Globally Guided Trajectory Planning in Dynamic Environments

de Groot, O.M.; Ferranti, L.; Gavrila, D.; Alonso-Mora, J.

DOI
10.1109/ICRA48891.2023.10160379

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2023)

2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Globally Guided Trajectory Planning in Dynamic Environments

Oscar de Groot, Laura Ferranti, Dariu Gavrila, Javier Alonso-Mora

***Abstract*— Navigating mobile robots through environments shared with humans is challenging. From the perspective of the robot, humans are dynamic obstacles that must be avoided. These obstacles make the collision-free space nonconvex, which leads to two distinct passing behaviors per obstacle (passing left or right). For local planners, such as receding-horizon trajectory optimization, each behavior presents a local optimum in which the planner can get stuck. This may result in slow or unsafe motion even when a better plan exists. In this work, we identify trajectories for multiple locally optimal driving behaviors, by considering their topology. This identification is made consistent over successive iterations by propagating the topology information. The most suitable high-level trajectory guides a local optimization-based planner, resulting in fast and safe motion plans. We validate the proposed planner on a mobile robot in simulation and real-world experiments.**

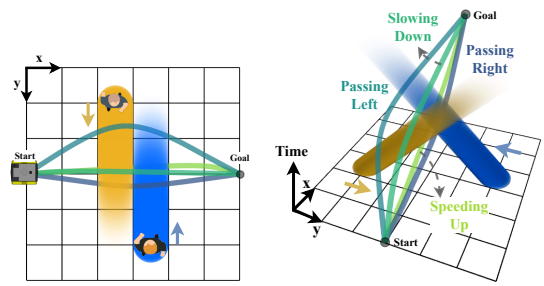
I. Introduction

Mobile robots have the potential to automate logistic tasks ranging from indoor transportation tasks, as found in automated warehouses and hospitals, to outdoor transportation tasks, such as package delivery. One of the major challenges for mobile robots is to move safely among humans.

Dynamic collision avoidance constraints are usually imposed on the motion of the robot, making its free configuration space nonconvex. In fact, each obstacle in a 2-D environment leads to at least two possible driving behaviors for the robot: passing left or right.

Existing motion planners are either *local* or *global*. Local planners typically remain in the driving behavior that they are initialized with. This may lead to poor performance (e.g. long travel times) if a higher performance driving behavior exists, but is not explored. Because of the limited scope, these planners are typically fast and can consider detailed dynamic models. Global planners explore and find the optimal driving behavior and motion plan, but can suffer from high computation times when the robot dynamical model is considered.

A common approach in the presence of static obstacles is to find a high-level path using a global planner. This path is passed to a local motion planner which improves the quality of the plan locally. The majority of existing works following this hierarchy do not consider dynamic obstacles in their global planner, they delegate dynamic collision avoidance to the local planner directly. Although this is computationally faster, it fails to address that when obstacles move, the set of driving behaviors becomes richer, since the planner needs to decide *when* and *how* to pass the obstacles. Fig. 1 illustrates these driving behaviors by looking at the planning problem in a 3-D state space consisting of 2-D position and bounded time. Each driving behavior travels through a distinct section of the state space and can therefore be identified by analyzing their topology in the collision-free space.



(a) Topview of a crossing scenario.

(b) Topologically distinct trajectories in the 3-D state space.

Fig. 1: An illustration of how topologically distinct guidance trajectories explore driving behaviors for a scenario where two pedestrians cross at the same time.

In this work we explicitly consider the possible driving behaviors in a dynamic environment by planning topologically distinct guidance trajectories in the state space. We select the most suitable guidance trajectory based on a high-level cost. This trajectory is passed to a Model Predictive Contouring Controller (MPCC) [1] that locally optimizes the motion of the robot while following the selected guidance trajectory, resulting in a fast and safe motion plan.

Related Work: Local optimization can be leveraged to plan locally optimal trajectories. Model Predictive Control (MPC) is often used to optimize planning performance (e.g., speed and comfort) while satisfying constraints (e.g., collision avoidance, vehicle model, actuator limits) offering a flexible and safe framework that can include road following [2] and dynamic collision avoidance in the deterministic [1] and uncertain [3], [4], [5] case. A limitation of MPC motion planners is that their solution is only guaranteed to be locally optimal, which can lead to unsafe or unexpected driving behavior when the local optimum corresponds to unsuitable (e.g., aggressive or slow) driving behavior.

Global planners such as Rapidly expanding Random Trees (RRT) [6], RRT* [7] and Probabilistic RoadMaps (PRM) [8] in principle resolve this issue, but are typically not fast enough to consider dynamic obstacles, especially when the robot's dynamic constraints are considered.

Topology-based planning methods identify driving behav-

The authors are with the Dept. of Cognitive Robotics, TU Delft, 2628 CD Delft, The Netherlands. Email: o.m.degroot@tudelft.nl

This work received support from the Dutch Science Foundation NWO-TTW, within the SafeVRU project (nr. 14667) and Veni project HARMONIA (nr. 18165).

