Planning and Decision Making

Project Report Group 19

{Marnix Enting ID: 4659147, Pavan Prasad H G ID: 5508053, Tishar Sinha ID: 5277876, Abhishek Kumar Kejriwal ID: 5217555}

*Abstract*— The traffic in a modern metropolitan city have become a topic of concern these days and it creates a sense of stress and discomfort in driver’s mind which eventually affects their driving style. It also makes the parking very difficult which works like blessing in disguise for the rapid development of automated parking systems. Automated parking systems or APS is a latest and reassuring area in the field of Automation system engineering. Our Motivation for this project was to create an automated valet parking system which generates the shortest path trajectory for the car avoiding all the collision after finding the empty parking slots. A perpendicular parking lot environment which constitutes of few obstacles is created with multiple parking slots. bi-Directional A\* method is used for path planning along with Model Predictive Control (MPC) was used as a controller. Results have been presented for the same.

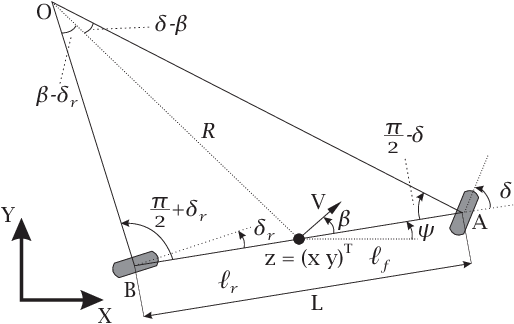
# Introduction

The concept of vehicle automation, automated mobility or connectivity has brought the change of how vehicle transportation was looked upon and interpreted. Similarly, a new field of study has arisen in the recent times called Automated Parking System or APS. Automated Parking system was introduced in 1905 in Paris, France at the Garage Rue de Ponthieu, this APS had elevators that transported unaccompanied cars to an upper deck which was then driven to an empty parking slot by an attendant. Since then numerous APS systems have been developed and introduced in different parts of the world by making the system more robust, smooth and fully automated. Automatic car parking systems can be broadly divided into two classes: Semi-Automatic parking systems and Fully Automatic Parking systems. Under Semi-automatic the human operator is still a part of the loop, like pushing a button to reach the desired parking space whereas this isn’t the case for fully automatic systems. This system consists of various electro-mechanical structures, sensors, and computer-controlled robots. Few advantages of such systems are that the parking space is utilized completely, and it provides a significant amount of space dedicated to parking. It also provides a safe and secure environment as there’s no human intervention.

Under this project we successfully developed the environment and implemented path planning algorithm along with MPC controller for tuning. To get a gist of the topic further literatures study, blogs and preexisting implemented algorithms helped us solidify our knowledge on the different methods of path planning and controllers. One of the papers which caught our attention was *Automatic parking and path following control for a heavy duty-vehicle* (2017). Under this work an automatic parking system was created using occupancy grid maps. A kinematic model of vehicle was utilized, and a gain scheduled linear quadratic controller with feedforward control action was considered. Further MPC control was used to limit the rapid changes with a maximum constraint.

1. Robot Model

For a robot model, we have chosen a car (without a trailer). A car is a non-holonomic system (no-slip conditions) that presents some non-holonomic constraints (cannot be expressed as a function of configuration alone). For example, if the velocity is required, e.g., f(q, d(q)/dt) = 0. Since the car is performing a parking maneuver i.e., moving at very low speeds, the effect of dynamics such as roll, slip angle etc. can be neglected. The steering wheel angle, throttle and brakes will be our primary control inputs to control the position, velocity, and acceleration of the model. For this motion planning experiment, we will be using the bicycle model to control the kinematics of the car.



*Figure 1: Kinematic Bicycle Model*

# Motion planning/Environment

The environment that we worked on for this project is a parking garage (*Figure2* for reference). The workspace is R2 and the configuration space is R2 x S1 for ground robots in a 2D planar world. The environment is built on an occupancy-based grid map. The parking space consists of multiple perpendicular spaces which along with the walls act as static obstacles. The number of empty parking slots along with the available parking slots are chosen randomly. The entire environment is rendered using OpenCV for visualization. Perpendicular parking is done when the cars are spaced in a line and there’s 90 degrees angle to the curb of the building and is an easier task when compared to other forms of parking. The static obstacles include the parked vehicles and the walls built in and around the parking garage area. For additional safety of the path, we added margin around the obstacles. The vehicle manouevring was developed in such a way to ease the complexity of the environment and augment the overall efficiency of the automated valet system. Multiple parking slots are left empty for the actual path planning.

