approach and CIAO-star allow for continuous-time models. In addition, the approach adopted by Thirugnanam et al. (2022) is relatively slow and does not consider the procedure of obtaining the polygon data. Some other works (Blackmore et al., 2011; Szmuk et al., 2017) also consider a given set of simple-shaped obstacles. CIAO can work with an arbitrary obstacle map. Among the versions of this algorithm, we choose original CIAO by Schoels et al. (2020a) as it aims to follow the global plan (further versions by Schoels et al. (2020b), contributes to rapid movement toward the goal configuration), and shows the best success rate (Schoels et al., 2020b). The original paper also provides results that outperform the GuSTO algorithm (Bonalli et al., 2019). Using Bench-mr framework, we also compare our method with different sampling and searching-based path planners, including Theta*(Daniel et al., 2010), RRT (LaValle and Kuffner, 2001), RRT*(Karaman and Frazzoli, 2010) and BFMT (Starek et al., 2014): this comparison seeks to highlight the difference between optimization-based and sampling- or searching-based path planners. We have made a comparison for ten planning cases from the MovingAI 2D Pathfinding Benchmark (Sturtevant, 2012) and for one case from the real environment (an artificial environment with two rooms and a small passage between them; the task is to move from one room to another; the polygons are shown in Fig. 9). Comparison results are given in Table 2. We use standard metrics calculated via the Bench-mr framework (Heiden et al., 2021): angle-over-length (AOL), maximum and normalized curvature, path length, planning time, success rate, and number of collisions. The first three metrics are designed to represent the smoothness of the generated trajectories (normalized curvature is measured along the path segments between instant turns, while AOL is measured along the entire path). AOL is calculated via dividing the total heading change by the path length (Heiden et al., 2021)

$$AOL = \frac{1}{l}\sum_{i=1}^{n-1}(|\theta_{i+1} - \theta_i|)$$

where θ_i is the orientation of the ith path segment, l is path length. Note that CIAO has not been able to plan a collision-free trajectory in two cases out of ten. This is due to the fact that the CIAO uses inequality constraints to provide collision avoidance and requires a collision-free initial guess (Schoels et al., 2020a). PolySAP and Nearest Point approximator were both able to converge to collision-free trajectories in all cases. It can be seen that PolySAP significantly outperforms CIAO and nearest point approximations in computation time, slightly outperforms them in path length, and edge them in curvature metrics. RRT, RRT*, BFMT and Theta* show much longer length due to non-optimal planning. Besides, Theta* demonstrates better AOL, while PolySAP shows better normalized curvature. The PolySAP planner provides a

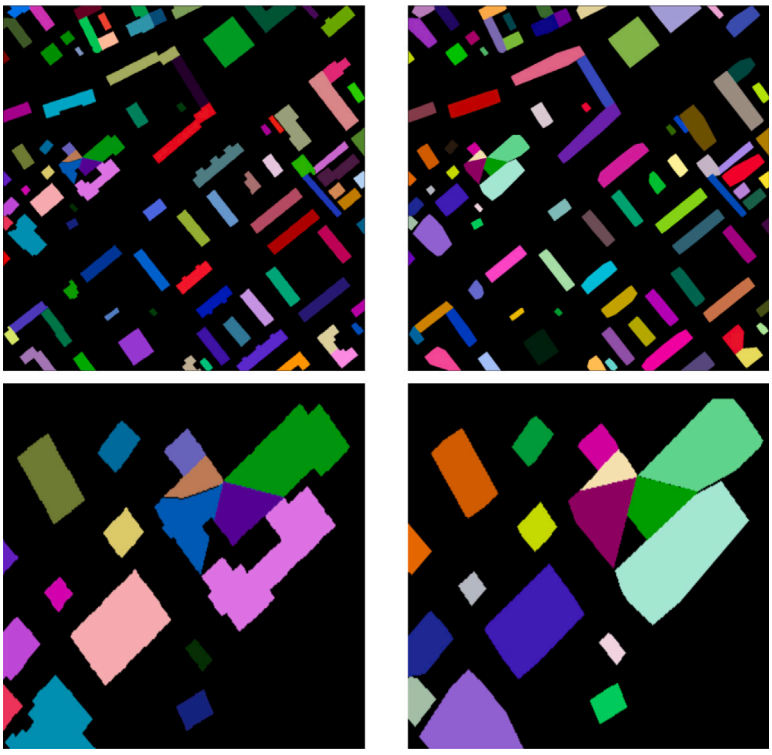


Fig. 13. Decomposition example for a BenchMR map: results of decomposition (left column) and convexification (right column). The lower row shows an enlarged fragment of the same map.

Table 2
Metrics comparison for PolySAP, CIAO, Nearest Point, Theta*, BFMT, RRT and RRT* on 10 testcase scenarios.

Metric	Theta*	BFMT	RRT	RRT*	CIAO	Nearest Point	**PolySAP (ours)—**
AOL	**0.313**	0.656	0.423	0.316	0.379	0.624	0.368
Maximum curvature	0.938	1.0	0.956	0.852	**0.757**	1.28	0.865
Normalized curvature	3.095	12.01	4.068	2.39	2.11	2.6	**1.634**
Path length (m)	10.79	23.81	12.15	9.99	7.012	7.07	**6.95**
Planning time (s)	0.11	1.855	0.0127	0.0157	0.049	0.017	**0.0034**
Success rate	**10/10**	8/10	**10/10**	**10/10**	8/10	**10/10**	**10/10**
Collisions	5/10	6/10	**0/10**	2/10	2/10	**0/10**	**0/10**

instant turn toward the destination point early into the motion, which is considered within AOL and ignored within the normalized curvature.

