Practical Course Project Report

Learning for self driving cars and intelligent systems

Yining Ma, Marc Brede

Technical University Munich

Contents

1 Week One

This week's task is concerned with Model Predictive Control of a vehicle. Using an optimization algorithm, a vehicle is to be guided from a starting position to the desired position. The first step was to come up with the vehicle dynamic equations that model the car's movement. The equations that we have used for this can be seen in the following.

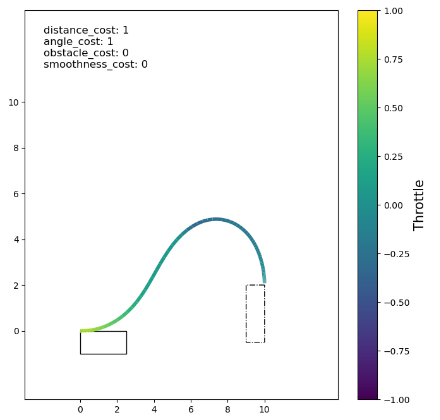
$$x_{t+1} = x_t + v_t \cdot \cos(psi_t) \cdot dt \tag{1}$$

$$y_{t+1} = y_t + v_t \cdot \sin(psi_t) \cdot dt \tag{2}$$

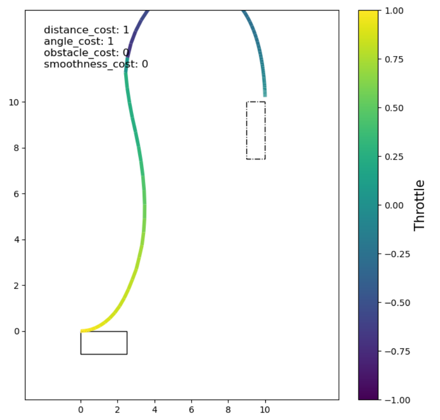
$$psi_{t+1} = psi_t + v_t \cdot \tan(steering) \cdot 0.5 \cdot dt \tag{3}$$

$$v_{t+1} = 0.99 \cdot v_t + pedal \cdot dt \tag{4}$$

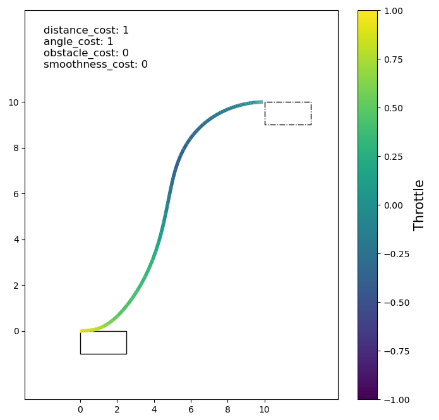
We want to optimize for the two variables that we can directly influence: steering and pedal. We came up with a cost function that penalizes the distance and the angle difference from the current position of the vehicle to the desired position, see equation 12. Both of these types of costs have their weighting factor w_{pos} and w_{orient} which we initially set to one. Using different desired end-position p_{des}, figure 1 shows the trajectory and pedal input that the vehicle took during the optimization of the cost function.



(a) Link to video



(b) Link to video

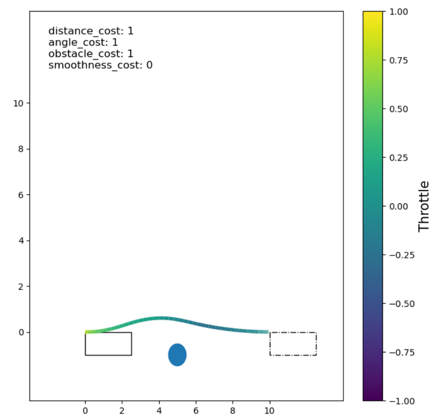


(c) Link to video

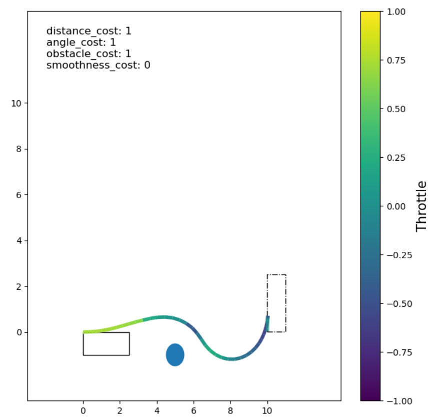
Figure 1: Trajectory and pedal input visualized for different desired positions

In the next step we introduced obstacles that the vehicle has to go around. If the vehicle gets closer to an obstacle than a minimum distance dis_{min}, it gets penalized by the inverse of its distance to the obstacle times a weighting factor w_{obs}, see 13. The minimum distance dis_{min} was set to 1.5 and the weighting factor w_{obs} to 1. The results can be seen in Figure 2.

