Motion Planning in Non-Convex Corridors: A Guided MPC Approach

Bernhard Wullt[1], Per Mattsson[2], Thomas B. Schön[2], Mikeal Norrlöf[1]

I. Introduction

Motion planning is an important problem in robotics, arising in many applications such as bin picking, autonomous driving, etc. We present a novel motion planner for cluttered environments, where corridor planning and model predictive control (MPC) are integrated into one solution. The corridor planner's task is to return a non-convex collision-free region that connects a given start and goal configuration. This region is referred to as a corridor and is constructed using a signed distance function (SDF), from which collision-free regions can be represented as spheres. From the resulting corridor, the MPC then controls the robot to stay within the corridor and to reach the goal state. Our main contribution is in the last part, i.e. how to guide the MPC within a non-convex corridor such that it reaches a global goal state. The main idea is to first select a sequence of spheres in the corridor such that the future motion is feasible and to force it to a local goal state, possibly unreachable in one time step. Our idea is similar to [1], which also uses spheres for collision-avoidance, but instead of selecting spheres along a corridor, they optimize the sphere radius by a line search method at each iteration. Furthermore, their approach requires a reference trajectory to be tracked at each iteration, which adds computation time. Our approach removes the need for a reference trajectory, we only track a set-point, similar to [2]. However, the set-point is sequentially computed, therefore allowing our solution to be used in cluttered environments. From simulations with double integrator dynamics and randomized obstacles, we observe an average computation time of 20 ms per iteration.

A. Background

Consider a robot operating in the world space, $\mathcal{W} \subset \mathbb{R}^3$. The exact description of the robot body is given by its configuration $\mathbf{q} \in \mathcal{C}$, where $\mathcal{C} \subset \mathbb{R}^{n_c}$ is the configuration space. We define the state of the system as $\mathbf{x} = [\mathbf{q}^\top, \dot{\mathbf{q}}^\top]^\top \in \mathcal{X} \subset \mathbb{R}^{n_x}$, where $\mathcal{X}$ is the state space, defined as $\mathcal{X} = \mathcal{C} \times \mathbb{R}^{n_c}$. We access the configuration and velocity of the state using the following notation, $\mathbf{q}(\mathbf{x})$ and $\dot{\mathbf{q}}(\mathbf{x})$, respectively. The state evolves according to the discrete dynamics, $\mathbf{x}_+ = \mathbf{f}(\mathbf{x}, \mathbf{u})$, where $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^{n_c}$ is the robot's control input and $\mathbf{x}_+$ is the state at the next time step.

The robot is surrounded by n_o obstacles $\mathcal{O} = \bigcup_{i=1}^{n_o} \mathcal{O}_i$, where $\mathcal{O}_i \subset \mathcal{W}$. The set of points covered by the robot body at a certain configuration is expressed as $\mathcal{FK}(\mathbf{q}) \subset \mathcal{W}$. The representation of the ith obstacle in the configuration space is given by the set $\mathcal{CO}_i = \{\mathbf{q} \in \mathcal{C} \mid \mathcal{B}(\mathbf{q}) \cap \mathcal{O}_i \neq \emptyset\}$. The obstacle space is formed as $\mathcal{C}_\text{o} = \bigcup_{i=1}^{n_o} \mathcal{CO}_i$. The complement is referred to as the free space, $\mathcal{C}_\text{f} = \mathcal{C} \setminus \mathcal{C}_\text{o}$. We define the SDF expressed in the configuration space as

$$\psi(\mathbf{q}, \mathcal{O}) = \begin{cases} -\phi(\mathbf{q}, \mathcal{O}) & \text{if } \mathbf{q} \in \mathcal{C}_\text{o}, \\ \phi(\mathbf{q}, \mathcal{O}) & \text{otherwise,} \end{cases} \tag{1}$$

where $\phi : \mathcal{C} \times \mathcal{W} \mapsto \mathbb{R}_+$ is a function that returns the closest distance measured using the 2-norm, $||\cdot||_2 : \mathbb{R}^m \mapsto \mathbb{R}_+$, to the boundary of the obstacle region, $\partial\mathcal{C}_\text{o}$, that is

$$\phi(\mathbf{q}, \mathcal{O}) = \min_{\mathbf{q}_\text{c} \in \partial\mathcal{C}_\text{o}} ||\mathbf{q} - \mathbf{q}_\text{c}||_2. \tag{2}$$

From now on, we remove the explicit dependency of the obstacle $\mathcal{O}$, only adding it when needed. With the SDF, we can construct a collision-free region

$$\mathcal{B}(\mathbf{c}) = \{\mathbf{q} \in \mathcal{C} \mid \|\mathbf{q} - \mathbf{c}\|_2 \leq \psi(\mathbf{c})\}. \tag{3}$$

Thus, it is a Euclidean norm ball centered around the center point $\mathbf{c}$ with radius defined by the signed distance. If the signed distance is positive, then the sphere is contained in the free space, i.e. $\mathcal{B}(\mathbf{c}) \subset \mathcal{C}_\text{f}$. In the following, we drop the explicit dependence on the ball center.

