Lilou Gras

Optimization for Robot Motion Planning and Control – report

Replication study: Optimal Trajectory Planning for Autonomous Driving with Logical Constraints: An MIQP Perspective

Introduction:

This report presents a replication of the research conducted in the paper "Optimal Trajectory Planning for Autonomous Driving Integrating Logical Constraints: An MIGP Perspective". The objective of this study was to reproduce the methodology presented in the original paper.

Brief description of the paper:

This paper tackles the challenge of generating optimal trajectory for autonomous driving. They introduce their paper discussing the current methods of model predictive control (MPC) and present their contribution as a novel approach using Mixed Integer Quadratic Programming (MIQP). Then, they demonstrate their result in different scenarios recognized as challenging in autonomous driving.

The scenarios:

In this report, only two out of four scenarios were replicated. For all the scenarios, the same cost function is applied, here is a screenshot directly from the paper:

$$J = \sum_{k=0}^{K} q_1(v_x(k) - v_r)^2 + q_2 a_x^2(k) + q_3(y(k) - y_r(k))^2 + q_4 v_y^2(k) + q_5 a_y^2(k) + r_1 j_x^2(k) + r_2 j_y^2(k), \quad (9)$$

The first scenario is simple, the car is starting at a velocity of 15 m/s and encounters a speedbump further down the road that starts at x = 30 and ends at x = 50. The car has to reduce its velocity to under 10m/s when driving over the speedbump. Following is a screenshot from the paper describing this scenario through the big-M method:

$$\begin{aligned}
&\delta_1(k) = 1 \Rightarrow x(k) \geq 30, \\
&\delta_2(k) = 1 \Rightarrow x(k) \leq 50, \\
&\delta_3(k) = 1 \Rightarrow v_x(k) \leq 10, \\
&-\delta_1(k) + \delta_3(k) \leq 0, \\
&-\delta_2(k) + \delta_3(k) \leq 0, \\
&\delta_1(k) + \delta_2(k) - \delta_3(k) \leq 1.
\end{aligned}$$

In here, we have δ to be binary arrays. We can understand from this equation that δ_1 is 1 when the car is further than 30 meters, δ_2 is 1 when the car has not reached 50 meters yet and 0 then and δ_3 is 1 when the speed of the car is below 10m/s. Finally, you add constraints on the binary arrays such that when the car is at the speed bump, the velocity is less than 10m/s.

The second scenario concerns obstacle avoidance. In this scenario, first the obstacles are static and can be viewed as parked vehicles. Following is a screenshot from the paper describing this scenario through the big-M method:

$$\delta_1^\nu(k) = 1 \Rightarrow x(k) \leq x^\nu(k) - L^\nu, \quad \text{(10a)}$$
$$\delta_2^\nu(k) = 1 \Rightarrow x(k) \geq x^\nu(k) + L^\nu, \quad \text{(10b)}$$
$$\delta_3^\nu(k) = 1 \Rightarrow x(k) \leq y^\nu(k) - W^\nu, \quad \text{(10c)}$$
$$\delta_4^\nu(k) = 1 \Rightarrow x(k) \geq y^\nu(k) + W^\nu, \quad \text{(10d)}$$
$$\delta_1^\nu(k) + \delta_2^\nu(k) + \delta_3^\nu(k) + \delta_4^\nu(k) = 1. \quad \text{(10e)}$$

In this scenario, we have 4 binary arrays. L^v and W^v corresponds to the length and width of the obstacle respectively while x^v and y^v is the position of the obstacle. Here, $L^v = 10m$ and $W^v = 2m$. We can see from the constraints that the car cannot be simultaneously at the same $x \pm L$ location and $y \pm W$ location as the obstacle. Once we achieved a good result with the static obstacles, we made them dynamic with a constant velocity.