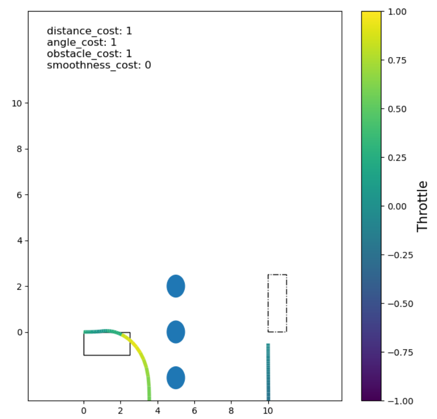
As the final adjustment, we introduced the notion of smoothness, i.e. the perceived comfortableness of the vehicle's passengers. It can either be modeled using the derivative of the vehicle's velocity w.r.t. time, or it can be modeled with the derivative of the pedal input w.r.t. time. The time-derivative of the pedal input is proportional to the change in acceleration, i.e. second time derivative of the vehicle's velocity. We argue that the change in acceleration is the perceived comfortableness of the passengers. The same principle applies to the perceived comfortableness of the steering. Therefore, we modeled the smoothness by penalizing a change in the steering and the pedal input from t to t+1 with a weighting factor w_{smooth}, see equation 7. Different values for this weight lead to different trajectories, as can be seen in figure 3.



(a) Link to video

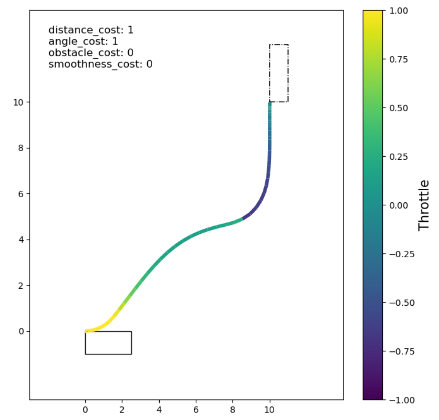


(b) Link to video

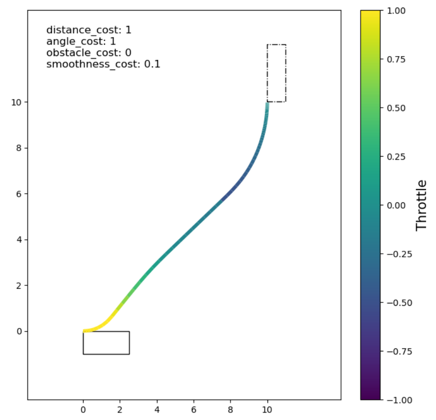


(c) Link to video

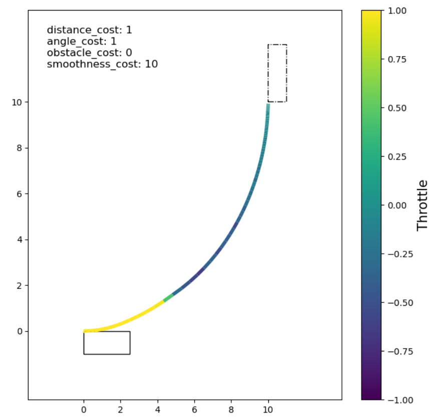
Figure 2: Vehicle avoiding obstacles while going to the desired position



(a) Link to video



(b) Link to video



(c) Link to video

Figure 3: Trajectory of the vehicle for different smoothness values

$$cost_1 = \|p_{des} - p_{curr}\|_2 \cdot w_{pos} + \|\angle_{des} - \angle_{curr}\| \cdot w_{orient} \tag{5}$$

