Obstacle Avoidance in Dense Environments using MPC

Pei An Hsieh
School of Engineering and Applied Science
University of Pennsylvania
Email: pahsieh@seas.upenn.edu

Zhenzhen Shao
School of Engineering and Applied Science
University of Pennsylvania
Email: zhens@seas.upenn.edu

Zi-Yan Liu
School of Engineering and Applied Science
University of Pennsylvania
Email: lzi@seas.upenn.edu

***Abstract*—In this final project, we compared two obstacle avoidance methods: Local Model Predictive Contouring Control (LMPCC) and Dynamic Window Approach (DWA). In addition, we present Obstacle Window Filtering, which improves the safety performance of LMPCC in dense environments. Obstacle Window Filtering filters obstacles set as constraints in LMPCC by a safety distance. The optimization solver produced better solutions by reducing the number of constraints, allowing robots to avoid more collision scenarios. The analysis also showed that although the current LMPCC method outperforms the traditional DWA, both do not scale well in dense environments.**

I. Introduction

Studies in robotics have evolved dramatically in past decades, however, one of the greatest challenges for robots is to safely and efficiently plan trajectories that are in crowded dynamic environments.

Traditional control policies often struggle in such scenarios, leading to issues like the freezing robot problem[1], where a robot becomes immobilized or acts aggressively facing complex obstacles, which increases the risk of collisions. Furthermore, the unstructured and non-convex constraints of environments make the optimization computationally expensive, often resulting in slow responses that not suitable for real-time applications.

This study focuses on implementing and experimenting with advanced local planning approaches to address the challenges above. This includes Local Model Predictive Contouring Control (LMPCC), which provides us with a framework that considers system dynamics and future control inputs[2], and Dynamic Windows Approach (DWA)[3], which is a traditional approach for real-time dynamic obstacle avoidance. We also propose Obstacle Window Filtering (OWF), which filters obstacles by distance and improves the performance of LMPCC.

II. Methods

A. Local Model Predictive Contouring Control (LMPCC)

The LMPCC collision avoidance method proposed in [2] consists of the following steps, which are executed in every planning loop.

1) Identify a collision-free area centered around the robot in the static map and establish constraints on the control problem to keep the robot within this region.
2) Anticipate the future positions of dynamic obstacles and employ boundaries to guarantee collision avoidance.
3) Address the control problem by solving a tailored Local Model Predictive Contouring Control (LMPCC) formulation designed for mobile robots.

1) Static Collision Avoidance: The available space in an unstructured scene is typically non-convex, posing challenges for optimization tasks due to slow and difficult computations. Our objective is to determine a set of convex rectangular areas centered around the robot and its predicted trajectory within the unoccupied regions of the environment's static map. To derive the set of convex regions at time t, we first retrieve the optimal MPC trajectory computed and recorded at time t-1 to calculate collision-free areas for the current optimal trajectory computation. For every point q_k (k=1,...,N) we compute a convex region in free space defined by a set of four linear constraints[2] $c_k^{stat}(q_k) = U_{l=1}^4 \, c_k^{stat,l}(q_k)$, a rectangular region aligned with the orientation of the trajectory at q_k, with each linear constraint obtained through a search routine.

2) Dynamic Collision Avoidance: Each moving obstacle i is characterized by its position $\mathbf{p}_i(t)$ and a circular shape with a radius r_{obs}. For each obstacle $i \in 1, .., N$, and prediction step k, we enforce the condition that the robot's circle wouldn't intersect with the ocuppied area defined by the obstacle's circle. The inequality constraint for each disc of the robot concerning the obstacles is expressed as follows:

$$c_k^{obst,i}(\mathbf{z}_k) = \begin{bmatrix} \Delta x_i^k \\ \Delta y_i^k \end{bmatrix}^T \begin{bmatrix} \Delta x_i^k \\ \Delta y_i^k \end{bmatrix} > 1 \tag{1}$$

3) Cost function for optimization: Here we define a contour error, angular and velocity penalty for tracking of the reference path, and obstacle distance cost and repulsive penalty for obstacles.

