Robotics Planning and Navigation
Model Predictive Control for Holonomic Robots
Assignment 2
- Gowri, Sivani

---

1 Model Predictive Control (MPC)

Model Predictive control(MPC) is an iterative process of optimizing the predictions of robot states in the future limited horizon while manipulating inputs for a given horizon. We try to find an optimized path from start to goal, by first finding a path that follows all the constraints and then optimizing it again from the starting point to find a better path.

In this assignment, we implemented Model Predictive Control (MPC) to obtain an optimized trajectory for a holonomic robot for two different cases :

1. With obstacles
2. Without obstacles

We made use of optimization solvers like cvx in MATLAB for implementing MPC.

1.1 Holonomic MPC without Obstacles

1.1.1 Objective Function

We try to minimize the goal reaching cost function J:

$$(x_{robo}(t) - x_g)^2 + (y_{robo}(t) - y_g)^2$$

where, the $(x_{\text{robo}}, y_{\text{robo}})$ is a function of 't' or discrete time steps, and

$$(x_g, y_g)$$

is the goal in world coordinates.

Now, if a holonomic motion model is considered i.e. $\text{f}(\text{n1}, \text{n2} \ldots; \text{t}) = 0$, we get the following form for motion model:

$$x_{t+dt} = x_t + v_{xt} * dt$$
$$y_{t+dt} = y_t + v_{yt} * dt$$

velocity constraints are given by

$$0 \leq v_{xi} \leq v_{\max} \quad \forall i \in [1, n]$$

$$0 \leq v_{yi} \leq v_{\max} \quad \forall i \in [1, n]$$

1.1.2 Procedure

1. **Initialisation:** Coordinates and controls were initialized with the following values -

$$(x_{initial}, y_{initial}, vx_{initial}, vy_{initial}) = (0, 0, 0, 0)$$

We also initialized four waypoints through which the robot must pass, along with their time stamps and the number of steps into future as 61.

$$\begin{aligned}(wp_{x_0}, wp_{y_0}, wp_{t_0}) &= (2, 4, 10)\\ (wp_{x_1}, wp_{y_1}, wp_{t_1}) &= (5, 8, 15)\\ (wp_{x_2}, wp_{y_2}, wp_{t_2}) &= (6, 9, 30)\\ (wp_{x_3}, wp_{y_3}, wp_{t_3}) &= (10, 10, 60)\end{aligned}$$

Similarly, we included constraints on velocity and acceleration -

$$\begin{aligned} v_{\min} \leq v_{optimal_i} \leq v_{\max} \quad \forall i \in [1, n] \qquad & 0 \leq v_{optimal_i} \leq 10\\ a_{\min} \leq a_{optimal_i} \leq a_{\max} \quad \forall i \in [1, n] \qquad & -2 \leq a_{optimal_i} \leq 2\end{aligned}$$

We precompute the matrices A,Q,C before passing them into the cvx optimizer to account for the waypoints provided by the user. The following equations are used :

```
x_t = [(x_init - curr_step_x); ones_arr; zero_arr];
y_t = [(y_init - curr_step_y); ones_arr; zero_arr];

zero_arr = zeros((no_of_steps-1),1);
s_t_x = [x_t(2:end,1); zero_arr]*del_t;
s_t_y = [zero_arr; y_t(2:end,1)]*del_t;

%initial values of x_t(0) and y_t(0)
x_t_0 = x_t(1,1);
y_t_0 = y_t(1,1);

s_t=[x_t_0 *x_t(2:end); y_t_0 *y_t(2:end)]*del_t; %(x0-xg)*del_t

%Updating the values of A, Q and C matrices
A = A + s_t_x*s_t_x' + s_t_y*s_t_y';
Q = Q + 2*s_t;
C = C + x_t_0*x_t_0  + y_t_0*y_t_0;
```

Figure 1: Code snippet for calculation of A,Q,C

2. **Optimization:** We try to solve for a set of velocities with x and y components : $\dot{x_0}$,$\dot{x_1}$,$\dot{x_2}$,$\dot{x_3}$, ... $\dot{x_{n-1}}$ and $\dot{y_0}$,$\dot{y_1}$,$\dot{y_2}$,$\dot{y_3}$, ... $\dot{y_{n-1}}$ that would push the robo from any position in the world towards the goal (x_g, y_g).

The goal reaching cost function is quadratic in nature, so we linearize some of the constraints to turn it into a convex optimization problem

$$min(J) = min(x^T Ax + Q^T x + C)$$

These matrices are passed into the convex optimization solver cvx of MATLAB to obtain the optimal velocities with which the robot has to travel. Along with optimizing the cost function J, we also pass the above declared waypoints, velocity and acceleration constraints along with collision constraints to the solver.

1.1.3 Results

The optimal trajectory was computed using the procedure mentioned above.