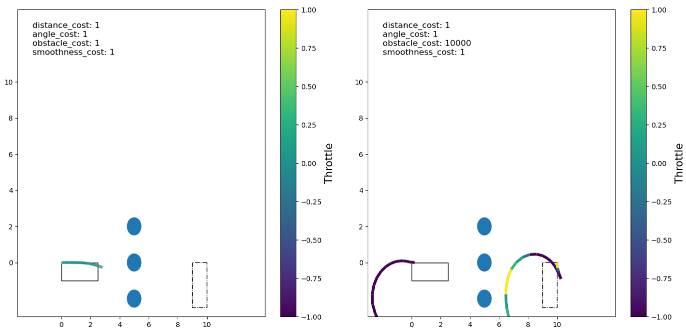
$$cost_2 = cost_1 + \sum_o \begin{cases} \|p_o - p_{curr}\|_2^{-1} \cdot w_{obs} & \text{if } \|p_o - p_{curr}\|_2 < dis_{min} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$cost_3 = cost_2 + \|steering_{t-1} - steering_t\| \cdot w_{smooth} + \|pedal_{t-1} - pedal_t\| \cdot w_{smooth} \tag{7}$$

There are some limitations to these optimizations and whether the algorithm can guide the vehicle to the desired position depends on the setting. Additionally, it highly depends on the weighting factors that we choose and we realized that some weighting factors work better than others. One observation that we made is that the vehicle tends to get stuck in obstacles often. Choosing a weighting factor w_{obs} that is much higher than 1 improved this, see figure 4.

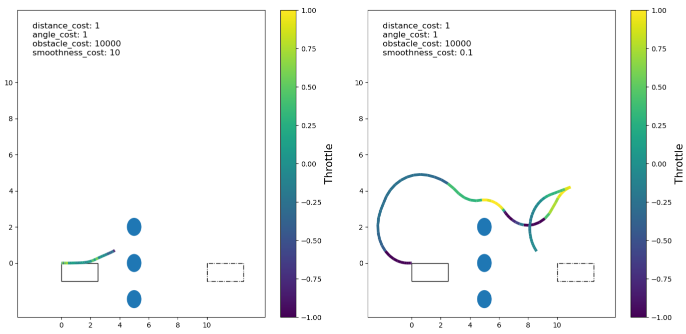
Another parameter that we were able to improve is the smoothness weight w_{smooth}. While higher values yields a smoother trajectory, often the algorithm gets stuck in situations in which the vehicle does not move anymore. Choosing a lower value for w_{smooth} improved this problem, see figure 5.

Finally, we have also observed that the optimization can get stuck in local minima if the value for the horizon is not big enough. Often, the optimization algorithm has to take steps against the gradient to get out of local minima. The optimization will only go against the gradient if it finds a position in the search range that yields an even lower value. The higher the horizon, the more likely is the optimization to find such an lower value, see figure 6.



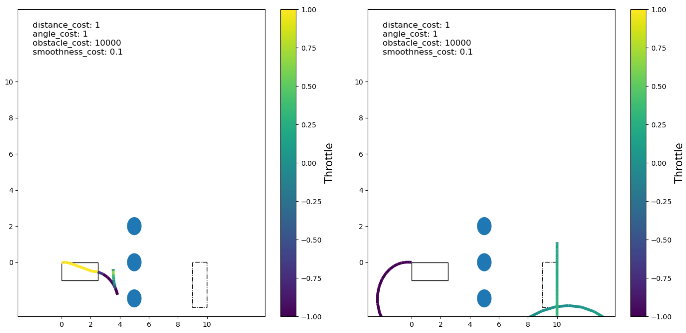
(a) Link to video (b) Link to video

Figure 4: Vehicle getting stuck in obstacle for low values for w_{obs} while for higher values it does not



(a) Link to video (b) Link to video

Figure 5: Vehicle getting stuck/not stuck depending on the value for w_{smooth}



(a) Link to video (b) Link to video

Figure 6: Vehicle getting stuck/not stuck in local minima depending on the horizon. Image a with a horizon of 5 and image b of 30.

