Autonomous Crowd-crossing Navigation with Hybrid Path Planning

Zhenya Guo, Kian Ming Phua, Yu Song, Yi Jie Tan, Peizhang Zhu

***Abstract*—This report presents an MPC controller design for local path planning of a delivery robot in an environment with static and moving obstacles. We referenced the work in the paper "Socially Aware Motion Planning with Deep Reinforcement Learning", and applied the model in a 2D vehicle. To assist the path planning algorithm, MATLAB Robotic System Toolbox was used. Given a start and end location, the pre-computation phase evaluates the sub-optimal global path which leads the robot to the target area. Each sub-step is controlled by a local MPC controller for obstacle avoidance and optimization. A static map with moving obstacles to simulate humans was used in the simulation. The controller design was tested with perfect model knowledge. In the simulation video, we can observe the delivery robot taking an optimal path to the destination with obstacle avoidance. The advantage of the design lies in the integration of an MPC controller for optimal obstacle avoidance with a global planned path.**

I. Presentation

Video Link: https://youtu.be/kRw6bl3umm8

II. Introduction

In 2017, revenue of the courier and delivery service industry is expected to be $247.9 billion globally [1] and $92.5 billion in the US [2]. While the consumer spending and the internet usage is growing extremely, the delivery industry is limited by volatile fuel prices and rising labor costs. [1] For most courier industry operators in US, the expenditure on labor is more than eight times than on capital investment. [2] Huge development space is left for automation in the industry. Using of autonomous delivery robot could replace most of the heavy labor work and save fuel energy by using electrical energy. Researchers from MIT developed a robot vehicle for navigating in pedestrian-rich environment. [3] The paper shows possibilities in hardware sensing and controller design for autonomous driving in crowded environment. [3] Our project will follow work in this paper, and design a Model Predictive Control (MPC) controller for a 2D vehicle. Different from other autonomous vehicle, safety is more important for delivery robot than optimized path planning. Thus, besides path planning to a predefined destination, the project will also emphasize the static and dynamic obstacle avoidance.

III. Model

In this project, a unicycle robot is used as our robot model as shown in Figure 1. The state variables include the robot's current position (x, y) in the 2D X-Y plane and the orientation of the robot (θ). The input to drive the robot comprises linear velocity and angular velocity (v, ω), which could be computed by the Pure Pursuit controller included in MATLAB Robotics System Toolbox. By setting the desired target location in the 2D plane, the Pure Pursuit controller sets the necessary input variables. The 2D map used to represent an actual environment is displayed in Figure 2. The white areas in the map are empty regions for the robot to traverse freely while black areas are walls. This is achieved with the use of a 2D Binary Occupancy Grid, where each cell has a value of 1 or 0 to indicate occupancy status. Furthermore, the 3 black boxes are dynamic obstacles moving along a linear path within the white areas. These blocks simulates the human presence in a crowded environment to achieve our aim of autonomous static and dynamic obstacle avoidance. To simulate a robot equipped with Light Detection and Ranging (LIDAR), the robot in simulation is only aware of obstacles within the radius of the blue lasers. Any obstacle beyond the laser is deemed too far to be relevant for the controller's obstacle avoidance system in the current computation time step. Also, the LIDAR works within a 180 degrees frame relative to the robot's current orientation. We will employ the above model and control input to simulate the successful navigation process of a delivery robot from start location to target location with obstacle avoidance via a local MPC.

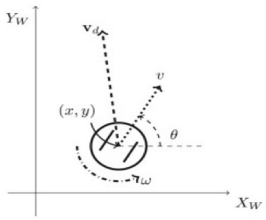

Figure 1: Robot Pose Tracking in 2D X-Y Plane [4]

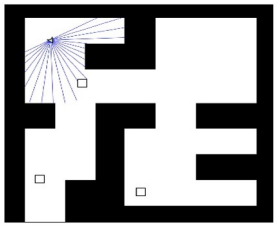

Figure 2: Map with LIDAR Equipped Robot and 3 Moving Obstacles

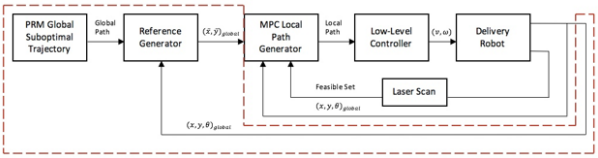

Figure 3: Robot Pose Tracking in 2D X-Y Plane

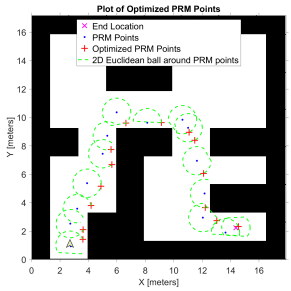

Figure 4: Optimized Probabilistic Roadmaps

IV. Techniques

A. Global Sub-optimal Path Planning (Figure 3)

1) Drawbacks in early iterations: The dynamic car model in MPC slowed down its optimization. In the first version, we included a discretized form of traditional dynamic model with time-step T_s

$$x(k+1) = x(k) + v(k)T_s\cos(\theta(k))$$
$$y(k+1) = y(k) + v(k)T_s\sin(\theta(k))$$
$$\theta(k+1) = \theta(k) + w(k)Ts$$

to calculate the velocity and angular velocity needed for driving the car. [5] However, this model added complexity to the optimization process and prolonged the MPC's calculation time, so the car's motion is discontinuous. Thus, we excluded the traditional dynamic model in the later iterations and let MPC calculate only the optimal positions where the car should go and feed it to another position-tracking controller to execute the drive operation. The MPC controller then functions like a navigator, and each optimization takes about 50% less time.