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10160379

iors through the environment by comparing trajectory topologies. When two trajectories can be smoothly transformed into each other without colliding with an obstacle, they are *homotopy* equivalent. In [9], methods using this measure are divided in three groups. The first group plans in a given homotopy class, for example, road rules are used in [10] to motivate a desired homotopy class in highway driving.

The second group leverages structure in the environment to enumerate possible homotopy classes after which trajectories in a subset of the classes are computed. An example of this approach is [11] where a 2-D workspace is decomposed with a trapezoidal decomposition. Similarly, in [12], [13] homotopy classes are derived from the road structure. The works [14], [15] compute a homology [1] invariant that for each obstacle identifies the rotations of a path around it. Graphs extended with this invariant can be used to plan a path in each homology class.

The third group evaluates the homotopy of a trajectory after it is found. In [16], [17], the homology invariant from [14] is applied to extend a PRM graph around static obstacles and the resulting high-level trajectories are optimized by a local planner. An alternative to homotopy, Universal Visibility Deformation (UVD) (based on VD [18]), introduced in [19] efficiently compares the topology of two trajectories. A UVD aware visibility-PRM (see [20]) is presented to plan mulitple distinct trajectories in real-time for drone flight. The same method is leveraged in [21] and [22] to achieve state-of-the-art results for drone flight in static environments.

Previous works [10], [12], [13] in the second group plan among dynamic obstacles in the state space, all of which assume and leverage road structure. Works [19], [21], [22] in the third group plan drone flight in unstructured 3-D environments, but do not consider dynamic obstacles.

Contribution: In this work, we present a method in the third group, based on [19] to plan trajectories through dynamic environments without relying on road structure. The contributions of this work are as follows:

1) A planner that considers multiple topologically distinct high-level trajectories in dynamic environments, without assuming a structured environment. The most suitable trajectory is selected as initialization (guidance) for a local planner.
2) An algorithm to identify and propagate topology information of trajectories to successive iterations, making the planner behavior stable and consistent.

We validate the proposed planner on a mobile robot both in simulation and in real-world experiments. Our results indicate that the addition of the high-level planner results in faster trajectories, where this improvement increases as the driving scenarios become more crowded. In addition, we observe less collisions in crowded scenarios. Our planner is implemented in ROS/C++ and will be released open-source.

[1]Homology differs slightly from homotopy. See [14] for definitions.

II. Problem Formulation

We model the robot motion by the deterministic discrete-time nonlinear dynamics

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ are the state and input at discrete time instance k, n_x and n_u are the state and input dimensions respectively and the state contains the 2-D position of the robot $\boldsymbol{p}_k = (x_k, y_k) \in \mathbb{R}^2 \subseteq \mathbb{R}^{n_x}$. Obstacles move in the same space as the robot. The position of obstacle j at time $k = 0$ is denoted $\boldsymbol{o}_0^j \in \mathbb{R}^2$ and we assume that for each obstacle predictions of its motion over the next N time steps (i.e., $\boldsymbol{o}_1^j, \ldots, \boldsymbol{o}_N^j$) are available to the robot at each time instance. We model the obstacles and robot area with a single disc each, with radius r_{obs} and r, respectively. We further assume that the robot is given a high-level reference path to follow, e.g., a straight line to the goal. The robot tracks the reference path through the workspace (it can deviate from it) without colliding with the obstacles.

III. Global Guidance

The proposed planner consists of two components, the first computes a global guidance trajectory to guide the second component, a local planner, to the global optimum in the dynamic environment. We design the guidance trajectory search to be light-weight and fast, giving an estimate in the vicinity of the global optimum. The local planner is initialized from the guidance trajectory and leverages more accurate robot models and constraints to locally obtain a safe high quality trajectory.

A. Dynamic Trajectory Planning

We explicitly consider multiple local optimal driving behaviors in the presence of dynamic obstacles by planning high-level collision-free paths in the state space. The state space, composed of the workspace and time, is described by $\mathcal{X} = \mathbb{R}^2 \times [0, T]$, with $[0, T]$ a continuous domain. A trajectory is a continuous path through the state space, $\boldsymbol{\tau} : [0, 1] \to \mathcal{X}$. The area of the workspace occupied by the union of obstacles at time t is denoted by $\mathcal{O}_t \subset \mathbb{R}^2$ and the obstacle set in the state space is thus $\mathcal{O} := \bigcup_{\forall t \in [0,T]} (\mathcal{O}_t, t) \subset \mathcal{X}$. Obstacles puncture holes in the state space, which make the collision-free space nonconvex and results in the existence of multiple locally optimal trajectories.

