MEAM 523 Final Report

Obstacle Avoidance: Path Planning and MPC Tracking

Jiacan XU, Weiyi TANG, Suyu ZHOU

1 Statement of the problem

In the area of unmanned ground vehicles, obstacle avoidance would always be the first problem that we should mainly concern. We have seen many previous works that tried to track predefined reference with fixed obstacles, but these methods would be too fragile to put in real scenario, since most of times we would never know where and when the obstacle will appear at the driving environment. Therefore, in our project, we tried to solve the problem of obstacle avoidance with randomly distributed unknown obstacles by using path planning methods and receding horizon tracking control.

In our project, the cart is running from a predefined source point to a destination point on a map with some fixed x and y boundaries. On the map we generate some randomly distributed obstacles with random sizes and shapes unknown to the cart. When the cart "detects" the obstacle, it will follow the path we planned to avoid collision.

We will present our project following three main features, including vehicle modeling, path planning and controller design. We simplified the model of a unmanned vehicle to have two driving wheels for vehicle modeling, and we used a simple A* method and artificial potential field method for path planning. As for controller design, we used receding horizon (RH) or to say, Model Predictive Control (MPC) for the tracking control of our nonlinear problem.

2 Vehicle Modeling

The first part of our project is to simplify the model of a vehicle to solve our problem. Therefore, we considered our cart as a differential driving mobile single-axle vehicle with front-steered wheel (two front driving wheels) and rolling rear wheels, and we put the center of the mass at the midpoint of two driving wheels. Assume there is no slip. The wheels have controllable speeds v_l and v_r, and track width equals to B.

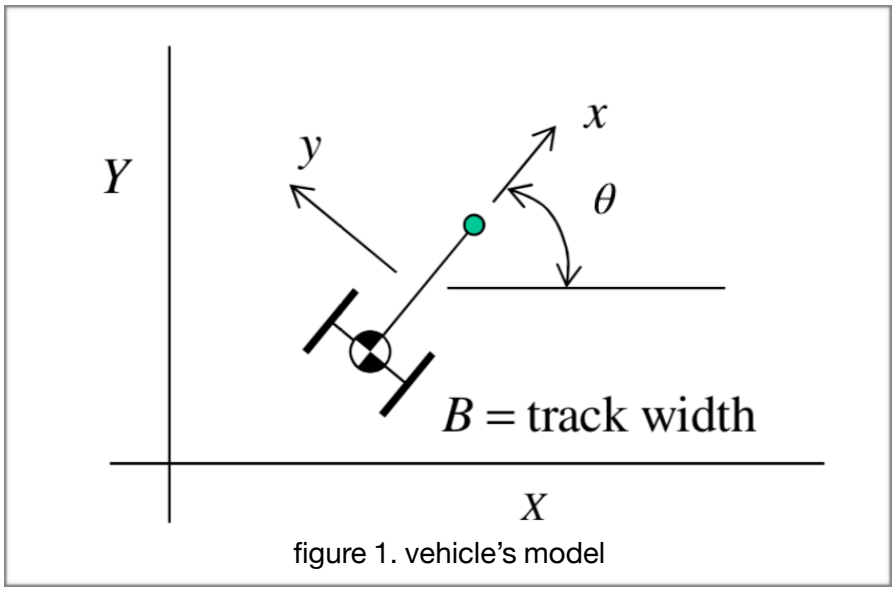


figure 1. vehicle's model

The speed control of the two front wheels leads to the control of linear speed and angular speed. The motion state of the cart can be described by the coordinates of vehicle (x, y). which is the position of the center of mass, and its orientation θ.

Linear Speed: $v = (v_l + v_r)/2$
Angular Speed: $v = (v_l - v_r)/B$

The kinematics equation is as follows:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} vcos(\theta) \\ vsin(\theta) \\ \omega \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} u$$

The state and input controls are denoted as $q = (x, y, \theta)^T$ and $u = (v, \omega)^T$.

3 Obstacle avoidance

3.1 Path Planning

To avoid random obstacles on a map with fixed x and y boundaries, we need to find a suitable path or trajectory with some given constraints, including obstacles and map boundaries. Moreover, we would like to generate an optimal trajectory which allows us to reach our destination with highest efficiency and shortest amount of time. In the next part of our project, we used two different method to do the path planning. One is A* Method, and another is Artificial Potential Field Method.

3.2 A* Methods

Let's first start with A* Method. A* Search Method is a classical informed search method in path planning. To begin with, we will make some notations.

$n \rightarrow$ node/state
$c(n_1, n_2) \rightarrow$ the length of an edge connecting between n1 and n2
$b(n_1) = n_2 \rightarrow$ backpointer of a node n1 to a node n2.

The basic concept of A* Method is to find the minimum of the evaluation function $f(n)$, which is the evaluated cost for an optimal path from start point to node/state n. The evaluation function as added by operating cost function $g(n)$ and heuristic function $h(n)$. The operating cost function denotes the cost from start point to node/state n, and the heuristic function denotes the cost from node/state n to destination node. In this method, we always want $h(n)$ to be less than or equal to the actual cost.
The core process of the A* Method is to choose from all expandable node for the next time step and pick the one with smallest evaluation value $f(n)$, and compute its

neighborhoods' $g(n)$, $h(n)$ and $f(n)$ until there is no more expandable node or we finally reach our destination node.

Then how do we use this method to avoid the obstacles? In our project, we simplified our environment to have only static obstacles and fixed starting point and goals. Using this method, we need no prior knowledge of the obstacles, but to make judgment for every time step when searching for the next node. And the search requires 2 lists to store information about nodes. One is the open list (O), where we store all nodes for expansions, and another is the closed list (C) where we store nodes which we have explored. To choose the next expansion, we need to choose in the open list and find the one with minimum $f(n)$. The following figures is one example to set and compute the cost. In our project, we used manhattan distance to compute the heuristic function.