B. Motion planning problem

Given a start state, $\mathbf{x}_\text{s} \in \mathcal{X}$, and a goal state, $\mathbf{x}_\text{g} \in \mathcal{X}$. The problem is to compute a sequence of inputs, $(\mathbf{u}_1, \ldots, \mathbf{u}_{n-1})$, such that the corresponding sequence of states, $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, starts and ends in $\mathbf{x}_1 = \mathbf{x}_\text{s}$ and $\mathbf{x}_n = \mathbf{x}_\text{g}$, respectively. Furthermore, the states are required to be collision-free $\mathbf{q}(\mathbf{x}_i) \in \mathcal{C}_\text{f}, \quad \forall i \in \mathbb{N}_{1:n}$, where $\mathbb{N}_{1:n}$ is the set of integers from 1 to n.

II. Method

The following section presents our suggested motion planner for static environments. Our approach is decoupled into corridor planning and control, which we describe in the following sections, ending with the integrated solution.

A. Corridor planning

We use Probabilistic Bubble RoadMap (PBRM) described in [3] as corridor planner. The planner is based on Probabalistic RoadMap (PRM) [4], with the difference that the vertices in the graph consists of collision-free regions, i.e. the sphere defined in (3), instead of points. The following section gives a brief overview how the planner works, see [3] for more details. The planner constructs a graph representing the free space, which is done by distributing n_v spheres that

[1] ABB Robotics, Sweden.
[2] Department of Information Technology, Uppsala University, Sweden.

cover as much space as possible. The edges in the graph are formed between spheres that intersect, that is

$$\mathcal{E} = \{(\mathcal{B}_i, \mathcal{B}_j) \in \mathcal{V}^2 \,|\mathcal{B}_i \cap \mathcal{B}_j \neq \emptyset\}. \tag{4}$$

Once this is done, the planner can be queried for corridors connecting a start and goal configuration, $\mathbf{q}_\text{s}$ and $\mathbf{q}_\text{g}$. This is done by a weighted graph search, resulting in a path, $\boldsymbol{\rho}(s) : [0,1] \mapsto \mathcal{C}_\text{f}$, which passes through a subset of the vertices. Since the edges are formed over intersecting spheres, the path is by construction guaranteed to lie in the free space. From the path a corridor is constructed, $\boldsymbol{P}(s) = \mathcal{B}(\boldsymbol{\rho}(s))$, which connects the query points, $\exists (s,l) \in [0,1]^2$, $(\mathbf{q}_\text{s}, \mathbf{q}_\text{g}) \in (\boldsymbol{P}(s), \boldsymbol{P}(l))$.

B. Corridor Control

Having computed a corridor, our next goal is to steer the robot to the goal state in a receding horizon fashion. We use a standard set-point tracking MPC defined as

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_{n_h}} \quad \sum_{i=1}^{n_h+1} \|\mathbf{x}_i - \tilde{\mathbf{x}}_\text{g}\|^2_{\mathbf{Q}_i} + \sum_{i=1}^{n_h} \|\mathbf{u}_i\|^2_{\mathbf{R}} \tag{5a}$$

$$\text{s.t.} \quad \mathbf{x}_1 = \mathbf{x}, \tag{5b}$$

$$\mathbf{x}_{i+1} = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i),\ i \in \mathbb{N}_{1:n_h}, \tag{5c}$$

$$\dot{\mathbf{q}}(\mathbf{x}_{n_h+1}) = 0, \tag{5d}$$

$$(\mathbf{x}_i, \mathbf{u}_i) \in \mathcal{X} \times \mathcal{U},\ i \in \mathbb{N}_{1:n_h}, \tag{5e}$$

$$\mathbf{q}(x_i) \in \mathcal{B}^\star_i,\ i \in \mathbb{N}_{1:(n_h+1)}. \tag{5f}$$

In the above, our decision variables are n_h control inputs. The cost aims to pull the predicted future trajectory towards a set-point, $\tilde{\mathbf{x}}_\text{g}$, which is a local goal state, while minimizing the control inputs. The predicted future trajectory starts in the current state (5b), evolves according to the dynamics (5c), ends in a terminal state (5d), while respecting the bounds on the state and control inputs (5e). Our collision-avoidance constraint is expressed in (5f), where we enforce each configuration to stay within a corresponding sphere. The following sections presents how we allocate the spheres for the predicted states and compute the local goal state to be tracked.