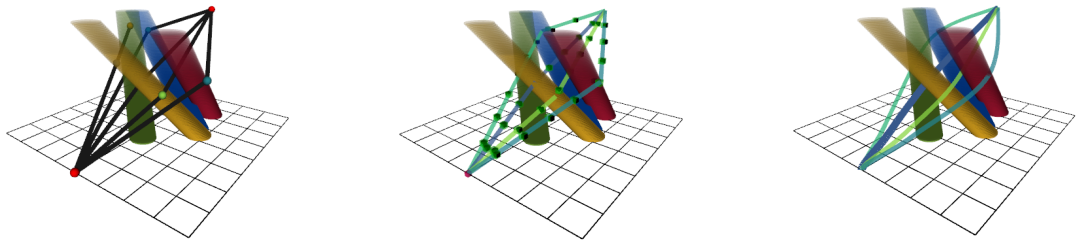
The topology of trajectories can be analyzed to identify a single trajectory for each local optimum. To distinguish trajectory topologies, we require a comparison function,

$$\mathcal{H}(\boldsymbol{\tau}_i, \boldsymbol{\tau}_j, \mathcal{O}) \begin{cases} 1 & \boldsymbol{\tau}_i, \boldsymbol{\tau}_j \text{ topologically equivalent,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

We then seek to find the set of topologically distinct trajectories $\mathcal{T}^* = \{\boldsymbol{\tau}_0, \boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{N_\tau}\}$ in the workspace, where

$$\mathcal{H}(\boldsymbol{\tau}_i, \boldsymbol{\tau}_j, \mathcal{O}) = 0, \ \forall \boldsymbol{\tau}_i, \boldsymbol{\tau}_j \in \mathcal{T}^*, \ i \neq j. \tag{3}$$

In this work, we adopt the topology measure UVD [19]. Two trajectories are in the same UVD class (topology equivalent)



(a) Visibility-PRM graph, with spheres denoting guards (orange), start and goal (red) and connector nodes (colored).

(b) Geometric paths (colored lines) and cubic spline points (cubes) before (green) and after optimization (black).

(c) Guidance trajectories with the picked trajectory (dark blue) thickened.

Fig. 2: Our high-level guidance method, viewed in the state space $\mathcal{X}$. (a) A graph is constructed using visibility-PRM. (b) Geometric paths are found through graph-search and are sampled and smoothed. (c) Cubic splines are fitted through the resulting points. Based on a high-level cost, one guidance trajectory is selected, to be tracked by a local planner.

if points along the trajectories can be connected, without intersecting with obstacles.

Definition 1. [19] *Two trajectories $\boldsymbol{\tau}_1(s), \boldsymbol{\tau}_2(s)$ parameterized by $s \in [0,1]$ and satisfying $\boldsymbol{\tau}_1(0) = \boldsymbol{\tau}_2(0)$, $\boldsymbol{\tau}_1(1) = \boldsymbol{\tau}_2(1)$, belong to the same uniform visibility deformation class, if for all s, line $\overline{\boldsymbol{\tau}_1(s)\boldsymbol{\tau}_2(s)}$ is collision-free.*

In practice, we check collisions for s at discrete intervals along the trajectories.

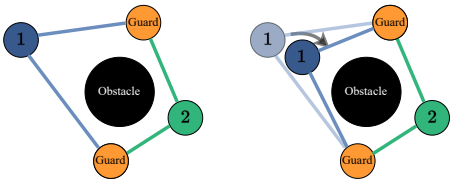
B. Visibility-PRM

We build on Visibility-PRM [19] to compute a sparse representation of the paths from start to goal, with distinct topologies. We introduce important adaptations to repurpose the approach in the state space domain, that includes time, to ensure that the algorithm gives consistent outputs over successive time steps. The modified Visibility-PRM algorithm is given in Algorithm 1 and is detailed below.

PRM initializes its graph with start ($\boldsymbol{x}_0$) and goal ($\boldsymbol{x}_N$) nodes. States for new nodes are drawn at random from a *feasible* state distribution $\boldsymbol{x}^{(i)} \sim \mathbb{P}_{\text{PRM}}$ (**NewSample** line 8) and if possible, the new node is connected to the graph.

Visibility-PRM distinguishes between *Guard* and *Connector* nodes, as follows, to ensure that the graph is sparse. The start and goal nodes are initialized as guards (lines 1, 2). For each new sample, we find the guards that it can be connected to without colliding (**VisibleGuards** line 9). If it connects to 0 guards, it is added as *guard* (**AddGuard** line 11). When a sample connects to exactly 2 guards, it becomes a *connector* (**InitializeConnector** line 17). For connectors we verify that the connection is dynamically feasible (**ConnectionInvalid** lines 13, 15). We then construct a piecewise linear path (**Path**) between a connector and the guards. Lines 22-29 replace an existing connector when the newly sampled connector is in the same UVD class and the new path is shorter. New connectors with a distinct UVD class are added by **AddConnector** in line 32.

A depth-first graph search augmented with a visited node list, similar to [16], computes paths on the graph from start to goal, giving the UVD distinct trajectories $\mathcal{T}^*$. We refer to these trajectories as the *geometric trajectories*. An example of the result is visualized in Fig. 2a.