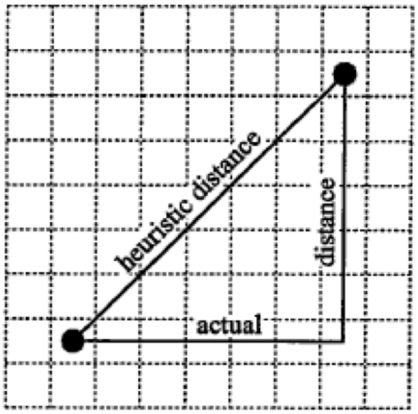


Figure 2. Heuristic Distance

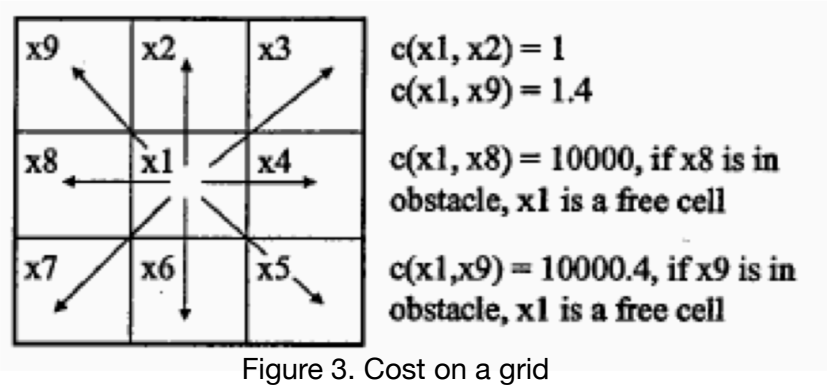


Figure 3. Cost on a grid

The algorithm is as follows,

1) Start with the starting node, and place the node into open list (O);
2) Search all possible neighborhoods of starting node and add them into list (O), and denote them as child nodes of the starting node. Then delete the parent node from the open list (O) and place it into close list (C);

3) Choose from the open list the one with smallest $f(n)$, delete it from (O), and add it into (C). Check its neighborhood, ignore those in close list or not expandable nodes, and add them into open list. Denote all new nodes as child nodes and the node we choose as parent node. If one of its neighborhood already exists in the open list, check if this path is better, or to say, with smaller $g(n)$. If yes, change the neighborhood's parent node into current node, and recalculate $g(n)$ and $f(n)$. If not, do nothing;
4) Repeat step (3), until we find our destination node;
5) From the destination node, generate a path to the starting point and find our final path.

In our implementation, we always wanted to increase our efficiency and avoid redundant calculation. Therefore, in the first step, we simply create a vector to point from the cart's current location to the destination, until the cart come across an obstacle. Then implement the algorithm to let the cart rotate by some angle for the next time step, until there would be no possible collision for the next time step, and repeat this process until we reach our destination.
The final trajectory of the cart is as follows.

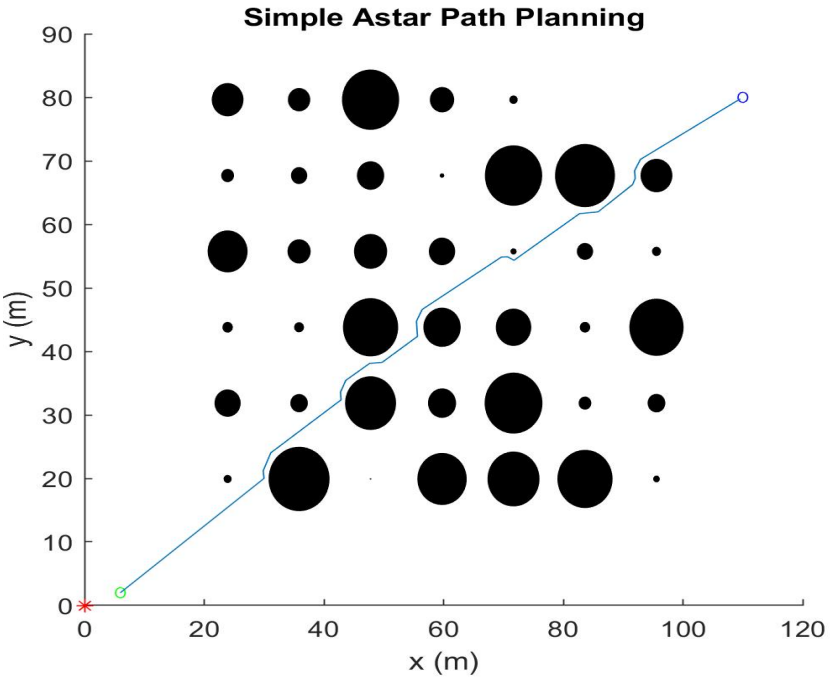
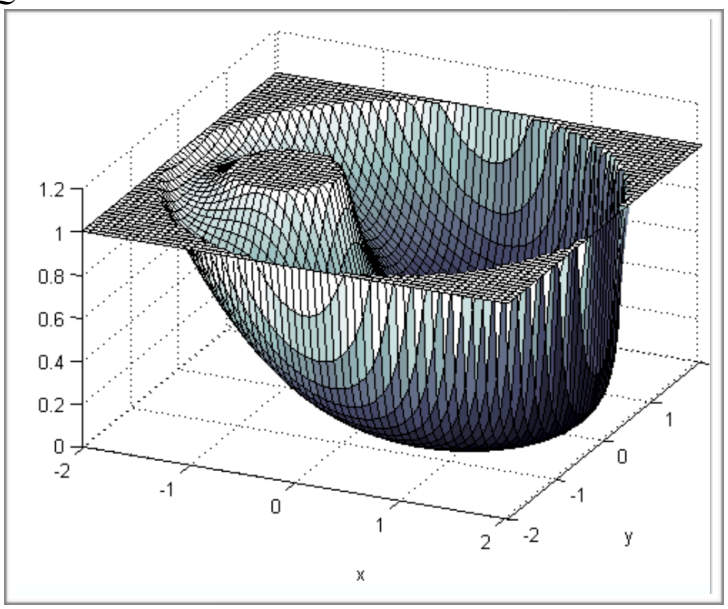