The visualizations of examples of optimized trajectories is given in Fig. 15 on the left. Significant difference of our approach comparing to CIAO is the capacity to handle some cases, when the global path is intersected by obstacles. Such case is shown in Fig. 15 on the right. CIAO has not been able to provide the path as it requires a collision-free initial guess. Fig. 16 shows the comparison of generated path between PolySAP, CIAO and other planners implemented in Bench-mr framework. The planned paths with the last three planners avoid the turns, while taking an unwarranted roundabout to reach the goal position.

5.4. Real robot experiment

As a proof of concept, we made a planning experiment on a real robot. We implement our planner as a Robotic Operation System package[3] and deploy it on the Husky UGV[4] mobile platform equipped with Velodyne VLP 16 LIDAR. RTAB-MAP SLAM by Labbé and Michaud (2019) is used for creating and updating the Occupancy Grid. The set of experiments were made within an experimental area in the office building at Moscow Institute of Physics and Technology. The entire map of the environment has a size of 114 by 17 m. A central part of the map is shown in Fig. 17. This is the most cluttered part of the map, while other parts include twisty wide corridors. In the experiments our platform was able to successfully navigate between various points of this environment. We use $\theta*$ algorithm by Nash et al. (2007) to obtain the global plan. Local planning is executed on a laptop with Intel Core i3-7020U CPU running Ubuntu. The laptop is connected to the robot via Ethernet. Replanning rate was set to 10 Hz, i.e. 100 ms for one planning operation. Sequential quadratic programming is an iterative optimization technique, therefore one can set the iteration limit in order to guarantee the required planning time. This was not necessary in our experiment: real order of the planning time is several milliseconds. CIAO* (Schoels et al., 2020a) and MPPI (Williams et al., 2016) applied for the similar environment requires hundreds of miliseconds to obtain the trajectory.

Fig. 18 and the supplementary video[5] show the most challenging task, when the robot is to move through the narrow passage between two rooms in an artificial indoor environment. This environment can be considered a simple maze. Its polygon decomposition is shown in Fig. 9. The robot passing through this environment can be seen in the and in Fig. 18. The top left cell shows the projection of the global path and

[3] https://wiki.ros.org/.
[4] https://opencv.org/.
[5] https://youtu.be/bb6M6Sgaqq8.

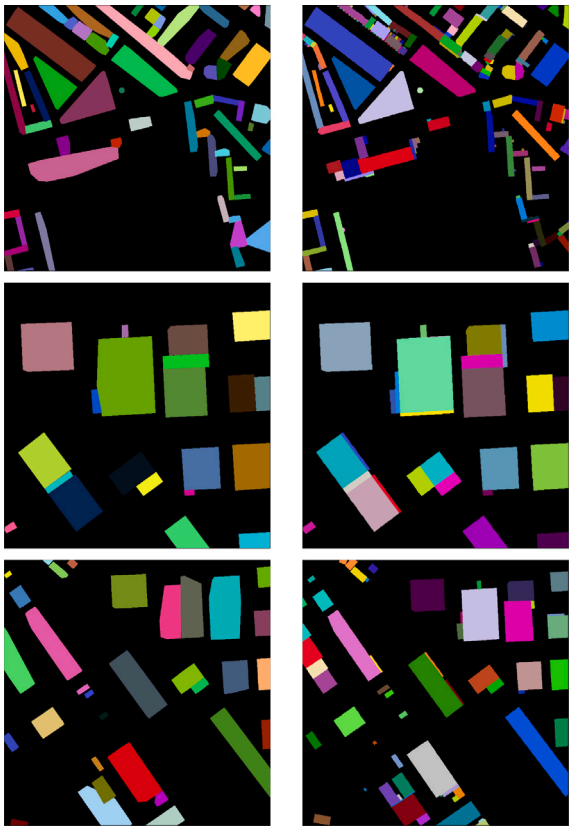


Fig. 14. Comparison of decomposition results for the interior extension of edges and PolySAP.

the local trajectory onto the occupancy grid. As is seen, a dangerous linear global path (it collides with the passage due to the robot size) is transformed into a smooth curve, which allows the robot to safely pass through the second room.

6. Discussion

In general, our approach is intended for local MPC trajectory planning for a mobile robot in a known 2D environment. Specifically, we consider scenarios where the exact obstacle map (occupancy grid) is provided by the sensor system and the SLAM algorithm. Our approach has been tested in environments where polygon decomposition is relevant, such as city maps and artificial indoor environments. We believe that polygon decomposition can be applied to many types of environments, including offices, storage facilities, mazes, and more. Most indoor environments, as well as many outdoor environments, consist of flat surfaces and can be effectively represented using polygons. Our approach may be less effective in environments with complex, "non-polygonal" features, such as those with numerous small obstacles. In such cases, polygon decomposition might be redundant and computationally expensive. Nonetheless, it still has the capability to produce accurate results.

We consider local planning as a model predictive control task. However, it is worth noting that generally, we use MPC as a framework for planning, not for direct control of the robotic system. We obtain an optimized trajectory and reference control inputs as a solution of the MPC task. These data are then submitted to the robot's low-level controller for execution. The direct use of generated control inputs is also an option, but it would be wise to combine it with a feedback control strategy. We use the standard view of the MPC statement (4) and propose a novel method for calculating one specific term of the cost function (4a). The system dynamics(4b) is out of scope here. We use a standard differential drive model as it corresponds to the dynamics of our real robot. However, it can be replaced with other suitable models of the mobile platforms, which meet the kinodynamic constraints.