2 Week Two

Last week we implemented the Model Predictive Control for one vehicle to guide it to the goal avoiding the obstacles with a good driving behavior. However, the obstacles in this scenario are all static obstacles which are unable to move. This week, we do one step further, meaning we have multiple vehicles scenario and each of the vehicle has its own start and goal. Moreover, we have to make sure that there is no collision happening among these vehicle so every vehicle can be regard as a dynamic obstacle of other vehicles. The dynamic equation of each vehicle is the same as the equation (1), (2), (3) and (4) shown in the report of week1, namely for vehicle i among N vehicles, we have following equations:

$$x_{t+1}^{(i)} = x_t^{(i)} + v_t^{(i)} \cdot \cos(psi_t^{(i)}) \cdot dt \tag{8}$$

$$y_{t+1}^{(i)} = y_t^{(i)} + v_t^{(i)} \cdot \sin(psi_t^{(i)}) \cdot dt \tag{9}$$

$$psi_{t+1}^{(i)} = psi_t^{(i)} + v_t^{(i)} \cdot \tan(steering^{(i)}) \cdot 0.5 \cdot dt \tag{10}$$

$$v_{t+1}^{(i)} = 0.99 \cdot v_t^{(i)} + pedal^{(i)} \cdot dt \tag{11}$$

We optimize the steering and pedal for all the vehicles at the same time based on the current state (position, orientation and velocity) and goal of each vehicle. Basically, we first need to make sure that all the vehicles drives towards their respective goal, this is achieved by the target cost shown in equation (12). Another requirement is that the vehicles should not collide with each other. So, the collision cost is introduced to penalize the situation when two vehicles are too close to each other as shown in equation (13). The total cost is just the sum of the target cost and the collision cost. Similar to what we did in the report of week 1, we have weighting factor w_{pos}, w_{orient} for target cost and w_{col} for collision cost, which adjusts how much we penalize for each cost. Initially, all the weighting factors are set to 1.

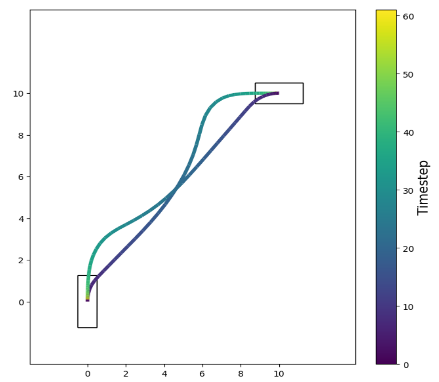
$$cost_1 = \sum_{i=1}^{N} \|p_{des}^{(i)} - p_{curr}^{(i)}\|_2 \cdot w_{pos} + \|\angle_{des}^{(i)} - \angle_{curr}^{(i)}\|_1 \cdot w_{orient} \tag{12}$$

$$cost_2 = cost_1 + \sum_{1 \le i < j \le N} \begin{cases} \|p_{curr}^{(i)} - p_{curr}^{(j)}\|_2^{-1} \cdot w_{col} & \text{if } \|p_{curr}^{(i)} - p_{curr}^{(j)}\|_2 < dis_{min} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

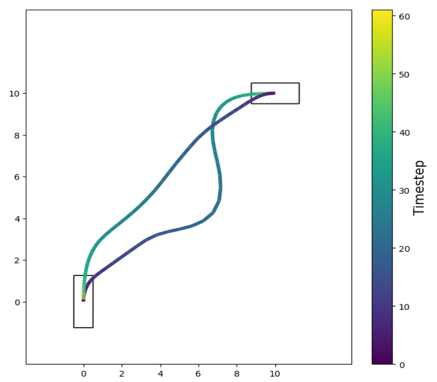
To start with an easy case, we first consider the 2-vehicle scenario. Since we want to check that the introduction of collision cost really keeps the vehicles from collision, we need to first make the 2 vehicles collide without collision cost and then introduce the collision cost. The result is shown in figure 7 (a) and (b). In the figure, we can see that all the trajectories are plotted with the color from blue to yellow, representing the moving of the vehicle from start time $t = 0$ to end time $t = t_{end}$. Although the trajectories lines intersect with each other, the real crash happens only when the lines intersect with the similar color and intersecting of the trajectories with different colors does not mean the crash of the vehicle.

Then, we notice that before we introduce the collision cost, we already have the trajectories and control input history of the two vehicle under the case without collision cost. We consider to use these control inputs as an initialization for optimizing the case with collision cost in the hope that it will lead to a faster and better solution.