Figure 4. Path planning using A*

3.3 Artificial Potential Field

From the section above, we can generate a possible path that allows the cart to go from source point and destination point without collision. However, using A* star method is not very efficient, and we might get into trouble if the obstacles are not convex. Therefore, we tried to use the artificial potential field to generate a more efficient path. The main part of forming an artificial potential field is by forming a navigation function

φ that has only one global minimum at our desired goal state $q^* \in Q$, Where Q is the configuration space, which we defined as a sphere world —— a disc punctured by an arbitrary number of smaller disjoint discs representing obstacles. The navigation function is at least twice continuous and is Morse on Q, and it is admissible, which means $\partial Q = \varphi^{-1}(1)$ or $\partial Q = 0$.



Fugure 5. Model sphere world

Before forming the navigation function, we need to make some notations. Denote q as the position of the cart, q^* as the coordinate of our destination, ρ_0 as the radius of the sphere boundary, q_i as the coordinate of the obstacle, ρ_i as the radius of corresponding obstacle. Then we can form some preliminary functions. Define the current distance to goal function as $\gamma_k = \|q - q^*\|^{2k}$, where k is a design parameter and $k > 0$, and define obstacle functions and combined obstacle function as follows:

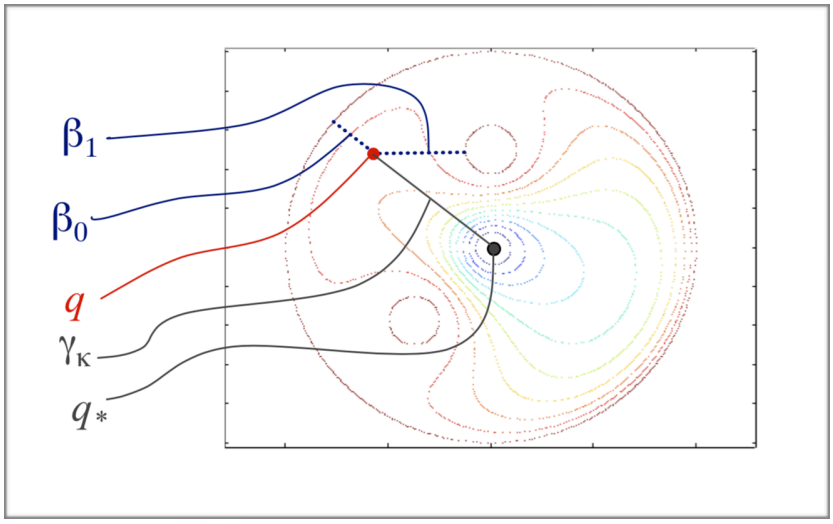


Figure 6. Configuration space

$$\beta_0 = -\|q - q_0\|^2 + \rho_0^2$$
$$\beta_i = \|q - q_i\|^2 - \rho_i^2, i = 1,2,...,M$$
$$\beta = \prod_{i=0}^{M} \beta_i$$

Then we can design our navigation function,

$$\varphi(q) = \frac{\gamma_1(q)}{(\gamma_k(q) + \beta(q))^{1/k}}$$

Implement this method using Matlab, we get the following results.