$$\begin{aligned} \mathbf{e}_k &= [p_k - \mathbf{p}_k] \\ J_{track}(p_k, \mathbf{p}_k) &= \mathbf{e}_k^T Q \mathbf{e}_k \\ J_{ang}(\phi_k, \boldsymbol{\phi}_k) &= (\phi_k - \boldsymbol{\phi}_k)^2 \\ J_{vel}(vel_k, \mathbf{vel}_k) &= (vel_k, \mathbf{vel}_k)^2 \\ J_{repulsive}(p_k, p_k^{obs}) &= \sum_{i=1}^{n} \frac{1}{(\delta x_k)^2 + (\delta y_k)^2 + \gamma} \end{aligned} \tag{2}$$

4) Solving LMPCC Problems: Local Model Predictive Contouring Control (LMPCC) is a variant of MPC that focuses on following a predefined path as closely as possible while also considering local dynamic constraints.

With current models and obstacles, the function is as follows:

$$\begin{aligned} J^* &= \min_{u} \sum_{k=0}^{N-1} J(z_k, u_k, \theta_k) + J(z_N, \theta_N) \\ \text{s.t.:} \quad & ([2]), \quad (1) \\ & z_{k+1} = f(z_k, u_k), \quad \theta_{k+1} = \theta_k + u_k \tau \end{aligned} \tag{3}$$

In which, the objective function quantifies the cost of control effort and deviation from the desired path. The conditions include the dynamic model of the system and both static and dynamic obstacle constraints. By solving this problem, the system manages to follow the reference path closely till the goal.

B. Obstacle Window Filtering (OWF)

We employ the cost function and constraint function discussed in Section II-A2 and II-A4 for optimization. However, through multiple experiments, it became evident that LMPCC tends to undergo slow optimization and, in some cases, fails to find feasible solution when the number of dynamic obstacles increases. Consequently, we have introduced a filter II-B for dynamic obstacles, excluding those that are far away. The rationale behind this is to eliminate obstacles that the robot may not reach within a certain timeframes, thereby allowing the LMPCC to focus only on the obstacles that are possible to cause collision.

Algorithm 1 Obstacle Window Filtering (OWF)

```
constraints_obs = {}
obs_remaining = N, thres_dis = a number
while obs_remaining > 0 do
    obs_remaining −= 1
    if dist(p_k, p_k^i) < thres_dis then
        Calculate C_k^{obst,i}
        Add C_k^{obst,i} to constraints_obs
    end if
end while
```

C. Dynamic Window Approach

The Dynamic Windows Approach (DWA) is designed to deal with the constraints imposed by limited velocities and accelerations of the robot. It is a sampling-based optimization in the velocity space.

A valid search space is reachable within the time interval while allowing the system to fully stop before the system collides with the closest object in the context of admissible velocities. All possible trajectories considered in optimization should fall in the search space and be uniquely determined by linear velocity(v) and angular velocity(ω).

Following is the objective function that we want to minimize over v and ω.

$$G(v,\omega) = \sigma(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot vel(v,\omega)) \tag{4}$$

In which, $heading(v,\omega)$ measures to what extent the robot is progressing towards the target, $dist(v,\omega)$ measures the distance away from the nearest obstacle and $vel(v,\omega)$ measures the forward velocity of the robot. The function σ smooths the weighted sum for better trajectories optimized from the objective function.

III. Results

A. Experiment Setup

1) Dynamic Models: Since the algorithm is aimed at UGVs navigating through crowds, we assume the robot has a radius of 0.5 m and moves at a reference speed of 1 m/s. It has a top speed of 4 m/s, a max acceleration of 0.85 m/s^2, and a max steering angle rate of 30 $^\circ/s$. We assume the UGV follows a bicycle kinematics model with the following dynamic equations.

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ \theta \\ v \end{pmatrix} = \begin{pmatrix} vcos\,(\theta) \\ vsin\,(\theta) \\ v\frac{tan(\delta)}{L} \\ a \end{pmatrix} \tag{5}$$

The dynamic obstacles are modeled to have a radius of 0.2 m and to move linearly in the room with a random initial speed and angle between 0.1 m/s and 1 m/s and 0 and 2π, respectively. If it hits the wall, it bounces off it, following the reflection law.