1) Sphere selection: To compute the corresponding spheres in (5f), we assume that we have access to an initial feasible trajectory, $X = (\mathbf{x}_1, \ldots, \mathbf{x}_{n_h+1}) \in \mathbb{R}^{n_x \times (n_h+1)}$. For each state in the trajectory, $\mathbf{x}_i \in X$, we find a sphere in the corridor that contains the state and is the furthest along the corridor. We compute this by solving

$$s_i^\star = \max\{s \in [0,1] \,|\, \mathbf{q}(\mathbf{x}_i) \in \boldsymbol{P}(s)\}, \tag{6}$$

resulting in the sphere $\mathcal{B}_i^\star = \boldsymbol{P}(s_i^\star)$. Repeating the process results in the sequence of spheres, $B = (\mathcal{B}_1^\star, \ldots, \mathcal{B}_{n_h+1}^\star)$.

2) Virtual goal state: To compute a virtual local goal state, $\tilde{\mathbf{x}}_\text{g}$, we set its velocity to zero, $\dot{\mathbf{q}}(\tilde{\mathbf{x}}_\text{g}) = 0$. Its configuration is computed by selecting the furthest point along the path that is contained in the last sphere belonging to B. This is computed as

$$s_g = \max\{s \in [0,1] \,|\, \boldsymbol{\rho}(s) \in \mathcal{B}^\star_{n_h+1}\}, \tag{7}$$

from which we compute the configuration as $\mathbf{q}(\tilde{\mathbf{x}}_\text{g}) = \boldsymbol{\rho}(s_g)$.

Algorithm 1 A receding horizon control approach to steer a robot within a corridor to reach a goal state.

Require: ψ, $\mathbf{x}_\text{s}$, $\mathbf{x}_\text{g}$

1: $\mathbf{x} \leftarrow \mathbf{x}_\text{s}$
2: $X \leftarrow (\mathbf{x})_{i=1}^{n_h+1}$
3: **while** $\varepsilon < \|\mathbf{x} - \mathbf{x}_\text{g}\|_2$ **do**
4: $\quad \boldsymbol{P} \leftarrow$ compute a corridor connecting $\mathbf{x}$ and $\mathbf{x}_\text{g}$
5: $\quad B \leftarrow$ compute spheres, Section II-B.1
6: $\quad \tilde{\mathbf{x}}_\text{g} \leftarrow$ compute virtual goal, Section II-B.2
7: $\quad X, U \leftarrow$ solve (5) with $\mathbf{x}, \tilde{\mathbf{x}}_\text{g}, B$
8: $\quad$ apply first control in sequence, $\mathbf{x} = \mathbf{f}(\mathbf{x}, U_1)$
9: $\quad$ shift trajectory X
10: **end while**

C. Motion Planning Solution

Having presented all the necessary parts, we can now integrate all parts into one solution, presented in Algorithm 1. We initialize our predicted future trajectory by repeating the start state for the whole horizon. In the online part, we compute a corridor connecting our current state and the goal state, line 4. Then, we compute the spheres and the virtual goal state, line 5 and 6. Next, we solve the optimization problem, line 7, and retrieve the optimized predicted future trajectory with input controls. The first control input in the sequence is applied to the system, line 8, and we end by shifting the trajectory. By repeating the process we end up in the goal state.

III. Conclusions

We have presented a novel motion planning solution, where corridor planning has been integrated with MPC. The corridor planner finds a corridor that connects the robot's current configuration with the goal configuration. To stay within the corridor and reach the goal state, we select a sequence of spheres that contain our future predicted trajectory and compute a local goal state that is the furthest along the path and within the last sphere. By solving a set-point tracking MPC formulation, we steer the robot inside the corridor to the global goal state. From simulated experiments with linear dynamics in cluttered environments, we observe fast execution, 20 ms on average per iteration, demonstrating the practical usefulness of our approach.

References

[1] T. Schoels, L. Palmieri, K. O. Arras, and M. Diehl, "An NMPC approach using convex inner approximations for online motion planning with guaranteed collision avoidance," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[2] T. Schoels, P. Rutquist, L. Palmieri, A. Zanelli, K. O. Arras, and M. Diehl, "CIAO*: MPC-based safe motion planning in predictable dynamic environments," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6555–6562, 2020.

[3] B. Wullt, M. Norrlöf, P. Mattsson, and Thomas B. Schön, "Probabilistic Bubble Roadmap." https://arxiv.org/abs/2502.16205, 2025.

[4] 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, 1996.