(a) Segments with new topology

(b) Segments with existing topology

Fig. 3: Each segment is associated with a topological ID. (a) Two segments with new distinct topologies are given unique IDs. (b) A new connector creates a shorter segment within an existing topology, the ID is transferred.

C. Propagating Guidance Trajectories

Because the search for guidance trajectories is repeated at each time step, consistency between successive iterations must be guaranteed to prevent the robot from switching between different trajectories. We address this by marking nodes in the graph with a topology identifier, then reintroducing the nodes in the next iteration with their identifiers. This allows us to reidentify and favour the selected guidance trajectory from the previous iteration.

Each segment consisting of a connector and two guards is in a distinct UVD class by construction. We assign to each connector i a *segment ID*, $\alpha_i \in \mathbb{Z}^+$, that uniquely identifies this segment (line 31). When a connector is replaced, the ID is transferred (line 27). This is illustrated in Fig. 3.

Each geometric trajectory is composed of one or more segments. We associate each geometric trajectory i with a *trajectory ID*, $\beta_i \in \mathbb{Z}^+$ and maintain a mapping between each trajectory and its segments, making it possible to reidentify each trajectory in successive iterations. For example if trajectory $\boldsymbol{\tau}_1$ with ID $\beta_i = 1$ contains segments $\alpha_1 = 1, \alpha_2 = 4$, we first save the mapping $1 \rightarrow \{1, 4\}$. In the next iteration, a trajectory consisting of segments $1, 4$ is reassigned the ID 1.

Algorithm 1: Proposed Visibility-PRM

Input: $\mathcal{O}$, $\boldsymbol{x}_0$, $\boldsymbol{x}_N$, Previous nodes $\mathcal{G}^-$

1 **AddGuard**($\boldsymbol{x}_0$, $\mathcal{G}$)
2 **AddGuard**($\boldsymbol{x}_N$, $\mathcal{G}$)
3 **while** Below sample and time limit **do**
4 reintroduce $\leftarrow$ Not all samples $\mathcal{G}^-$ were reintroduced
5 **if** reintroduce **then**
6 $\boldsymbol{x}$, $\mathcal{N}^- \leftarrow$ **ReintroduceSample**($\mathcal{G}^-$)
7 **else**
8 $\boldsymbol{x} \leftarrow$ **NewSample**($\boldsymbol{x}_0$, $\boldsymbol{x}_N$)
9 $\mathcal{L} \leftarrow$ **VisibleGuards**($\boldsymbol{x}$, $\mathcal{G}$, $\mathcal{O}$)
10 **if** $|\mathcal{L}| = 0$ *(No guards visible)* **then**
11 **AddGuard**($\boldsymbol{x}$, $\mathcal{G}$)
12 **else if** $|\mathcal{L}| = 2$ *(Exactly 2 guards visible)* **then**
13 **if ConnectionInvalid**($\mathcal{L}_0$, $\boldsymbol{x}$) **then**
14 **Continue**
15 **if ConnectionInvalid**($\boldsymbol{x}$, $\mathcal{L}_1$) **then**
16 **Continue**
17 $\mathcal{N} \leftarrow$ **InitializeConnector**($\boldsymbol{x}$)
18 **if** reintroduce **then**
19 Transfer segment ID of $\mathcal{N}^-$ to $\mathcal{N}$
20 $\boldsymbol{\tau} \leftarrow$ **Path**($\mathcal{L}_0, \boldsymbol{x}, \mathcal{L}_1$)
21 distinct $\leftarrow$ True
22 **for** Shared neighbour $\boldsymbol{x}^j, \mathcal{N}^j$ of $\boldsymbol{x}, \mathcal{N}$ in $\mathcal{L}$ **do**
23 $\boldsymbol{\tau}_j \leftarrow$ **Path**($\mathcal{L}_0, \boldsymbol{x}^j, \mathcal{L}_1$)
24 **if** $\mathcal{H}(\boldsymbol{\tau}, \boldsymbol{\tau}_j, \mathcal{O}) = 1$ **then**
25 distinct $\leftarrow$ False
26 **if Length**($\boldsymbol{\tau}$) $<$ **Length**($\boldsymbol{\tau}_j$) **then**
27 Transfer the segment ID of $\mathcal{N}^j$ to $\mathcal{N}$
28 In $\mathcal{G}$, replace connector $\mathcal{N}^j$ with $\mathcal{N}$
29 **Break**
30 **if** distinct **then**
31 **if** reintroduce is False **then**
32 Initialize segment ID of $\mathcal{N}$ with unused ID
33 **AddConnector**($\mathcal{N}$, $\mathcal{G}$)

Output: $\mathcal{G}$

In line 6 of Algorithm 1, **ReintroduceSample** reintroduces all nodes of the previous graph ($\mathcal{G}^-$) before sampling new states. Because a sampling time passes between these iterations, the time coordinate of each node (x, y, t) in the previous graph needs to be updated to $(x, y, t-h)$, where h is the sampling time. When the time coordinate becomes zero, we resample a new node halfway along the trajectory.