2) Simulation Environment and Performance Evaluation: We tested the performance of the methods by Python simulations and ran our experiments on an Intel Core i5-11300H processor. All the algorithms to have a time step of 0.1 s and a prediction horizon n = 10 (1 s). We also added an extra 0.05 m safety margin to increase safety. To solve the LMPCC, we used the SLSQP method and scipy.optimize.minimize as the solver. We sampled the DWA dynamic window with a resolution of 0.01 m/s and 0.1/s. We assume that the robot knows where every obstacle is and recorded five indexes to evaluate safety and efficiency performance:

- Task Time(s): Time it took to reach the goal.
- Collisions(s): The number of time frames with collisions.
- Collisions Percentage (%): Collisions over the Task time.
- Collision Speed (m/s): Average speed when collisions happen.
- Computation Time (s): Time to solve the optimization problem.

We set the Collision Percentage as a metric instead of considering only Collisions because moving slowly and spending more time on the task will also lead to collisions in more time frames. However, it does not mean to be unsafe since colliding at a slower speed is better than high-speed collisions.

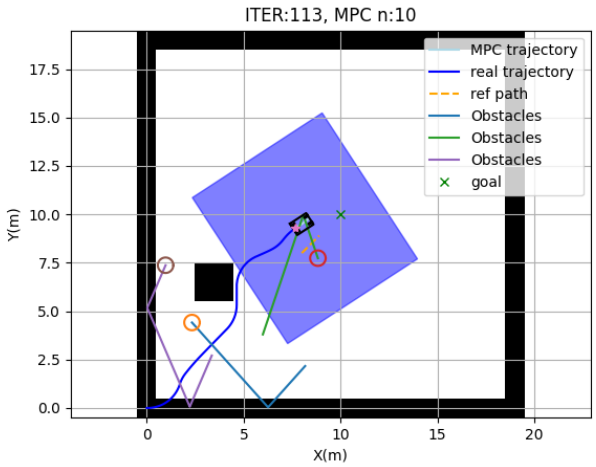


Fig. 1. The figure shows the robot (black hollow rectangle) navigating a room with a static obstacle(black square), and three dynamic obstacles(red, orange, and brown circles). It is targeted to reach a goal in the middle of the room. The blue rectangular region is the collision free space calculated by static obstacle constraints.

B. Static and Dynamic Obstacles

We achieved essential static and dynamic obstacle avoidance in simple environments as shown in 1 by implementing the LMPCC method[2]. The goal is placed at the middle of the room (10, 10). It averaged 0.20 s computation time for each time step, spent 12.8 s to reach the goal, and did not collide with any obstacles.

C. Dense Dynamic Obstacles

To test the effectiveness of the algorithms, we test them with 5, 10, 20, and 40 dynamic obstacles in a 10 m x 10 m room. The room is set as linear constraints in LMPCC and LMPCC with OWF but is treated as a surrounding obstacle in DWA due to the nature of the algorithm.

1) Local Model Predictive Contouring Control (LMPCC): Table I shows the result of LMPCC. It can consistently produce reasonable trajectories as in Fig. 2(a) & 2(d) although Task Time, Collisions, Collisions Percentage, and Computation Time all increases when the number of obstacles increases. We can observe that the collision speed is much lower than the reference speed, which is what we desire.

TABLE I
PERFORMANCE EVALUATION OF THE LMPCC SIMULATION RESULTS.

LMPCC				
Obstacles	5	10	20	40
Task Time (s)	18.5	21.8	22.1	64.1
Collisions (s)	1.8	1.4	1.8	29.2
Collision Percentage (%)	9.73	6.42	8.14	45.6
Collision Speed (m/s)	0.35	1.11	0.35	-0.31
Computation Time (s)	0.217	0.376	0.503	1.094

2) Dynamic Window Approach (DWA): DWA produces decent trajectories when the environment is simple as in Fig. 2(b), but may fail to produce feasible trajectories in crowded environments, as shown in Fig. 2(e). It can calculate the solution relatively fast since its speed mainly depends on the sampling resolution. The results in Table II show that the approach crashes for more than 10 obstacles, which leads to high crashing speeds when it fails to find a solution.