Chart, table

Description automatically generated

*Figure2: Environment Outline*

Few of the planning algorithms were selected for this project work like A\*, RRT\* and bi-Directional A\*. It was found out that bi-Directional A\* algorithm works the fastest and is the most efficient for such applications. A\* algorithm is a simple algorithm which is used to find the best path between the initial node and goal node. A\* algorithm uses the weighted graphs and heuristic functions in its implementation. Best path here represents the optimal path which is the combination of nodes to create the shortest path with the least cost outcome in terms of distance and time. The cost here is the weight of each node.

F(n) = G(n) + H(n) -equation(1)

Where, F(n) = Final Cost.

G(n) = Cost of traversing from one node to another.

H(n) = Heuristic estimation of different nodes.

Unidirectional A\* algorithm guarantees an optimal route but requires considerable search time. On the other hand, bidirectional A\* algorithm, usually known faster than unidirectional A\*, does not guarantee the route found to be optimal, if the search ends when the forward and backward search meet in the middle (*Taeg-Keun Whangbo*(2007)). This signifies that the bi-directional A\* provides a quasi-optimal path and reduces the complexity of the search space. The implemented planner works in such a way that it computes the path for all the empty spots and then selects the one with the least route cost.

Similar to A\* algorithm bi-directional A\* also uses Heuristic functions to guide the search. We are using Euclidean distance as the heuristic function for our implementation. The parking maneuver was performed based on the relative position of the car with respect to the parking spots. We implemented our algorithm in such a way that it offsets the goal node present at the centre of the lane in front of the parking spot and then based on the path of the robot system it select the pre computed spline required to park the vehicle. In order to make the manoeuvring realistic we have defined a turning radius that mimics the real world cornering. The route between two nodes were interpolated to make it a smooth continuous path.

The kinematic model of the robot (car) was controlled using Model Predictive Control (MPC) controller. The basic working ideology of MPC is that it’s a feedback control algorithm that uses a model to make predictions about future outputs of a process. The advantages of using MPC are:

* A multivariable controller that controls the output simultaneously by considering all the interactions between system variables. Few of the benefits of choosing MPC controller are:
* MPC can handle multi input multi-output (MIMO) systems.
* It can regulate the constraints (in our case tracking down the desired trajectory while satisfying all the constraints).
* MPC has preview capabilities, which means it can integrate the future reference information to boost the system performance.

**Equations used:**

= v cos( - equation(2)

= v sin( - equation(3)

v' = a - equation(4)

= v tan (/ L - equation(5)

The state vector is z = [x, y, v, ψ]

The input vector is u = [a, δ]

Where,

= Yaw angle

= Steering angle

L = Wheelbase

v = heading velocity

a = acceleration

x = x position

y = y position

# Results

In this project, we design an automated valet parking system on a parking garage environment which was developed using a 1000 x 1000 occupancy grid map. Perpendicular parking was chosen over other forms of parking because of its easy and handy nature. Further Multiple parking slots were made available and randomly to check for the actual performance of the implemented path planner method. Along with successful implementation of bi-directional A\* algorithm it was also established that of all the selected path planners used for the project, bi-Directional A\* works the fastest for this application. The average computational time taken by the bi-directional A\* algorithm was around 10 seconds considering there were more than 10 spots empty.

MPC controller was used to control the Kinematic model of the car (non-holonomic system). It provides us a fairly robust and smooth trajectory for our robot system. A suitable iterative parking operation in the designed small environment is developed and modified by implementing multiple path planner methods and the one with the optimal solution was chosen. The definition of optimal solution for this project can be defined by the computational time, shortest path and the distance from the robotic system to the parking slot.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

*Figure3: Car X and Y position in time with*

*respect to reference.*

From the graphs above, it can be clearly visualized that the path planned by the planner algorithm is fairly followed in the real time by the robotic system in the simulated environment.