Since the connectors possess a segment ID, the segment associations are carried over from the previous iteration (line 19). After new paths are found, we reidentify previous trajectories from their segments. These steps result in consistent identification of UVD distinct trajectories and improve the overall planner performance, since trajectories and their topology are propagated to successive iterations.

D. Spline Optimization

The geometric trajectories do not satisfy the kinematic constraints of the robot (they are discontinuous) and therefore cannot be followed by a low-level controller. We smoothen these trajectories in two steps as visualized in Figs. 2b and 2c. The first step samples the trajectories and optimizes the resulting points. The second step fits a smooth curve through the optimized points.

Step 1: We first convert the state space paths to time parameterized trajectories, $\boldsymbol{\tau}_{geometric} : [0, T] \rightarrow \mathbb{R}^2$. Each trajectory is sampled with regular intervals Δt to obtain a set of control points $\boldsymbol{Q} = [\boldsymbol{Q}_0, \ldots, \boldsymbol{Q}_N]$, with $\boldsymbol{Q}_i \in \mathbb{R}^2$.

We optimize these control points to improve smoothness both in position and velocity. The optimization problem is designed to be quadratic and unconstrained to minimize computation times. We define the cost as

$$J_{\text{points}} = J_{\text{geo}} + J_{\text{smooth}} + J_{\text{obst}} + J_{\text{vel}}, \tag{4}$$

where the geometric cost [19] penalizes distance to the original control points $\bar{\boldsymbol{Q}}$ on the geometric trajectory,

$$J_{\text{geo}} = \sum_{i}^{N} ||\boldsymbol{Q}_i - \bar{\boldsymbol{Q}}_i||^2. \tag{5}$$

The smoothness cost [19]

$$J_{\text{smooth}} = \sum_{i=1}^{N-1} ||\boldsymbol{Q}_{i-1} - 2\boldsymbol{Q}_i + \boldsymbol{Q}_{i+1}||^2, \tag{6}$$

functions as an elastic band that smoothens the trajectory.

The obstacle cost is designed to penalize the linear distance d_i^j from control point i to the obstacle j by an exponential penalty. To keep the cost quadratic however, we use the second order Taylor expansion. Let $\boldsymbol{A}_{i,j} = \bar{\boldsymbol{Q}}_i - \boldsymbol{o}_k^j$, where time index k matches the time associated with $\boldsymbol{Q}_i$. Then the cost is given by $J_{\text{obst}} = \sum_i \sum_j \boldsymbol{Q}_i^T \boldsymbol{H}_{i,j} \boldsymbol{Q}_i + \boldsymbol{f}_{i,j}^T \boldsymbol{Q}_i$, with

$$\boldsymbol{H}_{i,j} = \frac{\boldsymbol{A}_{i,j}\boldsymbol{A}_{i,j}^T}{2}, \ \boldsymbol{f}_{i,j} = -(1 + r + r_{\text{obs}})\boldsymbol{A}_{i,j} - 2\boldsymbol{H}_{i,j}\boldsymbol{o}_k^j.$$

The velocity cost penalizes an offset with respect to a tracking velocity v_{ref}. We compute this cost by constructing a trajectory that satisfies the velocity tracking cost everywhere, based on the geometric trajectory. We then penalize the distance to that trajectory. The velocity control points of the geometric path can be computed using

$$\boldsymbol{V}_i = \frac{\boldsymbol{Q}_{i+1} - \boldsymbol{Q}_i}{\Delta t}, \ i = 1, \ldots, N-1. \tag{7}$$

Then the associated velocity optimized path is given by

$$\boldsymbol{Q}_{\boldsymbol{i+1}}^{\boldsymbol{v}} = \boldsymbol{Q}_i^v + v_{\text{ref}} \frac{\boldsymbol{V}_i}{||\boldsymbol{V}_i||} \Delta t, \tag{8}$$

with $\boldsymbol{Q}_0^v$ equal to the current velocity of the robot. The velocity cost matches (5) with $\bar{\boldsymbol{Q}}$ replaced by $\boldsymbol{Q}^v$.

Step 2: Since the optimization problem is quadratic and unconstrained, we obtain the solution in closed-form. We fit cubic splines separately through the optimized x and y position of the control points to obtain a continuous trajectory, consisting of segments $\boldsymbol{\tau}^i(t) = \begin{bmatrix} \tau_x^i & \tau_y^i \end{bmatrix}^T$, with τ_x^i (and τ_y^i) given by

$$\tau_x^i(t) = a_x^i t^3 + b_x^i t^2 + c_x^i t + d_x,$$

which is twice continuously differentiable and passes through the control points. We impose a boundary condition on the initial velocity of the trajectories to ensure that it respects the robot's current velocity. For more details on fitting the cubic splines, we refer to [23]. The cubic splines together form a smooth trajectory $\boldsymbol{\tau} : [0, T] \rightarrow \mathbb{R}^2$ from the current robot position to the goal (see Fig. 2c). Guidance trajectory $\boldsymbol{\tau}$ is not guaranteed to be collision-free, but is smooth enough to be used by a local planner that enforces collision avoidance.