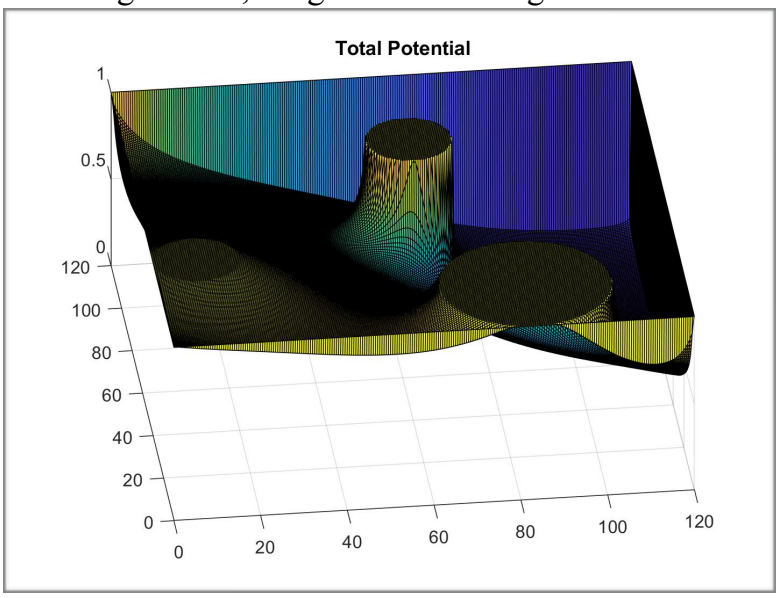


Figure 7. Configuration space with 3 obstacles

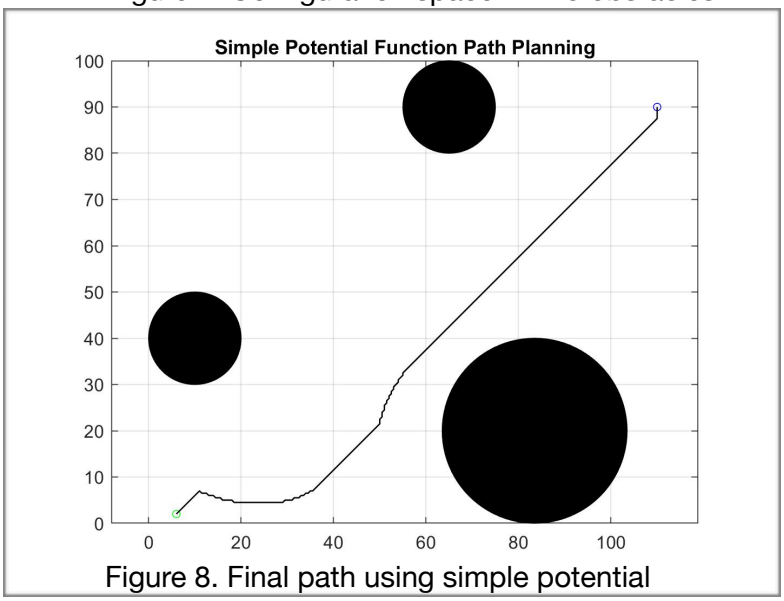


Figure 8. Final path using simple potential

However, we might also get into some troubles using this method. For example, we might get stuck to a local minima and we have to iterate hundreds of times before the cart

finally find a way out of the local minima. We might also face the situation that the cart goes along the boundary or goes to the saddle points.

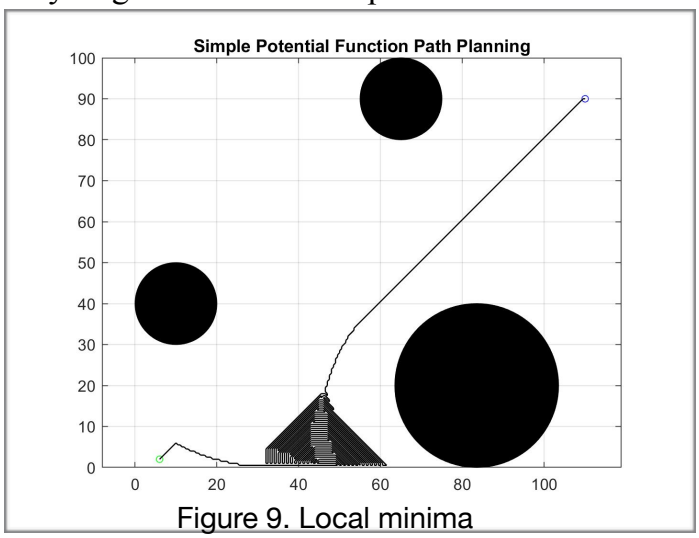


Figure 9. Local minima

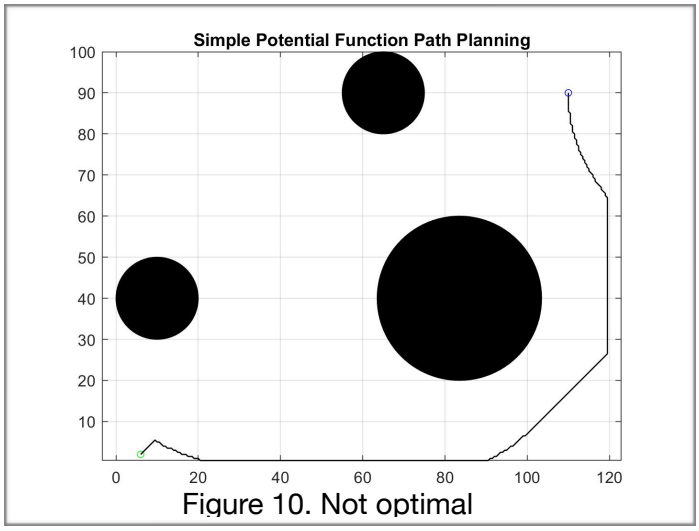


Figure 10. Not optimal

To solve these problems and make our method more robust, we need to make some changes to this method.

First, in the simulation, we will not using a sphere model. Instead, we generate our configuration space as binary image, and compute the immediate distances to the obstacles and goals. In such a manner immediate actions lead to motion of the cart, ultimately leading to the goal. All obstacles repel the cart with a magnitude inversely proportional to the distance, and the fixed goal point attracts the robot. The attraction should be strong enough or we will get stuck. The resultant potential, accounting for the attractive and repulsive force adding to the cart is measured and used to move the cart, therefore generate a trajectory of the cart.

To avoid the problems presented above, we need to set a distance threshold when computing the immediate distance between cart and obstacles. Denote this distance as d_0.

Repulsive function: $R = \nu(\frac{1}{d} - \frac{1}{d0})^2$; $R = 0$ when $d > d_0$;
Attractive function: $A = \xi\|p - p^*\|^2$

Where ν and ξ are scaling factors for repulsive potential and attractive potential. Therefore we get the following trajectory.