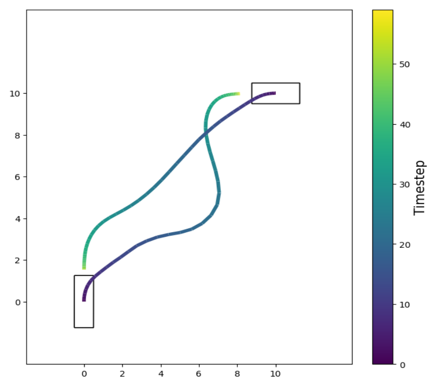
To our disappointment, when we use the no-collision-cost control inputs as initialization, the vehicle will always get stuck or overshoot near the goal (see figure 7 (c)), which will never happens if we use all-zero-array as initialization (see figure 7 (b)).



(a) no collision cost (link to video)

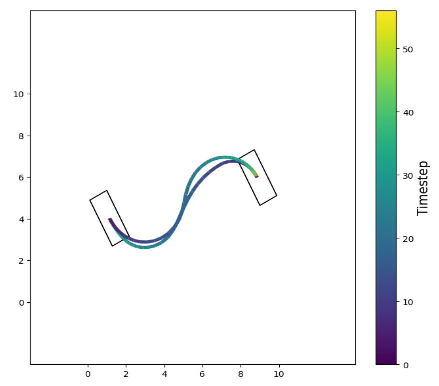


(b) with collision cost (link to video)

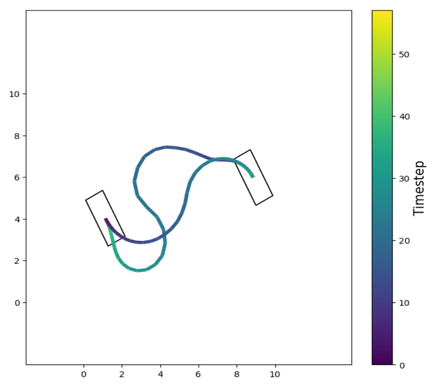


(c) with initialization (link to video)

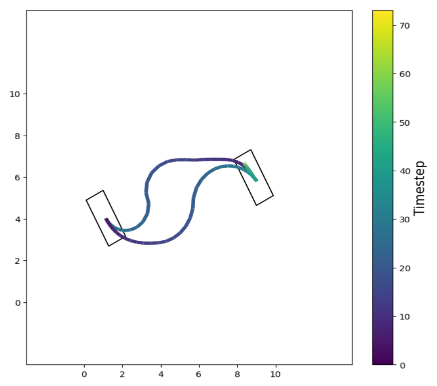
Figure 7: Trajectory of the vehicle with or without collision cost (case 1)



(a) no collision cost (link to video)

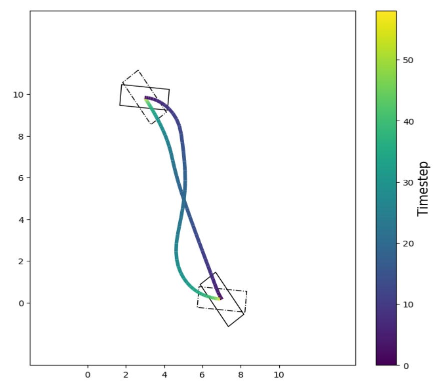


(b) with collision cost (link to video)

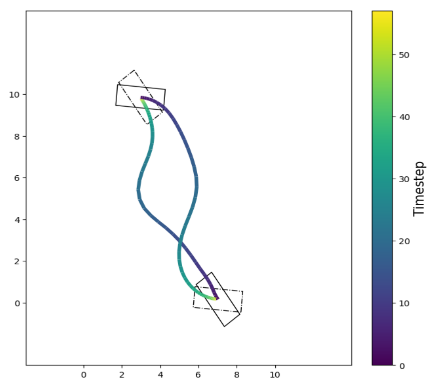


(c) with initialization (link to video)

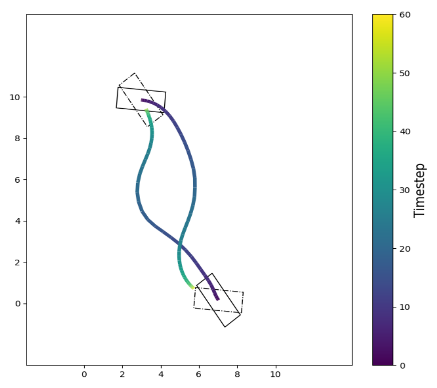
Figure 8: Trajectory of the vehicle with or without collision cost (case 2)