But still, MPC is shortsighted. In the first two versions, MPC merely tries to get closer to the destination at all time, without viewing the map as a whole and making higher-level path plans. Therefore, there is a need for a higher path plan strategy.

2) Global Path Plan Strategy: MPC controller does not give the car a real path plan, so we employed the optimized probabilistic roadmaps approach to produce an approximate global guidance for the car to follow. This approach first generates 50 randomly scattered nodes all over the obstacle-free space of the map, then a set of nodes with a given relative distance from each other is selected. These nodes connect the start and end location of the path. We generated the basic probabilistic roadmap by using the PRM class in the robotics toolbox in MATLAB.

The basic probabilistic roadmap has a problem: since the nodes are randomly located, the path formed by the selected nodes can be not a very direct path to the destination. We addressed this problem by (1) creating a 2-dimensional Euclidean ball around selecting nodes, (2) employing a reference generator that can produce a reference point on the circle which is closest to the destination and does not fall in obstacles, and (3) setting a large enough (2 meters) threshold radius around the reference point. As shown in figure 4, these reference points (Optimized PRM Points) are the short-term targets for MPC. The generator feeds the nearest reference point to MPC, which then calculates a 2 by N optimal coordinates vector and outputs it to the position tracking controller. This controller takes the middle time-step of the optimal vector as the target of current driving operation. As the car enters the threshold radius of the next reference point, the generator receives this threshold feedback and immediately feeds this new reference point to MPC, which keeps the car from trying to precisely hit each reference point. This strategy results in a more straightforward and optimal path, and provides the car with a global path plan.

B. MPC Local Path Planner with Obstacle Avoidance

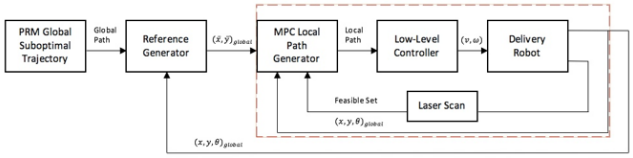

Figure 5: Robot Pose Tracking in 2D X-Y Plane

The inner-loop control system consists of four block : MPC Local Path Generator, Low-Level Controller, Delivery Robot and Laser Scan. The Laser Scan, using sensor data, would provides MPC a feasible region. MPC is used to find the optimal feasible local path to track the reference point. And the Low-Level Controller is responsible to drive the delivery robot to follow the local path. At the same time, the Delivery Robot would send the current states to each generators. Construction of Feasible Region In the part of Laser Scan, as showed in the picture, the delivery robot would emit 21 laser beams from its forward half space. The laser beams separates each other by 9 degrees and would sense the obstacle at the most of 5 meters distance from robot itself. The data from the laser scan would be the range of the laser beam and its local position angle, which are made of local polar coordinate system. Our method is to use these data to construct a feasible region by using the MATLAB function - $polyhedron()$. This function would automatically construct the polyhedron for you when each laser beams' position has been input to it. [6]

1) Local MPC Path Planner: The biggest problem of the feasible region is that the region is not a convex set which will lead the MPC to get a local other than global optimum path to track the reference points and even sometimes it fail to find one. To address this problem, we introduce a new method to

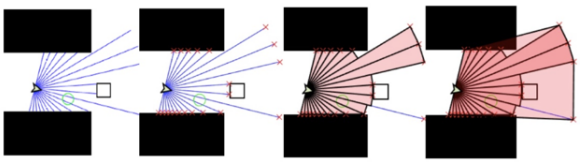

Figure 6: Generating feasible sets with LIDAR

solve this problem. The solution to this problem we introduced is to make the feasible region into a minimum convex set by MATLAB function, as showed in the picture. Since the cost function is quadratic function $(x_i - x)^2 + (y_i - y)^2$, and the feasible set is convex polyhedron which can be made of inequalities $Ax \leq b$, the problem become a linear quadratic convex problem. This will ensure the MPC path planner always has a global optimal solution to the optimal control.

Due to the manipulation of convex set, the optimal path generated by MPC could be out of feasible region. However, it is obvious that the path is always begin at the local original point and extend from the feasible region into the infeasible region. This condition would bring a great solution to the previous problem. What MPC compute out is a branch of discrete points, and what we need to do is only take data from first point and the point which its next point would not be inside the feasible region. The method could be implemented by MATLAB function - $inpolygon()$. As a result, the MPC would compute out a obstacle-free local path for low-level controller to track the reference point.