6.1. Limitations

It is worth pointing out the following limitations of our approach:

- The proposed approach considers the robot with a circular footprint. We inflate the polygons with the radius of the robot and, therefore, can consider it pointwise. If the robot is oblong, one can resort to the approach explored in Ziegler et al. (2014). Under this approach, the robot footprint is covered with a set of circles, and the distance to obstacles is calculated separately for each circle.
- Measured computation times of the MPC solver suggest that re-planning with precomputed polygons is feasible at a rate of 100 Hz. The runtime of the polygon decomposition algorithm depends on the size and complexity of the map. It takes less than two seconds to process relatively large city maps from the MovingAI dataset. If the map is continuously updated, polygons can be recalculated while the robot is moving through a large environment. One set of polygons can be used for several dozen MPC re-plannings. In such cases, a sliding window around the robot can be employed to focus on a relatively small area for fast polygon decomposition. Our experiments use a 7 m × 10 m area around the robot.
- Under our method, obstacle avoidance is "localized" within the specific term of the cost function. Therefore, our approach can potentially be combined with other penalty-based approaches within the single MPC solver. E.g., PolySAP can be applied for avoiding obstacles from the static map together with additional terms for avoiding dynamic or previously unknown obstacles.

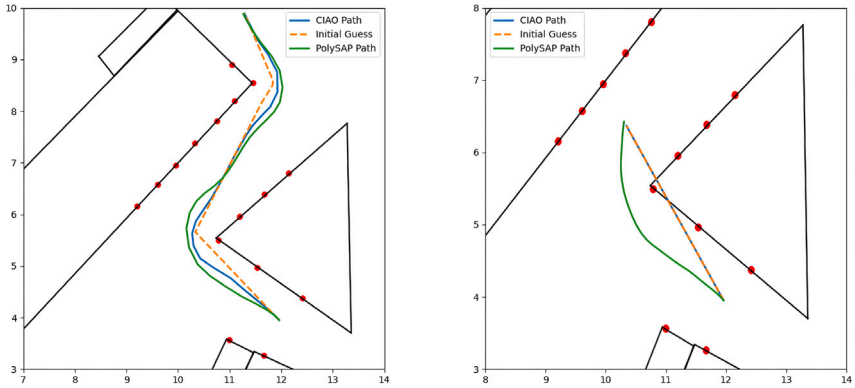


Fig. 15. Visualization of CIAO (blue) and PolySAP (green) trajectories on two scenarios. Orange dash correspond to the global path.

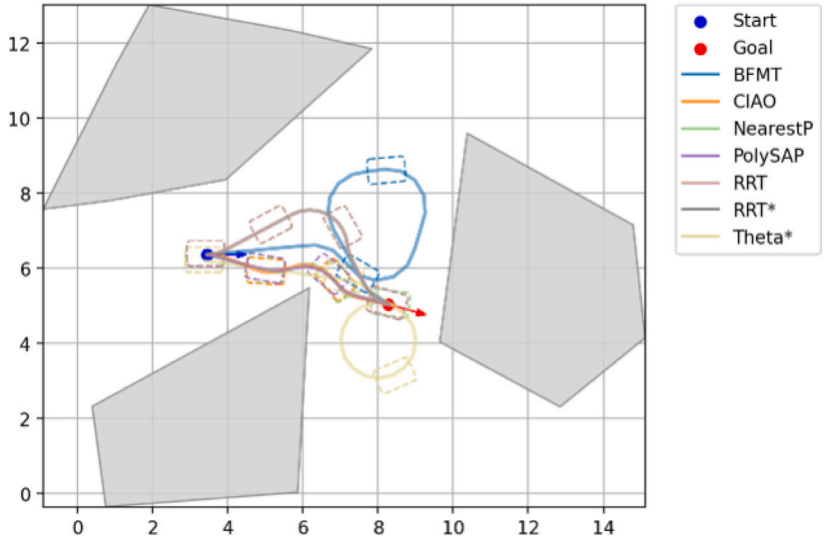


Fig. 16. One scenario for comparison of PolySAP, CIAO and different path planners in Bench-mr framework.

Fig. 17. Occupancy Grid map of the environment for real robot experiments. Artificial narrow passages may be seen in the top left area.

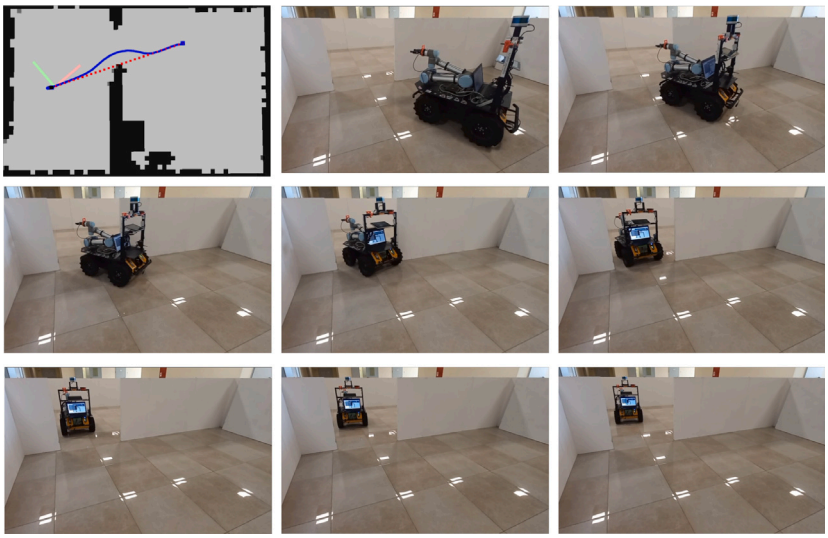


Fig. 18. Real robot moving through the narrow passage with PolySAP local planner.