E. Spline Selection

From the candidate splines generated in each control iteration, we need to select the guidance trajectory that best represents the performance indicators for the robot's motion. We consider the following criteria:

1) Minimize path length - preferring short paths.
2) Minimize difference to preferred velocity - penalizing too fast or slow driving.
3) Minimize acceleration - preferring smooth trajectories.
4) Consistency - ensuring that when another trajectory improves over the selected trajectory of the previous control iteration, it should significantly outperform it.

We compute the costs for each trajectory by taking samples of positions $\boldsymbol{p}_i$, velocities $\boldsymbol{v}_i$ and accelerations $\boldsymbol{a}_i$ at constant time intervals along the trajectory, resulting in the objective

$$\begin{aligned} J_{\text{select}} = &\sum_{i \in \mathcal{I}} w_L ||\boldsymbol{p}_i - \boldsymbol{p}_{i-1}|| + w_V || \, ||\boldsymbol{v}_i|| - \bar{v}|| \\ &+ w_a \alpha^i ||\boldsymbol{a}_i|| + w_c C, \end{aligned} \tag{9}$$

where $\mathcal{I}$ is the number of samples, w denotes weights, $\bar{v}$ is the reference velocity, $\alpha \approx 1$ to discount accelerations later in the trajectory and C denotes a constant penalty if this trajectory was not selected in the previous iteration. We select the lowest cost trajectory (thickened in Fig. 2c).

IV. Local Planning

The local planner needs to plan a kinematically feasible motion that is collision-free, since this is not guaranteed by the guidance trajectory. We introduce the guidance trajectory to the local planner in two ways. First, the solver is initialized with the guidance trajectory. This in itself may not be sufficient to converge to the desired optimum. We therefore also follow the guidance trajectory by using an MPCC [1] as local planner. This planner is designed to follow the path traced by the guidance trajectory while tracking its velocity.

The objective of the MPCC is given by

$$J_{\text{local}} = \sum_{k=0}^{N_{\text{MPCC}}} J_{c,k} + J_{l,k} + J_{v,k} + J_{a,k} + J_{\omega,k}, \tag{10}$$

where $J_{c,k}$ and $J_{l,k}$ denote the lag and contouring costs as in [1], the velocity reference tracking is enforced via $J_{v,k} = ||v_k - \bar{v}_k||$, where $\bar{v}_k$ is the velocity on the guidance trajectory at time step k, and the costs $J_{a,k}$ and $J_{\omega,k}$ penalize actuation.

To avoid obstacles one would typically use nonconvex constraints $||\boldsymbol{p}_k - \boldsymbol{o}_k|| \geq r + r_{\text{obs}}$ [1]. However, in practice these may result in switching between local optima, without giving a consistent solution. We employ the linearized version of these constraints, the linear constraint orthogonal to the vector between the robot and obstacle, $\boldsymbol{A}^T \boldsymbol{p}_k \leq b$, where

$$\boldsymbol{A} = \frac{\boldsymbol{o}_k - \boldsymbol{p}_k}{||\boldsymbol{o}_k - \boldsymbol{p}_k||}, \quad b = \boldsymbol{A}^T(\boldsymbol{o}_k - A(r + r_{\text{obs}})). \tag{11}$$

These constraints result in consistent and smooth motion[2].

V. Results

We validate our approach in simulation and real-world experiments, comparing in both cases against a local planner without guidance. Our planner is implemented in ROS/C++ and will be released open-source. A video of the simulations and experiments is available in [24].

A. Notes on Implementation

Experimental settings are given in Table I. The high-level planner is fast enough to plan over a longer horizon than the local planner while remaining real-time (i.e., $N_{\text{PRM}} > N_{\text{MPCC}}$). In crowded scenarios, this leads to smoother trajectories as the robot can adapt its high-level maneuvre earlier. We select the goal for PRM as the point along the reference path reached when the robot drives at the preferred velocity. It is projected to the nearest collision-free position if necessary. When the goal cannot be reached, we reduce the horizon. PRM nodes are sampled (i.e., $\mathbb{P}_{\text{PRM}}$) in a forward directed arc considering velocity and acceleration limits.