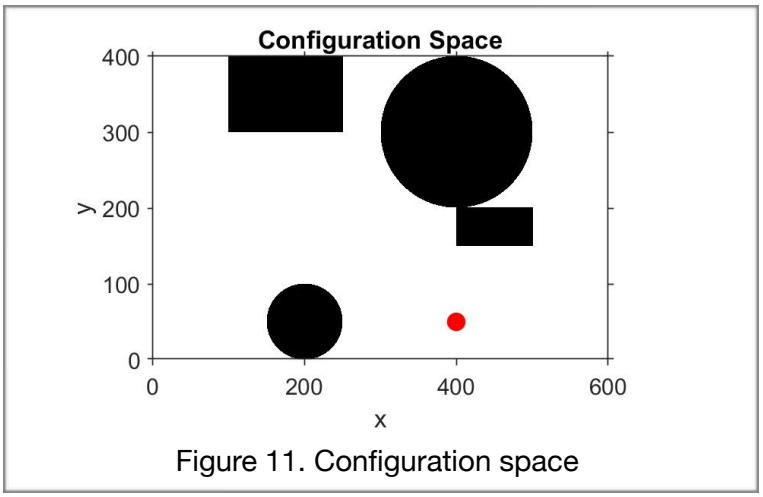


Figure 11. Configuration space

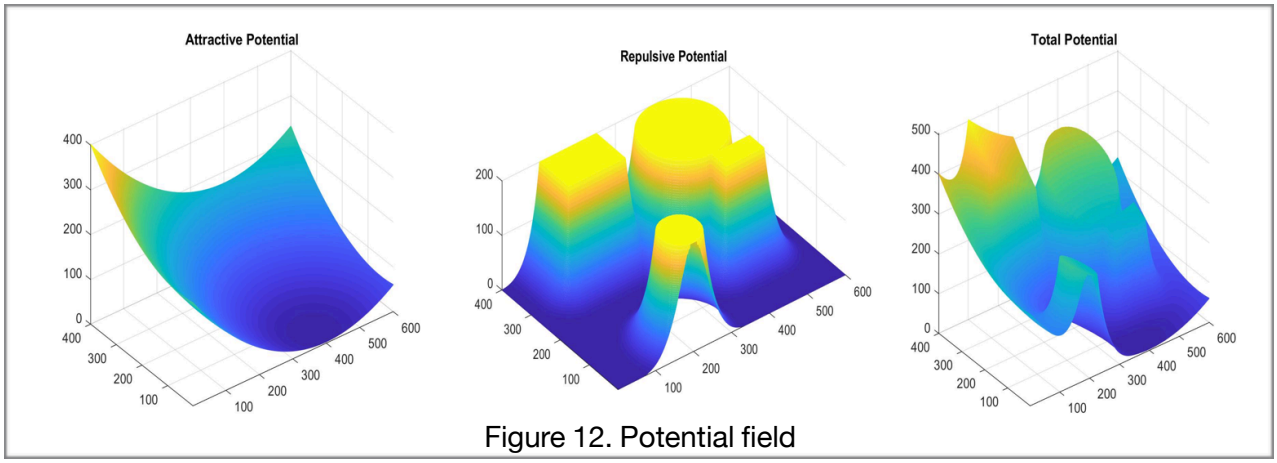


Figure 12. Potential field

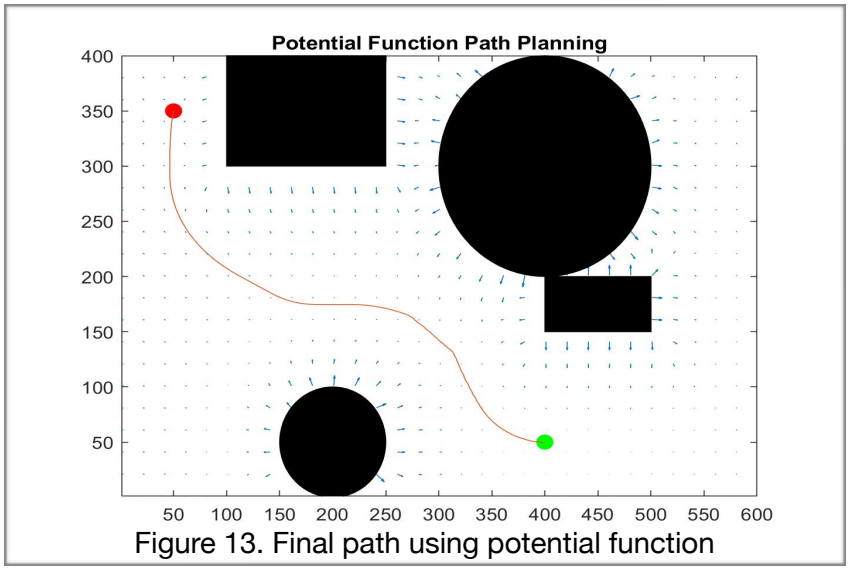


Figure 13. Final path using potential function

4 Model Predictive Control

4.1 Concept of MPC

After we generate an ideal trajectory of the cart, we need to track the trajectory using controller. Here we will apply Model Perspective Control. This method is very effective when we have constraints on states and input. It will give us an optimal solution with these constraints. Furthermore, Receding horizon control (RH) will be much helpful when we want to avoid obstacle without knowing the whole map and facing fast moving obstacles, which is our future works.

4.2 Error dynamic model

First, we need to form the dynamic equation of our system. We will use error dynamic equation and we will show that $\mathbf{x_e} = 0$ is the stable EQL of this system when we apply specific control law. So our error will eventually come to zero.

Instead of using the common definition of error state, we are going to use the form below.

$$\mathbf{x_e} = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix}$$

Where $\mathbf{x_r} = (x_r, y_r, \theta_r)^T$ is the reference state vector.

In other words, the error states is the transformation of the reference states in the local coordinate with the origin (x, y), which you can see from the picture below.