This further provides proof for our work that the planned trajectory for the robotic system was smooth. Additional insights can be inferred from the graphs on provided (to the right) showing car’s acceleration, position, angle, steering angle, and velocity with time.

Chart, line chart

Description automatically generated

*Figure4: Car acceleration with time*

Chart, line chart

Description automatically generated

*Figure5: Car position with time*

Chart, line chart

Description automatically generated

*Figure6: Car’s angle with time*

Chart, line chart

Description automatically generated

*Figure7:Car’s steering angle with time*

Chart, line chart

Description automatically generated

*Figure8: Car’s Velocity with time*

##### 

##### Chart, bar chart Description automatically generated

*Figure 9: Car Navigating to the nearest parking slot*

# Discussion

Few of the properties of bi-directional A\* is that it is complete and probabilistically complete. Though, the method is not necessarily optimal. RRT\* does guarantee an optimal solution, hence it theoretically has the advantage. However, practically RRT\* was not feasible due to high computational time. RRT\* required close to 100k iterations to find the route in such an obstacle space which takes over 20 mins of computation (even with a bias of 0.2). For this application, since the priority is computational time over the optimized path, we chose bi-directional A\*. The bi-directional A\* algorithm takes approximately 1-2 seconds to compute the path. This is well suited for the given problem.

Studies have been conducted in the field of finding optimal route for different robotic systems in various applications. Due to the rapid growth of ITS (Intelligent Traffic Systems), CNS (Car Navigation System) and APS (Automated Parking System) etc. it has become a necessity to look out for and work on efficient and optimal systems. This Project work provided us insights into APS, various path planning methods and Controllers. Our work also provides us further room for improvements in future work such as:

* Parking maneuver can be smoother and can be optimized for collision avoidance.
* Directionality of the car can be added (At the beginning of the manoeuvre the orientation of the car is not considered).
* Simultaneous implementation for multiple cars.
* Inclusion of dynamic obstacles
* Planning can be localized
* Can be implemented on a larger grid size such as airports, malls etc.

In our implementation we are assuming that all the vehicles are autonomous and an alternative which can be proposed for non-autonomous vehicles is to develop a platform which works on the same principle and navigates the car to the available parking slots.

##### References

1. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
2. Anastasia Razinkova, An Intelligent auto parking system for Vehicles, Kore university of technology and education, 2012.
3. Chenxu Li, Automatic Parking path planning and tracking control research for Intelligent vehicles, School of Automotive and Traffic Engineering, 2020.
4. Dr. Gene Yeau-Jian Liao, Automatic Parking vehicle systems, Wayne State University, 2016.
5. Mohammed Azher Therib, Design and implementation of smart car parking systems, Al-Furat Al-Awsat Technical University, 2016.
6. Joakim Mörhed and Filip Östman, Automatic parking and path following control for a heavy duty vehicle, Linkoping University, 2017.
7. René B.M. De Koster, Optimal design and planning for compact automated parking system, European Journal of Operational Research , 2018.
8. Letian Lin Jim Zhu, Path planning for autonomous car parking, Ohio University, 2018.
9. Zheng Li, A path planning algorithm for parallel automatic parking. Harbin Institute of Technology, 2013
10. Taeg-Keun Whangbo, Efficient modified bi-directional A\* algorithm for optional route finding, Kyungwon University, 2007
11. <https://github.com/pparmesh/Autonomous-Parking-AV>
12. <https://github.com/Pandas-Team/Automatic-Parking>
13. <https://nl.mathworks.com/help/driving/ug/automated-parking-valet.html>
14. <https://www.redblobgames.com/pathfinding/a-star/implementation.html>
15. <https://colab.research.google.com/github/RussTedrake/underactuated/blob/master/exercises/planning/rrt_planning/rrt_planning.ipynb>
16. [https://github.com/AtsushiSakai/PythonRobotics/blob/master/PathPlanning/](https://github.com/AtsushiSakai/PythonRobotics/blob/master/PathPlanning/RRTStar/rrt_star.py)
17. <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FKinematic-bicycle-model-of-the-vehicle_fig1_289523352&psig=AOvVaw2WQjbV-aDfPkhCKQ0DvynL&ust=1642087619983000&source=images&cd=vfe&ved=0CAwQjhxqFwoTCLDEiZzDrPUCFQAAAAAdAAAAABAF>