B. Simulated Guidance Ablation Study

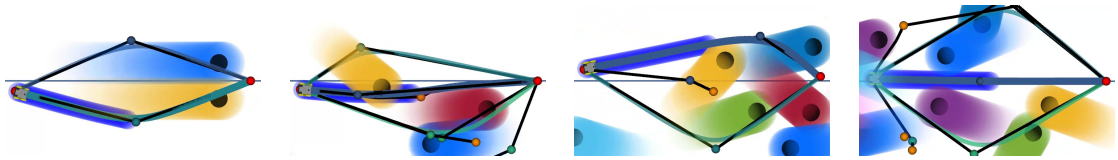
Our simulations consider four environments with pedestrians. The first environment (*head-on*) consists of a straight road with two pedestrians moving towards the robot. The other environments contain $4, 8$ and 16 pedestrians with random start positions and velocities near the reference path. The random scenarios are identical for all planners (i.e., we use the same random seed). In all simulations, pedestrians move with constant velocity. We reset the simulation when the robot reaches the end of the road in the x-direction and repeat each experiment 200 times.

Statistical results are given in Table II and snapshots are shown in Fig. 4. The guidance allows the robot to consistently navigate around the pedestrians in the head-on scenario. Without guidance, indecisiveness of the planner leads to infeasibility and collisions in most simulations. In the randomized environments, the guidance reduces the task

TABLE I: Experimental settings. Weights are denoted "(w)".

Overall	N_{PRM}	N_{MPCC}	h	$\bar{v}$		
	120	40	0.05 s	2 m/s		
Points (w)	J_{geo}	J_{smooth}	J_{obst}	J_{vel}		
	25	10	0.5	0.01		
Splines	Points	w_L	w_V	w_a	w_c	α
	20	1	100	100	25	0.95
MPCC (w)	$J_{c,k}^{\text{MPCC}}$	$J_{c,k}^{\text{Guidance}}$	$J_{l,k}$	$J_{v,k}$	$J_{a,k}$	$J_{\omega,k}$
	0.01	0.5	0.01	0.3	0.05	0.05

[2]In principle we could run a local planner for each guidance trajectory in parallel and choose the best one. This is left for future work.

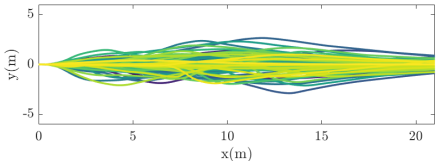


(a) Head-on 2 (b) Random 4 (c) Random 8 (d) Random 16

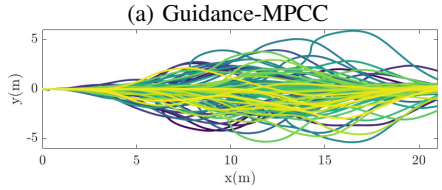
Fig. 4: Snapshots of the simulations, viewed in the state-space. Pedestrians (black discs) are visualized with their predicted area (colored disc) inflated by the robot area. Visualization of the graph and guidance trajectories are identical to Fig. 2. The robot's local motion plan is denoted by blue discs, indicating the predicted area occupied by the robot.

TABLE II: Statistical results for the task duration, number of collisions and computation times when comparing MPCC with and without guidance over 200 simulations each in 4 different scenarios. Results are denoted as "avg (std)" over experiments except for collisions. We denote with "High-Level Planning" the time spent to compute the guidance trajectory.

Scenario	Method	Task Duration [s]	Collisions	Computation Time [ms]	Computation Time High-Level Planning [ms]
Headon 2	LMPCC	13.3 (0.3)	160	17.1 (8.9)	-
	Guidance MPCC	**12.3** (0.1)	**0**	**16.5** (2.9)	6.5 (2.1)
Random 4	LMPCC	12.3 (0.3)	**0**	17.2 (3.0)	-
	Guidance MPCC	**12.2** (0.2)	1	**16.2** (2.6)	6.0 (1.9)
Random 8	LMPCC	12.7 (0.6)	3	**17.2** (3.1)	-
	Guidance MPCC	**12.4** (0.4)	3	17.6 (3.7)	6.5 (2.3)
Random 16	LMPCC	13.5 (1.1)	**16**	**19.4** (4.3)	-
	Guidance MPCC	**13.2** (1.0)	17	22.1 (6.7)	8.5 (4.6)



(a) Guidance-MPCC



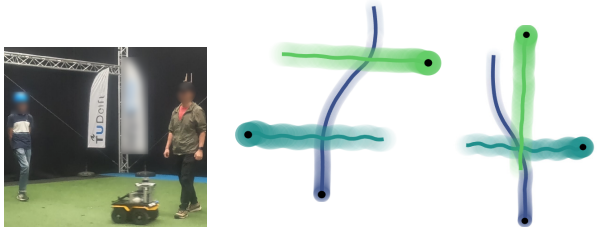
(b) MPCC

Fig. 5: A comparison of trajectories with and without guidance in a randomized environment with 8 pedestrians.

duration by 2, 3 and 5%, with a larger improvement in more crowded environments. Trajectories for the 8 pedestrian case, visualized in Fig. 5, show that guidance allows the planner to choose faster driving behaviors. Guidance MPCC collides slightly more often than the baseline. When obstacles block the path, the high-level planner may not find a guidance trajectory. We then use the last computed guidance trajectory, which can lead to collisions. Future work may resolve this problem by considering multiple goals.