(a) no collision cost (link to video)



(b) with collision cost (link to video)

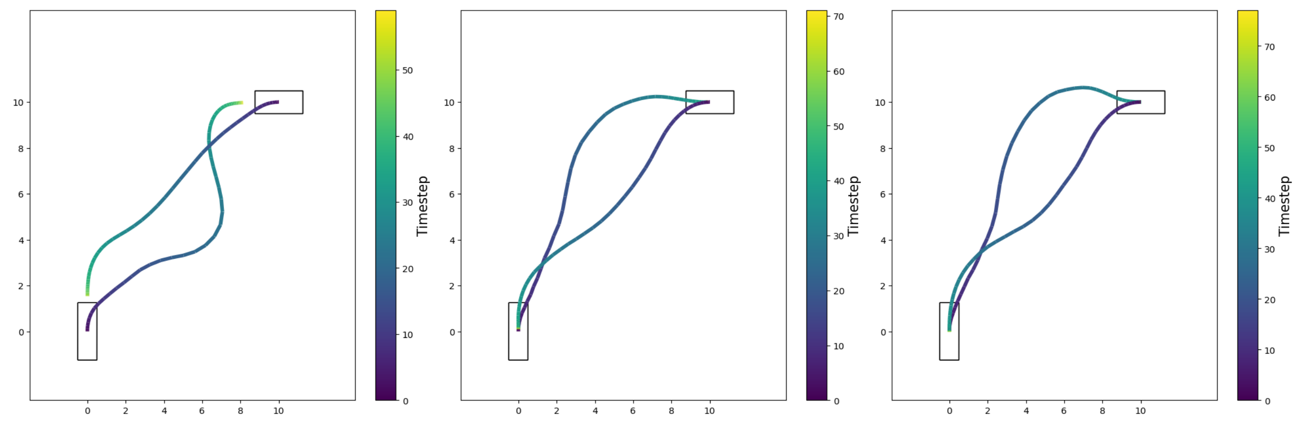


(c) with initialization (link to video)

Figure 9: Trajectory of the vehicle with or without collision cost (case 3)

To make sure the above mentioned case is not a coincidence, we try some other random chosen start positions and target positions. However, we notice that if we choose random start positions and target positions for both two vehicles, it is most likely that they will not collide with each other at all. To increase the probability that the two vehicles will collide and need our collision avoiding algorithm, we just choose the start position and target position for vehicle 1. For vehicle 2, we just use the target position for vehicle 1 as its start position and the start position for vehicle 1 as its target. Then we run the simulation and the results is shown in figure 8, 9.

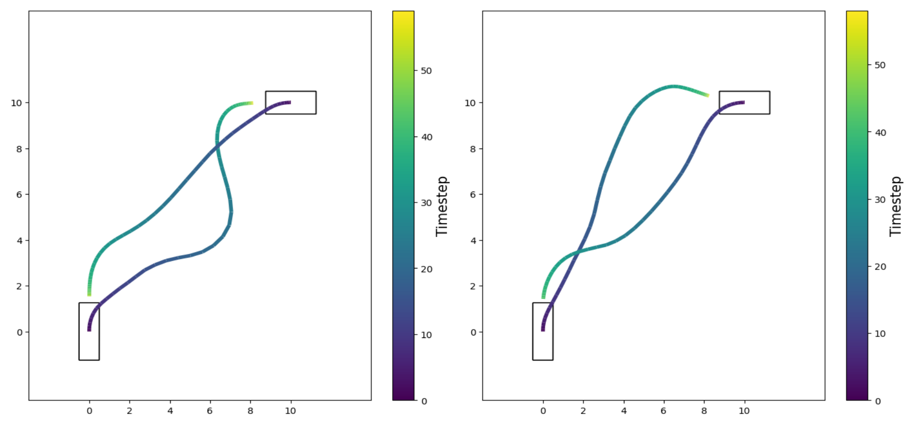
From the results of figure 7, 8 and 9, we notice that there is always an offset to goal when we use no-collision-cost control inputs as initialization for the case with collision cost. Actually, this offset appears also when we use all-ones-array as initialization. Intuitively, we want to fix this problem by adjusting our cost function. We increase all the weighting factor w_{pos}, w_{orient} and w_{col} from 1 to 100 and 10000, the result in figure 10 shows that the offset to the goal decrease when we use a large weighting factor. Basically, a heavy penalize of the target cost will lead to the vehicle closer to the goal.