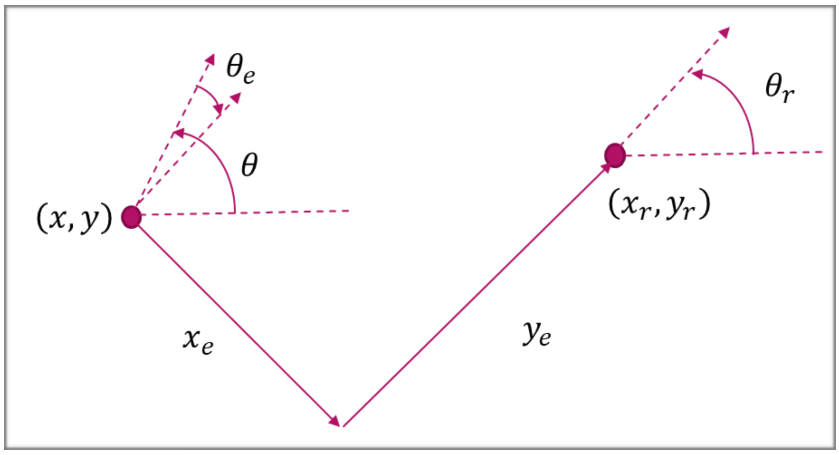


Figure 14. Error state and reference state

Then we can take the derivate of our states, then we will get:

$$\dot{x}_e = w y_e - v + v_r \cos\theta_e$$
$$\dot{y}_e = -w x_e + v_r \sin\theta_e$$
$$\dot{\theta}_e = w_r - w.$$

Here we will define a special control law:

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_r cos\theta_e + K_x x_e \\ \omega_r + v_r(K_y y_e + K_\theta sin\theta_e) \end{bmatrix}$$

Where $K_x, K_y, K_\theta > 0$.
Then we will have the combine expression of our dynamic:

$$\begin{bmatrix} \dot{x_e} \\ \dot{y_e} \\ \dot{\theta_e} \end{bmatrix} = \begin{bmatrix} \left(\omega_r + v_r\left(K_y y_e + K_\theta sin\theta_e\right)\right) y_e - K_x x_e \\ -\left(\omega_r + v_r\left(K_y y_e + K_\theta sin\theta_e\right)\right) x_e + v_r \sin\theta_e \\ -v_r(K_y y_e + K_\theta sin\theta_e) \end{bmatrix}$$

Then we will show that if we use the control rule we define, $\mathbf{x_e} = 0$ is a stable EQL when the reference velocity $v_r > 0$.
The way to prove this is to find a function V as a Lyapunov function:

$$V = \frac{1}{2}({x_e}^2 + {y_e}^2) + \frac{1}{K_y}(1 - cos\theta_e)$$

Let us test this function, first it is obvious that $V > 0$ for all points and $V = 0$ only holds at the EQL. Then we compute its derivate:

$$\dot{V} = \dot{x_e} x_e + \dot{y_e} y_e + \frac{\dot{\theta_e} sin\theta_e}{K_y} = -K_x x_e^2 - v_r K_\theta sin^2\theta_e / K_y \leq 0$$

So till now, we have proved that $\mathbf{x_e} = 0$ is a stable EQL when the reference velocity $v_r > 0$.
Then we do linearization of the nonlinear system we have before:

$$\dot{X_e} = \begin{bmatrix} -K_x & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & -v_r K_y & -v_r K_\theta \end{bmatrix} X_e$$

It is easy to see that $Re(\lambda_i) < 0$.
So we can show the uniformly asymptotically stability around $\mathbf{x_e} = 0$.
Finally, we will have our final dynamic by adding our input and B matrix:

$$\dot{\bar{\mathbf{x}}}_e = \begin{bmatrix} 0 & w_r & 0 \\ -w_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix} \bar{\mathbf{x}}_e + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_e$$

4.3 Cost function and Constraints

Then it is very important to figure out what kind of things we should optimize. So we need to form cost function which can be seen below:

$$J(t, x_e, u_e) = \frac{1}{2} x_e(t+T)^T x_e(t+T) + \int_t^{t+T} x_e(\tau)^T Q x_e(\tau) + u_e(\tau)^T R r_e(\tau) d\tau$$

Where $Q = Q^T \geq 0$ and $R = R^T \geq 0$

The first part at the right hand side is our terminal cost, and the rest part is stage cost.

In order to solve this optimal problem (it is continuous so it is hard to solve), we discrete our system and we will use Euler Discretization to form our new dynamic:

$$x(k+1) = (I + T_s A)x(k) + T_s B u(k)$$

Then we can write our optimal problem as:

$$J^*_{j \to N}(x_e, u_e) = \min_{u_e} \frac{1}{2} x_e(N)^T x_e(N) + \sum_{k=j}^{N-1} (x_e(k)^T Q x_e(k) + u_e(k)^T R r_e(k))$$

S.t.

$x_{j+k+1} = A x_{j+k} + B u_{j+k}, k = 0,1,...,N-1$

$x_{j+k} \in X, u_{j+k} \in U, k = 0,1,...,N-1$

$x_{j+N} \in X_f$

$x_j = x(j)$

In order to guarantee our terminal cost is control Lyapunov function, we need to satisfy this inequality:

$$\dot{g}(\mathbf{x}_e(t)) + L(t, \mathbf{x}_e(t), \mathbf{u}_e(t)) \leq 0$$

It will give us the constraint of our terminal state:

$$\frac{(v_r \cos\theta_{eT} - v_{\max})}{\alpha} \leq x_{eT} \leq \frac{(v_r \cos\theta_{eT} + v_{\max})}{\alpha}$$
$$-\frac{(w_{\max} + w_r)}{\beta} \leq \theta_{eT} \leq \frac{(w_{\max} - w_r)}{\beta}.$$

4.4 Receding horizon control

The overview of our track strategy can be seen below.