C. Real-World Experiments

We deploy the proposed planner experimentally on a mobile robot (Clearpath Jackal) navigating among pedestrians. The robot is equipped with an Intel i5 CPU@2.6GHz. Localization of the robot and pedestrians is obtained from a marker based tracking system and pedestrian predictions assume constant velocity, where the velocity is obtained from Kalman filtered position data.



(a) Setup (b) Crossing Case (c) Mixed Case

Fig. 6: Trajectories recorded in the real-world experiments for the robot (dark blue) and pedestrians (green). Start positions are denoted in black.

Trajectories for 2 scenarios and the setup are visualized in Fig. 6. In the first scenario, the guidance allows the robot to pass behind the last pedestrian. In the second scenario, the robot moves left to evade both pedestrians efficiently.

VI. Conclusion

This work presented a novel planner for autonomous navigation in dynamic environments. The planner finds distinct high-level trajectories to guide a local optimization-based planner to a global optimal plan. Guidance trajectories are computed and tracked over successive control iterations.

We showed that the resulting planner leads to shorter average task duration times than the local planner in isolation, with larger improvement in crowded environments. Real-world experiments further validated the proposed approach.

Further research could explore applications of the guidance trajectory search to predict human motion or to endow the local planner with socially compliant motion.

REFERENCES

[1] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4459–4466, Oct. 2019.

[2] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe Nonlinear Trajectory Generation for Parallel Autonomy With a Dynamic Vehicle Model," *IEEE Trans. Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2994–3008, Sep. 2018.

[3] H. Zhu and J. Alonso-Mora, "Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 776–783, Apr. 2019.

[4] A. Wang, X. Huang, A. Jasour, and B. Williams, "Fast Risk Assessment for Autonomous Vehicles Using Learned Models of Agent Futures," *arXiv:2005.13458 [cs, stat]*, Jun. 2020.

[5] O. de Groot, B. Brito, L. Ferranti, D. Gavrila, and J. Alonso-Mora, "Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments," *IEEE Robot. Autom. Lett.*, pp. 5389 – 5396, 2021.

[6] S. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Computer Science Department Iowa State University, Tech. Rep. TR 98-11, 1998. [Online]. Available: http://janowiec.cs.iastate.edu/papers/rrt.ps

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364911406761

[8] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[9] E. H. Bes, "Path Planning with Homotopic Constraints for Autonomous Underwater Vehicles," Ph.D. dissertation, 2012.

[10] P. Bender, O. c. Taş, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2015, pp. 1386–1392, iSSN: 1931-0587.

[11] J. Park, S. Karumanchi, and K. Iagnemma, "Homotopy-Based Divide-and-Conquer Strategy for Optimal Trajectory Planning via Mixed-Integer Programming," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1101–1115, Oct. 2015.

[20] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, Jan. 2000. [Online]. Available: https://www.tandfonline.com/doi/full/10.1163/156855300741960

[12] B. Yi, P. Bender, F. Bonarens, and C. Stiller, "Model Predictive Trajectory Planning for Automated Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 24–38, Mar. 2019.

[13] F. Altché and A. de La Fortelle, "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec. 2017, pp. 2126–2133.

[14] S. Bhattacharya, R. Ghrist, and V. Kumar, "Persistent Homology for Path Planning in Uncertain Environments," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 578–590, Jun. 2015.

[15] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based Path Planning with Homotopy Class Constraints," Jul. 2010.

[16] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, Feb. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889016300495

[17] C. Rösmann, A. Makarow, and T. Bertram, "Online Motion Planning based on Nonlinear Model Predictive Control with Non-Euclidean Rotation Groups," in *2021 European Control Conference (ECC)*, Jun. 2021, pp. 1583–1590. [Online]. Available: http://arxiv.org/abs/2006.03534

[18] L. Jaillet and T. Simeon, "Path Deformation Roadmaps: Compact Graphs with Useful Cycles for Motion Planning," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1175–1188, Nov. 2008. [Online]. Available: https://doi.org/10.1177/0278364908098411

[19] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust Real-time UAV Replanning Using Guided Gradient-based Optimization and Topological Paths," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1208–1214, iSSN: 2577-087X.

[21] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning Minimum-Time Flight in Cluttered Environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, Jul. 2022. [Online]. Available: http://arxiv.org/abs/2203.15052

[22] R. Penicka and D. Scaramuzza, "Minimum-Time Quadrotor Waypoint Flight in Cluttered Environments," arXiv, Tech. Rep. arXiv:2202.03947, Feb. 2022. [Online]. Available: http://arxiv.org/abs/2202.03947

[23] T. Kluge, "Cubic Splines," Mar. 2021. [Online]. Available: https://kluge.in-chemnitz.de/opensource/spline/spline.pdf

[24] O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora, "Globally Guided Trajectory Planning in Dynamic Environments," 2023. [Online]. Available: https://www.youtube.com/watch?v=tkRbsAuSTrA