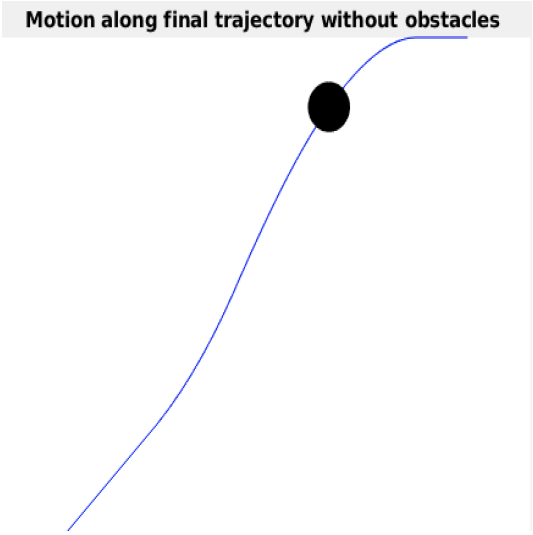

Figure 2: Motion along final trajectory without obstacles

1.2 Holonomic MPC with Obstacles

1.2.1 Objective Function

We try to find an optimal trajectory by adding circular obstacles. For obstacle avoidance, we use a euclidean distance constraint using the following formula for each obstacle in addition to the above optimization without obstacles. When a static obstacle of radius r_1 is located at (x_o, y_o) then the euclidean constraint is given by:

$$(x_i - x_o)^2 + (y_i - y_o)^2 \geq r_1^2 \quad \forall i \in [1, n] \tag{1}$$

1.2.2 Procedure

1. **Initialisation:** Similar to the case without obstacles the following initializations were being made. Coordinates and controls were initialized with the following values -

   $$(x_{initial}, y_{initial}, xvelocity_{initial}, yvelocity_{initial}) = (0, 0, 0, 0)$$

   We also initialized two waypoints through which the robot must pass, along with their time stamps and the number of steps into future as 61.

   $$(wp_{x_0}, wp_{y_0}, wp_{t_0}) = (5, 4, 25)$$
   $$(wp_{x_1}, wp_{y_1}, wp_{t_1}) = (10, 10, 60)$$

   Similarly, we included constraints on velocity and acceleration -

   $$v_{\min} \leq v_{optimal_i} \leq v_{\max} \quad \forall i \in [1, n] \qquad 0 \leq v_{optimal_i} \leq 10$$
   $$a_{\min} \leq a_{optimal_i} \leq a_{\max} \quad \forall i \in [1, n] \qquad -2 \leq a_{optimal_i} \leq 2$$

2. **Obstacles:** Four different circular obstacles of different radii - 0.8,0.4,0.5,0.6 were placed at various locations.
3. **Optimization:** We try to solve for a set of velocities with x and y components : $\dot{x_0}$,$\dot{x_1}$,$\dot{x_2}$,$\dot{x_3}$, ... $\dot{x_{n-1}}$ and $\dot{y_0}$,$\dot{y_1}$,$\dot{y_2}$,$\dot{y_3}$, ... $\dot{y_{n-1}}$ that would push the robo from any position in the world towards the goal (x_g, y_g).

   The goal reaching cost function is quadratic in nature. So we linearize some of the constraints and make it a convex optimization problem

$$min(J) = min(x^T Ax + Q^T x + C)$$

   These matrices are passed into the convex optimization solver cvx of MATLAB to obtain the optimal velocities with which the robot has to travel. Along with optimizing the cost function J, we also pass certain waypoint, velocity and acceleration constraints to the solver.
   Additionally, considering the obstacles , we also added obstacle avoidance constraints as given in equation (1).

1.2.3 Results

First, the optimal trajectory was computed using the procedure mentioned above along with the configuration space for all the obstacles.

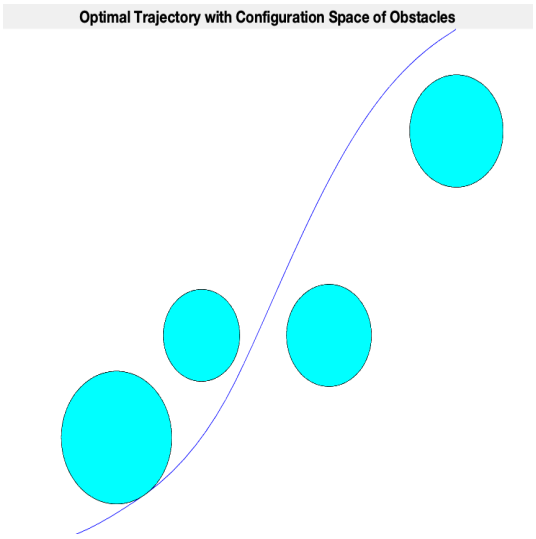

Figure 3: Computed Optimal Trajectory

A final optimal trajectory was obtained by plotting all the obstacles and showing the motion of the robot along the path.

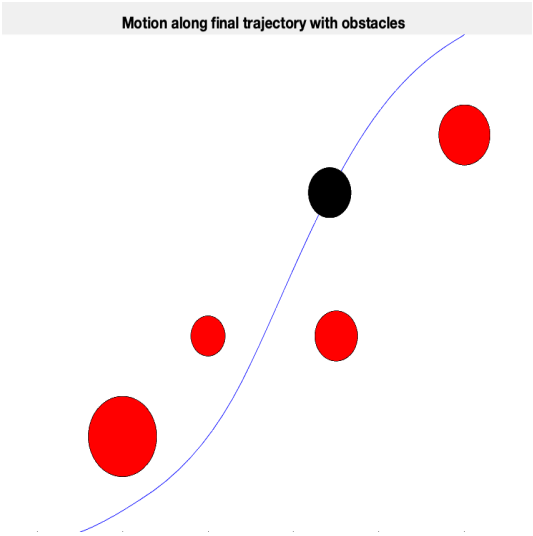

Figure 4: Motion along final trajectory with obstacles