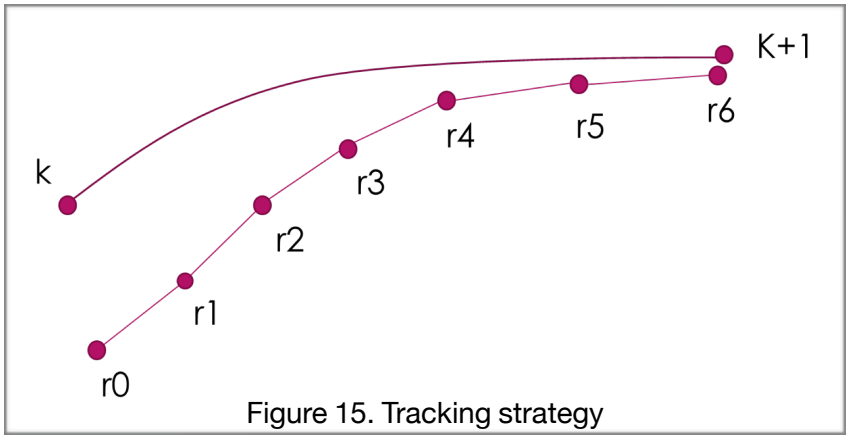


Figure 15. Tracking strategy

K-K+1 is the path controller we need to track. r_0 is the initial condition or the previous state we compute. When the cart is at r_0, the optimal problem above is solved then return a sequence of input, then we pick the first one and apply it to our system to compute next state. So we will move forward to r_0. We do this calculation over and over again to approach reference path.

4.5 Solver

Batch approach is introduced here to solve the Constraints Finite Time Optimal Control. Here we will use this method with substitution.

First, substitute all our dynamic into one equation which is the function of initial condition and input sequence:

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} I \\ A \\ \vdots \\ \vdots \\ A^N \end{bmatrix} x(0) + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix}$$

The equation above can be represented as

$$\mathcal{X} := \mathcal{S}^x x(0) + \mathcal{S}^u U.$$

Define:

$$\bar{Q} := blockdiag(Q, \ldots, Q, P) \text{ and } \bar{R} := blockdiag(R, \ldots, R)$$

Then the finite horizon cost function can be written as:

$$J_0(x(0), U) = \chi^T \bar{Q} \chi + U^T \bar{R} U$$

Eliminating χ and the above equation becomes:

$$J_0(x(0), U) = (S^x x(0) + S^u U)^T \bar{Q}(S^x x(0) + S^u U) + U^T \bar{R} U = U^T H U + 2x(0)^T F U + x(0)^T S^{x^T} \bar{Q} S^x x(0)$$

Then we need to rewrite our constraints:
If χ, $\mathcal{U}$ and χ_f are given by:

$$\chi = \{x \,|\, A_x x \leq b_x\}; \mathcal{U} = \{u \,|\, A_u u \leq b_u\}; \chi_f = \{x \,|\, A_f x \leq b_f\}$$

Then $G_0 U_0 \leq w_0 + E_0 x(0)$, where G_0, E_0 and w_0 are defined as follows,

$$G_0 = \begin{bmatrix} A_u & 0 & \dots & 0 \\ 0 & A_u & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_u \\ 0 & 0 & \dots & 0 \\ A_x B & 0 & \dots & 0 \\ A_x AB & A_x B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_f A^{N-1} B & A_f A^{N-2} B & \dots & A_f B \end{bmatrix}, E_0 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -A_x \\ -A_x A \\ -A_x A^2 \\ \vdots \\ -A_f A^N \end{bmatrix}, w_0 = \begin{bmatrix} b_u \\ b_u \\ \vdots \\ b_u \\ b_x \\ b_x \\ b_x \\ \vdots \\ b_f \end{bmatrix}$$

Finally, we can rewrite our optimal problem as:

$$\begin{aligned} J_0^*(x(0)) = \min_{U_0} \quad & [U_0' \; x(0)'] \begin{bmatrix} H & F' \\ F & Y \end{bmatrix} [U_0' \; x(0)']' \\ \text{subj. to} \quad & G_0 U_0 \leq w_0 + E_0 x(0) \end{aligned}$$

It is a quadratic optimal problem with constraint, so that we can easily solve this problem by using **quadprog** in MATLAB.

5 Matlab implementation and simulation results

Now we can implement all three features of our project and do some simulations using Matlab.

First, we used A* path planning method and MPC tracking.