Methodology:

In order to solve and implement their approach, they used a powerful optimizer: Gurobi. They added the aforementioned constraints and cost function as well as the following constraints for lower and upper bonds:

$$\underline{\mathbf{x}} = [0, 0, -4\,\mathrm{m/s^2}, 0, -2\,\mathrm{m/s}, -1\,\mathrm{m/s^2}]^T,$$
$$\overline{\mathbf{x}} = [\text{free}, 20\,\mathrm{m/s}, 3\,\mathrm{m/s^2}, 5\,\mathrm{m}, 2\,\mathrm{m/s}, 1\,\mathrm{m/s^2}]^T,$$
$$\underline{\mathbf{u}} = [-3\,\mathrm{m/s^3}, -2\,\mathrm{m/s^3}]^T, \overline{\mathbf{u}} = [3\,\mathrm{m/s^3}, 2\,\mathrm{m/s^3}]^T,$$
$$\underline{\theta} = -0.4\,\mathrm{rad}, \overline{\theta} = 0.4\,\mathrm{rad}, \underline{\omega} = -0.26\,\mathrm{rad/s}, \overline{\omega} = 0.26\,\mathrm{rad/s},$$
$$q_1 = 1, q_2 = 2, q_3 = 1, q_4 = 2, q_5 = 4, r_1 = 4, r_2 = 4.$$

TABLE I: Parameters used for case study

Results:

For the speedbump scenario, here are the result I was able to obtain (all the codes are in the same folder as this report):

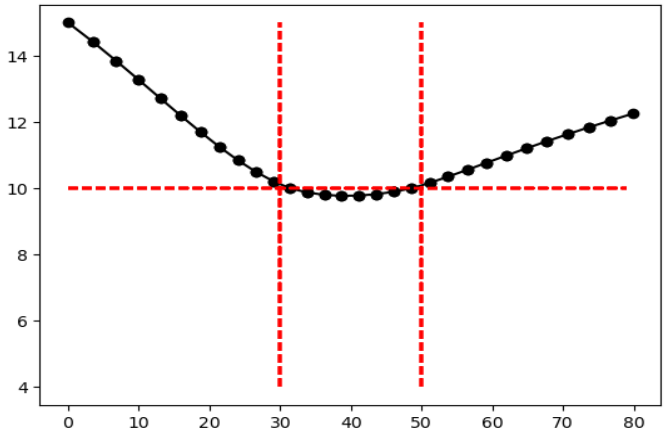
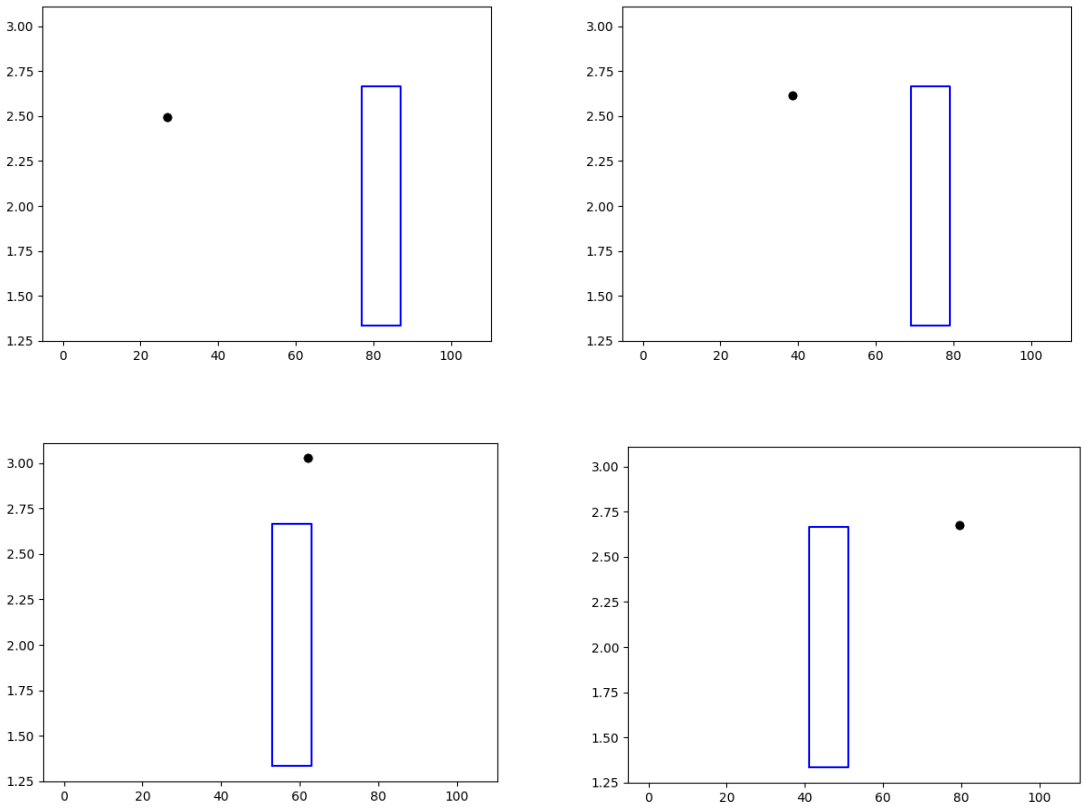