- Our current algorithm considers an obstacle map to be static within the trajectory prediction horizon. For the general case of the dynamic environment, the procedure of obstacle decomposition has to be repeated for each instantaneous map at timesteps $k...k+m$. The setting of the dynamic environment is often more specific. We have a given static map and a given set of dynamic obstacles with predicted trajectories. One specific case of this setting is when dynamic obstacles may relate to other autonomous

agents in the room (Franze and Lucia, 2015). With such a setting, we can use our approach to avoid static obstacles with an addition of specific J term to dodge dynamic obstacles.

- Our approach is intended for local planning and requires an initial guess provided by the global planner. This initial guess can be a rough estimate. In some cases, it may even intersect some polygons, and the PolySAP will provide a safe trajectory (see Fig. 15, right part). The optimizer will push the trajectory toward the *nearest* border of the polygon, which may lead to an incorrect or a non-optimal solution when the initial guess goes through the polygon center.

7. Conclusion

In this paper, we propose PolySAP, a novel approach for incorporating obstacle avoidance into the Model Predictive Control (MPC) task statement. This approach is based on representing obstacles as a set of convex polygons. We introduce a new algorithm for obstacle segmentation and a novel artificial potential function that repels the trajectory from the obstacles. Our polygon decomposition algorithm operates five times faster than the baseline method of Interior Extension of Edges. Experiments using planning datasets and a real mobile robot demonstrate that PolySAP can generate smooth and safe trajectories from rough global paths. Compared to the baseline CIAO MPC local planner, our approach delivers a significantly faster solution with only a minor increase in path length. Unlike CIAO, PolySAP can provide correct trajectories even when the global plan touches or slightly intersects obstacles. Overall, our approach offers fast, collision-aware trajectory optimization by representing environments as a set of convex polygons. It is a promising technique for improving rough and unsafe trajectories of indoor mobile robots.

We believe that a promising direction of future work is to combine PolySAP local planner with the global planner based on route decomposition. A relevant example of such method is the Visibility Graph (VG) global planner (Lozano-Pérez and Wesley, 1979). Here the map is represented as a set of polygons. A visibility graph connects mutually visible vertices of these polygons, and the shortest path can be found via routing within this graph. A VG planner provides fast but unsafe global plans, which touch the polygons' borders. Another specific downside of VG is high complexity of its formation for large terrains. Indoor environments are typically relatively compact. The combination of PolySAP and VG looks promising for the following reasons. First, VG exploits the polygon decomposition of the space. Second, our MPC solver can turn unsafe VG plans into a safe and executable trajectories. Therefore, we can compensate for the drawbacks of the two approaches (the unsafety of VG and local convergence of MPC) by combining them.

CRediT authorship contribution statement

Aleksey Logunov: Software, Investigation. **Muhammad Alhaddad:** Software, Investigation, Data curation. **Konstantin Mironov:** Methodology, Investigation, Conceptualization. **Konstantin Yakovlev:** Formal analysis, Conceptualization. **Aleksandr Panov:** Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT in order to improve formulations in the text. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Funding

This work was supported by the Ministry of Science and Higher Education of the Russian Federation under Project 075-15-2024-544.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Aleksandr Panov reports financial support was provided by Analytical Center for the Government of the Russian Federation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

Acar, Ercan U., Choset, Howie, Rizzi, Alfred A., Atkar, Prasad N., Hull, Douglas, 2002. Morse decompositions for coverage tasks. Int. J. Robot. Res. 21 (4), 331–344. http://dx.doi.org/10.1177/027836402320556359.

Adamkiewicz, Michal, Chen, Timothy, Caccavale, Adam, Gardner, Rachel, Culbertson, Preston, Bohg, Jeannette, Schwager, Mac, 2022. Vision-only robot navigation in a neural radiance world. IEEE Robot. Autom. Lett. 7 (2), 4606–4613. http://dx.doi.org/10.1109/LRA.2022.3150497.

Aichholzer, Oswin, Aurenhammer, Franz, 1995. A novel type of skeleton for polygons. J. Univers. Comput. Sci. 1, 752–761. http://dx.doi.org/10.1007/978-3-642-80350-5_65.

Alhaddad, Muhammad, Mironov, Konstantin, Staroverov, Aleksey, Panov, Aleksandr, 2024. Neural potential field for obstacle-aware local motion planning. In: 2024 IEEE International Conference on Robotics and Automation. ICRA, pp. 9313–9320. http://dx.doi.org/10.1109/ICRA57147.2024.10611635.

Blackmore, Lars, Ono, Masahiro, Williams, Brian C., 2011. Chance-constrained optimal path planning with obstacles. IEEE Trans. Robot. 27 (6), 1080–1094. http://dx.doi.org/10.1109/TRO.2011.2161160.

Bojadžić, Damir, Kunze, Julian, Osmanković, Dinko, Malmir, Mohammadhossein, Knoll, Alois, 2021. Non-holonomic RRT & MPC: Path and trajectory planning for an autonomous cycle rickshaw. http://dx.doi.org/10.48550/arXiv.2103.06141, arXiv preprint arXiv:2103.06141.

Bonalli, Riccardo, Cauligi, Abhishek, Bylard, Andrew, Pavone, Marco, 2019. GuSTO: Guaranteed sequential trajectory optimization via sequential convex programming. In: 2019 International Conference on Robotics and Automation. ICRA, pp. 6741–6747. http://dx.doi.org/10.1109/ICRA.2019.8794205.

Bormann, Richard, Jordan, Florian, Li, Wenzhe, Hampp, Joshua, Hägele, Martin, 2016. Room segmentation: Survey, implementation, and analysis. In: 2016 IEEE International Conference on Robotics and Automation. ICRA, pp. 1019–1026. http://dx.doi.org/10.1109/ICRA.2016.7487234.