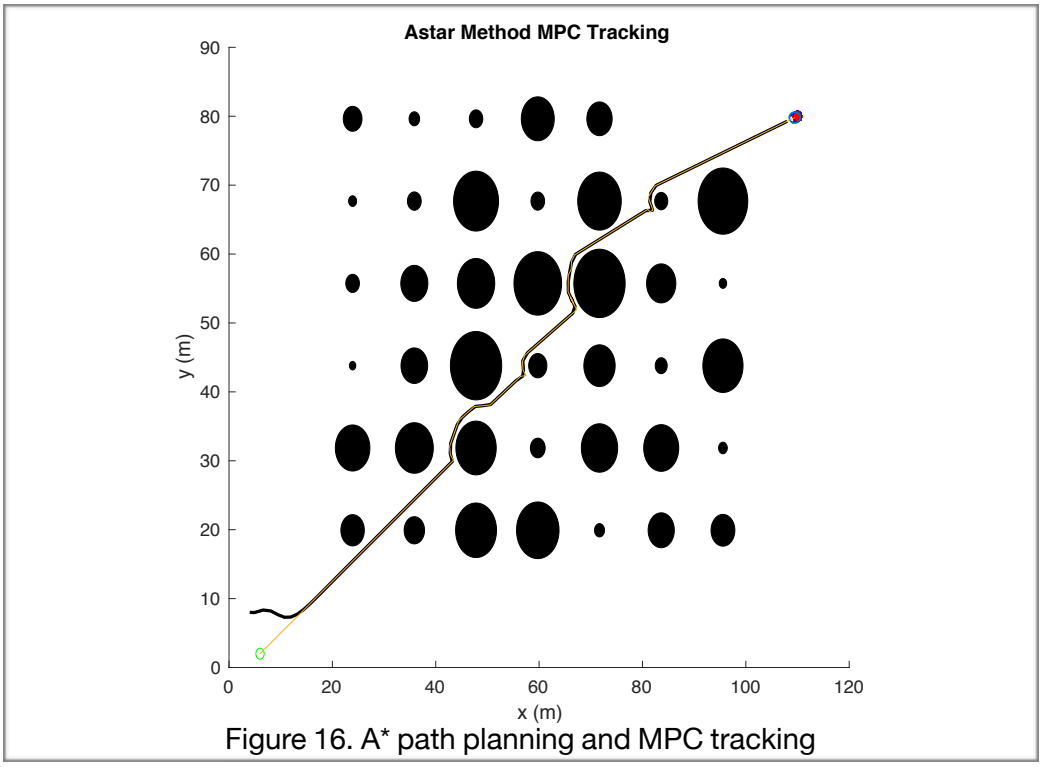


Figure 16. A* path planning and MPC tracking

Then, we used artificial potential field method for path planning and used MPC for tracking.

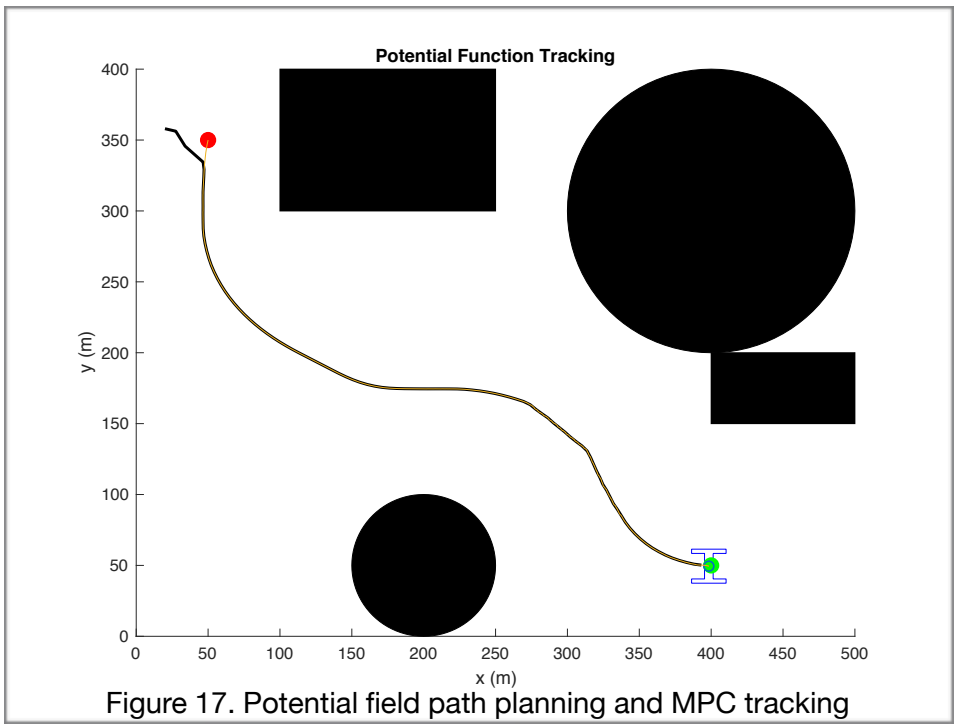


Figure 17. Potential field path planning and MPC tracking

We can see that using MPC tracking we can accurately follow the right trajectory.

6 Future works

We have successfully solved the problem of obstacle avoidance with randomly distributed obstacles unknown to the cart. However, in reality, we might face the kind of situation that the obstacles or other vehicles are moving in a very high speed in the environment. Also, we might face the problem of suddenly appearing obstacles or vehicles. Therefore, our future works should be designing local dynamic obstacle avoidance MPC to detect the obstacles, generate avoidance path and do path following.

7 References

[1] Elon Rimon and Daniel E. Koditschek, “Exact robot navigation using artificial potential fields,” *IEEE Transactions on Robotics and Automation* 1991

[2] Nicoletta Bloise, Elisa Capello, Matteo Dentis and Elisabetta Punta, “Obstacle Avoidance with Potential Field Applied to a Rendezvous Maneuver”, *Appl. Sci.* 2017, *7*, 1042; doi:10.3390

[3] Y. bayma. A. Nilipour and C. Lelm, "A Locomotion Control Method for Autonomous Vehicles", *Ym. IEBE Conference on Robolics slid Autanation*, pp. 1315-1317.1988.

[4] W. Nelson and L Cox. "Local Path Control for an Autonomous Vehicles", *Pm. lEEE Conference on Robotics and Automation*, pp. 1504-1510,1988.

[5] Y. J. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, “A stable tracking control method for an autonomous mobile robot,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 1990, pp. 384–389.

[6] Dongbing Gu and Huosheng Hu, “Receding Horizon Tracking Control of Wheeled Mobile Robots”, *IEEE Transaction on Control System Technology*, Vol. 14, No. 4, July 2006