(a) weight factor 1 (link to video) (b) weight factor 100 (link to video) (c) weight factor 10000 (link to video)

Figure 10: Trajectory of the vehicle with initialization using different weighting factor

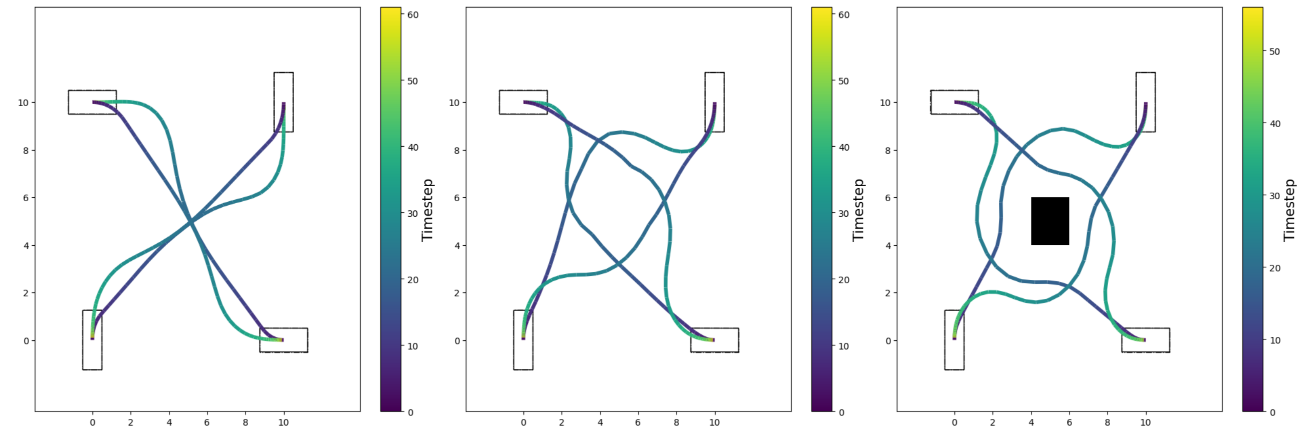
Additionally, we change the L2-Norm $\|p_{des}^{(i)} - p_{curr}^{(i)}\|_2$ in equation (12) to be L1-Norm $\|p_{des}^{(i)} - p_{curr}^{(i)}\|_1$ to see if it will help to reduce the offset to the goal when using small weighting factor (weighting factor $= 1$), since the L2 Norm penalizes the cost heavily when the vehicle far away from the goal but penalize less than L1 Norm when the vehicle near the goal. However, the result in figure 11 shows an obvious offset to the goal no matter we using L1 Norm or L2 Norm.



(a) L2 Norm (link to video) (b) L1 Norm (link to video)

Figure 11: Trajectory of the vehicle with L1 or L2 norm in target cost

Finally, we move to the 4-vehicle scenario, namely the 4 vehicles are initially placed on the 4 corners and each vehicle are set to target to its goal at another side of the diagonal. The result is just what we expected. Without introduction of collision cost, the 4 vehicles crash into each other at the center (see figure 12 (a)). However, after we add the collision cost, the 4 vehicles are all able to reach the goal successfully without collision. In the figure 12 (b), we observe that the trajectories of the vehicles roughly simulates a driving behavior at a roundabout. To simulate how will the vehicles drive at a roundabout, we remove the collision cost, introduce a static obstacle in the middle and set up the obstacle cost mentioned in equation (6) of report week 1. The simulation result is shown in figure 12 (c). Although we cancel the collision cost in this simulation, the vehicles obtained a similar result as if we have the collision cost, which indicates that using a roundabout is an efficient way to guide the flow of traffic and somehow solve the traffic problem at crossroad instead of using traffic lights which would mean that some of the cars have to stop, which is exactly what we do in the real world.



(a) no collision cost (link to video) (b) with collision cost (link to video) (c) with roundabout (link to video)

Figure 12: Trajectory of the vehicle with or without collision cost (4 vehicles)