BrainCorp, 2023a. Automated inventory management, BrainCorp. https://braincorp.com/industries/commercial-offices/.

BrainCorp, 2023b. Commercial offices, BrainCorp. https://braincorp.com/applications/shelf-scanning/.

Brown, Stanley, 2017. Coverage path planning and room segmentation in indoor environments using the constriction decomposition method. In: A thesis presented to the University of Waterloo in fulfillment of the thesis requirement for the degree of Master of Applied Science in Mechanical and Mechatronics Engineering.

Butzke, Jonathan, Sapkota, Krishna, Prasad, Kush, MacAllister, Brian, Likhachev, Maxim, 2014. State lattice with controllers: Augmenting lattice-based path planning with controller-based motion primitives. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 258–265.

Choset, Howie, Pignon, Philippe, 1998. Coverage path planning: The boustrophedon cellular decomposition. Field Serv. Robot. 203–209. http://dx.doi.org/10.1007/978-1-4471-1273-0_32.

Daftry, Shreyansh, Abcouwer, Neil, Del Sesto, Tyler, Venkatraman, Siddarth, Song, Jialin, Igel, Lucas, Byon, Amos, Rosolia, Ugo, Yue, Yisong, Ono, Masahiro, 2022. Mlnav: Learning to safely navigate on martian terrains. IEEE Robot. Autom. Lett. 7 (2), 5461–5468. http://dx.doi.org/10.1109/LRA.2022.3156654.

Daniel, Kenny, Nash, Alex, Koenig, Sven, Felner, Ariel, 2010. Theta*: Any-angle path planning on grids. J. Artificial Intelligence Res. 39, 533–579.

Dijkstra, Edsger W., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271. http://dx.doi.org/10.1145/3544585.3544600.

Ding, Hui, 2020. Motion path planning of soccer training auxiliary robot based on genetic algorithm in fixed-point rotation environment. J. Ambient. Intell. Humaniz. Comput. 11 (12), 6261–6270. http://dx.doi.org/10.1007/s12652-020-01877-4.

Douglas, David, Peucker, Thomas, 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Can. Cartogr. 10 (2), 112–122. http://dx.doi.org/10.3138/FM57-6770-U75U-7727.

Erdmann, M., Lozano-Perez, T., 1986. On multiple moving objects. In: Proceedings. 1986 IEEE International Conference on Robotics and Automation. vol. 3, pp. 1419–1424. http://dx.doi.org/10.1109/ROBOT.1986.1087401.

Franze, Giuseppe, Lucia, Walter, 2015. A receding horizon control strategy for autonomous vehicles in dynamic environments. IEEE Trans. Control Syst. Technol. 24 (2), 695–702. http://dx.doi.org/10.1109/TCST.2015.2440999.

Gammell, Jonathan D., Strub, Marlin P., 2021. Asymptotically optimal sampling-based motion planning methods. Annu. Rev. Control. Robot. Auton. Syst. 4, 295–318. http://dx.doi.org/10.1146/annurev-control-061920-093753.

Gilbert, Elmer G., Johnson, Daniel W., Keerthi, S. Sathiya, 1988. A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE J. Robot. Autom. 4 (2), 193–203. http://dx.doi.org/10.1109/56.2083.

González, David, Pérez, Joshué, Milanés, Vicente, Nashashibi, Fawzi, 2016. A review of motion planning techniques for automated vehicles. IEEE Trans. Intell. Transp. Syst. 17 (4), 1135–1145. http://dx.doi.org/10.1109/TITS.2015.2498841.

Hart, Peter E., Nilsson, Nils J., Raphael, Bertram, 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4 (2), 100–107. http://dx.doi.org/10.1109/TSSC.1968.300136.

Heiden, Eric, Palmieri, Luigi, Arras, Kai O, Sukhatme, Gaurav S, Koenig, Sven, 2020. Experimental comparison of global motion planning algorithms for wheeled mobile robots. http://dx.doi.org/10.48550/arXiv.2003.03543, arXiv preprint arXiv:2003.03543.

Heiden, Eric, Palmieri, Luigi, Bruns, Leonard, Arras, Kai O., Sukhatme, Gaurav S., Koenig, Sven, 2021. Bench-MR: A motion planning benchmark for wheeled mobile robots. IEEE Robot. Autom. Lett. 6 (3), 4536–4543. http://dx.doi.org/10.1109/LRA.2021.3068913.

Houska, B., Ferreau, H.J., Diehl, M., 2011a. ACADO toolkit – an open source framework for automatic control and dynamic optimization. Optim. Control. Appl. Methods 32 (3), 298–312. http://dx.doi.org/10.1002/oca.939.

Houska, B., Ferreau, H.J., Diehl, M., 2011b. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. Automatica 47 (10), 2279–2285. http://dx.doi.org/10.1016/j.automatica.2011.08.020.

Huang, W.H., 2001. Optimal line-sweep-based decompositions for coverage algorithms. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164). vol. 1, pp. 27–32 vol.1. http://dx.doi.org/10.1109/ROBOT.2001.932525.

Ji, Jie, Khajepour, Amir, Melek, Wael William, Huang, Yanjun, 2016. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. IEEE Trans. Veh. Technol. 66 (2), 952–964. http://dx.doi.org/10.1109/TVT.2016.2555853.

Jian, Zhiqiang, Zhang, Songyi, Chen, Shitao, Nan, Zhixiong, Zheng, Nanning, 2021. A global-local coupling two-stage path planning method for mobile robots. IEEE Robot. Autom. Lett. 6 (3), 5349–5356. http://dx.doi.org/10.1109/LRA.2021.3074878.