TABLE II
PERFORMANCE EVALUATION OF THE DWA SIMULATION RESULTS.

DWA				
Obstacles	5	10	20	40
Task Time (s)	15.0	27.4	21.5	21.6
Collisions (s)	0.2	4.9	6.2	11.9
Collision Percentage (%)	1.33	17.8	28.8	55.1
Collision Speed (m/s)	0.95	2.25	2.59	3.30
Computation Time (s)	0.104	0.0607	0.0482	0.0443

3) LMPCC with Obstacle Window Filtering (OWF): We used a 3 m obstacle window ($\text{thres}_{\text{dis}}$ = 3) for the test cases. The performance indexes of LMPCC with OWF shown in Table III have similar trends as the results of LMPCC. The trajectories are shown in as in Fig. 2(c) & 2(f).

TABLE III
PERFORMANCE EVALUATION OF THE LMPCC WITH OWF SIMULATION RESULTS.

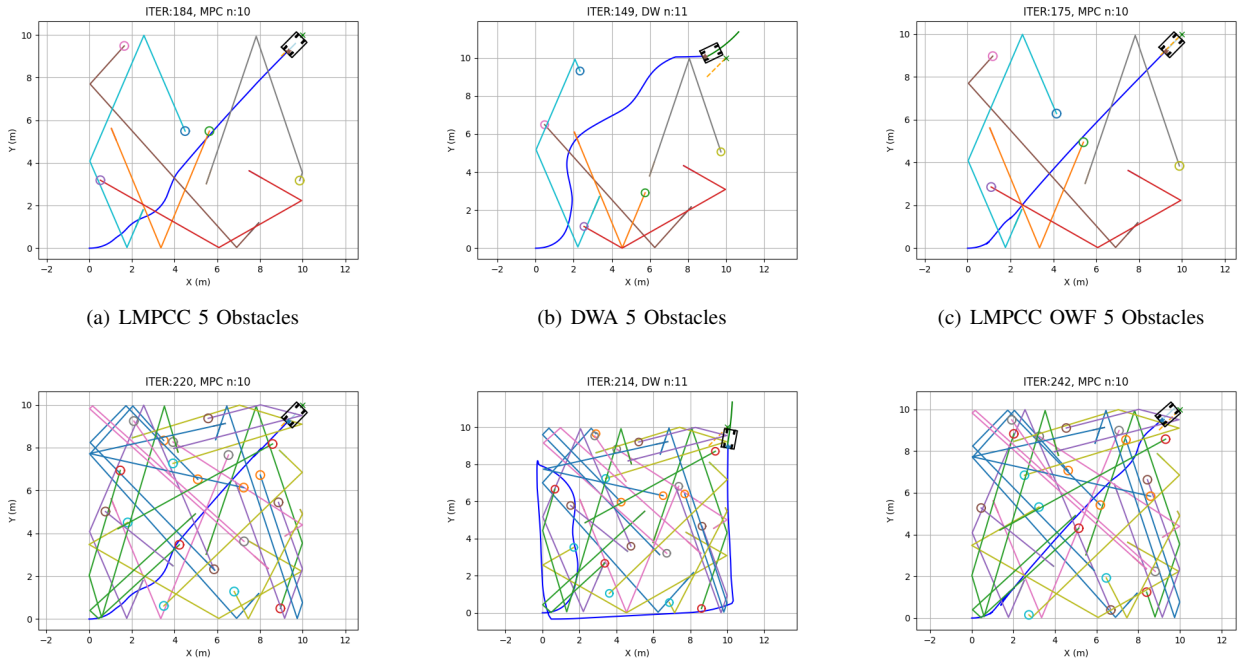
LMPCC with OWF				
Obstacles	5	10	20	40
Task Time (s)	17.6	21.6	24.3	49.3
Collisions (s)	0.9	0.9	1.3	12.5
Collision Percentage (%)	5.11	4.17	5.35	25.4
Collision Speed (m/s)	-0.22	-0.22	0.13	-0.14
Computation Time (s)	0.188	0.185	0.279	0.385