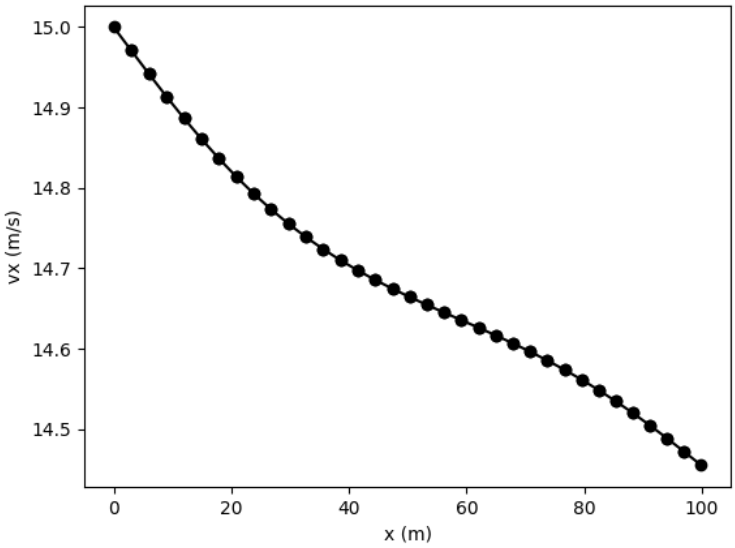
Figure 1: Velocity (y axis m/s) with respect to position (x axis, in m)

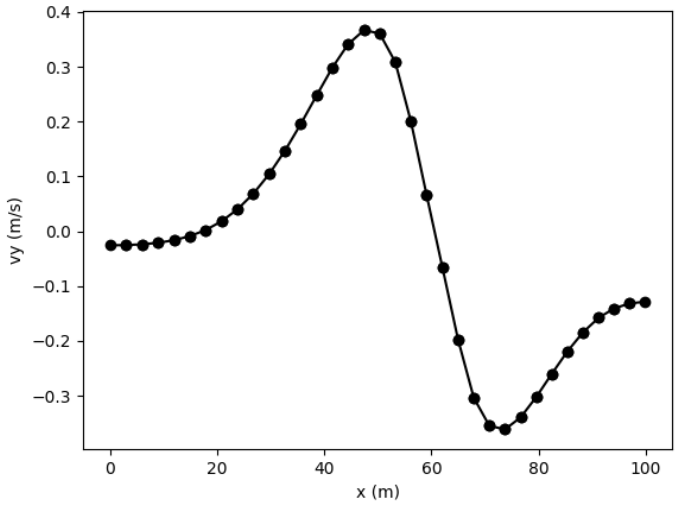
Here, we can see that the velocity goes below 10m/s when reaching the speedbump and increases again to the expected 12m/s after the car passes the speedbump.

Then, we have the second scenario with moving obstacles, the results are clearer if you launch the code but here is an attempt to show them:



We can see that the obstacle is moving from around x = 90 to x = 50 and that our vehicle (black dot) is avoiding it while also moving forward. Here are the x and y velocities of our vehicle:





We can see our vehicle is slightly slowing down and going up then down while avoiding the obstacle.

Note:

The code corresponding to the speedbump is speedbumpv2.ipynb and the code corresponding to the second scenario is avoiding_ncars.py