Kalakrishnan, Mrinal, Chitta, Sachin, Theodorou, Evangelos, Pastor, Peter, Schaal, Stefan, 2011. STOMP: Stochastic trajectory optimization for motion planning. pp. 4569–4574. http://dx.doi.org/10.1109/ICRA.2011.5980280,

Karaman, Sertac, Frazzoli, Emilio, 2010. Optimal kinodynamic motion planning using incremental sampling-based methods. In: 49th IEEE Conference on Decision and Control. CDC, IEEE, pp. 7681–7687.

Katerishich, Maksim, Kurenkov, Mikhail, Karaf, Sausar, Nenashev, Artem, Tsetserukou, Dzmitry, 2023. DNFOMP: Dynamic neural field optimal motion planner for navigation of autonomous robots in cluttered environment. In: 2023 IEEE International Conference on Systems, Man, and Cybernetics. SMC, IEEE, pp. 1984–1989. http://dx.doi.org/10.1109/SMC53992.2023.10394025.

Kavraki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.H., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. 12 (4), 566–580. http://dx.doi.org/10.1109/70.508439.

Kayacan, Erkan, Chowdhary, Girish, 2019. Tracking error learning control for precise mobile robot path tracking in outdoor environment. J. Intell. Robot. Syst. 95, 975–986. http://dx.doi.org/10.1007/s10846-018-0916-3.

Khatib, O., 1985. Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings. 1985 IEEE International Conference on Robotics and Automation. vol. 2, pp. 500–505. http://dx.doi.org/10.1109/ROBOT.1985.1087247.

Kim, Dong Hun, Shin, Seiichi, 2006. Local path planning using a new artificial potential function composition and its analytical design guidelines. Adv. Robot. 20 (1), 115–135. http://dx.doi.org/10.1163/156855306775275530.

Kurenkov, Mikhail, Potapov, Andrei, Savinykh, Alena, Yudin, Evgeny, Kruzhkov, Evgeny, Karpyshev, Pavel, Tsetserukou, Dzmitry, 2022. NFOMP: Neural field for optimal motion planner of differential drive robots with nonholonomic constraints. IEEE Robot. Autom. Lett. 7 (4), 10991–10998. http://dx.doi.org/10.1109/LRA.2022.3196886.

Labbé, Mathieu, Michaud, François, 2019. RTAB-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. J. Field Robot. 36 (2), 416–446. http://dx.doi.org/10.1002/rob.21831.

Latombe, J.C., 1991. Robot motion planning. Kluwer Acad. Publ. Boston.

LaValle, Steven M., Kuffner, Jr., James J., 2001. Randomized kinodynamic planning. Int. J. Robot. Res. 20 (5), 378–400. http://dx.doi.org/10.1177/02783640122067453.

Li, Xinming, 2020. Robot target localization and interactive multi-mode motion trajectory tracking based on adaptive iterative learning. J. Ambient. Intell. Humaniz. Comput. 11 (12), 6271–6282. http://dx.doi.org/10.1007/s12652-020-01878-3.

Li, Y., Chen, H., Joo Er, M., Wang, X., 2011. Coverage path planning for UAVs based on enhanced exact cellular decomposition method. Mechatronics 21 (5), 876–885. http://dx.doi.org/10.1016/j.mechatronics.2010.10.009.

Li, Linjun, Miao, Yinglong, Qureshi, Ahmed H, Yip, Michael C, 2021. Mpc-mpnet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints. IEEE Robot. Autom. Lett. 6 (3), 4496–4503. http://dx.doi.org/10.1109/LRA.2021.3067847.

Li, Zhaoying, Zhang, Zhao, Liu, Hao, Yang, Liang, 2020. A new path planning method based on concave polygon convex decomposition and artificial bee colony algorithm. Int. J. Adv. Robot. Syst. 17 (1), http://dx.doi.org/10.1177/1729881419894787.

Lozano-Pérez, Tomás, Wesley, Michael A., 1979. An algorithm for planning collision-free paths among polyhedral obstacles. Commun. ACM 22 (10), 560–570.

Luis, Carlos E., Vukosavljev, Marijan, Schoellig, Angela P., 2020. Online trajectory generation with distributed model predictive control for multi-robot motion planning. IEEE Robot. Autom. Lett. 5 (2), 604–611. http://dx.doi.org/10.1109/LRA.2020.2964159.

Mohamed, Ihab S., Allibert, Guillaume, Martinet, Philippe, 2020. Model predictive path integral control framework for partially observable navigation: A quadrotor case study. In: 2020 16th International Conference on Control, Automation, Robotics and Vision. ICARCV, IEEE, pp. 196–203. http://dx.doi.org/10.1109/ICARCV50220.2020.9305363.

Morato, Marcelo Menezes, Bernardi, Emanuel, Stojanović, Vladimir, 2021. A qLPV nonlinear model predictive control with moving horizon estimation. Complex Eng. Syst. 1 (5), 1–24. http://dx.doi.org/10.20517/ces.2021.09.

MOXI, 2023. Diligent robotics. https://www.diligentrobots.com/.

Nash, Alex, Daniel, Kenny, Koenig, Sven, Felner, Ariel, 2007. Theta*: Any-angle path planning on grids. In: AAAI. vol. 7, pp. 1177–1183.

Nielsen, Lasse Damtoft, Sung, Inkyung, Nielsen, Peter, 2019. Convex decomposition for a coverage path planning for autonomous vehicles: Interior extension of edges. Sensors 19 (19), http://dx.doi.org/10.3390/s19194165.

Panov, Aleksandr I., 2019. Goal setting and behavior planning for cognitive agents. Sci. Tech. Inf. Process. 46 (6), 404–415. http://dx.doi.org/10.3103/S0147688219060066, URL: https://link.springer.com/article/10.3103/S0147688219060066.