IV. COMPARATIVE ANALYSIS

1) Safety: From the results above, we can observe that LMPCC with OWF performs best considering safety. The Collision Percentage is the lowest for more than 10 obstacles, and it collides at the lowest speed in most cases. We speculate that the trajectories produced by LMPCC with OWF are better optimization results by discarding irrelevant constraints and costs. LMPCC also outperforms DWA, which shows that modeling future control inputs is crucial to navigating crowds. However, with only five obstacles, DWA collided the least, although at a higher speed. This result shows a potential to combine the two algorithms with hierarchical control.

2) Robustness: For real-life applications, performing in all kinds of scenarios is essential. Considering the results from our experiments, LMPCC and LMPCC with OWF are more robust in different scenarios. While experimenting with these algorithms, we found that tuning parameters is critical for DWA to produce feasible results. Therefore, research [5] has been conducted to fine-tune the parameters, and it showed massive performance improvements, which would be interesting to compare with LMPCC in the future.

3) Computation Efficiency: The calculation speed of the LMPCC is not fast enough to meet the targeted update rate; that is a 0.1 s time step (10 Hz) in this experiment. The lack of speed is not only because we coded it in Python and did not



(a) LMPCC 5 Obstacles (b) DWA 5 Obstacles (c) LMPCC OWF 5 Obstacles

(d) LMPCC 20 Obstacles (e) DWA 20 Obstacles (f) LMPCC OWF 20 Obstacles

Fig. 2. The results of these figures are the three algorithms tested with 5 and 20 obstacles. In the images above, LMPCC generates a smooth trajectory to reach the goal. DWA performed well under 5 obstacles but failed to find a feasible solution midway and followed the wall to the goal under 20 obstacles. LMPCC with OWF also provides a smooth trajectory to the goal with better reactive motions.

use a specialized optimization toolkit to solve for the solution but also because we set naive dynamic obstacle constraints. Its ability to solve problems in real-time simple environments has been shown in [2], but more innovative ways to design constraints should be investigated for applications in crowded environments.

4) Task Efficiency: The LMPCCs took less than 1 minute to move about 15 m in an area with 0.4 $obstacles/m^2$, which is acceptable without the environment reacting. The DWA Task Time is short because it fails to find a solution that avoids obstacles and speeds its way to the goal.

V. Conclusion & Future Works

LMPCC is better than DWA in more complex environments. However, how it deals with dynamic obstacles is naive and can be improved. By applying OWF, one can achieve safer results than considering every obstacle. Future extensions include research into why OWF affects the result and consider human reactions to the robot's presence in MPC since multiple research[1][4] show that cooperative planners work better than noncooperative planners.

VI. Acknowledgment

The DWA code was modified from Python Robotics, and great thanks to Professor Posa and TAs for the fantastic MEAM 5170 course.

References

[1] Trautman, P., & Krause, A. (2010, October). Unfreezing the robot: Navigation in dense, interacting crowds. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 797-803). IEEE.

[2] Brito, B., Floor, B., Ferranti, L., & Alonso-Mora, J. (2019). Model predictive contouring control for collision avoidance in unstructured dynamic environments. IEEE Robotics and Automation Letters, 4(4), 4459-4466.

[3] D. Fox, W. Burgard and S. Thrun, "The dynamic window approach to collision avoidance," in IEEE Robotics & Automation Magazine, vol. 4, no. 1, pp. 23-33, March 1997, doi: 10.1109/100.580977.

[4] Trautman, P., Ma, J., Murray, R. M., & Krause, A. (2013, May). Robot navigation in dense human crowds: the case for cooperation. In 2013 IEEE international conference on robotics and automation (pp. 2153-2160). IEEE.

[5] M. Dobrevski & D. Skočaj, "Adaptive Dynamic Window Approach for Local Navigation," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 6930-6936, doi: 10.1109/IROS45743.2020.9340927.