To its optimality, the solution might not be a global optimal one given the manipulation of convex set. However, we consider the safety issue and success of each MPC iterations are our priority for this project. Our result could be very useful for delivery robot where the optimality issue is not so important.

2) Implemented Algorithm: The MPC algorithm is implemented as follows:

$$\min_{x_{\text{local}}^{1\to N}, y_{\text{local}}^{1\to N}} 10\Big((x_{\text{global}}^{N} - \bar{x})^2 + (y_{\text{global}}^{N} - \bar{y})^2\Big) + \sum_{k=1}^{N-1}\Big((x_{\text{global}}^{N} - \bar{x})^2 + (y_{\text{global}}^{N} - \bar{y})^2\Big)$$

$$\text{s.t.} \begin{bmatrix} x_{\text{global}}^{k} \\ y_{\text{global}}^{k} \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x_{\text{local}}^{k} \\ y_{\text{local}}^{k} \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\forall k = 1, ..., N$$

$$A \begin{bmatrix} x_{\text{local}}^{k} \\ y_{\text{local}}^{k} \end{bmatrix} \leq b, \forall k = 1, ..., N$$

$$(x_{\text{local}}^{1} - 0)^2 + (y_{\text{local}}^{1} - 0)^2 \leq a$$

$$(x_{\text{local}}^{k} - x_{\text{local}}^{k-1})^2 + (y_{\text{local}}^{k} - y_{\text{local}}^{k-1})^2 \leq a, \forall k = 2, ..., N$$

where (x, y) is the robot's global current state and feedback to the MPC and reference generator at any time. The matrix A, and vector b that defines the convex polyhedral set could be retrieved from MATLAB function $polyhedron()$. The last two inequality is to ensure the path approximately continues and starts at original point in the local coordinate system, where a could be defined as any small number.

For each MPC iteration, we choose half path as our controller's waypoints to reduce the computation time and strengthen the ability to resist uncertainty.

V. Results

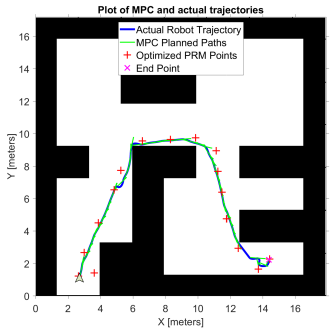

Figure 7: MPC paths and actual trajectories

The path planned by the MPC controller at each stage, as well as the actual trajectory taken by the robot is shown in Figure 7. The video of the entire simulation can be found here (insert link), with moving obstacles emulating walking pedestrians. The simulation shows that the controller achieves the objective of reaching the destination while avoiding moving pedestrians and the fixed obstacles in the map.

On average, the MPC controller takes about 3 seconds of CPU time to solve each optimal control problem in the simulation with YALMIP. This can be problematic if this technique were to be implemented real-time in an embedded system, for example. A potential way to circumvent this problem would be to use a more direct numerical method to solve the optimal control problem, such as the batch approach, instead of a solver package.

VI. Future Work

Our group has identified two areas for development: model implementation and random movement of dynamic obstacles. Currently, the simple robot model adapted from the MATLAB Robotics Toolbox tracks only the global pose of the robot (x, y, θ) where these state space variables were used in the MPC design. Thus, the robot model can be improved by implementing a 2D car-like kinematic model as shown below. This will improve the effectiveness of the MPC in driving an actual robot through the steering angle and velocity.

Furthermore, to achieve the aim of obstacle avoidance in a crowded environment, more random "human" obstacles can be added to the map. These humans can traverse in random and non-linear paths to add dynamism to the simulation.

References

[1] IBISWorld, "IBISWorld Industry Report: Global Courier & Delivery Services," IBISWorld, 2017.

[2] D. Kalyani, "IBISWorld Industry Report 49222: Couriers & Local Delivery Services in the US," IBISWorld, 2017.

[3] Y. F. Chen, M. Everett, M. Liu and J. P. How, "Socially Aware Motion Planning with Deep Reinforcement Learning," 26 March 2017. [Online]. Available: https://arxiv.org/abs/1703.08862 . [Accessed 11 December 2017].

[4] Hasan A. Poonawala, Mark W. Spong, "Time-optimal velocity tracking control for differential drive robots," November 2017. [Accessed 11 December 2017].

[5] Adrian Filipescu, Viorel Minzu, Bogdan Dumitrascu and Adriana Filipescu, "Trajectory-Tracking and Discrete-Time Sliding-Mode Control of Wheeled Mobile Robots," June 2011. [Accessed 02 December 2017].

[6] Jiechao Liu, Paramsothy Jayakumar, Jeffrey L. Stein and Tulga Ersal, " A Multi-stage Optimization Formulation for MPC-based Obstacle Avoidance in Autonomous Vehicles Using A LIDAR Sensor," October 2014. [Accessed 04 December 2017].