Papaioannou, Savvas, Kolios, Panayiotis, Theocharides, Theocharis, Panayiotou, Christos G, Polycarpou, Marios M, 2023. Distributed search planning in 3-d environments with a dynamically varying number of agents. IEEE Trans. Syst. Man Cybern.: Syst. 53 (7), 4117–4130. http://dx.doi.org/10.1109/TSMC.2023.3240023.

Parikh, Priyam, Trivedi, Reena, Dave, Jatin, Joshi, Keyur, Adhyaru, Dipak, 2023. Design and development of a low-cost vision-based 6 dof assistive feeding robot for the aged and specially-abled people. IETE J. Res. 70 (2), 1716–1744. http://dx.doi.org/10.1080/03772063.2023.2173665.

Preparata, Franco P., Shamos, Michael Ian, 1989. Computational Geometry: an Introduction. Springer-Verlag, New York.

Ratliff, Nathan, Zucker, Matt, Bagnell, J. Andrew, Srinivasa, Siddhartha, 2009. CHOMP: Gradient optimization techniques for efficient motion planning. In: 2009 IEEE International Conference on Robotics and Automation. pp. 489–494. http://dx.doi.org/10.1109/ROBOT.2009.5152817.

Ren, Jing, McIsaac, K.A., Patel, R.V., 2006. Modified Newton's method applied to potential field-based navigation for mobile robots. IEEE Trans. Robot. 22 (2), 384–391. http://dx.doi.org/10.1109/TRO.2006.870668.

Salzmann, Tim, Arrizabalaga, Jon, Andersson, Joel, Pavone, Marco, Ryll, Markus, 2024. Learning for CasADi: Data-driven models in numerical optimization. In: Abate, Alessandro, Cannon, Mark, Margellos, Kostas, Papachristodoulou, Antonis (Eds.), Proceedings of the 6th Annual Learning for Dynamics & Control Conference. In: Proceedings of Machine Learning Research, vol. 242, PMLR, pp. 541–553.

Schoels, Tobias, Palmieri, Luigi, Arras, Kai O., Diehl, Moritz, 2020a. An NMPC approach using convex inner approximations for online motion planning with guaranteed collision avoidance. In: 2020 IEEE International Conference on Robotics and Automation. ICRA, pp. 3574–3580. http://dx.doi.org/10.1109/ICRA40945.2020.9197206.

Schoels, Tobias, Rutquist, Per, Palmieri, Luigi, Zanelli, Andrea, Arras, Kai O., Diehl, Moritz, 2020b. CIAO*: MPC-based safe motion planning in predictable dynamic environments. IFAC- Pap. (ISSN: 2405-8963) 53 (2), 6555–6562. http://dx.doi.org/10.1016/j.ifacol.2020.12.072, 21st IFAC World Congress. URL: https://www.sciencedirect.com/science/article/pii/S2405896320303281.

Schulman, John, Duan, Yan, Ho, Jonathan, Lee, Alex X., Awwal, Ibrahim, Bradlow, Henry, Pan, Jia, Patil, Sachin, Goldberg, Ken, Abbeel, P., 2014. Motion planning with sequential convex optimization and convex collision checking. Int. J. Robot. Res. 33 (9), 1251–1270. http://dx.doi.org/10.1177/0278364914528132.

Song, Xiaona, Sun, Peng, Song, Shuai, Stojanovic, Vladimir, 2023. Quantized neural adaptive finite-time preassigned performance control for interconnected nonlinear systems. Neural Comput. Appl. 35 (21), 15429–15446. http://dx.doi.org/10.1007/s00521-023-08361-y.

Starek, J., Schmerling, Edward, Janson, Lucas, Pavone, Marco, 2014. Bidirectional fast marching trees: An optimal sampling-based algorithm for bidirectional motion planning. In: Workshop on Algorithmic Foundations of Robotics.

Stoican, Florin, Prodan, Ionela, Grøtli, Esten Ingar, Nguyen, Ngoc Thinh, 2019. Inspection trajectory planning for 3d structures under a mixed-integer framework. In: 2019 IEEE 15th International Conference on Control and Automation. ICCA, IEEE, pp. 1349–1354. http://dx.doi.org/10.1109/ICCA.2019.8899514.

Sturtevant, N., 2012. Benchmarks for grid-based pathfinding. Trans. Comput. Intell. AI Games 4 (2), 144–148. http://dx.doi.org/10.1109/TCIAIG.2012.2197681, URL: http://web.cs.du.edu/~sturtevant/papers/benchmarks.pdf.

Szczepanski, Rafal, Tarczewski, Tomasz, Erwinski, Krystian, 2022. Energy efficient local path planning algorithm based on predictive artificial potential field. IEEE Access 10, 39729–39742. http://dx.doi.org/10.1109/ACCESS.2022.3166632.

Szmuk, Michael, Pascucci, Carlo Alberto, Dueri, Daniel, Açikmeşe, Behcet, 2017. Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 4862–4868. http://dx.doi.org/10.1109/IROS.2017.8206363.

Szot, Andrew, Clegg, Alexander, Undersander, Eric, Wijmans, Erik, Zhao, Yili, Turner, John, Maestre, Noah, Mukadam, Mustafa, Chaplot, Devendra Singh, Maksymets, Oleksandr, et al., 2021. Habitat 2.0: Training home assistants to rearrange their habitat. Adv. Neural Inf. Process. Syst. 34, 251–266.

Tang, Gang, Tang, Congqiang, Zhou, Hao, Claramunt, Christophe, Men, Shaoyang, 2021. R-DFS: A coverage path planning approach based on region optimal decomposition. Remote. Sens. 13 (8), http://dx.doi.org/10.3390/rs13081525.

Thirugnanam, Akshay, Zeng, Jun, Sreenath, Koushil, 2022. Safety-critical control and planning for obstacle avoidance between polytopes with control barrier functions. In: 2022 International Conference on Robotics and Automation. ICRA, pp. 286–292. http://dx.doi.org/10.1109/ICRA46639.2022.9812334.

Thrun, S., 1998. Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence 99 (1), 21–71. http://dx.doi.org/10.1016/S0004-3702(97)00078-7.

Tong, Chunyu, 2020. Three-dimensional reconstruction of the dribble track of soccer robot based on heterogeneous binocular vision. J. Ambient. Intell. Humaniz. Comput. 11 (12), 6361–6372. http://dx.doi.org/10.1007/s12652-020-02039-2.

Verschueren, Robin, Frison, Gianluca, Kouzoupis, Dimitris, Frey, Jonathan, Duijkeren, Niels van, Zanelli, Andrea, Novoselnik, Branimir, Albin, Thivaharan, Quirynen, Rien, Diehl, Moritz, 2022. Acados—a modular open-source framework for fast embedded optimal control. Math. Program. Comput. 14 (1), 147–183. http://dx.doi.org/10.1007/s12532-021-00208-8.

Vogel, Jorn, Leidner, Daniel, Hagengruber, Annette, Panzirsch, Michael, Bauml, Berthold, Denninger, Maximilian, Hillenbrand, Ulrich, Suchenwirth, Lioba, Schmaus, Peter, Sewtz, Marco, Bauer, Adrian Simon, Hulin, Thomas, Iskandar, Maged, Quere, Gabriel, Albu-Schaffer, Alin, Dietrich, Alexander, 2021. An ecosystem for heterogeneous robotic assistants in caregiving: Core functionalities and use cases. IEEE Robot. Autom. Mag. 28 (3), 12–28. http://dx.doi.org/10.1109/MRA.2020.3032142.

Waechter, A., Biegler, L.T., 2005–2022. IPOPT. https://github.com/coin-or/Ipopt.

Williams, Grady, Drews, Paul, Goldfain, Brian, Rehg, James M, Theodorou, Evangelos A, 2016. Aggressive driving with model predictive path integral control. In: 2016 IEEE International Conference on Robotics and Automation. ICRA, pp. 1433–1440. http://dx.doi.org/10.1109/ICRA.2016.7487277.

Williams, Grady, Wagener, Nolan, Goldfain, Brian, Drews, Paul, Rehg, James M, Boots, Byron, Theodorou, Evangelos A, 2017. Information theoretic MPC for model-based reinforcement learning. In: 2017 IEEE International Conference on Robotics and Automation. ICRA, IEEE, pp. 1714–1721. http://dx.doi.org/10.1109/ICRA.2017.7989202.

Wu, YuXuan, Wang, Jing, Zhou, Meng, Dong, Zhe, Chen, YangQuan, 2021. Air-ground cooperative exploration of 3D complex environment with maximized visibility and obstacles avoidance. In: 2021 International Conference on Unmanned Aircraft Systems. ICUAS, IEEE, pp. 1416–1421. http://dx.doi.org/10.1109/ICUAS51884.2021.9476714.

Yang, Fan, Cao, Chao, Zhu, Hongbiao, Oh, Jean, Zhang, Ji, 2022. FAR planner: Fast, attemptable route planner using dynamic visibility update. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 9–16. http://dx.doi.org/10.1109/IROS47612.2022.9981574.

You, Wenwen, Xie, Xiangpeng, Wang, Hui, Xia, Jianwei, Stojanovic, Vladimir, 2024. Relaxed model predictive control of TS fuzzy systems via a new switching-type homogeneous polynomial technique. IEEE Trans. Fuzzy Syst. 32 (8), http://dx.doi.org/10.1109/TFUZZ.2024.3405078.

Zanelli, A., Domahidi, A., Jerez, J., Morari, M., 2017. FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. Internat. J. Control 93 (1), 13–29. http://dx.doi.org/10.1080/00207179.2017.1316017.

Zeng, Jun, Zhang, Bike, Sreenath, Koushil, 2021. Safety-critical model predictive control with discrete-time control barrier function. In: 2021 American Control Conference. ACC, IEEE, pp. 3882–3889. http://dx.doi.org/10.23919/ACC50511.2021.9483029.

Zhang, Zeqing, Zhang, Yinqiang, Han, Ruihua, Zhang, Liangjun, Pan, Jia, 2022. A generalized continuous collision detection framework of polynomial trajectory for mobile robots in cluttered environments. IEEE Robot. Autom. Lett. 7 (4), 9810–9817. http://dx.doi.org/10.1109/LRA.2022.3191934.

Ziegler, Julius, Bender, Philipp, Dang, Thao, Stiller, Christoph, 2014. Trajectory planning for bertha — A local, continuous method. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings. pp. 450–457. http://dx.doi.org/10.1109/IVS.2014.6856581.

Zimmermann, Simon, Busenhart, Matthias, Huber, Simon, Poranne, Roi, Coros, Stelian, 2022. Differentiable collision avoidance using collision primitives. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 8086–8093. http://dx.doi.org/10.1109/IROS47612.2022.9981093.

Zuo, Zhiqiang, Yang, Xu, Li, Zheng, Wang, Yijing, Han, Qiaoni, Wang, Li, Luo, Xiaoyuan, 2020. MPC-based cooperative control strategy of path planning and trajectory tracking for intelligent vehicles. IEEE Trans. Intell. Veh. 6 (3), 513–522. http://dx.doi.org/10.1109/TIV.2020.3045837.